Learning While Filtering Documents

Jamie Callan
Center for Intelligent Information Retrieval
Computer Science Department
University of Massachusetts
Amherst, MA 01003-4610, USA
callan@cs.umass.edu
www.cs.umass.edu/~callan/

Abstract This paper examines the problems of learning queries and dissemination thresholds from relevance feedback in a dynamic information filtering environment. It revisits the EG algorithm for learning queries, identifying several problems in using it reliably for information filtering, and providing solutions. It also presents a new algorithm for learning dissemination thresholds automatically, from the same relevance feedback information used to learn queries.

1 Introduction

An information filtering system monitors information sources, for example newswires, to find documents that satisfy a person's information need. The central problems in information filtering are creating an accurate representation of a person's information need, and then determining whether a given document satisfies the information need.

Past research has shown that it is difficult for people to state their information needs precisely, but easy for them to identify documents that satisfy their information needs. These results suggest that a person's original statement of the information need should be refined automatically, by a supervised machine learning algorithm, based upon user feedback. A large body of research addresses this problem, but does so based on assumptions that are becoming less relevant to current practice.

The predominant approach is to learn a "Routing" query. *Routing* queries are created by analyzing a large set of example documents that have been labeled relevant and non-relevant. The most accurate algorithms assume that documents can be examined repeatedly, usually in any order, or that potential changes can be tested on an archival collection [4, 15, 12].

A related problem is deciding how well a document must satisfy an information need in order to be disseminated. This problem of setting *dissemination thresholds* is usually addressed with algorithms that assume access to a large set of labeled example documents [7, 8].

These *batch-oriented* algorithms are becoming less relevant to current practice. Information filtering is increasingly deployed in interactive computing environments, where its users expect the system to respond immediately to any feedback they provide. There may not be sufficient time to analyze a batch of several hundred documents, or to carefully test the effects of several different adjustments to each query weight. An immediate response is required, preferably before very many additional documents are filtered.

This paper addresses the problems of learning filtering queries and their corresponding dissemination thresholds, automatically and incrementally. The work reported here was done with InRoute, a document filtering system based on a Bayesian inference network model of information retrieval and filtering [5]. However, the results reported here apply also to most statistical models of IR, including other probabilistic and vector space models.

2 Document Filtering

There are many types of document filtering environments, each with different assumptions and requirements [19, 14, 6, 18, 5]. We begin by defining what is meant by *document filtering* in this paper, and discussing how previously published research relates to it.

We are interested in a document filtering system that processes each document as soon as it arrives, deciding immediately whether it satisfies one or more information needs. Decisions can be influenced by information about documents already processed ("learning"); no information can be used from documents that have not yet been processed. The system should learn all it can about its environment, its users, and their needs, and it should react quickly whenever it has the opportunity to become more effective.

These requirements seem reasonable when viewed from the perspective of the filtering "consumer", who we define as a person who wants to see important information immediately, and with a minimum of work. However, these requirements present difficulties for many of the information filtering solutions proposed in the past.

For example, it would not be acceptable to store documents for some period of time, then index them and search the index with an information retrieval system [17, 6]. This "slow filtering" approach is only acceptable when information can be disseminated slowly.

When information must be disseminated quickly, one must use a filtering system that processes documents individually, as they arrive. Filtering systems such as SIFT [18] and InRoute [5] satisfy this requirement.

Some efficiency optimizations designed for large filtering services cause problems. For example, it might not be acceptable to "compile" a set of profiles to improve filtering speed, if doing so meant that people could not easily change their profiles [3]. However, it is a common user-requirement that software must run faster than it does, so optimizations that maintain flexibility and responsiveness are important.

The current state-of-the-art for creating and modifying queries from large amounts of training data is the Rocchio algorithm, augmented with Dynamic Feedback Optimization (DFO) [4]. Rocchio can be implemented as an incremental algorithm [1]. However, Dynamic Feedback Optimization is a batch-oriented algorithm that iteratively tests a large number of weight adjustments against an archival database. It is not clear how DFO can be applied in a more interactive environment.

Prior research suggested that the EG algorithm [10] is as effective at creating queries for filtering and routing tasks as Rocchio augmented with DFO [12]. This work was encouraging because EG is an incremental algorithm with well-defined theoretical properties. However, the algorithm was tested in a batch-oriented manner, in which training documents were examined repeatedly, and in random order. Later work on the TREC-5 Routing task produced less accurate results, raising doubts about the stability of the algorithm [2].

There is little published research on learning dissemination thresholds, perhaps because filtering systems have only recently been based on statistical models. Most methods tested in the TREC-5 and TREC-6 Filtering tracks depended on having access to a large set of training documents, and a cost function defining the user's requirements for precision vs recall [7, 8]. It is not clear how well these methods would work in a more interactive environment, particularly initially, when little training data is available. Nor is it clear how these approaches to threshold-learning interact with query learning or *idf* adjustments.

The state-of-the-art for meeting the requirements stated above can be summarized as follows. A few filtering systems are available that are based on statistical models and also capable of disseminating documents quickly. The best algorithms for learning Routing queries are not applicable. The incremental Rocchio algorithm, and perhaps EG, can be used. There is little guidance on how to learn dissemination thresholds automatically.

3 Learning Term Weights Incrementally With EG

Prior research suggested that the EG algorithm might be as effective as Rocchio augmented with Dynamic Feedback Optimization (DFO) [12], but the experiments were not truly incremental, and later work on a different corpus failed to support this result [2]. The best one can say is that EG appears to be a promising incremental alternative worth further examination.

3.1 The EG Algorithm

A detailed description and analysis of EG is beyond the scope of this paper. Details and in-depth analysis of the algorithm can be found in [10], while [12] provides a detailed comparison of the Rocchio, EG, and Widrow-Hoff algorithms for IR tasks. We confine our attention here to a study of possible problems in applying EG to information filtering tasks.

The EG algorithm, like Rocchio, attempts to find a linear classifier that minimizes the magnitude of classification errors [10]. One can consider the classifier to be a weight vector that is applied to a set of query terms. EG imposes the restrictions that all weights must be positive, and sum to 1, neither or which is a problem for filtering queries. EG is trained on one example at a time, adjusting the weights after seeing each example. The update rule for a weight vector w_i of d features is:

$$w_{i+1,j} = \frac{w_{i,j} \exp\left(-2\eta\left(\mathbf{w_i} \cdot \mathbf{x_i} - y_i\right) x_{i,j}\right)}{\sum_{j=1}^{d} w_{i,j} \exp\left(-2\eta\left(\mathbf{w_i} \cdot \mathbf{x_i} - y_i\right) x_{i,j}\right)}$$

where x_i is a vector of real-valued features representing the current document, and y_i represents the desired value of the classifier for the current document. Initial weights are typically set to $\frac{1}{d}$.

The learning rate η determines how rapidly EG learns from each example. Kivinen and Warmuth suggest setting it to $\eta = 2/(3R^2)$, where R is a value that satisfies the constraint $\max_i(\max_j x_{i,j} - \min_j x_{i,j}) \leq R$.

EG would appear to be a well-defined algorithm, with few opportunities for mistaken choices in how to apply it. However, such is not the case. Some choices that one must make in applying EG are listed below.

Feature Representation: EG is most easily used for Boolean problems, but can also be applied to non-Boolean features. Should one use Boolean features, or features based on *tf.idf* weights?

Target Values: What value should the classifier respond with for relevant and nonrelevant documents? Should 0 and 1 be used, or other values?

Learning Rate: Kivinen and Warmuth suggest one way of setting the learning rate. However, they also state that other choices may be more appropriate for noisy problems. What learning rate should one choose?

Answering these questions is our first priority.

3.2 Hypotheses

Several decades of IR research show that *tf.idf* features are consistently more effective than Boolean features. While it is important to verify that this remains true for the EG algorithm, our first hypothesis is that it does indeed remain true. This hypothesis is tested using InRoute's *tf.idf* features [5], which incorporate Robertson's *tf* weighting formula [16] and a scaled *idf*.

Our second hypothesis is that EG is sensitive to the choice of target values intended to signify relevant and nonrelevant documents. Some algorithms, for example the Perceptron algorithm [13], are insensitive to the values chosen; all that matters is the *direction* of the error gradient. However, EG follows an *exponentiated gradient*; small variations in how the gradient is determined might have large effects on algorithm behavior.

What, then, are the proper scores to assign relevant and nonrelevant documents? It is impossible for most *tf.idf* classifiers to produce document scores of either 0 or 1, so these may be poor choices. We note, for example, that Lewis, et al, were careful to choose target values of 0.42 and 0.49, which were roughly comparable to "low" and "high" scores in INQUERY [12]. When weights sum to 1.0, as they do for EG, the maximum score a linear classifier can achieve is equal to its highest feature; the

Corpus Name	Use	Queries	Collection	Size	Number of Documents
TREC-2 Routing	Train	50	CDs 1 & 2	2.1 GB	741,856
TREC-2 Routing	Test	50	CD 3	1.1 GB	336,310
TREC-6 Filtering	Train	47	FBIS volumes 3 & 4	493 MB	130,471
TREC-6 Filtering	Test	47	FBIS volume 6	452 MB	120,653

Table 1: Characteristics of training and testing corpora.

minimum score is equal to its lowest feature. Our third hypothesis is that the feature values with maximum and minimum values represent better target values than 1 and 0 for relevant and nonrelevant documents, respectively.

Finally, how should the learning rate be determined? Kivinen and Warmuth provide one option, but suggest that it may be too high for noisy problems. Document filtering is a noisy problem, so one might conclude that a lower rate make sense. However, the Kivinen and Warmuth suggestion is very conservative for real-valued features that cluster in a narrow range, as *tf.idf* features often do. For example, if the difference between the maximum and minimum features is 0.1, $\eta = 66.67$, yielding a learning rate of $\exp(-133.34\delta x_{i,j})$, where δ is the difference between the current document score and the desired document score. In other words, the learning rate converges to 0, and nearly no learning occurs on that example.

Our fourth hypothesis is that $R = 1.0$, the method suggested for Boolean features, is a reasonable way to set the learning rate for problems with *tf.idf* features.

3.3 Data

Two sets of queries and corpora were used for testing: the TREC-2 Routing queries and corpus, and the TREC-6 Filtering queries and corpus. These are referred to as the "TREC-2" and "TREC-6" corpora and queries throughout the paper. Corpora characteristics are shown in Table 1.

The initial queries for the TREC-2 task were the topic Description fields, with only stop words and leading stop phrases removed ("description-only" queries).

The initial queries for the TREC-6 task were the topic Description and Title fields, with only stop words and leading stop phrases removed, and expanded with 70 related words and phrases.

3.4 Experimental Methodology

The TREC-2 Routing task was converted into a filtering task by presenting judged documents to EG in a random order, without replacement; each judged document was seen just once.

Although we are interested eventually in the feature selection capabilities of EG, we confined our attention to learning query-term weights in these experiments. EG was able to eliminate query terms, by driving their weights to 0, but it could not add features selected from relevant documents.

Dissemination thresholds were not used. Every judged document was available for training.

Experiments were evaluated using average precision, calculated by the trec_eval program available from Cornell, as is common for TREC Routing-style tasks without dissemination thresholds. (We argue in Section 4.2 that other metrics are appropriate for filtering tasks involving dissemination thresholds.)

3.5 Experimental Results

The EG hypotheses discussed above would be difficult to test independently. For example, an experiment intended to compare Boolean and *tf.idf* features must train on target values using some learning rate. It only makes sense to hold independent variables constant when "reasonable" values are known, or when there is reason to believe that the specific choice of values doesn't matter. Such is not the case with this learning algorithm. A poor choice about learning rate, for example, could cripple learning irrespective of other choices.

The EG hypotheses can be mapped to three variables, representing feature type, method of setting target values, and method of setting learning rate.

Feature Type:

- Boolean.
- *tf.idf*.

Target Values:

- Boolean targets {0, 1}.
- {0.42, 0.49}, as used in [12].
- {MinFeature, MaxFeature}, from hypothesis 3.

Learning Rate:

- $R = \max_i(\max_j x_{i,j} - \min_j x_{i,j})$, as suggested in [10] (called KW, in this paper).
- $R = 1$.

There are 12 possible combinations of feature values, but only 7 are unique and make sense. All 7 combinations were tested. The experimental results are summarized in Table 2. The baseline values were obtained with unweighted queries.

The best combinations of parameter settings all used *tf.idf* features. However, the worst combination also used *tf.idf* features. One can conclude that *tf.idf* features do lead to more effective classifiers, as proposed by the first hypothesis, when other parameters are set appropriately.

Boolean target values were a poor choice for the *tf.idf* representation, as expected. The target values of 0.42 and 0.49 performed well, confirming previously published results [12]. Target values set dynamically, based on the minimum and maximum feature values, performed poorly with the KW learning rate, but performed well when the learning rate was calculated with $R = 1$. This result is consistent with our expectations; when the target values are farther from "typical" document scores, a larger learning rate is required.

In five out of six pairs of tests it was better to set the learning rate with $R = 1$ than to set it with the Kivinen & Warmuth method. We view this as confirming the fourth hypothesis, although we are puzzled by the one exception.

Test Description			TREC-2	TREC-6
Feature	Targets	R	AvgPrec	AvgPrec
N/A	N/A	N/A	0.1900	0.2631
Boolean	0 / 1	KW, 1	0.1901	0.2294
tf.idf	0 / 1	KW	0.1355	0.1751
tf.idf	0 / 1	1	0.1683	0.2625
tf.idf	Min/Max	KW	0.2093	0.2511
tf.idf	Min/Max	1	0.2401	0.2728
tf.idf	0.42/0.49	KW	0.2301	0.2834
tf.idf	0.42/0.49	1	0.2438	0.2483

Table 2: Results from an experiment comparing different methods of selecting feature type, target values, and learning rate. Each Average Precision (AvgPrec) figure represents an average of the trec_eval Average Precision (AvgPrec) from five tests. TREC-2 Routing queries (51-100) trained on TREC volumes 1 & 2 data, tested on TREC volume 3 data. TREC-6 Filtering queries, trained on TREC FBIS 3 & 4 data, tested on TREC FBIS 6 data.

The most consistent combinations of parameter settings were the one corresponding to our hypotheses (*tf.idf*, Min/Max, $R = 1$), and the one used in [12] (*tf.idf*, 0.42/0.49, KW). This experiment confirms each of the hypotheses presented in Section 3.2, but also confirms that the KW method of setting learning rates is appropriate when reasonable target values are known *a priori*.

The tested values for feature type and target values cover the space of possible choices well, but the same is not true for the tested values of learning rate. R could be set to other constant values, or it could be set as some multiple of the value suggested by Kivinen and Warmuth. We have no hypotheses about what might be best, so tests were conducted to explore a range of possible values.

For brevity, we focus only on tests using the *tf.idf* representation and the MinFeature / MaxFeature target values. Two types of tests were conducted. In one type, R was set to a constant value. In the other type, the value $\eta = 2/(3R^2)$ was multiplied by a constant. η and the learning rate are inversely correlated; reducing η increases the learning rate, and vice versa.

The experimental results are summarized in Table 3. In general, EG performs well and is stable over a wide range of learning rates. $R = 0.6$ and $R = 0.8$ were slightly better than $R = 1.0$. However, as the learning rate converges to 0.0 (1.0η), effectiveness falls off. There appears to be no advantage to setting R to a constant versus multiplying η by a small constant. One must make the choice based on other criteria. For example, one might prefer setting R to a constant because it is simpler and faster.

4 Learning Dissemination Thresholds Incrementally

Deciding whether a document matches a query is simple under a Boolean model, but complex under statistical models. In a vector space, any document has a (possibly small) similarity to any query. In a probabilistic model, any document has a (possibly small) probability of satisfying any information need. The important question is whether the similarity or probability is high enough that the document is worth showing to someone.

The question of whether a document is "good enough" is expressed as a problem of setting a dissemination threshold. Documents with a score (similarity, probability) that exceeds the threshold are disseminated (shown to someone); documents with scores below the threshold are discarded.

Test Description			TREC-2	TREC-6
Feature	Targets	R	AvgPrec	AvgPrec
N/A	N/A	N/A	0.1900	0.2631
tf.idf	Min/Max	0.4	0.2413	0.2657
tf.idf	Min/Max	0.6	0.2426	0.2775
tf.idf	Min/Max	0.8	0.2421	0.2785
tf.idf	Min/Max	1.0	0.2399	0.2728
tf.idf	Min/Max	1.2	0.2403	0.2610
tf.idf	Min/Max	1.4	0.2398	0.2483
tf.idf	Min/Max	0.02 η	0.2429	0.2315
tf.idf	Min/Max	0.04 η	0.2418	0.2532
tf.idf	Min/Max	0.06 η	0.2407	0.2671
tf.idf	Min/Max	0.08 η	0.2411	0.2752
tf.idf	Min/Max	0.1 η	0.2411	0.2779
tf.idf	Min/Max	0.2 η	0.2393	0.2776
tf.idf	Min/Max	0.4 η	0.2338	0.2711
tf.idf	Min/Max	0.6 η	0.2336	0.2617
tf.idf	Min/Max	0.8 η	0.2202	0.2572
tf.idf	Min/Max	1.0 η	0.2124	0.2535

Table 3: Results from an experiment comparing different methods of setting the EG algorithm's learning rate. The first figures (with N/A entries) are baseline values obtained with unweighted queries. Each subsequent Average Precision (AvgPrec) figure represents the average of five tests.

The problem is complicated by the users, of course. Some people don't want to be bothered with material unless it is highly likely to be relevant ("high precision user"). Others want to see anything that might be relevant ("high recall user"). Most fall somewhere in between.

A system can require people to specify dissemination thresholds manually, but few people can select a good threshold for a new query without significant effort. Even after observing several relevant and nonrelevant documents (obtained how?), it would be difficult for the average person to understand that a score of 0.429129 is predictive of relevance, but a score of 0.417535 is not.

A better choice is to let an algorithm set dissemination thresholds, for example based upon the same type of relevance information used for improving queries. Several batch-oriented solutions require a cost function describing a person's preferences, and then find a threshold that optimizes for it by testing different thresholds on a training set of relevant and nonrelevant documents [7, 8].

Our interest is a solution for interactive environments. "Corpus" statistics such as *idf* may be adjusted as each document passes through the system, making it less clear what a "good" document score might be. Training is interleaved with filtering, and the system must actually disseminate a document before obtaining a relevance judgement. In this already complex environment the problem of setting dissemination thresholds must be addressed at the same time that the relevance feedback algorithm is adding/deleting query terms or adjusting query term weights.

4.1 An Algorithm

The requirements for an algorithm are stated simply. The filtering system only has access to relevance judgements for documents that it has disseminated. In its initial state, no documents have been disseminated, so the threshold must start low enough to actually disseminate documents. The threshold rises over time, to exclude nonrelevant documents that only marginally match the query. How high it rises should be a function of common relevant and nonrelevant document scores, and of the user's preferences concerning recall and precision.

One simple solution, adopted here, is to track the average scores of disseminated relevant and nonrelevant documents, and then to place the threshold somewhere between the two averages. The user's preferences with respect to precision and recall can influence where between these averages to place the threshold.

The algorithm is most at risk of setting a poor threshold when the number of disseminated documents is low. For example, if the first disseminated document happened to have a very high score, the threshold could be set so high that no other documents would be disseminated. The risk can be reduced by learning more slowly from the first ten relevant and ten nonrelevant documents.

The algorithm is complicated, slightly, by its interaction with other learning that may occur simultaneously, for example query term additions and deletions, query term weight adjustments, and *idf* adjustments. All of these changes alter document scores, making the averages an outdated view of the previously disseminated relevant and nonrelevant documents. Therefore it is important to compensate for these changes, by enabling the averages to reflect the new information.

The document scores computed for vector-space queries and most probabilistic queries are a product of a weight vector and a vector of *tf.idf* scores. (The weight vector is included because of the presumed use of a relevance feedback algorithm such as Rocchio or EG.) The average of a set of relevant document scores can be decomposed as follows:

$$\begin{aligned}
&\text{Avg_Relevant} \\
&= \frac{1}{n}\left(\text{Doc_Score}_1 + \ldots + \text{Doc_Score}_n\right) \\
&= \frac{1}{n}\left((w_1 \cdot T_{1,1} \cdot I_1 + \ldots w_n \cdot T_{1,n} \cdot I_n) + \right. \\
&\qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
&\qquad \left.(w_1 \cdot T_{n,1} \cdot I_1 + \ldots w_n \cdot T_{n,n} \cdot I_n)\right) \\
&= w_1 \cdot I_1 \cdot \frac{1}{n} \cdot (T_{1,1} + \ldots T_{n,1}) + \\
&\qquad \vdots \qquad \vdots \qquad \vdots \qquad \vdots \\
&\quad w_n \cdot I_n \cdot \frac{1}{n} \cdot (T_{1,n} + \ldots T_{n,n})) \\
&= w_1 \cdot I_1 \cdot \text{Avg_T}_1 + \ldots + w_n \cdot I_n \cdot \text{Avg_T}_n
\end{aligned}$$

where $T_{i,j}$ is the term weight for the j'th term in document i, I_j is the *idf* weight for the j'th term, and w_j is the query term weight for the j'th term.

Instead of storing the average relevant document score as a single number, it can be stored as a vector of average relevant tf scores, one per query term. Whenever the average relevant document score is needed, it can be recomputed quickly using this vector, the current query term weights, and the current *idf* values. This approach requires slightly more effort, but keeps the average relevant and nonrelevant scores accurate during *idf* fluctuations, query term weight adjustments, and query term deletion. As long as dissemination thresholds are not recomputed often, the overhead is low.

The algorithm proposed above has the following characteristics:

- tolerant of changes to "corpus" statistics;
- tolerant of changes to query terms, and query weights;
- somewhat responsive to user preferences with respect to precision vs recall; and
- not misled by the occasional low-scoring relevant document or high-scoring nonrelevant document.

These characteristics make it well-suited to most interactive filtering environments.

4.2 Metrics

It is generally accepted that metrics for ranked retrieval are not adequate for measuring filtering systems that use dissemination thresholds [11]. The TREC-5 and TREC-6 Filtering tracks adopted set-based measures of the form:

$$\alpha \cdot R_f - \beta \cdot N_f - \gamma \cdot R_m$$

where R_f is the number of relevant documents found, N_f is the number of nonrelevant documents found, and R_m is the number of relevant documents missed. The TREC-6 F1 measure, for example, set $\alpha = 3$, $\beta = 2$, and $\gamma = 0$, to simulate a person interested in high precision. The TREC-6 F2 measure set $\alpha = 3$, $\beta = 1$, and $\gamma = 1$, to simulate a person interested in high recall.

The F1 and F2 measures reflect the cost/benefit ratio of reading the entire set of documents. However, because the components are not scaled, the numbers can be difficult to interpret. A score of -20 could reflect a system doing well on a large set, or a system doing poorly on a small set.

Precision and recall over a set of disseminated documents are easier for most people to understand. When a single metric is required for comparing results, a weighted average of precision and recall can be used. For example:

$$(\alpha \cdot P + \beta \cdot R)/(\alpha + \beta)$$

where α and β reflect the relative importance of precision and recall to a particular person.

In this paper, filtering experiments are measured by several metrics: TREC-6 F1, TREC-6 F2, precision, recall, and two weighted averages of precision and recall. We hope that one or more of them will be useful to the reader. Table 4 illustrates how each metric behaves on four artificial examples.

4.3 Experimental Methodology

The TREC-6 Filtering methodology [8, 9] was adopted for these experiments. The filtering system starts with an "Ad-hoc" query, and receives relevance feedback whenever it disseminates a document during the training phase. Queries and thresholds are then frozen and tested on the test set. This methodology is not ideal for simulating an interactive environment, but it is to some extent an accepted methodology. Using it makes the work described here comparable to work reported at TREC.

The default relevance feedback algorithm in InRoute, an incremental Rocchio algorithm [1], was used to learn queries. The default settings were used, which allowed up to 10 query terms to be added to the query.

	Small Set, Medium Recall	Small Set, High Recall	Large Set, Medium Recall	Large Set, High Recall
Retrieved:	20	30	200	300
Relevant:	8	8	80	80
Rel_ret:	4	6	40	60
Set Precision:	0.2000	0.2000	0.2000	0.2000
Set Recall:	0.5000	0.7500	0.5000	0.7500
TREC-6 F1:	-20	-30	-200	-300
TREC-6 F2:	-8	-8	-80	-80
3P,1R:	0.2750	0.3375	0.2750	0.3375
1P,2R:	0.4010	0.5685	0.4010	0.5685

Table 4: The behavior of four metrics on different types of experimental results.

Tests were conducted with the thresholds set at 0%, 25%, 50%, and 75% of the difference between the average nonrelevant and average relevant document scores. (A threshold at 0% is set at the average nonrelevant document score.) The goal was to investigate the results the system would produce when the threshold was placed at different points between the "High Precision" (75%) and "High Recall" (0%) ends of the spectrum.

A "batch-oriented" Routing query was also created, as a state-of-the-art reference point against which to compare the incremental queries. The Routing query contained 50 additional terms, and 50 proximity operators each for distances 1, 5, and 20 [2]. For this comparison *only*, filtering output, using thresholds, was converted to a ranked list by sorting on document score. The usual Routing output (no thresholds, top 1,000 documents per query) was used. Results were evaluated using the Average Precision metric.

Two sets of queries and corpora were used for testing purposes: the TREC-2 queries and corpus, and the TREC-6 queries and corpus. The corpora characteristics are described in Section 3.3.

4.4 Experimental Results

The experimental results are shown in Figures 1 and 2. The precision of the learned thresholds for the TREC-6 queries is 28% to 35%, while recall is 10% to 43%. The thresholds for the shorter TREC-2 "description only" queries yield precision of 17% to 24%, and recall of 24% to 42%.

These results indicate that reasonable dissemination thresholds were learned, in spite of the constant adjustment of *idf* values, the tuning of query term weights, and the addition/deletion of query terms. The results for the TREC-6 queries are preferable to the results for the TREC-2 queries, which may be due to the improved initial ad-hoc queries (recall that the TREC-6 queries were expanded with query expansion, whereas the TREC-2 queries were not).

Weak correlations between lower thresholds and improved recall were apparent in both tests; in each test, three out of four lower thresholds produced higher recall. Lower thresholds were also weakly correlated with lower precision on the TREC-2 data, but not on the TREC-6 data. The results can be viewed as weak support for the hypothesis that precision and recall vary depending upon how the threshold is placed, but they do not confirm the hypothesis.

It is unclear why precision is not affected more strongly by lower thresholds. Lower thresholds result in more documents being used for relevance feedback, which may result in more accurate queries.

Thresholds for several queries were so high that no documents were disseminated for those queries. For example, the 75% and 50% thresholds on the TREC-6 test disseminated no documents for one query (the "Num Queries" row in Figure 1). Disseminating no documents is the appropriate response when the system is unable to learn an accurate query. Recall on the overall test set is lower when no documents are disseminated for a query, but precision is unaffected.

The queries that were learned incrementally compared favorably with the batch Routing queries in both tests. Most of the incremental tests on the TREC-6 data produced average precision within 10% of the batch Routing queries. Ranked results at recall up to 100 documents (not shown) always favored the incremental queries by a substantial margin.

The TREC-2 batch queries were much more effective than their incremental counterparts, presumably because of the shorter, "description-only" queries used in the incremental tests. However, much of this difference occured at high recall in ranked tests. At low recall, the incremental queries were only 10-20% worse than batch queries (not shown).

One can conclude from this comparison that the filtering queries learned incrementally were not "crippled" with respect to the state-of-the-art Routing queries. It would be a mistake to conclude much more than that, given the many differences in how the queries were formed, and in how they were intended to be used.

5 Conclusions

Information filtering is becoming common in interactive computing environments, where people expect the system to respond quickly to any feedback they provide. One challenge for the research community over the next few years will be to shift from batch-oriented query- and threshold-learning algorithms to equally effective incremental algorithms suitable for dynamic environments.

This paper represents a first step down that path. It revisits the EG algorithm for learning queries, identifying several problems in using it reliably for information filtering, and providing solutions. It also presents a new algorithm for learning dissemination thresholds automatically, from the same relevance feedback information used to learn queries. Collectively, these results represent an "end-to-end" solution to problems encountered in realistic, dynamic, and responsive information filtering systems.

Although the work presented here is encouraging, much remains to be done. There is not yet a direct

Training:					
Threshold:	Batch	75%	50%	25%	0%
TrainedOn:	41,476	9,623	13,441	21,313	42,185
Train Prec:	0.1398	0.1047	0.1492	0.1405	0.0852
Train Recall:	1.0000	0.1739	0.3460	0.5164	0.6200
Testing: Total number of documents over all queries					
Num Queries:	47	46	46	47	47
Retrieved:	47,000	2,342	6,060	5,053	10,085
Relevant:	6872	6,872	6,872	6,872	6,872
Rel_ret:	4350	665	1,970	1,780	2,932
Testing: Filtering Metrics					
Set Precision:	0.0926	0.2839	0.3251	0.3523	0.2907
Set Recall:	0.6330	0.0968	0.2867	0.2590	0.4267
TREC-6 F1:	-72,250	-1,359	-2,270	-1,206	-5,510
TREC-6 F2:	-32,122	-5,889	-3,082	-3,025	-2,297
3P,1R	0.2277	0.2372	0.3155	0.3290	0.3247
1P,2R	0.4547	0.1585	0.2993	0.2898	0.3818
Testing: Average precision (non-interpolated)					
	0.2075	0.0846	0.1794	0.1812	0.2290

Figure 1: Effectiveness when thresholds are set at different points between the average relevant and average non-relevant document scores. (TREC-6 Filtering data.)

Training:					
Threshold:	Batch	75%	50%	25%	0%
TrainedOn:	89,179	14,179	24,192	48,948	277,381
Train Prec:	0.1837	0.1502	0.2028	0.1579	0.0312
Train Recall:	1.0000	0.1300	0.2994	0.4716	0.5284
Testing: Total number of documents over all queries					
Num Queries:	50	48	48	47	47
Retrieved:	50,000	10,843	19,081	25,818	19,020
Relevant:	10,981	10,981	10,981	10,981	10,981
Rel_ret:	6,722	2,545	3,481	4,313	3,839
Testing: Filtering Metrics					
Set Precision:	0.1344	0.2347	0.1824	0.1671	0.2018
Set Recall:	0.6121	0.2383	0.3259	0.4165	0.3573
TREC-6 F1:	-66,390	-8,961	-20,757	-30,071	-18,845
TREC-6 F2:	-27,371	-8,800	-12,358	-14,609	-10,570
3P,1R	0.2539	0.2340	0.2161	0.2235	0.2388
1P,2R	0.4545	0.2327	0.2726	0.3183	0.3008
Testing: Average precision (non-interpolated)					
	0.3494	0.1727	0.2216	0.2851	0.2350

Figure 2: Effectiveness when thresholds are set at different points between the average relevant and average non-relevant document scores. (TREC-2 Routing data.)

comparison of the effectiveness of incremental versions of Rocchio and EG. It is not clear how many query terms and proximity operators to add, nor when to add them, when relevance judgements occur only occasionally. The algorithm for learning dissemination thresholds was intended to be sensitive to user preferences with respect to precision and recall, but the experimental results suggest that it is only partially successful in this respect. Finally, we have still not shown how to satisfy a truly high precision user.

Acknowledgements

I thank Daniella Malin for her assistance in the work described here. This research was partially supported by the NSF Center for Intelligent Information Retrieval at the University of Massachusetts, Amherst, and by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623. Any opinions, findings, conclusions, or recommendations expressed in this material are the author's and do not necessarily reflect those of the sponsors.

References

[1] J. Allan. Incremental relevance feedback. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 270–278, Zurich, 1996. Association for Computing Machinery.

[2] J. Allan, J. P. Callan, W. B. Croft, L. Ballesteros, J. Broglio, J. Xu, and H. Shu. INQUERY at TREC-5. In D. K. Harman and E. M. Voorhees, editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 119–132. National Institute of Standards and Technology, Special Publication 500-238, 1997.

[3] T. A. H. Bell and A. Moffat. The design of a high performance information filtering system. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 12–20, Zurich, 1996. Association for Computing Machinery.

[4] C. Buckley and G. Salton. Optimization of relevance feedback weights. In *Proceedings of the Eighteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 351–357, Seattle, 1995. Association for Computing Machinery.

[5] J. P. Callan. Document filtering with inference networks. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 262–269, Zurich, 1996. Association for Computing Machinery.

[6] P. W. Foltz and S. T. Dumais. Personalized information delivery: An analysis of information filtering methods. *Communications of the ACM*, 35(12):51–60, 1992.

[7] D. Harman, editor. *Proceedings of the Fifth Text REtrieval Conference (TREC-5)*. National Institute of Standards and Technology Special Publication 500-238, Gaithersburg, MD, 1997.

[8] D. Harman, editor. *Proceedings of the Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology Special Publication, Gaithersburg, MD, (in press).

[9] D.A. Hull. The TREC-6 Filtering track: Description and analysis. In D. K. Harman and E. M. Voorhees, editors, *The Sixth Text REtrieval Conference (TREC-6)*. National Institute of Standards and Technology, Special Publication, (in press).

[10] J. Kivinen and M. K. Warmuth. Exponentiated gradient versus gradient descent for linear predictors. Technical Report UCSC-CRL-94-16, Baskin Center for Computer Engineering and Information Sciences, University of California, Santa Cruz, CA, 1994.

[11] D. D. Lewis. The TREC-5 filtering track. In D. K. Harman and E. M. Voorhees, editors, *The Fifth Text REtrieval Conference (TREC-5)*, pages 75–96, Gaithersburg, MD, 1997. National Institute of Standards and Technology, Special Publication 500-238.

[12] D. D. Lewis, R. E. Schapire, J. P. Callan, and R. Papka. Training algorithms for linear text classifiers. In *Proceedings of the Nineteenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 298–306, Zurich, 1996. Association for Computing Machinery.

[13] N. J. Nilsson. *Learning machines.* McGraw-Hill, 1965.

[14] K.H. Packer and D. Soergel. The importance of SDI for current awareness in fields with severe scatter of information. *Journal of the American Society for Information Science*, 30(3):125–135, 1979.

[15] S. E. Robertson, S. Walker, M. M. Hancock-Beaulieu, M. Gatford, and A. Payne. Okapi at TREC-4. In D. K. Harman, editor, *The Fourth Text REtrieval Conference (TREC-4)*, pages 73–96, Gaithersburg, MD, 1996. National Institute of Standards and Technology, Special Publication 500-236.

[16] S.E. Robertson and S. Walker. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In *Proceedings of the Seventeenth Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 232–241, Dublin, Ireland, 1994. Association for Computing Machinery.

[17] M. F. Wyle and H. P. Frei. Retrieving highly dynamic, widely distributed information. In *Proceedings of the ACM SIGIR International Conference on Research and Development in Information Retrieval*, pages 108–115, Boston, MA, 1989. Association for Computing Machinery.

[18] T. Yan and H. Garcia-Molina. SIFT – A tool for wide-area information dissemination. In *Proc. USENIX Winter 1995 Technical Conference*, New Orleans, January 1995.

[19] J. A. Yochum. A high-speed text scanning algorithm utilizing least frequent trigraphs. In *Proceedings of the IEEE International Symposium on New Directions in Computing*, pages 114–121, Trondheim, Norway, 1985. IEEE.