Real Time User Context Modeling for Information Retrieval Agents

Travis Bauer
Computer Science Department
Lindley Hall, Indiana University
150 S. Woodlawn Avenue
Bloomington, IN 47405, U.S.A.
trbauer@cs.indiana.edu

David B. Leake
Computer Science Department
Lindley Hall, Indiana University
150 S. Woodlawn Avenue
Bloomington, IN 47405, U.S.A.
leake@cs.indiana.edu

ABSTRACT

The success of personal information agents depends on their ability to provide task-relevant information. This paper presents WordSieve, a new algorithm that generates context descriptions to guide document indexing and retrieval. WordSieve exploits information about the sequence of accessed documents to identify words which indicate a shift in context. We have tested WordSieve in a personal information agent, Calvin, which monitors a user's document access, generates a representation of the user's task context, indexes the resources consulted, and presents recommendations for other resources that were consulted in similar prior contexts. In initial experiments, WordSieve outperforms *term frequency/inverse document frequency* at matching documents to hand-coded vector representations of the task contexts in which they were originally consulted, where the task context representations are term vectors representing a specific search task given to the user.

1. INTRODUCTION

By monitoring the user to infer a task context, personal information agents can assist users in intelligent ways, such as performing autonomous web searches, or suggesting knowledge resources that were useful in similar past contexts. We are investigating how personal information agents can capture contextual information as users seek information, and can use that information to suggest resources consulted in similar contexts in the past.

To provide real-time context-based retrieval, we have developed a new algorithm, WordSieve, which generates vector representations of documents as the user accesses them without requiring comprehensive statistics about word distributions in the documents accessed. Instead, working with a relatively small memory (in current tests, at most 650 unique words), it identifies task-specific keywords from document access sequences. WordSieve exploits information the user provides implicitly by virtue of accessing similar documents together. It does this by building access profiles which identify terms occurring frequently in sequences of document accesses and which are expected to be useful for distinguishing sets of documents related to the same task. In this way, WordSieve exploits extra knowledge about the document access context to generate indices that reflect the task context.

In our experiments, WordSieve outperformed term frequency/inverse document frequency (TFIDF) [9], a popular indexing algorithm, at matching documents to hand-coded vector representations of the task contexts in which they were originally consulted, where the task context representations are term vectors representing a specific search task given to the user. This paper presents WordSieve's architecture and performance.

2. WORDSIEVE

WordSieve identifies task-specific terms and generates representations of documents reflecting both their content and the context in which they were consulted. WordSieve identifies boundaries in user tasks from implicit information—the characteristics of document access subsequences—and describes tasks by keywords that are useful in distinguishing those subsequences. To illustrate, consider a user searching the web first for information about genetic algorithms, then for some other subject. The term "genetic" is a good task-specific term since it occurs frequently for a certain period of time, and then stops occurring. In doing so, "genetic" effectively segments the document accesses into two groups, the first group probably related to "genetic." (Note that the term "genetic" need not occur in all the documents accessed in the sequence, and that there may be some overlap between tasks as transitions occur).

WordSieve represents user access patterns in a 3-layer architecture, the lowest layer reflecting the frequency of words currently occurring in a document stream and the upper two layers reflecting a user access profile. Each layer consists of a set of units, each of which corresponds to a word from the input text stream. WordSieve uses a competitive learning algorithm to assign keywords to units while the document stream passes through it. The goal of WordSieve is to identify keywords that occur frequently during a given task, but infrequently during other tasks. WordSieve's information flow is shown in figure 1, and the function of each layer is described below.

2.1 Layer 1 - Most Frequent Words

The bottom layer (1) performs initial processing on the text stream. Through a competitive learning process, this layer assigns the units in this layer to the most frequently occurring words. In the current implementation, this layer contains 150 units, so it can be sensitive to only 150 unique words at a time. The number of units in this layer is significant, because it is a competitive network. If there are too many units, there is not enough competition and it will not iden-

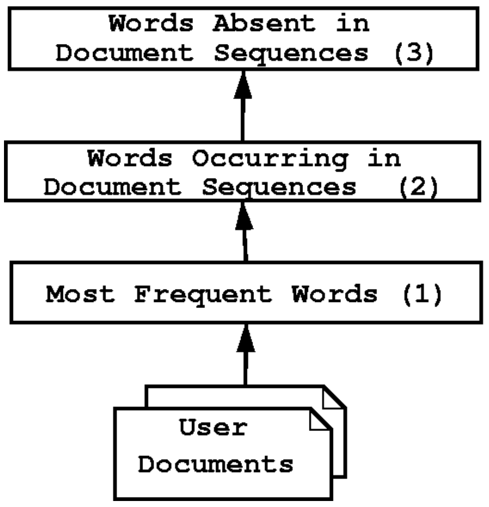
Words Absent in Document Sequences (3)
Words Occurring in Document Sequences (2)
Most Frequent Words (1)
User Documents

Figure 1: Information flow in WordSieve

tify the correct words. If there are too few units, the competition is so great that it will not learn all the words needed. The algorithm appears to work well under a reasonably large range of layer sizes, but we have not yet conducted a comprehensive analysis of the effects of changing the layer size.

Words are assigned to units as follows: Each of the 150 units is associated with a unique word. As the documents are read, each term passes through the bottom layer in the order in which it occurs in the document. If the word is already associated with a unit in the layer, the excitement of that unit is increased by a value α (see below). If the word is not associated with any term, it is given a chance to "take over" a randomly chosen unit. The term takes over the chosen unit with a probability of $0.0001(e-100)^2$, where e is the excitement of the unit and has a range of 0 to 100. The chance of "taking over" the unit thus decreases with the excitement of that unit. Also, the values of all the units decay at a rate β.

2.2 Layers 2 and 3 - Partition Words

Obviously, not all of the frequently occurring words are of use in characterizing the context. Terms such as "the," "and," and other common words occur frequently in many documents regardless of the context. Layer 1 learns frequently occurring words, including both context-determining words and common words with low information content. Layers 2 and 3 determine which of these words characterize the context.

The upper two layers (2 and 3) are continuously presented with the state of the units in the bottom layer. Each upper layer contains 500 units, which are paired such that units in both layers are sensitized to the same words. Layer 2 detects words that occur frequently in successive documents. Layer 3 detects which of the words in layer 2 tend to not occur for long periods of time. Each unit is associated with two values: excitement and priming. In layer 2, a unit's priming increases while its word is present in layer 1. The unit's excitement increases as a function of the priming. Excitement and priming will decay when the word is absent. In layer 3, almost the opposite happens: the priming increases while the word is absent in level 1. The excitement changes as a function of the priming.

Levels 2 and 3 build up user access profiles. When the excitement values of corresponding terms in layers 2 and 3 are multiplied together, the terms with higher values tend to be the terms which occur frequently for discrete periods of time in the user's document accesses. Such terms can characterize a user's information-seeking task, as is shown in the next section.

3. APPLYING WORDSIEVE IN CALVIN

WordSieve was tested in Calvin, a personal information agent we are developing [6]. Calvin uses WordSieve to suggest task-relevant resources to users as they browse documents. When a user accesses a document, Calvin passes its contents through WordSieve. Based on this information, WordSieve updates the current context and user access profile in real time, while the user is engaged in the task. The user access profile is persistent across user sessions, allowing Calvin to learn user browsing patterns in multiple contexts. Calvin stores profiles for any number of users, loading individual profiles when the user logs into Calvin with a name and password.

4. PERFORMANCE

To evaluate the performance of WordSieve, we performed an experiment to test its ability to match a document to a hand-coded vector representation of the web search task during which it was consulted. In particular, we wanted to see how closely WordSieve could correlate a document to the original search task given to the user. To test this, seven users (computer science graduate students) were asked to browse the Internet for twenty minutes each while being monitored by Calvin. For the first ten minutes, they were asked to find documents on the WWW that were about "The use of genetic algorithms in artificial life." For the second ten minutes, they were asked to search for information about "Tropical butterflies in Southeast Asia." The users were given no restrictions on how to find the pages. They were only instructed that the documents must be loaded into the web browser provided to them.

User access profiles were developed from this data by passing each set of data through WordSieve three times (in the order originally browsed by the user) to simulate one hour of browsing.[1]

Two term vectors were generated for each document, one using WordSieve, and one using TFIDF. To provide a search task characterization, vectors were created to represent the task description given to the users. The WordSieve and TFIDF vectors from each document were compared to their associated task description.

In our experiments, WordSieve generated indices (i.e. term vectors) for documents which were reliably more strongly correlated to the original task description than those produced by TFIDF (F(1,82)=91.1, repeated-measures ANOVA). This generally held across various subsets of the data. The average TFIDF similarity was 0.145 and the average WordSieve similarity was 0.224. This suggests that WordSieve is performing better at generating profiles that reflect a user's task.

In the experiments presented, the tasks were quite distinct. We have not yet conducted experiments on browsing where the tasks overlap. However, if the tasks overlapped, the keywords which overlapped would be treated both by TFIDF and WordSieve as non-discriminators and would have relatively low values in the term vectors generated, and the non-overlapping keywords would have relatively high values. Because these terms would have similar effects on both algorithms, we expect that performance would be equally affected in both algorithms and that the conclusions of the comparison would be the similar.

[1]WordSieve learns about contexts as the user switches tasks. In the twenty minutes of browsing, the user switched tasks only one time. By repeating the sequence of documents, multiple task switches were simulated.

		Num Docs	Standard Deviation		Mean			ANOVA	
			TFIDF	WordSieve	TFIDF	WordSieve	Diff	F	Significant
Overall		381	0.142	0.170	0.145	0.224	+54.48%	91.14	Yes
By User	User 1	64	0.130	0.121	0.142	0.182	+28.17%	6.66	Yes
	User 2	44	0.138	0.137	0.161	0.166	+3.11%	0.04	No
	User 3	44	0.146	0.108	0.132	0.119	-9.85%	0.38	No
	User 4	47	0.100	0.158	0.124	0.177	+42.74%	12.03	Yes
	User 5	87	0.117	0.147	0.117	0.277	+136.75%	117.24	Yes
	User 6	62	0.194	0.225	0.207	0.353	+70.53%	28.31	Yes
	User 7	33	0.138	0.122	0.128	0.210	+64.06%	7.80	Yes
By Query	Genetic	161	0.157	0.191	0.172	0.273	+58.72%	59.20	Yes
	Butterfly	220	0.126	0.142	0.124	0.188	+51.61%	35.76	Yes
Document Length	0-1000	112	0.114	0.209	0.097	0.213	+119.59%	71.30	Yes
	1001-2000	67	0.106	0.142	0.133	0.214	+60.90%	15.51	Yes
	2001-3000	50	0.144	0.154	0.147	0.224	+52.38%	11.62	Yes
	3001-4000	49	0.197	0.165	0.298	0.270	-9.40%	1.20	No
	4001-4999	103	0.108	0.146	0.130	0.221	+70.00%	37.74	Yes

Table 1: Comparison of TFIDF and WordSieve

5. MODELING IN RELATED WORK

Other agents have been designed to monitor user tasks and suggest resources to users. Some such agents are able to make much more specific suggestions than WordSieve is able, usually at the expense of needing a specific domain model, such as pre-created categories. Lumiere is an example of such an agent apporach [5]. Another example focused on information retrieval is found in Chen [3]. Other agents do not require a specific domain model and suggest documents based on vector comparisons with current document content [7, 2, 8]. Some agents use explicit feedback to enhance contextual information [4, 1]. One of the advantages of WordSieve is that it does not require user feedback. However, if desired, it could be modified to take utilize such information.

6. PERSPECTIVES AND FUTURE WORK

WordSieve is most appropriate for agents which monitor ongoing user document accesses, rather than for batch document indexing. Although it does not need to remember or refer to previously accessed documents to build a user access profile, it does need the documents to be presented in a sequence in which the user would actually access them. This is not a problem in a real-time agent environment, but would be in a large, unorganized corpus. Although TFIDF does not have this limitation, it cannot take advantage of the order of document use when that information is available.

7. CONCLUSIONS

In this paper, we have presented WordSieve, an algorithm for indexing documents for personal information agents. WordSieve builds a context profile by reading the documents the user accesses and extracting terms which tend to partition the user's browsing behavior into discrete tasks, in order to generate context-sensitive indices that can be used to suggest relevant documents in the future. In our experiments, WordSieve outperforms TFIDF at generating indices that associate documents with explicit task descriptions that were given to the user to guide browsing, but were unavailable to both programs when they generated their indices. We believe that automated context-based indexing techniques such as WordSieve are potentially useful for in many different kinds of environments where personal information agents must assist users in organizing and appropriately retrieving digital resources.

8. REFERENCES

[1] E. Bloedorn, I. Mani, and T. R. MacMillan. Representational issues in machine learning of user profiles. In *Proceedings of the Thirteenth National Conference on Artificial Intelligence and Eighth Innovative Applications of Artificial Intelligence Conference*, 1996.

[2] 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.

[3] H. Chen and S. T. Dumais. Bringing order to the web: automatically categorizing search results. In *Proceedings of CHI'00, Human Factors in Computing Systems*, pages 145–152, 2000.

[4] W. W. Cohen. Learning rules that classify e-mail. In *Papers from the AAAI Spring Symposium on Machine Learning in Information Access*, pages 18–25, 1996.

[5] E. Horvitz, J. Breese, D. Heckerman, D. Hovel, and K. Rommelse. Inferring the goals and needs of software users. In *Proceedings of the Fourteenth Conference on Uncertainty in Artificial Intelligence*, pages 256–265, 1998.

[6] D. Leake, T. Bauer, A. Maguitman, and D. 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.

[7] H. Lieberman, N. V. Dyke, and A. Vivacqua. Let's browse: A collaborative web browsing agent. In *Proceedings of the 1999 international conference on Intelligent user interfaces*, pages 65–68, 1999.

[8] B. 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.

[9] G. 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.