Detecting Context-Differentiating Terms Using Competitive Learning

Travis L. Bauer
trbauer@acm.org
David B. Leake
leake@cs.indiana.edu

October 2, 2003

Abstract

Personal information agents monitor ongoing user information accesses in order to provide users with context-relevant information. Providing the needed information requires effective methods for identifying the user's task context, based on available information. For user browsing tasks, one approach to context identification is to extract context-determining terms from the documents that the user consults. The thesis of this article is (1) that term extraction for personal information agents can be done by learning terms whose occurrence frequencies have a large variance over time, (2) that indexing and retrieval based on these terms can be at least as effective as standard information retrieval techniques, and (3) that this information can be learned without comprehensive corpus analysis, making it suitable for use in personal information retrieval.

We have developed an unsupervised term extraction algorithm, WordSieve, that learns individualized context-differentiating terms for document indexing and retrieval. This article presents a new version of WordSieve, compares its design and performance to our initial approach, and assesses its effectiveness for a controlled personal information retrieval task, compared to three common indexing techniques requiring statistics about the global corpus. In the experiments, the new version of WordSieve generates task-relevant indices of comparable or better quality to common indexing techniques, using only local information.

1 Introduction

A central problem for personal information agents is identifying the user's information access context, in order to provide context-relevant suggestions. The article presents an approach to term extraction for personal information agents, in which term extraction is based on learning terms whose occurrence frequencies have a large variance over time. We have implemented this approach in an unsupervised term extraction algorithm, WordSieve. WordSieve learns to differentiate task contexts of users engaged in

information access, and generates term vector representations of the users' task contexts for indexing and retrieval. WordSieve learns task contexts by discovering terms with certain occurrence patterns in the stream of documents that they consult. Our hypothesis, substantiated by experiments, is that these occurrence patterns can be used to identify information access contexts. In addition, WordSieve automatically adapts to noisy, real world input without requiring computationally intensive statistical summaries of large sets of data or explicitly defined categories.

There has been considerable prior research on personal information retrieval agents, some of which will be surveyed in this article. However, much current work in the field depends on applying or adapting techniques designed for analyzing corpora. Two attributes contribute to WordSieve's special suitability for personal information retrieval tasks. First, WordSieve uses a competitive, stochastic process to learn in real time using limited storage. Second, WordSieve takes advantage of the order in which the user accesses documents to dynamically identify changing task contexts based on individual users' needs. From this information, it builds a user profile of task identifying terms, as well as a context-identifying index for the document. These profiles are used for indexing and retrieval in Calvin, a personal information agent that we have developed [6]. They can also be used for query generation, to guide context-based retrieval from standard search engines.

In previous literature, we have described tests of an initial WordSieve algorithm [5, 4, 7]. In this article, we describe extensions of this research, discussing a new version of WordSieve and its benefits compared to the original, comparing the performance of the two, and showing how the new algorithm compares to three common indexing techniques based on global information, Term Frequency/Inverse Document Frequency, Log-Entropy, and Latent Semantic Indexing, for a controlled personal information retrieval task. In these tests, WordSieve generates task-relevant indices of comparable or better quality to these algorithms, using only local information.

2 Principles of WordSieve

The WordSieve approach is based on four central principles: local processing, competitive learning, real-time learning during processing, and exploitation of information clustering over time. This section introduces each of these principles as they apply to WordSieve, and the following section describes the algorithm in detail.

2.1 Local Processing

Rather than requiring information about the global document corpus (e.g., statistical summaries of term frequencies), WordSieve processes only the documents that the user consults, processing each term as it occurs in the user's document stream. Based on each occurrence, it modifies a small internal structure and then passes on to the next feature. The incremental effects of each term occurrence on internal structures accumulate over time, helping to identify significant terms.

WordSieve's approach differs from commonly-used techniques, in that global information about the statistical features of all terms is not necessary. In Log Entropy [2],

for example, information about the occurrences of all the terms in all the documents is necessary to compute global weights. In the WordSieve approach, all decisions about weighting are handled locally, without the benefit of this "bird's eye view" of the data. The WordSieve hypothesis is that it is possible to develop update rules such that a set of useful context-differentiating features will emerge from these local decisions.

2.2 Competition

Competition among units is central to WordSieve's function. Rather than keeping track of all the features, WordSieve keeps track of a limited set of candidates, in units associating those candidates with activation values used to estimate their value as context discriminators. In our tests, these units reflect only information for 150 terms, which effectively compete for positions in the layer of units.

During user browsing, the stream of terms from accessed documents is presented to the network, one term at a time. Some of the terms are "caught" in the net, becoming associated with one of the units. As new terms are processed, some of the terms may be replaced by new terms, with high-activation terms less likely to be replaced. In the long run, the terms that stay "caught" in the units are the ones that win the competition and are thus considered to be good candidates. Thus WordSieve can be seen as a "sieve" catching the valuable indexing terms and letting the rest "pass through."

The sieve is implemented by associating each unit with an activation value, which determines how strongly a term is bound to a unit. A term's "activation" level corresponds approximately to its rate of occurrence, and a term associated to a node with high activation is less likely to be replaced than one in a unit with a low activation. When a term is presented to the network, if the term is already in the network, the activation will be updated. If the term is not in the network, it will take over the existing unit with the lowest activation.

2.3 Relying on Approximate Information

Through the updating process, WordSieve learns statistical term occurrence tendencies over time. Its statistics are not exact; for example, the fact that a term does not occur within some window X does not entail that the term is occurring less than 1/X times in the slightly broader context. Thus, the use of this information must be resilient to the irregularities in the data set. For hysteresis, WordSieve maintains activations even when the term does not occur for short periods of time. However, the activation decays when a term stops occurring, and eventually the term is dropped. Experiments suggest that this imprecise measurement of how often a term is occurring is still good enough to achieve high performance, and requires only storing information for a small number of terms (those currently associated with units), rather than for each term in the corpus. Thus unlike traditional indexing algorithms, WordSieve does not need a comprehensive statistical summary of the document corpus to determine its indices, which we makes it well-suited for personal information retrieval.

2.4 Exploiting Temporal Clustering

Personal information agents have access to a type of information that may not be available for other kinds of information retrieval: ordering information about the user's document access sequence. The user implicitly provides temporal clustering by virtue of accessing the documents in a particular order, and WordSieve exploits this information. The clustering is not "discrete"—where one cluster ends and another begins is not explicit in the document stream, and clusters may overlap as a person slowly changes from one topic to another during browsing—but WordSieve reflects hypothesis that useful indexing and retrieval can still be based on these clusters.

In particular, if it is possible to extract terms correlating to these clusters, then those features can be used to identify the cluster to which a document belongs and, in turn, to suggest that document when related documents are being accessed, i.e., in similar future contexts. This context-based feature extraction WordSieve's goal.

3 Two approaches to implementing the WordSieve Principles

In previous literature [7], we described the first version of WordSieve in detail and illustrated its performance. In the first version, two different layers of units separately rewarded a term for occurring or not occurring, in order to—in combination—extract terms sometimes occurring frequently and sometimes occurring infrequently, with the hypothesis that such terms were good context-discriminators. Using the two different components, one to detect each situation, was a straightforward method to detect these context-differentiating terms.

However, we have devised a new approach with fewer layers, that has the same performance and exhibits less variation across subjects. This new version of the algorithm has a single component that rewards terms on their transitions in occurrence frequency, rather than having separate layers with units rewarded the entire time the term is occurring or absent or not occurring. In the new approach, a single layer rewards terms when they go from occurring frequently to occurring infrequently and vice versa. Another difference is the elimination of priming. In the first version of WordSieve, priming was used to make the algorithm resilient to small variations in term occurrence frequency. However, this was already provided by the competitive nature of the other parts of the algorithm and was unnecessary. The differences between the two algorithms are listed in Table 1. The advantage of this approach, first of all, is simplicity. There is a single component to deal with rather than two. Secondly, the learning takes place in relatively short periods of time rather than all the time, which makes it somewhat more stable.

Our tests will show that the new version of the algorithm achieves equivalent performance to the original algorithm, and that this performance is exceeds or is comparable to standard approaches, using only local information. The second approach will have less variance for reasons that will be discussed.

Revision 1	Revision 2
Three Layers	Two Layers
Uses priming	No Priming
Rewards Presence/Absence of term	Rewards changes in frequency

Table 1: Differences between the two versions of WordSieve

Size	Number of units
BaseFrequency	Minimum frequency to maintain activation level
Max Activation Rate	Highest rate at which activation can change
Initial Activation	Activation of a new unit
Level 2 Min Activation	Activation level necessary to move to level 2

Table 2: Free Parameters for level 1

3.1 Layer One

In the new WordSieve algorithm, the layer one tracks the terms which are currently occurring frequently and provides some data smoothing to account for small variations in the occurrence frequency. Activation changes directly as a result of the occurrences of the term itself. For each term presented, the activation of each node in layer one decreases by:

$A_i \leftarrow A_i - maxActivationRate \times BaseFrequency$

If a unit is associated with the term presented, its activation is increased by the following value:

$A_i \leftarrow A_i + maxActivationRate$

If the term is not present in the network, the lowest ranking term in layer one is removed and the term being presented gets its place.

There are a number of free parameters controlling how this level functions. They are shown and described in Table 2

3.2 Layer Two

The activation of a unit in level two alone is the algorithm's estimation of how well the term distinguishes document clusters. This activation value changes as a function of a curve that rewards the unit when the term starts occurring frequently or stops occurring frequently, and punishes it otherwise. In order to achieve a sharp spike in activation, we based activation changes on the "witch of agnesi" equation[15], described by:

$$y = \frac{8a^3}{x^2+4a^2}$$

single spike preceded and followed by near zero values makes it ideal for the present task of rewarding a unit for a short period of time. If the X axis is the amount of time that the unit has been present or absent in level 1 and the y value reflects the amount of change to the activation, a term's presence or absence for a short period of time will have no great effect on the activation. However, if the term occurs for a longer period of time, then the reward is spiked for a short period. Note that these spikes in

When term is present	
Broadness of activation spike	
Activation Rate	Maximum Rate at which the activation can increase
Decay Rate	Maximum Rate at which the activation can decrease
When term is absent	
Width	How wide the curve is
Activation Rate	Maximum Rate at which the activation can increase
Decay Rate	Maximum Rate at which the activation can decrease

Table 3: Free Parameters for second level

activation change are applied to the second level both when the corresponding term appears in the first layer and when it is dropped out. This lets units in the second layer get rewarded for both the presence and absence of a term in the document stream.

For use in the algorithm, the equation was scaled and translated along both the x and y axes, with these parameters were incorporated into the basic update equation. The modified form of the equation is:

$$(\frac{8a^3}{(((x-width)/width)*10.0)^2+4a^2} \times (ActivationRate + DecayRate)) - DecayRate$$

The free variables for the second layer are as shown in Table 3.

At first, it was hoped that a single curve could be used for both when the term was present and when it was absent. However, good performance required using two different curves, because rewards for terms that start occurring frequently need to be given sooner than for terms that stop occurring frequently. Thus, the translating and scaling had to be different for when the term was present and when it was absent. A plot of this equation with the parameters chosen for the "term present" curve for one of the experiments problems is shown in Figure 1. The Y axis is the amount of change to the activation of a unit. The X axis is the number of terms presented.

In conclusion, WordSieve is designed to use competitive, local learning to learn about statistical occurrence patterns, extracting features which have the contextual characteristics discussed earlier. We have introduced two implemented algorithms for this task. We now turn to the evaluation of algorithms.

4 Evaluation

4.1 Overview

The goal of our evaluation was to assess the relative ability of both versions of WordSieve and other algorithms to index documents so that they would be retrieved in relevant contexts. In order to test the algorithms in multiple situations, we used two different data sets. To gather the first, we conducted a controlled web browsing experiment to collect data from real web browsing. Sixteen paid subjects browsed the World Wide Web for information on four specific topics, listed in Table 4. The topics are intentionally overlapping, in order for them to have overlapping vocabularies. For

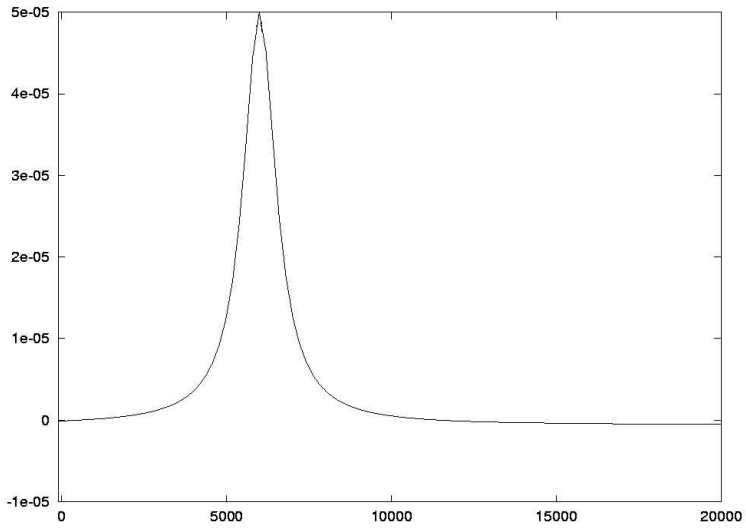

Figure 1: Plot of scaled and translated equation for problem 3

Browsing Topics
The political life of George Bush
The political life of Al Gore
Traditional food in Indonesia
Traditional food in Thailand

Table 4: Browsing Topics

example, the term "politics" would be significant in both the Bush and Gore topic. This makes it more difficult to differentiate the topic in which a web page is accessed.

During 40 minutes of browsing each, the users accessed a total of 2,279 unique documents. These documents were hand-tagged as either being "content" or "noise" documents. "Content" documents were all the documents that contained text which the user was meant to read; however, they may be irrelevant to the task. "Noise" documents were error messages and empty documents. Of these 2,279 unique documents, 891 were considered "noise." Users accessed an average of 221 documents each during their browsing sessions, with an average of 143 being content documents. Overall, there were 4,423 document accesses, 2857 of them being non-noise documents.

We assessed the algorithms' performance based on how well they could distinguish among the original contexts, as determined by the contexts within which the documents were originally accessed. Specifically, we measured how well, given a query generated from a document that was accessed by a user searching for information on some topic, the system could retrieve other documents from the set consulted for that topic.

If the training sets were on completely disjoint topics, we would expect such a problem to be trivial. It is the ability to make more fine distinctions among contexts that is difficult, so having an overlapping set of topics for training and testing makes the problem more interesting.

Performance of the algorithms was computed as a weighted average of precision/recall measurements for different levels of recall. The algorithms were trained using data from one user, with a genetic algorithm used to tune the WordSieve parameter setting. Next, all documents accessed by all users were indexed and put into a search engine. The testing was then accomplished by picking several documents from each of the categories, generating a query from its index and seeing how well that query recalled other documents from the same category. The precision was computed at various levels of recall and a weighted average computed as follows where p_x is the precision at a recall of x.:

$$\frac{2p_{0.0} + 1.5p_{0.1} + p_{0.2} + p_{0.3} + p_{0.4}}{6.5} \tag{1}$$

The average performance of various algorithms on the web browsing data are plotted in Figure 2. The statistical analysis of these results using a repeated measures ANOVA are shown in Table 5. These results show that WordSieve, on average, performs as well as LSI and generally better than TFIDF and Log-Entropy. The differences in the averages between either versions of WordSieve and LSI are not statistically significant. The differences between both versions of WordSieve and TFIDF and Log/Entropy are statistically significant.

The results are striking, especially considering the kind of information that the algorithms were given. TFIDF, Log-Entropy, and LSI need considerable information for their computations. They require corpus summaries of the entire data sets, and have access to statistical summaries of all terms in the sets. WordSieve, on the other hand, does not get this information, but has to build up its information through competitive learning.

The standard deviation between users for each algorithm is shown in Table 6. It is notable that the second version of WordSieve had a smaller standard deviation than the

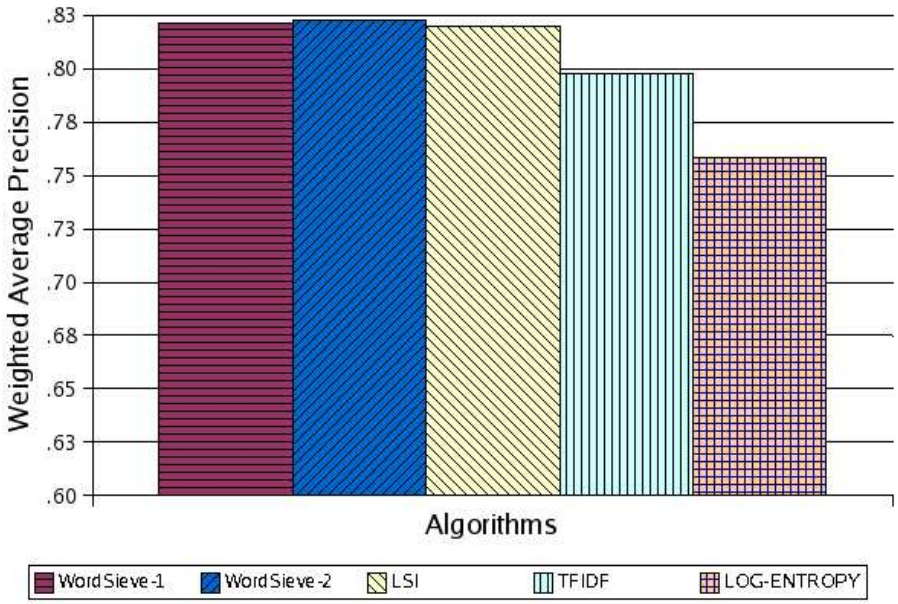

Figure 2: Comparison of average performance on individual users

	LSI	TFIDF	Log/Entropy
WordSieve Version 1	0.03	5.83	26.29
WordSieve Version 2	0.12	6.25	17.90

Table 5: F values of repeated measure ANOVA analysis. Values are statistical significant (at $\alpha = 0.05$) for the differences between both versions of WordSieve and TFIDF and Log/Entropy

WordSieve 1	0.04191
WordSieve 2	0.02180
LSI	0.02692
TFIDF	0.03349
Log-Entropy	0.04864

Table 6: Standard Deviation across subjects

first version. This supports our belief that the strategy chosen by the genetic algorithm for the first version, while producing good performance on the testing problem, does not necessarily lead to good results across the board. On average, the first version of WordSieve performed about the same, but with greater variance.

These results support the thesis of this article:

1. that term extraction for personal information agents can be done by learning terms whose occurrence frequencies have a large variance over time

2. that indexing and retrieval based on these terms can be at least as effective as standard information retrieval techniques

3. that this information can be learned without comprehensive corpus analysis, making it suitable for use in personal information retrieval.

We also ran the same tests with collected newsgroup data. A brief summary of the actual training data chosen is shown in Table 7. Although this does not represent web browsing, which was the target application of this study, the newsgroup data was useful because it subject matter was less constrained than with the web browsing data set. Note that the content of the documents was not considered when deciding what category they belonged to. So, for example, if there was a discussion in soc.religion.christian about atheism, the atheism posts were still considered part of soc.religion.christian and not put to alt.atheism. One could argue that such a thread really "belonged" to the alt.atheism category.

The performance of the algorithms on the Usenet problems is shown in Figure 3. The second version of WordSieve outperformed TFIDF and Log-Entropy on all three problems and outperformed LSI on two out of the three. Although there are not enough examples here to draw conclusions quite as general as with the browsing problem, these tests suggest that WordSieve performs well even when there is a great variety of topics in the dataset.

LSI is strikingly better than any of the other algorithms in problem 4. The strength of LSI is to find the implicit latent semantic structure in texts based on co-occurrence of terms in different documents. In a problem such as this where there are a great variety of different topics discussed in the same general category, one would expect LSI to do well.

ID	Training Data	Testing Data
Newsgroup Data		
3	alt.atheism talk.religion.misc soc.religion.christian rec.sport.baseball rec.sport.hockey	All posts to these newsgroups
4	comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware rec.autos rec.motorcycles	All posts to these newsgroups
5	talk.politics.guns talk.politics.misc sci.electronics sci.med sci.space	All posts to these newsgroups

Table 7: Newsgroup data sets

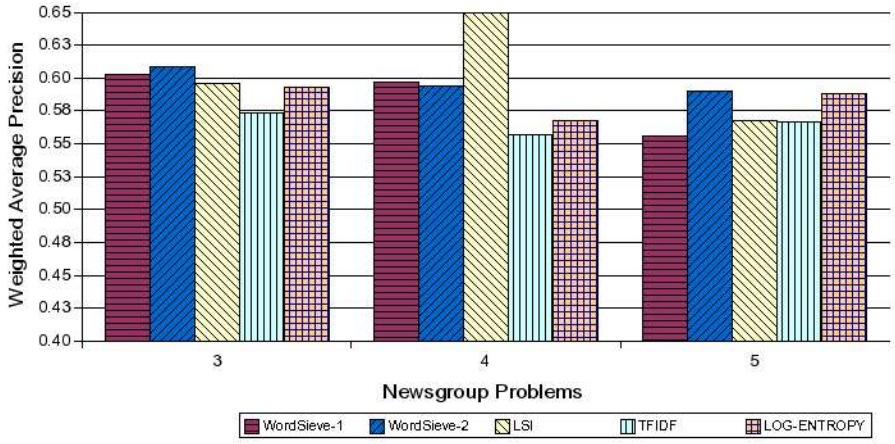

Figure 3: Comparison of average performance on Usenet problems

5 Related Work

There is a large body of literature discussing personal information retrieval agents. Many of these systems attempt to analyze users by considering the documents that they have accessed. However, these systems to not take advantage of the statistical properties of the document stream which the users generate. Instead, they collect a set of documents and consider them as a corpus, not taking advantage of the fact that the documents were accessed in a particular order. Webwatcher [9] and Letizia profile users based on the documents that they have accessed. Watson [8] does not keep a permanent profile of the user, but searches based on the recently accessed documents. For some similar systems [3, 13], the user must provide feedback to indicate which documents are useful. The Hotlist/Coldlist system [14] keeps a list of interesting and non-interesting documents as a user profile. Systems like WebGlimpse [12] assist user browsing by automatically performing searches for users and providing users with lists of documents related to what they were currently looking at.

Most of these systems are primarily indexer/retrievers. They may be customized for individual use, but their performance relies almost entirely on the effectiveness of statistical techniques for analyzing free text. Some systems, such as the Citation Finder [11] employs multiple strategies for finding useful documents. Some use data mining techniques to discover rules about users' behavior [1]. Most use terms for indexing, but not all. Some systems serve as a "meta service" to find the search engines that are most likely to be helpful to the user [10]. Other systems generalize the behavior of multiple users in the past to predict what the current user's interests [9].

These systems share a common assumption that the meaning of a document can be approximated primarily by looking at the documents themselves and comparing their content in some way. The selection of the documents in the corpus might be determined implicitly by recording the user's computer use, but once collected, the documents are treated as a corpus. The core assumption is that the usefulness of the document can be found by comparing the document's content to other documents; they do not take into account how the documents actually get used by the user.

Some systems use co-occurrence of document access, such as ProfBuilder [16]. However, this system requires individual pages to be remembered and stored for future reference. We propose a learning method that, on the one hand, learns about the user based on the sequence of document accesses, and on the other hand yields a term vector-based user profile that can be applied in the same way as those of standard algorithms.

6 Discussion and Conclusion

These tests suggest that WordSieve is a promising approach for extracting key terms for indexing documents according to the contexts in which they are used and for differentiating a user's different task contexts. Our experiments suggest that WordSieve's local, competitive approach enables discovering useful indexing terms, and it does so without requiring a global summary of an entire corpus. They also suggest that approach can perform on average as well as or even better than the existing algorithms.

In our tests on user browsing, the algorithm performed as well as LSI, often better than TFIDF, and usually better than Log-Entropy.

A remaining challenge for using WordSieve is tuning the algorithm. Both versions of WordSieve have a number of free parameters, and this makes it more difficult to tune than the other algorithms. We believe that more work on WordSieve can help identify ways to make the tuning more automatic or perhaps even reduce the number of free parameters it requires.

References

[1] Gediminas Adomavicius and Alexander Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, February 2001.

[2] Ricardo Baeza-Yates and Berthier Ribeiro-Neto. *Modern Information Retrieval*. ACM Press, 1999.

[3] Marko Balabanović and Yoav Shoham. Learning information retrieval agents: Experiments with automated web browsing. In *Proceedings of the AAAI Spring Symposium on Information Gathering from Heterogeneous, Distributed-Resources*, March 1995.

[4] Travis Bauer and David Leake. Real time user context modeling for information retrieval agents. In *Tenth International Conference on Information and Knowledge Management*, pages 568–570. ACM Press, 2001.

[5] Travis Bauer and David Leake. Wordsieve: A method for real-time context extraction. In *Modeling and Using Context: Proceedings of the Third International and Interdisciplinary Conference, Context 2001*, pages 30–44. Springer-Verlag, 2001.

[6] Travis Bauer and David Leake. Calvin: A multi-agent persoanl information retrieval system. In *Agent Oriented Information Systems 2002: Proceedings of the Fourth International Bi-Conference Workshop*, 2002.

[7] Travis Bauer and David Leake. Using document access sequences to recommend customized information. *IEEE Intelligent Systems*, 17(6):27–32, Nov/Dec 2002.

[8] J. Budzik, K. Hammond, and L. Birnbaum. Information access in context. In *Knowledge based systems*, 2001.

[9] T. Joachims, D. Freitag, and T. Mitchell. Webwatcher: A tour guide for the world wide web. In *Proceedings of IJCAI97*, August 1997.

[10] David B. Leake and Ryan Scherle. Towards context-based search engine selection. In *Proceedings on the International Conference on Intelligent User Interfaces*, pages 109–112, Santa Fe, NM, Jan 2001.

[11] Seng Wai Loke, Andrew Davison, and Leon Sterling. CIFI: An intelligent agent for citation finding on the world-wide web. In *Pacific Rim International Conference on Artificial Intelligence*, pages 580–591, 1996.

[12] U. Manber, M. Smith, and B. Gopal. Webglimpse: Combining browsing and searching. In *Proceedings of 1997 Usenix Technical Conference*, 1997.

[13] M. Pazzani, J. Muramatsu, and D. Billsus. Syskill & webert: Identifying interesting web sites. In *Proceedings of the National Conference on Artificial Intelligence*, Portland, OR, 1996.

[14] M. Pazzani, L. Nguyen, and S. Mantik. Learning from hotlists and coldlists: towards a www information filtering and seeking agent. In *Proceedings of AI Tools Conference*, Washington, DC, 1995.

[15] Murray R. Spiegel. *Mathematical Handbook of Formulas and Tables*. Shaum's Outline Series in Mathematics. McGraw-Hill Book Company, 1968.

[16] Ahmad M. Ahmad Wasfi. Collecting user access patterns for building user profiles and collaborative filtering. In *Proceedings of the 4th international conference on Intelligent user interfaces*, pages 57–64. ACM Press, 1999.