Adaptive Lightweight Text Filtering

Gabriel L. Somlo and Adele E. Howe

Colorado State University
Computer Science Dept.
Fort Collins, CO 80523, U.S.A.
{howe, somlo}@cs.colostate.edu
tel.: +1 (970) 491-7589
fax: +1 (970) 491-2466

Adaptive Lightweight Text Filtering

Gabriel L. Somlo and Adele E. Howe

Colorado State University
Computer Science Dept.
Fort Collins, CO 80523, U.S.A.
{howe, somlo}@cs.colostate.edu

Abstract. We present a lightweight text filtering algorithm intended for use with personal Web information agents. Fast response and low resource usage were the key design criteria, in order to allow the algorithm to run on the client side. The algorithm learns adaptive queries and dissemination thresholds for each topic of interest in its user profile. We describe a factorial experiment used to test the robustness of the algorithm under different learning parameters and more importantly, under limited training feedback. The experiment borrows from standard practice in TREC by using TREC-5 data to simulate a user reading and categorizing documents. Results indicate that the algorithm is capable of achieving good filtering performance, even with little user feedback.

1 Introduction

Text filtering makes binary decisions about whether to disseminate documents that arrive from a dynamic incoming stream. *Adaptive* filtering systems [3] start out with little or no information about the user's needs, and the decision of whether to disseminate a document must be made when the document becomes available. The system is given feedback on each disseminated document to update its user profile and improve its filtering performance.

In this paper, we present a lightweight filtering system designed for use in personal Web information agents. We assess how well the algorithm works with little feedback, a requirement for its application as a *personal* information gathering agent. Also, we assess the effect of algorithm parameters on robustness.

We make two key contributions in our research. First, our algorithm adapts standard filtering techniques to the needs of personalized web applications: lightweight, privacy protecting and responsive to user provided examples. The algorithm learns a profile of user information interests. Second, we adapt a rigorous evaluation method to web systems by using text filtering benchmarks to simulate user behavior. Traditionally, Web systems have been evaluated with user studies, with the disadvantages of slow data collection, little experimental control and decreased objectivity of conclusions. Relying on simulated user feedback allows us to test many alternative design decisions *before* subjecting the system to a user study, which means we are less likely to waste subjects' time and are more likely to produce a well tuned system.

2 Filtering Algorithm

At its core, our filtering algorithm uses TF-IDF vectors [8] to represent documents and topic queries. As documents arrive from the incoming stream, they are first transformed into TF (term frequency) vectors, by associating each distinct word in the document with its frequency count, and then are weighted by each word's IDF (inverse document frequency). The components of the vectors are computed using the well-known formula: $TFIDF_{d,t} = TF_{d,t} \cdot IDF_t$ where $TF_{d,t}$ is the term frequency of t in document d. For any given term, its IDF is computed based on the fraction of the total documents that contain this term at least once: $IDF_t = \log \frac{D}{DF_t}$ where D is the number of documents in the collection, and DF_t is the number of documents containing t (document frequency).

Traditionally, D and DF values assume the existence of a fixed corpus, in which documents are known a priori. In filtering, however, we do not assume random access to all documents. Thus, D and DF are approximated: every time a new incoming document is processed, we increment D and the entries into DF that correspond to the terms contained in the new document.

The similarity between two TF-IDF vectors (e.g., between a query and a document vector) is computed using the cosine metric [8], i.e., the normalized dot product between the two vectors. This measures the cosine of the "angle" between the two vectors, and ranges from 1 (when the vectors have the same direction) to 0 (when the vectors have nothing in common).

For filtering, topic queries generalize subjects of interest to a user. In addition to a TF-IDF vector, each topic query in the profile maintains a dissemination threshold, the minimum similarity between the vector of an incoming document and the topic's query vector required for disseminating the document. Dissemination thresholds range between $[0, 1]$. When a new document arrives, its vector is compared to each topic query vector; if the similarity exceeds the dissemination threshold for at least one topic, the document is shown to the user.

Filtering algorithms start with profile queries created by converting a description of each topic into a TF-IDF vector; the vectors are updated based only on feedback from documents that have been disseminated [3]. Our algorithm starts with an empty profile. Learning occurs from positive examples of relevant document-topic pairs provided by the user. For our application, new topics are created in the user profile when the user submits a relevant document for a non-existent topic; the document's TF-IDF vector forms the initial query vector for the topic and the dissemination threshold is initialized to 0.5. Earlier, pilot experiments showed that the initial value did not matter; feedback quickly moves the value within a short distance of its apparent ideal.

For existing topics, the query vector is learned using a mechanism inspired by relevance feedback [7]. Both the original query vector and the document vector are normalized to eliminate the influence of document length, after which the document vector is weighted by a document weight w and added to the query vector. The document weight controls how much influence the original query vector and the new document vector will each have on the updated query.

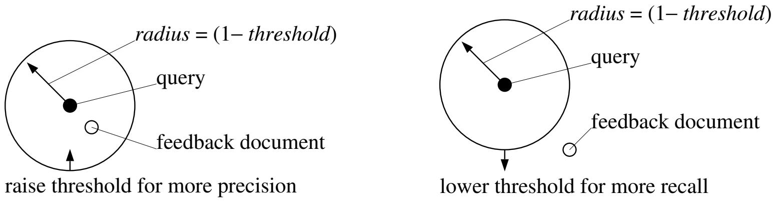


Fig. 1. Learning the dissemination threshold

```
AddDocToTopic(doc, topic)
    s = similarity(doc.vector, topic.vector);
    normalize(doc.vector);
    normalize(topic.vector);
    topic.vector += w · doc.vector;
    normalize(topic.vector);
    topic.threshold += α · (s − topic.threshold);
```

Fig. 2. Topic learning mechanism

Like the query vector, the dissemination threshold also needs to be adapted. Callan [1] proposes a dissemination threshold that starts low and grows in time to improve precision. The system presented in [9, 10] features dissemination thresholds that can both increase and decrease, but the algorithm heavily relies on numerous cached documents to optimize the dissemination threshold.

To keep our filtering system lightweight, we prohibit the algorithm from caching past document vectors. Instead, the threshold tracks the similarity between the original query vector and the latest feedback document, with a learning rate α. We can imagine the original query vector to be at the center of a "bubble", with a radius given by the dissemination threshold. If the new feedback document is inside this bubble, we can raise the threshold (shrink the bubble), to improve precision. For feedback documents outside the bubble, we must lower the threshold to improve recall. This process is illustrated graphically in Figure 1. Unlike Callan's method, ours permits non-monotonic changes to the threshold vector. The learning algorithm, for both the query vector and the dissemination threshold, is presented in pseudocode in Figure 2. The pseudocode includes two critical parameters: w (document weight) and α (learning rate).

3 Personal Web Information Agents

Personal Web information agents, such as Letizia [6], WebWatcher [4] and WebMate [2], can be tailored, over time, to help their users find information that matches their own interests. The helper agent intercedes between the user and their Web access, learning a model of user requests to be used in modifying requests and/or filtering results. Letizia and WebWatcher recommend links that are likely to lead to interesting documents by pre-fetching pages (Letizia) or comparing the profile to a fixed collection (WebWatcher). WebMate uses greedy incremental clustering to build its user profile.

Web information agents are similar to text filtering systems. Both types of systems examine an incoming document stream; agents often generate the stream for themselves by crawling the Web or by extracting links from a Web page during browsing. Both types of systems maintain user profiles. Both systems make binary decisions about the documents in the stream: whether to recommend them to the user based on their similarity to the user profile. Thus, we propose that filtering algorithms such as the one presented in Section 2 be used with Web helper agents, and that performance evaluation of these agents be done with rigorous methods borrowed from IR, and specifically from text filtering.

We simulate the behavior of the user with data from a text filtering benchmark test. The agent starts with an empty profile. The user can create new topics by showing the agent a relevant document and providing a topic name. The agent will compare all new documents from its incoming stream against each topic in the profile, and decide whether to disseminate them to the user. How the incoming stream is generated is beyond the scope of this paper, but any method used by the Web agents mentioned earlier in this section will do.

Usually, in text filtering, only disseminated documents are used for feedback [3]. Our scenario relaxes this requirement by allowing users to present documents they encountered independently when surfing the Web. At any time, a user can create a new topic or reinforce an existing one. It is likely though that users will stop providing feedback on a given topic after a certain number of documents, and expect the agent to continue its filtering task without further instruction.

4 Performance Evaluation

The purpose of our study is to determine the robustness of our filtering system, in particular its sensitivity to the amount of feedback and to parameter settings. The experiment proceeds as follows: For each new incoming document, the system predicts relevance based on the current user profile. The new document is compared to each internal profile topic vector, and a record is made when any of the dissemination thresholds have been passed. Then, the "user" provides feedback for relevant documents, which is simulated with the relevance ratings in the datasets. The new document is added to all topics for which it is known to be relevant. Our independent variables are:

- w document weight that controls how much influence the new document vector exerts on the updated query. Values below one favor the original query vector. Our experiments used: 0.1, 0.15, 0.2, 0.25, 0.5, 1.0, 2.0, and 4.0.
- α the learning rate for the dissemination threshold, with values of 0.1, 0.3, 0.5, 0.7, and 0.9.
- U_{max} the maximum number of updates (documents provided as feedback) for each topic, with values of 5, 10, 20, 30, 40, 50, 60, 70, 100, 200, and ∞ (as many as possible).

We use two datasets: Foreign Broadcasting Information Service (FBIS) collection, and Los Angeles Times (LATIMES) collection from TREC[1] disk #5. The FBIS and the LATIMES collections consist of 48527 and 36330 news articles, classified into 194 topics. Because the first document that is relevant to a topic is always used to create an entry for the topic in the user profile, we only use topics with more than one relevant document, which reduces the total number of topics to 186.

For each combination of independent variables, we record:

RF number of relevant documents found correctly as relevant;
RM number of relevant documents missed (incorrectly identified as non-relevant);
NF number of non-relevant documents found as relevant;

These values are used to compute the following performance metrics:

Recall	number of relevant documents correctly disseminated	$recall = \frac{RF}{RF+RM}$
Precision	number of relevant disseminated documents	$precision = \frac{RF}{RF+NF}$
LGF	performance metric proposed by Lewis and Gale [5], assume equal weights for recall and precision.	$LGF = \frac{2 \cdot precision \cdot recall}{precision+recall}$
LF1, LF2	linear utility functions from TREC-8 evaluation [3]	$LF1 = 3RF - 2NF$ $LF2 = 3RF - NF$

5 Results

First, we test the influence of algorithm parameters under ideal conditions, when all possible learning information is used ($U_{max} = \infty$). Then, we evaluate robustness by reducing the amount of feedback to the algorithm, i.e., smaller values of U_{max}. A single user building their own web profile (our application) cannot be expected to provide thousands of relevance judgments for every topic.

5.1 Impact of Parameters α and w when $U_{max} = \infty$

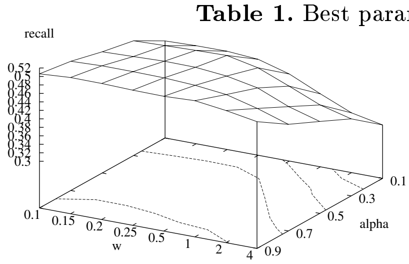
A two-way Analysis of Variance indicates that both α and w, individually and jointly, strongly influence the value of recall ($p < 0.00001$). Recall is consistently good for values of $\alpha \geq 0.7$ and $w \leq 0.5$, but strongly decreases outside this region. A surface plot as well as an ANOVA table for each dataset are shown in Figure 3. Performance depends primarily on α and is close to optimal as long as w is within a range of values. The best parameter values are in Table 1.

Precision depends strongly on α ($p < 0.00001$). As one would expect, precision is highest for the smallest tested value of $\alpha = 0.1$, the most conservative update. The document weight w also has a significant effect on precision, albeit much smaller. The best values for precision seem to occur for $w \in [0.25, 1.0]$. The interaction effects are negligible. 0.5104 at $\alpha = 0.1$ and $w = 1.0$ for LATIMES.

[1] TREC (Text REtrieval Conference) document collections can be ordered from `http://trec.nist.gov`

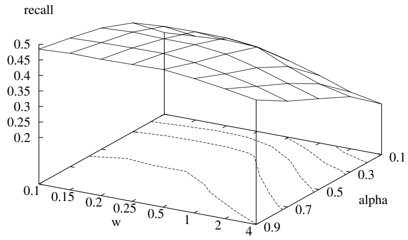
	FBIS					LATIMES				
Metric:	Rec.	Prec.	LGF	$LF1$	$LF2$	Rec.	Prec.	LGF	$LF1$	$LF2$
α	0.9	0.1	0.1	0.1	0.1	0.9	0.1	0.3	0.1	0.1
w	0.5	0.5	0.2	0.5	0.25	0.5	1.0	0.5	1.0	0.25
Value	0.511	0.516	0.453	4533	8441	0.493	0.510	0.386	1012	2097

Table 1. Best parameter values for each metric



ANOVA	F	p
α	403.36	0.00001
w	142.44	0.00001
α:w	27.59	0.00001

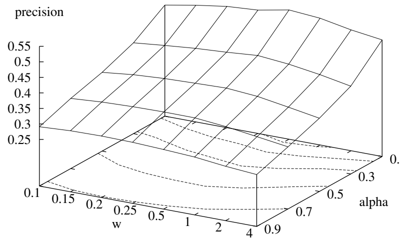
a) Dataset: FBIS



ANOVA	F	p
α	435.12	0.00001
w	96.07	0.00001
α:w	8.43	0.00626

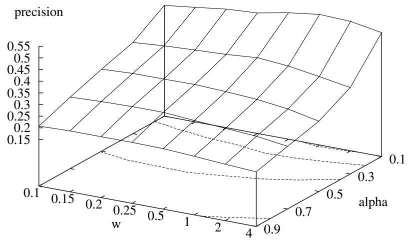
b) Dataset: LATIMES

Fig. 3. Effects of learning rate α and document weight w on recall



ANOVA	F	p
α	613.63	0.00001
w	9.52	0.00389
α:w	0.99	0.32598

a) Dataset: FBIS



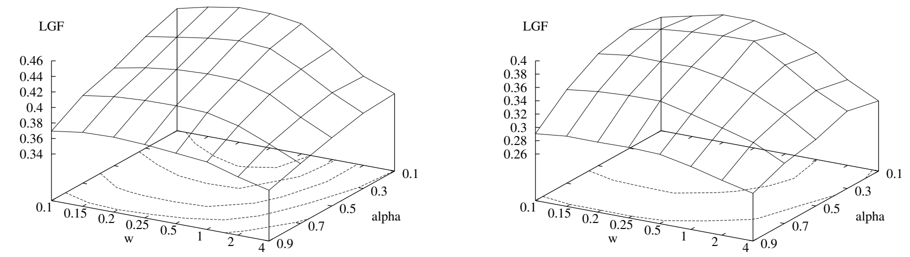
ANOVA	F	p
α	427.54	0.00001
w	3.53	0.06824
α:w	2.41	0.12896

b) Dataset: LATIMES

Fig. 4. Effects of learning rate α and document weight w on precision

The previous analysis considers recall and precision in isolation, which is clearly unrealistic for our application. The measures that encompass both recall and precision (LGF, $LF1$, $LF2$) show strong effects of w and α (see Figures 5,6,7). LGF and $LF1$ show interaction effects. LGF shows almost best performance within the interval, $\alpha \leq 0.3$ and $w \leq 0.5$.

In conclusion, with unlimited training, the best learning parameters for our algorithm are $w \in [0.2, 1.0]$, and $\alpha \leq 0.3$ for everything except recall, where val-



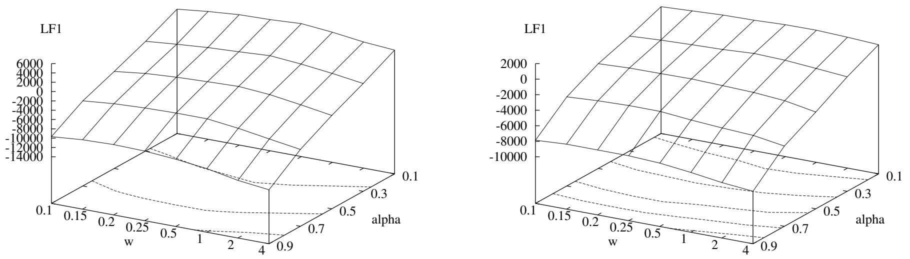
ANOVA	F	p
α	348.49	0.00001
w	209.67	0.00001
α:w	17.58	0.00017

a) Dataset: FBIS

ANOVA	F	p
α	127.47	0.00001
w	99.41	0.00001
α:w	6.42	0.01574

b) Dataset: LATIMES

Fig. 5. Effects of learning rate α and document weight w on LGF



ANOVA	F	p
α	1736.64	0.00001
w	36.18	0.00001
α:w	6.23	0.01726

a) Dataset: FBIS

ANOVA	F	p
α	1070.65	0.00001
w	10.36	0.00272
α:w	5.06	0.03063

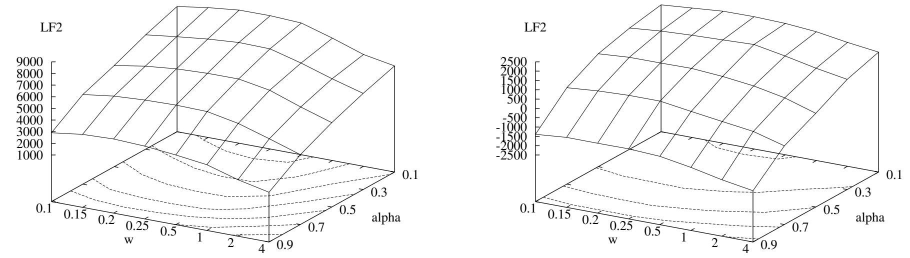
b) Dataset: LATIMES

Fig. 6. Effects of learning rate α and document weight w on TREC8 $LF1$

ues of $\alpha \geq 0.9$ give the best results. Fortunately for our application, performance is always pretty good within the same interval on *both* datasets; thus, we would expect our best parameter settings to generalize to other data sets.

5.2 Robustness with Less Feedback (U_{max})

The two datasets vary wildly in the number of relevant documents per topic. Histograms of the distribution of topics according to the number of known relevant documents they contain are given in Figure 8. To determine the effects of limiting training, we had to focus on topics for which many documents were available. We used the topics in the fourth quartile of the above distributions:



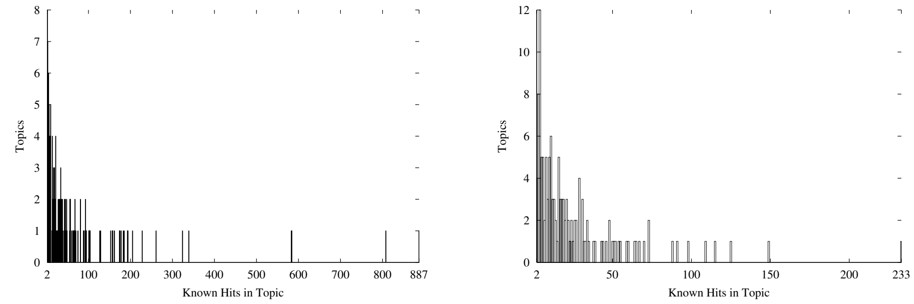
ANOVA	F	p
α	643.15	0.00001
w	109.24	0.00001
α:w	0.11	0.73996

a) Dataset: FBIS

ANOVA	F	p
α	426.38	0.00001
w	29.97	0.00001
α:w	0.99	0.32714

b) Dataset: LATIMES

Fig. 7. Effects of learning rate α and document weight w on TREC8 $LF2$



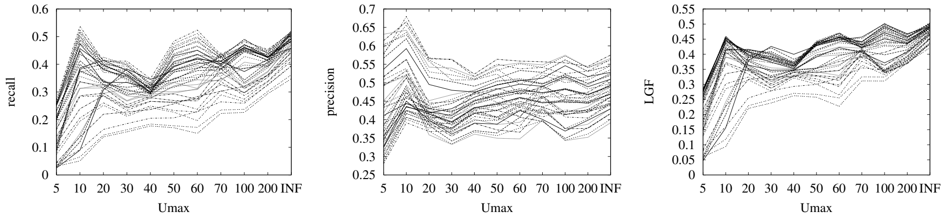
a) Dataset: FBIS b) Dataset: LATIMES

Fig. 8. Distribution of known relevant documents across topics

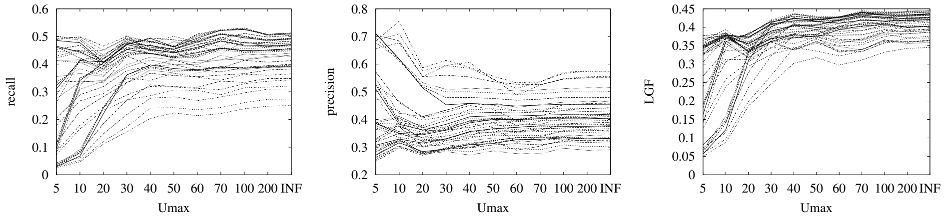
this includes 36 topics from the FBIS dataset, with 75 or more known relevant documents, and 36 topics from the LATIMES dataset, with 31 or more known relevant documents. These topics include the majority of known relevant documents in our datasets: 7,678 for FBIS, and 2,368 for LATIMES.

We plotted recall, precision, and LGF against U_{max} for each combination of α and w in Figure 9. These plots show that the interaction effects between U_{max} and the two parameters are small for both datasets. Additionally, the effect of U_{max} is non-monotonic, and that while no parameter combination dominates all values of U_{max}, some are consistently among the top few.

ANOVA tests indicate that U_{max} strongly influences recall: $F = 128.34, p < 0.00001$ for FBIS, and $F = 36.30, p < 0.00001$ for LATIMES. In comparison, the influence on precision is significant only on the FBIS dataset ($F = 47.36, p < 0.00001$ for FBIS, $F = 1.02, p < 0.31$ for LATIMES). The U_{max} on LGF is significant for both datasets: $F = 100.59, p < 0.00001$ for FBIS, and $F = 31.07, p < 0.00001$ for LATIMES. In contrast to the unlimited updates,

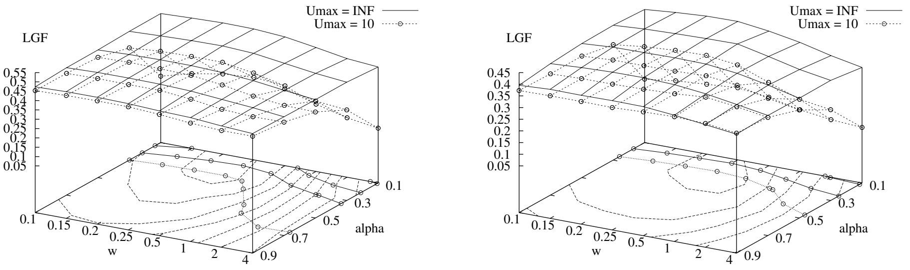


a) Dataset: FBIS



b) Dataset: LATIMES

Fig. 9. Influence of U_{max} on recall, precision, and LGF for combinations of α and w



a) Dataset: FBIS b) Dataset: LATIMES

Fig. 10. Influence of U_{max} on LGF for the FBIS dataset

limited updates favor high values of α, while w remains in the same range. This result follows from the need to make the most out of the available data.

Interestingly, our plots show that very good results can be obtained after as few as 10 updates per topic. After 10 updates per topic for FBIS, we reached 0.4489 recall and 0.4501 precision, resulting in 0.4495 LGF at $\alpha = 0.9$ and $w = 0.5$; LATIMES showed 0.4055 recall, 0.3818 precision, and LGF of 0.3933, at $\alpha = 0.7$ and $w = 1.0$. As an example, Figure 10 displays the surface plots of LGF at $U_{max} = 10$ and $U_{max} = \infty$.

6 Conclusions

We presented a lightweight filtering algorithm intended for use with Web information agents, and an evaluation method for such algorithms based on TREC benchmarks. We ran a factorial experiment to test the algorithm's robustness. We found that performance is robust within a relatively wide interval of parameter values. Importantly for our target application, we found that the algorithm is robust against limited training information; we observed a slow degradation in performance as the amount of feedback was reduced down to 10.

When used in a Web information agent, our algorithm will need to train with up to 10 or 20 positive examples per topic. In consequence, the algorithm must start out with a large learning rate for the dissemination threshold. An idea we intend to test in the future is to lower α as more documents are used to train a particular topic.

Using benchmarks instead of user studies has allowed us to collect large amounts of information in a short period of time. First, we intend to test our algorithm on more datasets, starting with the Financial Times document collection used in the filtering track at recent TREC conferences. Ultimately, when these algorithms have been incorporated into the web agent, we will test the efficacy of the design decisions derived here in a user study.

References

1. Callan, J. Learning While Filtering Documents *Proceedings of the 21st International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Melbourne, Australia, 1998.
2. Chen, L., Sycara, K. WebMate: A Personal Agent for Browsing and Searching. *Proceedings of the Second International Conference on Autonomous Agents*, Minneapolis, MN, 1998.
3. Hull, D.A., Robertson, S. The TREC-8 Filtering Track Final Report *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, 1999.
4. Joachims, T., Freitag, D., Mitchell, T. WebWatcher: A Tour Guide for the World Wide Web. *Proceedings of the 15th International Joint Conference on Artificial Intelligence (IJCAI-97)*, Nagoya, Japan, 1997.
5. Lewis, D.D., Gale, W.A. A Sequential Algorithm for Training Text Classifiers *Proceedings of the 17th International ACM-SIGIR Conference on Research and Development in Information Retrieval*, Dublin, Ireland, 1994.
6. Lieberman, H. Letizia: An Agent That Assists Web Browsing. *Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95)*, Montreal, Canada, 1995.
7. Rocchio, J.J. Relevance Feedback in Information Retrieval *Salton, G. (ed.), The SMART Retrieval System: Experiments in Automatic Document Processing* Prentice-Hall, 1971.
8. Salton, G. *Automatic Text Processing: The Transformation, Analysis, and Retrieval of Information by Computer* Addison-Wesley, 1988.
9. Zhai, C., Jansen, P., Stoica, E., Grot, N., Evans, D.A. Threshold Calibration in CLARIT Adaptive Filtering *Proceedings of the Seventh Text REtrieval Conference (TREC-7)*, Gaithersburg, MD, 1998.
10. Zhai, C., Jansen, P., Roma, N., Stoica, E., Evans, D.A. Optimization in CLARIT TREC-8 Adaptive Filtering *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*, Gaithersburg, MD, 1999.