SAND2008-1053P

# Learning by Reading RFI – A Swarm Approach

## Key Insights

**Natural text is what humans have evolved for representing deep semantic knowledge. A swarm-like approach can process data in its native format.**

**Natural text is highly structured**

Natural text is the best way nature has found to represent deep semantic human knowledge. We should learn to process it in that form.

**A swarm-like approach:**

- Is amenable to integration of multiple participants
- Naturally supports parallel processing
- Naturally leads to the creation of emergent properties that could be hard to reproduce by brute force.

## Key Aspects of Approach

**Key Testing Approach:**

The goal is to produce a highly interactive system with which a person can converse in a reasonably natural way about the text the computer has processed.

**How it works:**

- A swarm of agents, each capable of some identifying some simple aspect of text, processes a large corpus of background information.
- A person presents some textual question or statement to the system, perturbing the agents in the swarm.
- The swarm processes the new text and as the system settles, the system responds to the user with a textual answer.

**Assumptions and limitations:**

- Don't assume that the computer's representation will be human readable.
- It would be difficult to trace the computer's "reasoning"

## Key Technical Challenges

- **Developing a Common Platform**
- **Producing a unified output from the underlying swarm of processes.**

This program will achieve the ability for a computer to process text and provide a highly interactive system for a person to interact with

# Learning by Reading RFI – A Swarm Approach

## Program Constraints

- Output of the system is natural text, not formal logic statements.
- The system is not expected to come to neat, clean answers. It is conversational.

## Metrics

Multiple Metrics:

•Question Answering

•Turing Test-like conversation

## Appropriate Sequence of Program Objectives

1. Development of a common platform for processing text in a swarm-like manner.
2. Development of problem sets to be processed
3. Development of computational units to be included as part of the swarm
4. Testing and iteration

## Target Input and Output

**Target Input**

- A corpus of natural text
- Statement from some human participant

**Target Output**

- Natural language statements
- Potentially fragments of text from original corpus

# Curriculum Vitae

Travis Bauer
tlbauer@sandia.gov

## Education

| | |
|---|---|
| 2003 | PhD<br>Computer Science and Cognitive Science<br>Department of Computer Science<br>Indiana University, Bloomington |
| 1995 | Bachelor of Arts<br>Departments of Computer Science and Religion/Philosophy<br>Jamestown College, Jamestown ND |

## Publications

Chew, P., Verzi, S., Bauer, T., and McClain, J. Evaluation of the Bible as a Resource for Cross-Language Information Retrieval. *Proceedings of the Workshop on Multilingual Language Resources and Interoperability, pages 68-74, Sydney, July 2006.*

Bauer, T., Laham, Darrel, Benz, Zach, Dooley, Scott, Kimmel, Joe. Text analysis and dimensionality reduction for automatically populating a cognitive model framework. in "Cognitive Systems: Human Cognitive Models In Systems Design." ed. Chris Forsythe, Mike Bernard, Tim Goldsmith. Lawrence Erlbaum Associates (November 9, 2005).

Bauer, T and Leake, D. Detecting Context-Differentiating Terms Using Competitive Learning. SIGIR Forum, Fall 2003.

Bauer, T. WordSieve: Context Analysis for Personal Information Retrieval. PhD Dissertation. Indiana University. May 2003.

Bauer, T. and Leake, D., Using Document Access Sequences to Recommend Customized Information, *IEEE Intelligent Systems Special Issue on Information Customization*, Nov/Dec 2002.

Fu, Y. and Bauer, T. and Mostafa, J., Concept Extraction and Association from Biomedical Literature, *Fourth International Workshop on Web Information and Data Management*, 2002.

Bauer, T. and Leake, D., Calvin: A Multi-Agent Personal Information Retrieval System, *Agent Oriented Information Systems 2002: Proceedings of the Fourth International Bi-Conference Workshop AOIS-2002*

Bauer, T. and Leake, D., Exploiting Information Access Patterns for Context-Based Retrieval, *Proceedings of the 2002 International Conference on Intelligent User Interfaces,* ACM Press, 2002, 176-177.

Bauer, T. and Leake, D., Real Time User Context Modeling for Information Retrieval Agents, *Proceedings of the 2001 ACM CIKM: Tenth International Conference on Information and Knowledge Management*, Association for Computing Machinery, 2001.

Bauer, T. and Leake D., WordSieve: A Method for Real-Time Context Extraction. *Modeling and Using Context: Proceedings of the Third International and Interdisciplinary Conference, Context 2001*, Springer-Verlag, 2001.

Bauer, T. and Leake D., A Research Agent Architecture for Real Time Data Collection and Analysis. In *Proceedings of the Workshop on Infrastructure for Agents, MAS, and Scalable MAS.* Montreal, Canada, 2001.

Leake, D., Bauer, T., and Maguitman, A. Capture, Storage and Reuse of Lessons about Information Resources: Supporting Task-Based Information Search. *Proceedings of the AAAI-2000 Workshop on Intelligent Lessons Learned Systems*, AAAI Press, Menlo Park, CA, 2000.

## Presentations

Bauer, T., Verzi, S., Basilico, J. Automated Context Modeling through Text Analysis, Cognitive Systems: Human Cognitive Models in System Design, Sante Fe. July 6-9, 2005.

Bauer, T. and Abbot, R. Automated knowledge elicitation: Current technology and future research., Cognitive Systems: Human Cognitive Models in System Design, June 29-July 1, 2004.

Bauer, T. Creating Human-Machine Systems for which Interaction is Between Real & Synthetic Cognitive Entities. Presented at: NNSA Future Technologies Conference. Track C: Future Trends in Rapid Design & Build Technologies. May 19, 2004.

Bauer, T. WordSieve: Learning task differentiating keywords automatically. Presented at *SIGIR Workshop on Implicit Measures of User Interests and Preferences*. August 1, 2003. Toronto, Canada. http://research.microsoft.com/~sdumais/SIGIR2003/SIGIR2003-ImplicitWorkshop.htm

Bauer, T. Distributed, Competition-Based Feature Extraction: Application to Personal Information Retrieval. Presented at: *Cognitive Systems: Human Cognitive Models in System Design*. Hosted by Sandia National Laboratories and the University of New Mexico. Santa Fe, New Mexico, June 30-July 2, 2003. http://www.unm.edu/cognitive_systems/

## Other Writing

*Temporal Difference Learning and Transcendental Idealism: Exploring Possible Collaboration*, Senior Thesis, Jamestown College, 1995.

## Professional Experience

Nov 2006 - Present **Principal Member of Technical Staff**
Sandia National Laboratories

Analysis of raw textual data for the extraction and determination of associative knowledge for use in psychologically plausible cognitive

models. For more information, see
http://www.sandia.gov/cog.systems/Index.html.

Leading team of developers in designing and developing reusable software library for performing text analysis. Engaged in business development activities to determine software requirements and develop software for deployment.

May 2004 - Nov 2006 **Senior Member of Technical Staff**
Sandia National Laboratories

Analysis of raw textual data for the extraction and determination of associative knowledge for use in psychologically plausible cognitive models. For more information, see
http://www.sandia.gov/cog.systems/Index.html.

Leading team of developers in designing and developing reusable software library for performing text analysis. Engaged in business development activities to determine software requirements and develop software for deployment.

June 2003 - May 2004 **PostDoctoral Appointee**
Sandia National Laboratories

Analysis of raw textual data for the extraction and determination of associative knowledge for use in psychologically plausible cognitive models. For more information, see
http://www.sandia.gov/cog.systems/Index.html.

September 2002 - May 2003 **Research Assistant**
Indiana University Computer Science, David Leake
Developing intelligent support software using concept maps.

January-May 2002 **Associate Instructor**
Indiana University
Knowledge Based Computation course

January-May 2000 **Research Assistant**
Indiana University
Designing and writing information retrieval software agent for aiding researchers and engineers. Supported by NASA as part of a project for knowledge capture, storage, and reuse

August-December 1999 **Associate Instructor**
Indiana University
Introduction to Computer Science, scheme programming

| | |
|---|---|
| 1989-1999 | **System Administrator**<br>Dan's Production Service, Williston, ND<br>Developed and maintained company's central database system and local area network. |
| 1995-1996 | **Instructor in Computer Science**<br>Jamestown College, Jamestown, ND<br>Taught fifteen credits in computer science department.<br>Co-taught course in Object Oriented Programming using C++. |

**Research Experience**

| | |
|---|---|
| Information Retrieval: | Algorithm development, Corpus Analysis, Document Parsing, Latent Semantic Analysis, Indexing and Retrieval, Search Engine Construction. |
| Human/Computer Interaction: | Software development for data collection in controlled Information Retrieval experiments, user profile development. |
| Multi-agent systems: | Multi-Agent architecture design, encryption and security, coordination. |
| Software Development | Worked with experts at Dan's Production Service analyzing information flow for oil well production reporting. Designed and implemented database and reporting software for company. |

**Skills**

| | |
|---|---|
| Languages: | Java, C++, C# |
| Systems: | Windows, Linux, Unix |
| Topics: | XML, TCP/IP networking, encryption, database development (experience with Clarion and Postgres), Latent Semantic Analysis, Window API |

The attached publication shows some relevant previous research the submitter has done. In the attached work, a set of simple agent-like entities were used to analyze a corpus as a stream and accomplish performance similar to that of latent semantic analysis.

# Using Document Access Sequences to Recommend Customized Information

**Travis Bauer and David Leake,** *Indiana University*

***WordSieve, a text analysis algorithm, uses a competitive-network-learning approach to learn topic-relevant keywords in real time with no predetermined corpus. You can use these keywords to form search engine queries to suggest relevant documents to the user.***

Information about a user's task can greatly enhance automated information customization. However, customization agents should learn unobtrusively without requiring users to provide task descriptions. WordSieve, a competitive-learning text analysis algorithm, identifies context-relevant terms and uses them to construct a model of a user's access patterns. WordSieve works in real time, while the user accesses documents, and requires no explicit user feedback. Initial tests demonstrate WordSieve's ability to identify useful context descriptions.[1,2] WordSieve's competitive-learning component also lets the algorithm naturally adapt to a user's shifting interests.

We are developing a personal information retrieval system, Calvin, which observes users while they browse the World Wide Web, analyzes the documents they access, does background searches, and suggests related pages. During this time, it builds user profiles that model a user's document access patterns. Figure 1 shows a screenshot of Calvin's Windows interface.

Calvin uses WordSieve to learn over time the terms that tend to indicate topics that interest the user. Consider, for example, a person reading a page about Indonesian cooking. A system that only considers the current page content might have difficulty determining how to customize future information presentation: The person might be interested in oriental cooking or might be trying to learn information about Indonesia in general. WordSieve enables Calvin to learn to distinguish whether the user tends to access documents about Indonesia or documents about cooking. It differs from other personal information agents (such as Letizia[3] and Watson[4]) in that it uses a competitive network to characterize topic access patterns over time.

## Generating context representations

WordSieve represents users and tasks by *term vectors*, where the terms' values indicate their significance in describing users or tasks. WordSieve learns keywords and keyword weightings that characterize the contexts within which an individual user tends to access documents. By context, we mean the sequences of documents a user accesses while pursuing a topic.

Other methods such as TFIDF (term frequency inverse document frequency)[5] and LSI (latent semantic indexing)[6] also look at term occurrences in and among documents. However, unlike WordSieve, they do not use the sequence of accesses. By looking not only at whether and how often a term occurs, but also when it occurs in a sequence of documents, WordSieve can improve indexing and retrieval performance. WordSieve can extract this sequentially available information through a competitive network process.

We based WordSieve's design on our hypothesis that good indexing terms are terms that occur frequently at some times, but seldom occur at other times. Terms such as "a" and "the" are not good context terms: They occur frequently, but they occur in most documents. Likewise, for a user who primarily reads political news, the terms "political" and "congress" might appear in most documents consulted, but do not help differentiate the user's context.


1094-7167/02/$17.00 © 2002 IEEE

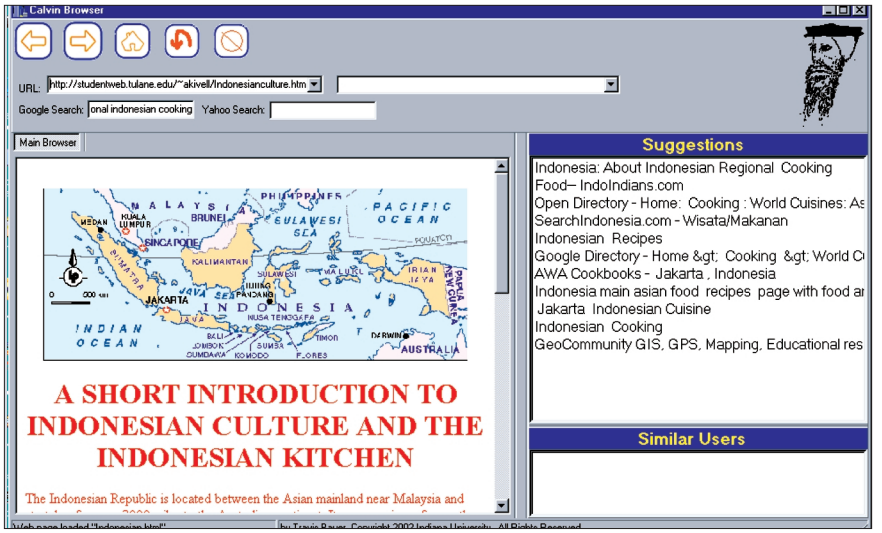


**Figure 1. The Calvin Windows interface.**

For example, Figure 2 shows a 40-minute period of Web browsing when a user spent 10 minutes exploring each of four different topics. The *x*-axis shows a point for every 75 terms. Each *y* value represents the number of times the term "george" occurred in the 75 terms preceding each point. As the figure shows, "george" occurs frequently during the Bush query, less frequently during the Gore query, and not at all elsewhere. Consequently, "george" works well in distinguishing the two contexts.

WordSieve selects terms that mark distinct document sequences for an individual user. This personal dictionary, which characterizes topics the user consults, also serves as a user profile. You can use this profile to generate vectors for various contexts and to generate queries for search engines. Each term chosen to represent a user serves as a dimension in a multidimensional vector space.

Other techniques also use the vector space model for representing documents. In this model, a term's weight in a particular context (a document, for example) is usually expressed as the product of a local and global weight for the term. The basic intuition is that a term's significance is a function of its significance in a document (the local weighting) and in general (the global weighting). A problem with TFIDF for information customization is that the global weights require global information about a larger corpus. In personal information retrieval, this global corpus might not be available and global weights must change as the user's interests change. WordSieve does not require global information to construct global weightings for term vector production. It learns global weights by reading user-accessed documents term by term.

## WordSieve architecture

WordSieve capitalizes on document sequences through a stochastic, competitive process. This process can identify which words occur frequently (for the local term weight) and which words indicate contexts (for the global frequency weight).

WordSieve uses a three-layer network (see Figure 3). WordSieve's basic model is a stream of terms passing across a network of units. Some terms get "caught" in the network, but most pass through. The caught terms form a user profile of user interests that you can use to generate queries and indices.

Each layer serves a different purpose. As a user browses online,

- Layer 1 identifies words that are occurring frequently in a stream of documents.
- Layer 2 identifies words that often occur frequently, even though they might not be now.
- Layer 3 identifies the words in layer 2 that tend to occur infrequently.

Each unit in layer 1 is associated with three different values:

- A *term* is learned from the documents passing through WordSieve.
- An *activation* indicates the unit's probability of becoming bound to a different term.
- A *priming* determines the rate at which the activation level changes. We choose this value to provide hysteresis to help overcome the effect of noise.

Activation varies between 0 and 1. Priming varies between –1 and 1.

Each layer has a limited number of units, and terms effectively compete for the units in the network. Because of the network's updating strategy, you can use the terms that win this competition and their associated activation levels to compute the terms that often vary between frequent and infrequent occurrence. We use the resulting set of terms as a profile of topics the user typically accesses, to generate queries and indices from documents. All three values persist across document presentations. The layers are connected in the following way:

- *Layers 1 and 2*. Units in layer 1 are connected to any units in layer 2 that are bound to the same term. Layer 2 units might not have corresponding layer 1 units, but if a term's unit has a high activation level in layer 1, its activation in layer 2 grows faster.
- *Layers 2 and 3*. Layers 2 and 3 are coupled together. For every unit in layer 2, a corresponding unit exists in layer 3. Their activation and priming values can vary independently, but if a unit gets bound to a new term in layer 2, its corresponding unit in layer 3 gets bound to the same term. Layers 2 and 3 differ in their update functions, however.

### Network processing

WordSieve processes each word individually, following the order of occurrence in each document.

***Layer 1.*** For every term presented to layer 1, WordSieve adjusts a unit's activation as a function of the priming so that $A_i \leftarrow A_i + P_i \times MaxActivationRate$, where $A_i$ is the activation of unit *i*, $P_i$ is the priming of unit *i*, and *MaxActivationRate* is the activation's highest rate of change. If the priming is positive, the activation will increase. If the priming is negative, the activation will decrease. If the unit's priming is at its extreme values (–1 or 1), the unit's activation will change at a rate of *MaxActivationRate* per term presented to the network.

The priming also changes for each term presented. For each term presented, the priming decreases: $P_i \leftarrow P_i - MaxPrimingRate \times BaseFrequency$, where *MaxPrimingRate* is the priming's maximum rate of change and the *BaseFrequency* represents a term's required minimum frequency in the document stream for the priming to remain constant. If a unit is sensitized to the term presented, WordSieve also increases its priming by $P_i \leftarrow P + MaxPrimingRate$.

If the term is not bound to any unit in the network, it takes over the unit with the lowest activation.

The effect of this policy is that if the term occurs at a rate of *BaseFrequency* per 100 terms, the unit's priming remains constant. If it occurs less, the priming will decrease, and if it occurs more, the priming will increase.

***Layers 2 and 3.*** Layer 2 functions somewhat differently. If the term is in layer 2 but not in layer 1, the level 2 unit's activation decreases at a constant rate. If the term is in layer 1, the priming and activation increase. Also, for layers 2 and 3, the priming values are limited to 0 or 1.

In layer 3, the opposite happens. If the term is in layer 1, the corresponding unit's activation in layer 3 decreases at a constant rate. If it's not in layer 1, the layer 2 unit's activation increases as a function of priming.

If the term is not in layer 2 and its activation in layer 1 is above a predefined constant, WordSieve removes the term with the lowest activation in layers 2 and 3 and replaces it with the new term.

## Applying the network

Figure 4 displays snapshots of some of WordSieve's units during information access. In this diagram, each block corresponds to a unit. The term that unit is associated with is printed in the block's center. The background's brightness indicates the term's weight in this layer. Figure 4a shows part of layer 1. The brightest terms (those terms that occur most frequently) include "texas," "political," "george," "bush," "son," "family," and "dynasty." This first layer comprises the local weights, identifying the words that are currently occurring frequently. This layer changes rapidly corresponding to the changing documents the user browses.

WordSieve calculates global weights from the second and third layers. By design, layers 2 and 3 change more slowly, serving as a "long-term memory." Figure 4b shows layers 2 and 3. The background's brightness corresponds to the term's activation level in layer 2. The text's brightness indicates the term's degree of activation in layer 3. This profile of context-discriminating terms builds up over time as the system observes document accesses. You can see that "texas," "political," and "george" have high activation levels in both layers 2 and 3 (indicated by brightly colored text and background).

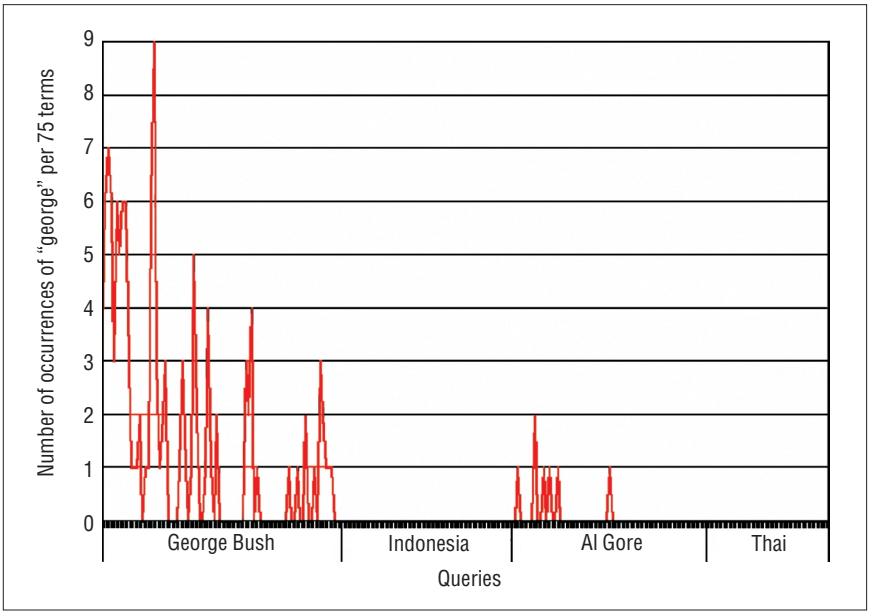


**Figure 2. Frequency of the term "george" per 75 terms encountered while browsing four topics.**

The term "dynasty," highly activated in level 1, is not in the second or third layers, indicating that it is not an important term for this user. Layer 2 records that the term "bush" occurs frequently during some document accesses, but not infrequently enough to have a high global weighting. This is because the term occurs too frequently in these browsing sessions to have a high weighting. Ideally, the system would identify "bush" as a good indexing term. If the user had a broader range of interests so that "bush" occurred less overall, WordSieve would identify it as a good indexing term.

Multiplying the values of corresponding terms in all three layers gives us the context vector (see Figure 4c). The system determined that the current context is characterized by the terms "george," "political," "life," "texas," and "president" and, to a lesser extent, by the terms "shrub" and "google." The fact that the system considered the presence of "google" and "shrub" to be potentially relevant terms is an artifact of the user's search process: Google searches for "Bush" often return pages about shrubs.

## Experiments

We have previously reported on the original WordSieve algorithm's ability to identify the terms in its search task.[1,2] This showed that WordSieve could automatically generate vectors that resembled the subjects' original task descriptions. The experiment described here tests a refined version of the algorithm with more users (16 instead of six) and different subject domains, and measures a different property. Although the initial pilot experiment compared vectors to original task descriptions, the new experiment aims to measure WordSieve's indexing and retrieval ability. Having indexed all documents, we wanted to know how well it could generate a query from a given document to retrieve the other documents accessed in the same context.

In this experiment, we asked 16 paid subjects responding to a public announcement to browse the Internet for 40 minutes, 10 minutes each on four different topics: George Bush's political life, Al Gore's political life, traditional Indonesian cooking, and traditional Thai cooking. We stored the contents and the order the Web pages were accessed in a database. We stripped punctuation, HTML code, and stop words from the Web pages.

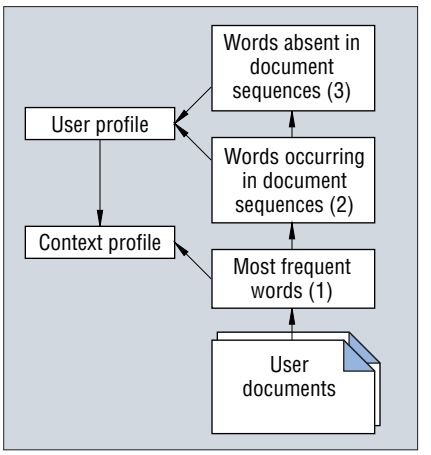


**Figure 3. The WordSieve three-layer network.**


**Figure 4. A snapshot of WordSieve's node activation: (a) layer 1, (b) layers 2 and 3, and (c) the context vector. These images do not show all of the units.**

We compared WordSieve to TFIDF and LSI. Some personal information agents use TFIDF for indexing and retrieval.[3,7,8] LSI operates on indices produced by techniques such as TFIDF. It takes a matrix composed of document vectors and, by reducing this matrix's rank, can typically fine-tune the indices to achieve higher information retrieval performance. The rank reduction effectively reduces the noise in the vectors and capitalizes on term co-occurrence to optimize vectors. LSI is useful for extracting information implied in the combination of document vectors, which can improve performance in retrieval contexts.

Both of these techniques get their power from their use of global information about a corpus: As illustrated in the following experiments, having comprehensive statistics about a large corpus improves their ability to generate appropriate vectors. However, the need for such statistics is problematic for personal information agents because they might only have access to limited data. Thus, we explore in our experiments whether WordSieve, using only local information, can achieve comparable performance. In these experiments, we used our own search engine, which uses the cosine similarity metric to compare the query to the vectors in the database. We generated queries from documents accessed by all the users. We used WordSieve, along with TFIDF and LSI, to index relevant documents accessed by users during browsing sessions.

## Parameter settings

We individually tuned each algorithm to achieve the highest-possible performance on a composite problem, simulating browsing by taking the pages accessed by the first five users and treating them as a single browsing session. We fed this browsing session into our implementation of the algorithms and then conducted the following test: We computed each algorithm's quality as a weighted average of the average precision at a recall of (0.0, 0.1, 0.2, 0.3, and 0.4, respectively). The average was $(2p_{0.0} + 1.5\,p_{0.1} + p_{0.2} + p_{0.3} + p_{0.4})/6.5$.

***Optimizing WordSieve.*** Twelve parameters must be set to determine WordSieve's functioning. We tuned the parameters manually. Once we had a good set of parameters, we employed a genetic algorithm to achieve greater performance in the given experiment. The genetic algorithm significantly increased the algorithm's performance (see Figure 5). The genetic algorithm settled on 130 units in level 1 and 117 in layers 2 and 3.

***Optimizing TFIDF.*** For TFIDF, the dictionary and vector sizes are the two main settings. The dictionary size is the number of terms that you could possibly use in a vector. The vector size is the number of terms that were actually used to index the documents. Dictionary size significantly affects performance. We compared the best performance that TFIDF could produce, found by brute force. Figure 6 shows the tests' results. We chose a dictionary size of 50 and a maximum vector size of 50.

***Optimizing LSI.*** For our LSI test, we used as input the TFIDF vectors generated. We then experimented with different rank reductions. We found that for these problems, the rank reduction only slightly affects performance, although we found that LSI can substantially increase performance on suboptimal TFIDF vectors. We chose to reduce the rank to 30.

## Performance comparison

We tested the algorithms to see how well they generated queries from documents the users accessed. In this experiment, we passed

a single user's browsing through WordSieve until 1,000 documents has been passed through to train it. Then, we used each algorithm to generate indices of some of the documents each user accessed during each search task. We used these indices as queries in the search engine, which resulted in 152 queries. We considered each document in the result set relevant if a user accessed it during the same search task as the document that generated the query. We computed the weighted averages for all result sets for each user.

We found that these results demonstrated WordSieve's promise for providing good performance without global information. We also found that WordSieve outperformed the other two algorithms for all users (see Figure 7). Figure 7 shows the precision-recall graphs for both users with the greatest and smallest differences in performance.

## Discussion

Although WordSieve outperforms the other algorithms in these results, we optimized the algorithms for the best performance possible on these specific problems. We have also achieved good results on different data used in previously published experiments.

Factors influencing performance include the variety of the vocabulary used in topics of interest and number of documents accessed for each topic. Also, other task domains might require other parameter settings.

We were surprised to find that LSI generally performed no better and sometime performed worse than TFIDF on these tests. It seems that, at least for this kind of problem and data set, LSI cannot improve on the optimal TFIDF parameter settings. We found, however, that LSI can substantially improve on the performance of suboptimally generated TFIDF vectors. We believe that a significant factor in LSI's performance on our data is the overlapping topics. Both the Gore-Bush topics and the Thai-Indonesian topic pairs share significant latent semantic structures because of the overlapping subject areas. Using the latent semantic structure will tend to make the documents from the different overlapping topics have similar vectors and thus increase wrong documents' chances for retrieval across overlapping topics. In Figures 7b and 7c, LSI can improve over TFIDF for the later recall values. Apparently, it created more accurate indices for some outlying documents that TFIDF did not detect.

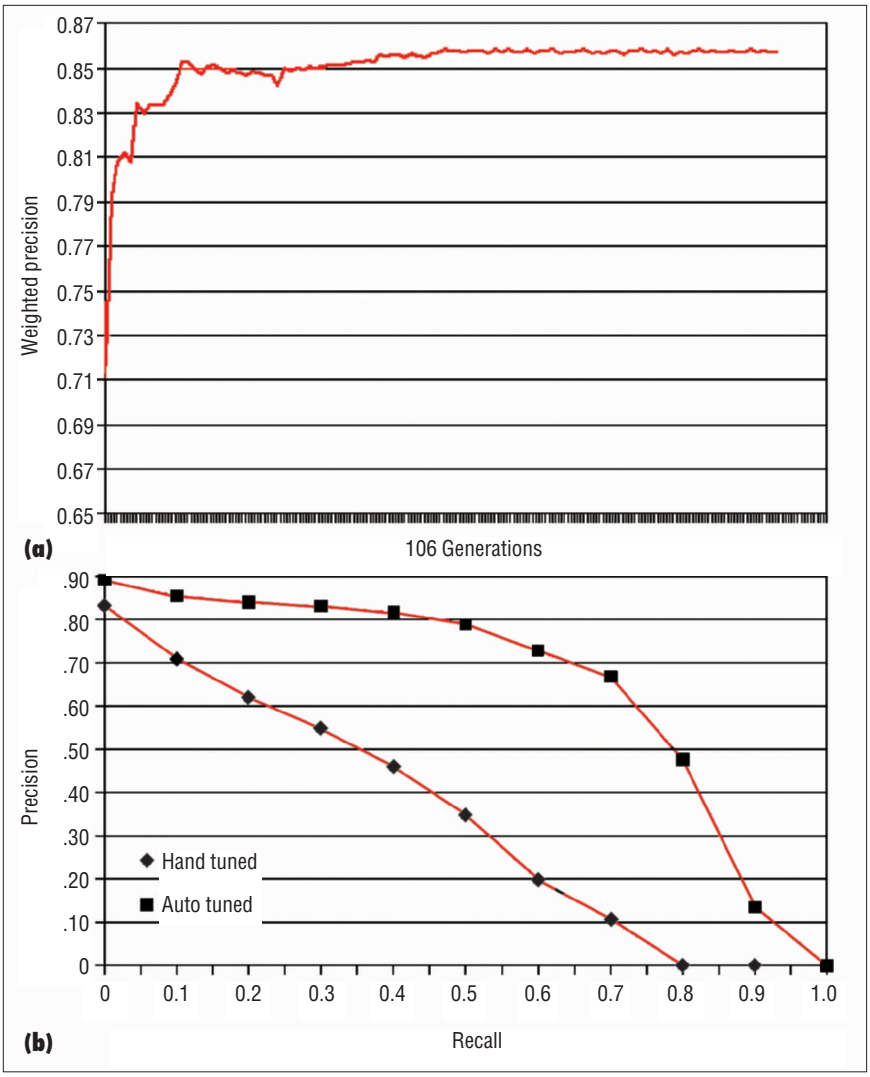


**Figure 5. WordSieve's optimization using a genetic algorithm: (a) change in fitness and (b) difference in performance.**

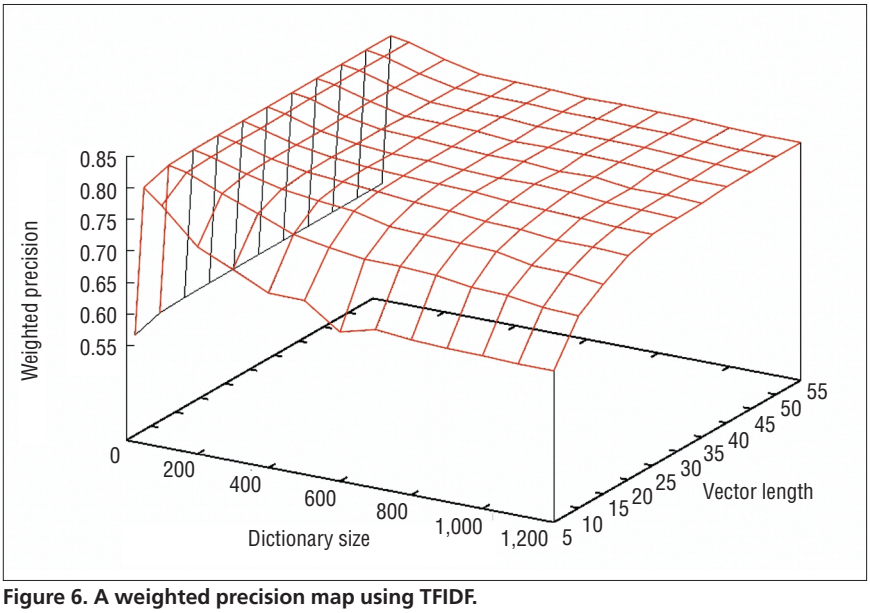


**Figure 6. A weighted precision map using TFIDF.**

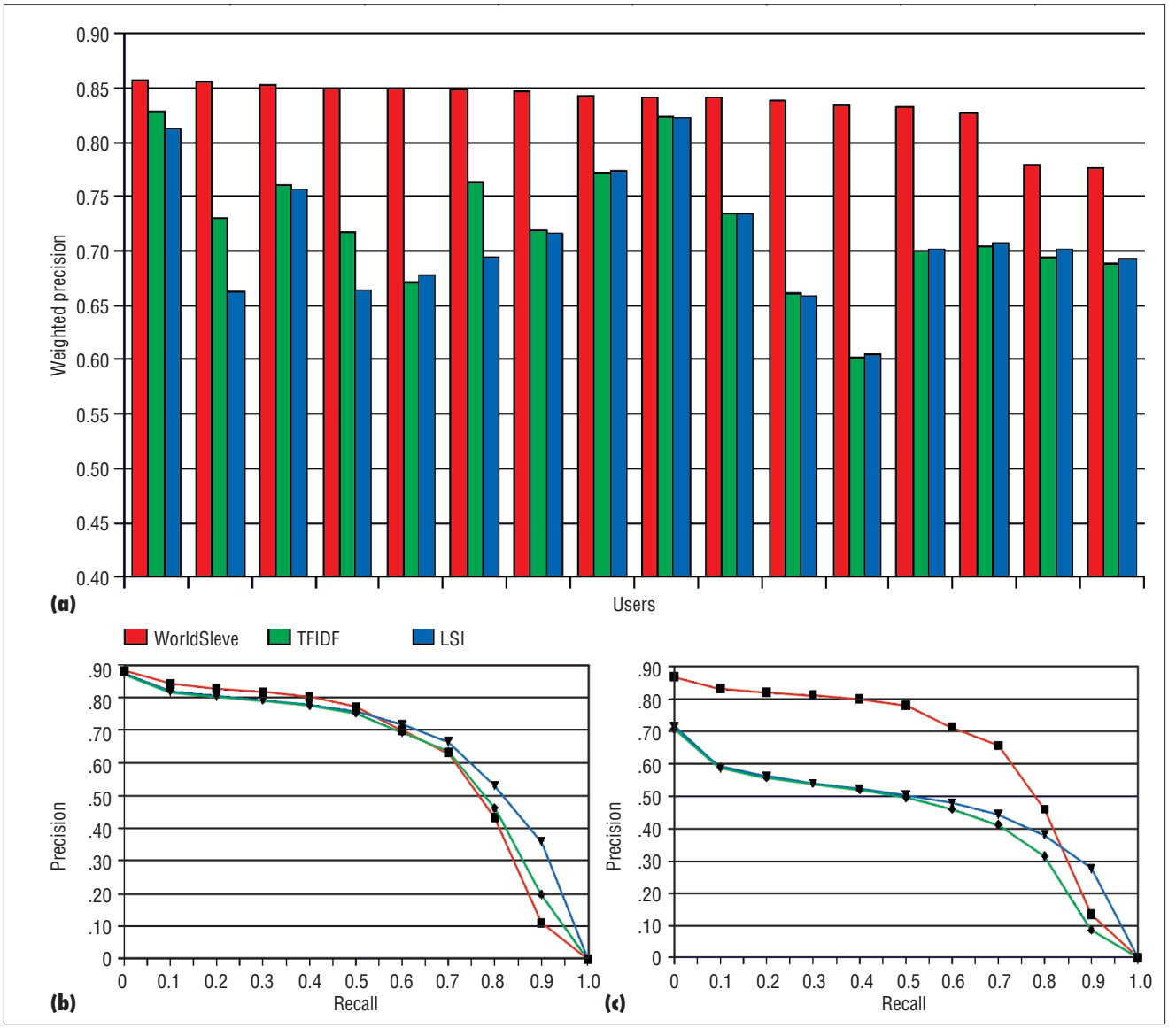


**Figure 7. WordSieve results: (a) overall user comparisons; (b) least different and (c) most different results.**

A primary issue for future study involves parameter tuning. In this experiment, we wanted to give each algorithm the best chance possible. Therefore, we optimized the parameters to give the highest performance on this particular data set. However, the preliminary work we have done with other data sets suggests that other parameters will be optimal for other data sets and that the parameters chosen here might give poor performance elsewhere. Initial experiments with other data sets having more varied topics suggest that more units are necessary in those situations. Particularly, the dictionary size needs to be larger for problems with a more varied vocabulary. In the future, we plan to work on an autotuning version of WordSieve, in which the algorithm adjusts its own parameters over time to fit the particular user.

We also intend to study how WordSieve adjusts to changing user interests. The activations in the units of each layer slowly decay over time if repeated term occurrences do not activate them. If a user's interests change, terms associated with the old interests that no longer occur will slowly decrease in their activation and eventually be replaced. We have plans for additional experiments to show how long it takes the algorithm to learn a user's interests. We will need to show how quickly the system can adjust to changing user interests and how temporary interest shifts will affect it. ■

## References


1. T. Bauer and D. Leake, "Wordsieve: A Method for Real-Time Context Extraction," *Modeling and Using Context: Proc. Third Int'l and Interdisciplinary Conf.* (Context 2001), Springer-Verlag, Berlin, 2001, pp. 30–44.

2. T. Bauer and D. Leake, "Real Time User Context Modeling for Information Retrieval Agents," *10th Int'l Conf. Information and Knowledge Management,* ACM Press, New York, 2001, pp. 568–570.

3. H. Lieberman, "Letizia: An Agent That Assists Web Browsing," *Proc. Int'l Joint Conf. Artificial Intelligence* (IJCAI 95), Morgan Kaufmann, San Francisco, 1995, pp. 924–929.

4. J. Budzik, K. Hammond, and L. Birnbaum, "Information Access in Context," *Knowledge-*

## The Author

**Travis Bauer** is a graduate student at Indiana University, where he will complete a PhD in computer science and cognitive science in May 2003. His research focuses on automatically extracting context descriptions of user interests from their document access patterns and applying this technology in autonomous, distributed, and multiagent systems. He codeveloped an open source Java library for building information retrieval systems. He received his BA in computer science from Jamestown College. He is a member of AAAI and the ACM. Contact him at the Computer Science Dept., Lindley Hall, Indiana Univ., 150 S. Woodlawn Ave., Bloomington, IN 47405; trbauer@acm.org.

**David Leake** is an associate professor of computer science and member of the Cognitive Science faculty at Indiana University. His research interests include artificial intelligence, cognitive science, case-based reasoning, and intelligent user interfaces. He is the chair of the Seventh International Conference on Intelligent User Interfaces and the editor of *AI Magazine*. He received his PhD in computer science from Yale University. He is a member of AAAI, the ACM, and the Cognitive Science Society. Contact him at the Computer Science Dept., Lindley Hall, Indiana Univ., 150 S. Woodlawn Ave., Bloomington, IN 47405; leake@indiana.edu.


*Based Systems*, vol. 14, nos. 1–2, 2001, pp. 37–53.

5. G. Salton, A. Wong, and C.S. Yang, "A Vector Space Model for Automatic Indexing," *Comm. ACM*, vol. 18, no. 11, Nov. 1975, pp. 613–620.
6. M. Berry, S. Dumais, and T. Letsche, "Computational Methods for Intelligent Information Access," *Proc. Supercomputing* (SC 95), IEEE CS Press, Los Alamitos, Calif., 1995, pp. 1–38.
7. M. Pazzani, J. Muramatsu, and D. Billsus, "Syskill & Webert: Identifying Interesting Web Sites," *Proc. 13th Nat'l Conf. Artificial Intelligence*, AAAI Press, Menlo Park, Calif., 1996, pp. 54–61.
8. D. Billsus and M.J. Pazzani, "A Personal News Agent That Talks, Learns and Explains," *Proc. 3rd Ann. Conf. Autonomous Agents*, ACM Press, New York, 1999, pp. 268–275.