WordSieve: A Method for Real-Time Context Extraction*

Travis Bauer and David B. Leake

Computer Science Department
Lindley Hall, Indiana University
150 S. Woodlawn Avenue
Bloomington, IN 47405, U.S.A.
{trbauer,leake}@cs.indiana.edu
http://www.cs.indiana.edu/~{trbauer,leake}

Abstract. In order to be useful, intelligent information retrieval agents must provide their users with context-relevant information. This paper presents WordSieve, an algorithm for automatically extracting information about the context in which documents are consulted during web browsing. Using information extracted from the stream of documents consulted by the user, WordSieve automatically builds context profiles which differentiate sets of documents that users tend to access in groups. These profiles are used in a research-aiding system to index documents consulted in the current context and pro-actively suggest them to users in similar future contexts. In initial experiments on the capability to match documents to the task contexts in which they were consulted, WordSieve indexing outperformed indexing based on *Term Frequency/Inverse Document Frequency*, a common document indexing approach for intelligent agents in information retrieval.

1 Introduction

Intelligent information retrieval agents are an emerging class of software designed to aid computer users in various aspects of their work. By observing the user's interaction with the computer to determine contextual cues, these agents can suggest information that is likely to be useful in the current context. Much research has studied the formalization of context (e.g., [2, 5, 4])), and rich context representations have been proposed (e.g., [18, 19]). However, the circumstances in which information retrieval agents function strongly constrain the context extraction methods and representations that are practical for them to use. In order to provide robust support, information retrieval agents must automatically generate context descriptions in a wide, unpredictable range of subject areas; in order to provide information when it is most useful, they must make context-related decisions in real time, as rapidly as possible.

Because of their need to handle unpredictable subject areas, intelligent information retrieval agents generally forgo carefully-crafted, pre-specified context representation

* Travis Bauer's research is supported in part by the Department of Education under award P200A80301-98. David Leake's research is supported in part by NASA under awards NCC 2-1035 and NCC 2-1216.

schemes in favor of representations gathered implicitly by observing the user working in some task context. In systems that seek to aid a user by retrieving documents relevant to the user's current context, a common approach is to analyze the content of the current document being accessed and retrieve similar documents, under the assumption that documents with similar content will be useful in the current context.[1] These systems often index documents based on Term Frequency/Inverse Document Frequency [17], a term-based approach which focuses on the relationship of the document to the corpus. In particular, TFIDF does not take into account another contextual cue, the order in which the documents were accessed. Our goal is to develop a practical method for using information about document access patterns to help improve on standard techniques for selecting context-relevant documents.

We are developing a context extraction algorithm, called WordSieve, to take advantage of implicit contextual information provided by considering the user's document access patterns over time. By considering information developed from the order in which the documents are accessed, the algorithm is able to make suggestions that reflect not only the content of documents, but also when documents with that content tend to be consulted. WordSieve has been implemented in an intelligent information retrieval agent, CALVIN, which observes users as they browse the world wide web, indexes the documents they access in a given context, and suggests those documents in similar contexts in the future [13]. The success of this approach can be measured by whether it selects documents that are appropriate to the user and task context, even when no explicit description of the task is available to the system. In initial experiments, with CALVIN monitoring users who were given two browsing tasks, WordSieve outperformed TFIDF at matching documents to the tasks in which they were accessed.

This paper is organized as follows. We first discuss issues involved in selecting a context model for intelligent agents. Next we describe the WordSieve algorithm and its performance in our initial experiments. Finally, we discuss some issues raised by this work and its relationship to previous research.

2 Content or Context?

One of the issues addressed by context theory is disambiguating sentence meanings with respect to a specific context (such as disambiguating the sentence "Now I am here") [15]. Similar issues arise in determining the relevant content of documents in context. For example, the explicit content of a document written in the middle ages by a Benedictine monk about food preparation is a constant in any context, but its relevant content changes depending on the context in which it is used. A person interested in finding out about the Benedictine order would use the document in a different context from a person interested in cooking, and in each context, there would be a different answer to the question "What is this document about?"

Information retrieval agents generally retrieve based on document content [3, 6, 7, 14, 16], and commonly treat a document as an independent entity, taking into account its relationship to the entire corpus but not to the immediate group of recently-consulted

[1] For a contrasting view, see [8].

documents within which it was accessed. For example, in TFIDF-based indexing, an index vector is created for each document; the magnitude of each element indicates how well that term represents the contents of the document, based on its frequency within the document and the inverse of the frequency of documents in the corpus containing that term. Thus if a term occurs very frequently in some document, but rarely occurs in the corpus as a whole, it will be heavily weighted in the term vector. In practice this approach provides robust results for many collections. However, it does not reflect potentially-useful information about access context: it considers only the relationship between documents and a corpus, not the context in which those documents are used.

Our research on WordSieve is guided by the hypothesis that the relevant features of a document depend not only on what makes it different from every other document (which is captured by TFIDF), but what makes it similar to the documents with which it was accessed. In other words, the context of a document is not reflected only in its content, but in the other documents with which it was accessed.

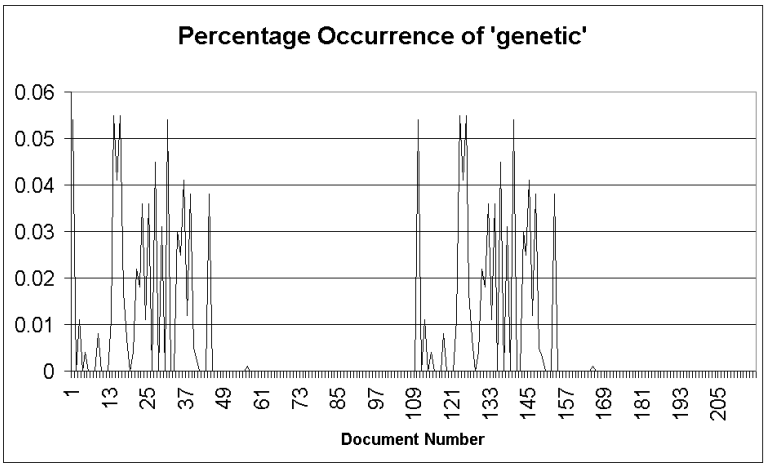

Fig. 1. Percentage occurrence of the term ’genetic’ during a user’s search session.

As an illustration, consider figure 1. The graph represents the occurrence of the word ’genetic’ in web pages during the course of a web search. This access pattern shows four clear partitions, distinguishable by the occurrence pattern of the word “genetic.” When the word ’genetic’ occurred frequently the user was searching for pages about “the use of genetic algorithms in artificial life software.” Our hypothesis is that terms with this type of access pattern are useful indicators of the context of a user’s browsing task.

TFIDF works by exhaustively analyzing the corpus, extracting and comparing word occurrences. There are two reasons why this can be undesirable. First, it is time consuming. In an information agent, the system needs to respond and learn in real time. Second, in an information agent, it is not desirable to store an entire full-text library of all the documents a user has accessed. WordSieve overcomes both of these weaknesses

by being able to learn these terms in real time, and building up information incrementally rather than requiring the entire corpus to analyze at one time.

3 The WordSieve Algorithm for Context Extraction

WordSieve's goal is to find terms associated with document access patterns. The WordSieve algorithm finds groupings of documents which tend to be accessed together, and indexes documents according to frequently occurring terms which also partition the documents. Our hypothesis is that these terms are good indicators of task context. We evaluate this hypothesis in section five.

Because WordSieve automatically extracts terms associated with sets of document accesses, rather than using explicit task descriptions, WordSieve does not require a user to specify when one task is finished and another has begun. Thus there is no need for the user to artificially limit browsing behavior to provide task-related information (which a user would be unlikely to do in practice).

WordSieve's context representation and its system design reflect several constraints affecting real-time information retrieval agents that assist users as they perform other tasks:

1. The system must be relatively compact and should consume only limited resources. It should not, for example, require storing and re-processing previously accessed documents, and consequently must accumulate its contextual information along the way.
2. The system must run in real time. It must make its suggestions while the user is performing the task for which they are relevant.
3. The system should develop a user profile, reflecting the access patterns of the particular user, in order to provide personalized recommendations likely to be useful for that user.
4. The system should be able to use the user profile to produce a context profile when the user is accessing documents, reflecting both the user and the current task.

4 The WordSieve Architecture

The WordSieve network consists of three small, interdependent levels of nodes. Their function is analogous to a sieve, "trapping" partitioning words that reflect a user's context. Each of the nodes in each level contains a small number of attributes: a word and one or two real number values. The word associated with any node can change over time.

The architecture of the current version of WordSieve is shown in figure 2. WordSieve processes documents by first passing each word through the MostFrequentWords level as the words are encountered in a document. Levels 2 and 3 then get their information solely from what survives in the MostFrequentWords level. We will examine each level individually, then consider how well the system performs.

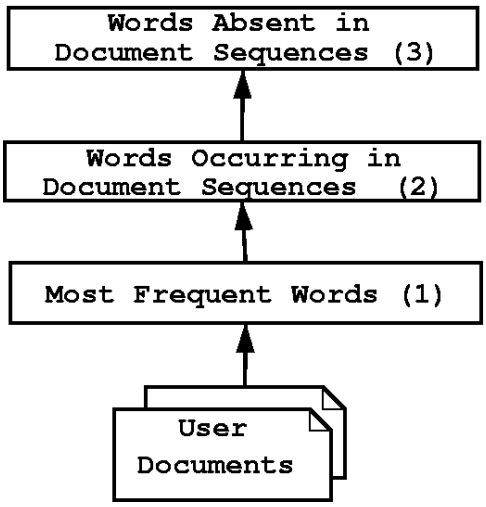

Fig. 2. Word Sieve Diagram

4.1 Level 1: MostFrequentWords

The MostFrequentWords level (level 1) learns which words are currently occurring frequently in the stream of text passing through it, without needing to refer to a database of pre-indexed documents. It will trap any word that occurs frequently, including non-discriminators, such as "and," "or," and "not." The intention is for this layer to select a small number of common terms; in our experiments, this level contains 150 nodes, each of which is associated with a single term.

Nodes in this level do not directly record how many times their words occur. Instead, each occurrence of a word in the text stream increases the excitement of its associated node. To control growth, at each processing step the excitement of the nodes in level 1 decreases by $r = \frac{b}{100}$ where *r* is the rate of decay per word passed into the sieve and *b* is a base rate for decay. In short, for every 100 words presented to the sieve, the excitement of every node is decreased by *b*.

The desired effect is that the excitement of a node will be related to the frequency of occurrence of its corresponding word in recent texts. More precisely, node activation levels are characterized by a threshold of frequency, the "rate goal" g, $0 < g < 1$, representing the percentage occurrence of a term at which the excitement of that term's associated node should increase. If a word occurs with a frequency of g, its node will tend to maintain the same level of excitement. If a word occurs with a frequency less than g, then its node will decay more than it is excited, and over time, its excitement will decrease. If a word occurs with a frequency greater than g, the opposite occurs: its node will be excited more than it decays, and its excitement level will tend to increase.

When a word enters level 1, WordSieve does one of two things, depending on whether there is already a corresponding node for it in level 1.

1. If there is a node corresponding to that word, its excitement is increased by $e = \frac{b}{g}$ where *b* and *g* are as defined above.
2. If there is no node corresponding to that word, the word gets a chance to "take over" a node already assigned to another word. In this process, a node is randomly chosen from the sieve. Then, that node will be re-assigned to the new word with a probability of $.0001 * (E - 100)^2$. This way, a word with a high excitement value is unlikely to be replaced. A word with a low excitement value has a greater chance of being replaced.

The proper operation of this level depends on *g*. If *g* is too low, then too many terms will have increasing values in level one, and the network will be unable to distinguish the ones occurring frequently. If *g* is too high, then the words will decay too fast, resulting in the decay of even the frequently occurring words. The correct value of *g* depends on the distribution of word frequencies in the incoming text stream. Level 1 continuously adjusts the value of *g* to keep the excitement levels of about 10% of the nodes increasing, and the rest decreasing. After every document, *g* is reset to a new value *g'*, given by the following equation.

$$g' = \frac{(o - \frac{N}{10})}{\frac{N}{10} * 0.1} \tag{1}$$

In this equation, *o* is the number of nodes with excitement values greater than the initial value given to the nodes when they are assigned to a word, and N is the number of nodes in level 1.

The result is that level 1 can identify an approximation of the most frequently occurring words in the stream of text, without keeping track of all the words occurring and without keeping a database of the documents which have been accessed. In return for this, WordSieve sacrifices absolute precision and complete determinism. Because of the probabilistic element in the algorithm, the system will produce slightly different values each time it is run. However, as will be shown later, it is sufficiently precise that the system as a whole produces reliably similar results each time it is run.

4.2 Level 2: Words Occurring in Document Sequences

The nodes in level 2 identify words that tend to occur together in sequences of document accesses. In our experiments, this level contained 500 nodes. Because this level does not have direct access to the stream of text, it is only sensitized to words corresponding to nodes in level 1.

Each node in this layer is associated with a word and two real values, excitement and priming. The excitement of the node increases as a word continues to have high excitement values in level 1. Priming determines how fast the excitement of the node can change. Both excitement and priming are allowed to have values between 0.0 and 1.0. At each pass, all the words in the word sieve are presented to this level sequentially. For each word presented to this level, every node's excitement decays by 0.5% and each node's priming decays by 0.1%. Then, if a node is already sensitized to the given word, its priming is increased by 1.75 $\times$ the decay amount.

The node's excitement is increased as a function of its priming. If the given word has not yet been trapped by this level, it probabilistically replaces a node in a manner similar to that described above.

While level 1 is sensitized to the terms that are currently frequent in the text stream, level 2 keeps a record of the words that have tended to occur frequently at different times. Level 2 remembers words even after they stop occurring in the text stream, and only slowly forgets them. Nodes for non-discriminators will get high values at this level, as will nodes for terms which partition the set.

4.3 Level 3: Words Absent in Document Sequences

The function of level 3 is similar to that of level 2. However, its nodes achieve high excitement when they correspond to words that occur infrequently for periods of time. Specifically, its nodes are automatically sensitized to exactly the same words as level 2. However, the nodes keep a low level of excitement until the word stops occurring and increase the excitement value for as long as the word does not occur. Non-discriminators will not get a high level of activation in this level, although nodes will exist for them. The partition words which reflect the user's context will achieve higher activation levels here.

4.4 WordSieve Output

The last two criteria for our architecture specified that user and context profiles should be generated by the system. These are derived from the nodes in the three levels as follows.

User Profiles We define the user profile to be the set of words which tend to partition the user's document accesses into groups. These are words that have occurrence patterns such as that shown in figure 1. Once WordSieve has experienced a reasonably diverse set of document accesses, the user profile can be extracted from the nodes of levels 2 and 3. The user profile consists of the set of words corresponding to nodes in those levels, each word associated with the product of its excitement values in levels 2 and 3. Words with high products tend to have the desired occurrence patterns.

Note that this user profile will be different for every user, because it depends on the user's patterns of document access. These profiles enable the system to learn about the kinds of documents the user accesses as a group, and use them to identify terms which indicate the context in which a user is working. Having identified the context, the system can index a document by that context for suggestion later, and can identify previously indexed documents which may be helpful to the user, aiding personalization of recommendations.

Context Profiles While the user is accessing documents, we can use WordSieve to build a profile of the current context. Intuitively, the context profile should consist of the words which are frequent in that particular document stream and have high values

in the user profile. We achieve this in WordSieve by multiplying the excitement values of words in level 1 and their values in the upper levels. These words should be characteristic of the sets of document sequences.

5 Evaluation

We have evaluated the performance of the algorithm for the ability to index documents according to a search task given to the user, where the context is defined as a topic given to the users, but unavailable to WordSieve. In our tests, WordSieve is able to more strongly match documents to task context than TFIDF. The software used for data collection in our experiment is CALVIN, an Intelligent Agent for research assistance [13]. This system is designed to observe a person using a computer and suggest resources that have been helpful in similar contexts in the past. CALVIN uses WordSieve to index documents.[2]

5.1 Experiment

This experiment examined the comparative performance of WordSieve and TFIDF at matching a document, when seen out of context, to its original search task, described by a term vector representation of the search phrase. The search phrase vectors were compared to the vectors produced by WordSieve profiles and TFIDF.

Seven subjects separately searched the WWW for 20 minutes each. During that time, they were asked to perform two tasks. For the first ten minutes, they were told to load into their browser pages about "The use of Genetic Algorithms in Artificial Life Software." After ten minutes, they were asked to search for information about "Tropical Butterflies in Southeast Asia." Every document they accessed through the browser was automatically recorded. The HTML tags were stripped, as well as punctuation, but no filtering was done on the pages accessed. "PAGE NOT FOUND" pages and pages which contained advertisements were not filtered out. Thus the data collected represents a realistic sampling of the kinds of pages that an Intelligent Agent must handle when observing a user accessing the WWW. To provide an evaluation criterion, documents which did not pertain to the user's search (such as "PAGE NOT FOUND" documents) were then hand-tagged as "noise" documents, and the other documents were hand tagged as either belonging to the genetic algorithm or the butterfly task. These tags were not available to WordSieve.

Users accessed an average of 124 documents per 20 minute session. On average, they accessed 69 "noise" documents and 54 relevant documents. A total of 590 different documents were accessed, 381 of which were determined to be relevant. There were 135 documents which were accessed by more than one user, 88 of which were determined to be relevant documents.

During the user's browsing, the documents were stored. This data was then run through the WordSieve in a series of "simulated" browsing sessions. The sessions were

[2] Some of the classes used in Calvin and WordSieve are released as an open source Java package, IGLU, available at http://www.cs.indiana.edu/~trbauer/iglu.

simulated only in the sense that the user was not actually at the browser; the data was processed by WordSieve in the same order in which the user accessed it, and no pages were omitted. To simulate multiple task changes, a single simulated browsing session consisted of passing data from one user session through WordSieve three times. Thus information presented to the system was as if the user alternated searching for information about genetic algorithms, and information about butterflies three times, for ten minutes each.

Having built up a context model with WordSieve, a vector for each relevant document in each simulated run was computed by running the document through an empty level 1 and multiplying the resulting node values by their values in the other two levels. The vector for the TFIDF was computed as per Salton [17]. For each word in the document, the weights of each word in the vector were defined by equation 2.

$$w_{ij} = tf_{ij} \cdot \log \frac{N}{df_i} \tag{2}$$

Then, each resulting vector was compared to the original query. Similarity to the original query was calculated via the cosine similarity metric shown in equation 3 [17].

$$\frac{\sum_{i=1}^{t} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{t} x_i^2 \cdot \sum_{i=1}^{t} y_i^2}} \tag{3}$$

In all cases, when computing vectors and doing comparison, the algorithms only had access to the documents in one user's session. Each user's session was treated independently.

5.2 Consistency of Results

Because WordSieve is a probabilistic algorithm, it does not produce exactly the same results every time it is run. To test its performance variation, all simulated user sessions were run through WordSieve four times and the results were compared. Although there was variation among individual rankings of documents, the variation was not statistically significant according to a repeated measures ANOVA test ($F(3, 1146)=2.26$, $p<.05$). This suggests that although WordSieve works probabilistically, it is consistent across even a relatively small set of data (in our experiment, 60 minutes worth of browsing). This also suggests that the results of the experiment are not the results of "lucky" runs of the simulation, but that our data set is large enough to faithfully represent the abilities of the algorithm.

5.3 Performance vs. TFIDF

On the whole, when viewing a document in isolation, WordSieve is better able to match the document to its original context than TFIDF. Using a repeated-measures analysis of variance shows that these findings are statistically reliable, $F(1, 382)=91.03$, $p<.05$.

The overall comparison of WordSieve and TFIDF are shown below. The following table reports the average similarity of all relevant documents of all users to their original

	TFIDF	WordSieve
Mean	0.145	0.224
Standard Deviation	0.142	0.170

context as measured by TFIDF and WordSieve. WordSieve's mean performance at this task surpassed that of TFIDF by 54%.

As shown in figures 3 and 4, the overall patterns generally hold true when breaking down the comparisons by context and user. WordSieve outperformed TFIDF in all cases except for users 2 and 3 where it performed almost as well. Without a larger set of users, it is difficult to determine why it did not do as well in those cases. However, it may be significant that both had accessed relatively small number of relevant documents (only user 7 accessed fewer). Overall, these analyses show that the results are reproducible across diverse subsets of the data.

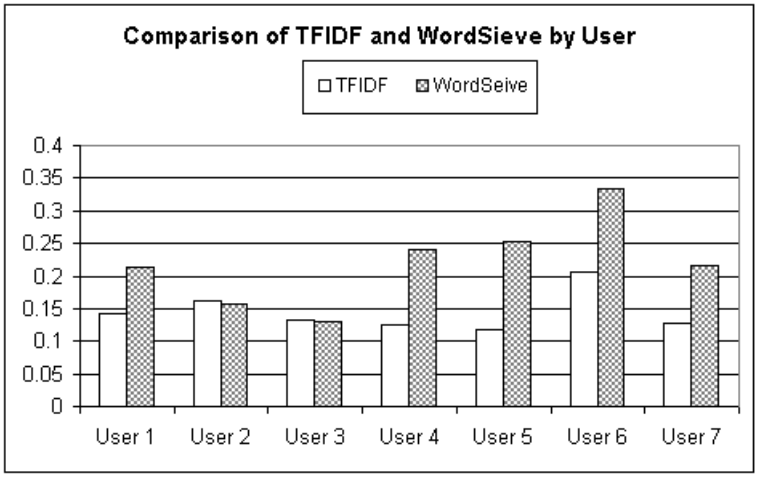

Fig. 3. Comparisons by User

5.4 Discussion

WordSieve extracts context information from documents in the form of keywords. This is standard for many intelligent information agents, and for the purposes of document retrieval, seems to be a natural approach. However, the results of these experiments suggest the benefit of taking into account extra contextual information available in user document access patterns, and the effectiveness of the WordSieve algorithm for this task. The experiments suggest that keyword occurrence patterns exist in user document accesses over time, and that those patterns can help characterize the context within

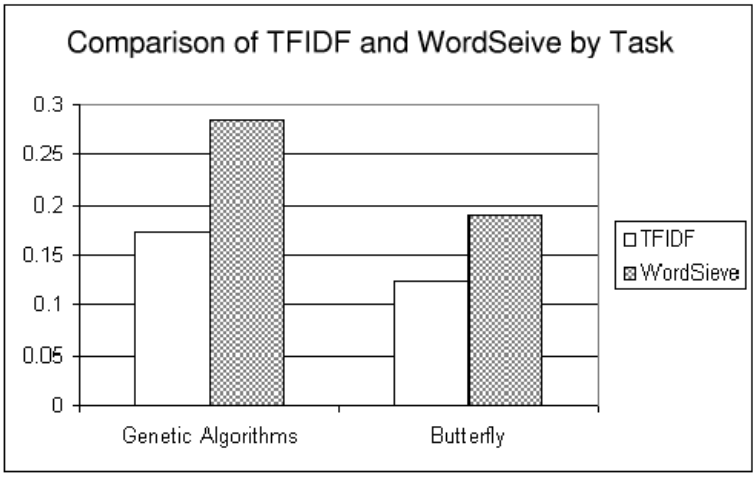

Fig. 4. Comparisons by Query

which the document was accessed, compared to a method such as TFIDF that is unable to capitalize on this information. The results also suggest the viability of using a small, short term memory (the MostFrequentWords level) and probabilistic networks in learning about the user's context, though more experiments and long-term studies are needed.

This research also raises the issue of what constitutes the context of a given instance of a web search. For this experiment, we defined the context as the relevant terms from the topic for which the users were asked to search. However, a number of comments we received from users after they had performed the experiment question those assumptions. The users all seemed to have more trouble with the butterfly query than with the genetic algorithm query.[3] Two users commented that they had found much information about a particular kind of butterfly, but not about Asian butterflies in general. This suggests that the user's perception of the context was informed both by the explicit task given the user and the kind of information the user actually found, and suggests that perhaps the context should be defined not only as what the users were asked to find, but what they were asked to find plus what they actually did find. That type of context description could be generated by asking the users to write five keywords that they thought characterized the web pages they actually found, but that were not in the original question, and using those to augment the task description. However, we expect that both TFIDF and WordSieve would benefit from these expanded queries, so that the comparative performance patterns for the two algorithms would still hold.

We believe we can improve this algorithm's performance. We have not yet sufficiently explored the values assigned to the free parameters in the system to see how we can increase the accuracy of this model and decrease the variance in its perfor-

[3] Figures 3 and 4 shows that documents for the butterfly query tended to rank lower on average. That may indicate a lack of web pages which clearly map to that context.

mance. Also, another level could be added which would become sensitized to only non-discriminators. These words could then be banned from the upper two levels, or perhaps from the lower levels altogether, assuring that space in the profile is reserved for useful words only.

Obviously, there are many forms of contextual information that WordSieve does not yet take into account. For example, WordSieve does not take into account the location of a term on a page, nor is it able to treat a document differently if a user keeps returning to it. For example, consider a situation where a user performs a web search, then clicks on many links from that page, returning frequently to make another choice. That page is probably useful in determining the context, but WordSieve would not treat it in any special way (other than re-reading it many times). Thus we see WordSieve as a useful tool to reflect one aspect of context, to be augmented with others for richer descriptions.

Another interesting question concerns how to account for the quality of a document in the subject domain being searched. WordSieve's analysis does not address this. In Calvin, users can set up customized filters to stop certain documents from being indexed (e.g., to filter out advertisements and error pages), but this increases the need for user configuration.

6 Relationship to Other Work

Calvin is an intelligent agent for learning about a user's context and making suggestions based on that information. Perhaps the best known agent in this class is the Microsoft Office Assistant [12]. Using a Bayesian network, the office assistant infers what a user's goals might be from the user's behavior, and makes suggestions based on those inferences. However, the Office Assistant has a very specific model of how the underlying application works, and its suggestions deal with how to use the software. Thus, this agent's understanding of context is tailored to its special-purpose task. The advantage of this approach is the ability to give very specific advice. However, the advantage is gained at the cost of generality. (See [11] for comments on the potential difficulty of applying a problem-specific notion of context to form a broader theory.) CALVIN could potentially make suggestions based on the documents loaded into any application. Although it does not claim a formal theory of context, its architecture is easily transferred across document access applications.

Margin Notes [16] is another agent that suggests previously accessed documents. Unlike Calvin, which indexes documents accessed at runtime, this system pre-indexes documents and email stored on the user's computer. Then, while the user is accessing information on the WWW, the user's web pages are automatically annotated with references to related pre-indexed files. Margin Notes uses TFIDF for pre-indexing documents, so its understanding of context is based purely on the content of a set of documents and not on how the user accesses them. Other approaches also index documents in isolation from the context in which they were accessed [10].

Watson [6] observes uses of standard software tools, such as browsers and word processors, and generates queries to WWW search engines for context-relevant information. Watson, like Calvin, uses a vector representation of the context. However, it focuses on information about the immediate task context, rather than information about

the user's task sequence. Its method of document parsing helps it find significant words without a larger corpus, in a spirit similar to WordSieve, and its automated use of WWW search engines gives it an advantage over both Calvin and Margin Notes in that it can suggest documents which system users have not yet seen. In should be noted, however, that as document indices are generated by WordSieve for multiple users, those indices can enable cross-user retrievals, providing an individual user with new documents expected to be of interest in the current context.

Other techniques, such as data mining with rule learning, have been employed to record user profiles [1]. Rule learning methods would be well-suited to finding the kinds of terms that WordSieve learns. However, rule learning usually requires a database of user activity to analyze in order to find the rules, while WordSieve works incrementally in real time.

Finally, the idea of learning user profiles for making suggestions has been applied to areas other than text document indexing and retrieval, such as recommending television programs [9]. The PTV system builds a user profile using information such as explicit descriptions of preferences, as well as user watching practices. This system also makes use of the preferences of other users who have similar profiles. One design challenge facing such systems is that the actual contents of the television programs are opaque to the system, unlike the contents of text documents for systems such as WordSieve.

7 Conclusions

In this paper, we have presented WordSieve, a new algorithm for characterizing a user's context by analyzing documents which the user is accessing. The algorithm builds a user profile during document access to reflect the range of user interests, and generates context profiles to reflect the user's current context. WordSieve outperforms TFIDF in initial experiments on associating documents to the contexts in which they were accessed. This performance gain does not seem to be specific to some subset of the overall data, but appears to generalize well over various subsets of the data which we have examined. Research on WordSieve and its application suggests a number of questions for context studies research, especially concerning the use of context in Intelligent Information Agents and the kinds of information about a user's context that can be learned automatically from implicit feedback.

References

1. Gediminas Adomavicius and Alexander Tuzhilin. Using data mining methods to build customer profiles. *Computer*, 34(2):74–82, February 2001.
2. V. Akman and M. Surav. Steps toward formalizing context. *AI Magazine*, 17(3):55–72, 1996.
3. Marko Balabanović. An interface for learning multi-topic user profiles from implicit feedback. In *AAAI-98 Workshop on Recommender Systems*, 1998.
4. M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Philosophical Foundations of Artificial Intelligence*, 12(3), July 2000.
5. Bruno Bouzy and Tristan Cazenave. Using the object oriented paradigm to model context in computer go. In *Proceedings of the First International and Interdisciplinary Conference on Modeling and Using Context*, 1997.

6. J. Budzik and J. K. Hammond. Watson: Anticipating and contextualizing information needs. In *Proceedings of the Sixty-second Annual Meeting of the American Society for Information Science*, Medford, NJ, 1999. Information Today, Inc.
7. J. Budzik, K. Hammond, and L. Birnbaum. Information access in context. In *Knowledge based systems*, 2001.
8. J. Budzik, K. Hammond, L. Birnbaum, and M. Krema. Beyond similarity. In *Working Notes of the AAAI-2000 Workshop on AI for Web Search*. AAAI Press, Menlo Park, 2000.
9. P. Cotter and B. Smyth. PTV: Intelligent personalised tv guides. In *Proceedings of the 12th Innovative Applications of Artificial Intelligence (IAAI-2000) Conference*. AAAI Press, 2000.
10. Marti A. Hearst. *Context and Structure in Automated Full-Text Information Access*. PhD thesis, University of California at Berkeley, 1994.
11. Graeme Hirst. Context as a spurious concept. In *Proceedings, Conference on Intelligent Processing and Computational Linguistics*, pages 273–287, Mexico City, February 2000.
12. Eric Horvitz, Jack Breese, David Heckerman, David Hovel, and Koos Rommelse. Inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, 1998.
13. David Leake, Travis Bauer, Anna Maguitman, and David Wilson. Capture, storage and reuse of lessons about information resources: Supporting task-based information search. In *Proceedings of the AAAI-2000 Workshop on Intelligent Lessons Learned Systems*, Menlo Park, CA, 2000. AAAI Press.
14. Henry Lieberman, Neil Van Dyke, and Adriana Vivacqua. Let's browse: A collaborative web browsing agent. In *Proceedings of the 1999 international conference on Intelligent user interfaces*, pages 65–68, 1999.
15. Carlo Penco. Objective and cognitive context. In Paolo Bouquet, Luigi Serafini, Patrick Brézillon, Massimo Benerecetti, and Francesca Castellani, editors, *Modeling and Using Contexts: Proceedings of the Second International and Interdisciplinary Conference*, pages 270–283. Springer-Verlag, 1999.
16. Bradley J. Rhodes. Margin notes: Building a contextually aware associative memory. In *Proceedings of the 2000 international conference on Intelligent user interfaces*, pages 219–224, Jan 2000.
17. Gerard Salton. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer*. Addison-Wesley Series in Computer Science. Addison-Wesley Publishing Company, Inc., 1989.
18. Roy Turner. Context-sensitive reasoning for autonomous agents and cooperative distributed problem solving. In *Proceedings of the IJCAI Workshop on Using Knowledge in its Context*, Chambery, France, 1993.
19. Roy Turner. Context-mediated behavior. In J. Mira, A.P. del Pobil, and M. Ali, editors, *Lecture Notes in Artificial Intelligence 1415: Methodology and Tools in Knowledge-Based Systems*, pages 538–547, Benicassim, Spain, June 1998. Springer, New York.