Incremental Relevance Feedback for Information Filtering

James Allan
allan@cs.umass.edu
Center for Intelligent Information Retrieval
Department of Computer Science, University of Massachusetts
Amherst, MA 01003-4610 USA

Abstract

We use data from the TREC routing experiments to explore how relevance feedback can be applied incrementally—using a few judged documents each time—to achieve results that are as good as if the feedback occurred in one pass. We show that relatively few judgments are needed to get high-quality results. We also demonstrate methods that reduce the amount of information archived from past judged documents without adversely affecting effectiveness. A novel simulation shows that such techniques are useful for handling long-standing queries with drifting notions of relevance.

1 Introduction

An information filter monitors a stream of documents and selects those that match a query. Information filtering differs from more traditional information retrieval in several respects: the documents arrive continuously rather than residing in a collection; the query is long-lived rather than one-shot (it might exist for days, weeks, or even years); and the matching process requires a yes/no decision rather than a ranked list.

Relevance feedback techniques provide a means for automatically correcting a query to more accurately reflect the user's interests: a set of "good/bad" relevance judgments on documents are "fed back" into the query to generate a better query. Researchers interested in feedback generally explore the problem using a test collection such as that provided by the TREC routing task,[Har96] which consists of some queries and a collection of documents that are known to be relevant or not relevant for each query. The documents are broken into training and test sub-collections, the queries are improved based upon the training sub-collection, and the results are checked using the test collection.

We are interested in a combination of those two ideas—*i.e.*, relevance feedback for information filtering—and the new questions that arise in that setting:

1. Do standard feedback techniques work when relevance judgments arrive a few at a time rather than in a batch? If so, how quickly do we achieve a "high quality" query?
2. How much information from past relevance judgments must be archived for a query in order to maintain that "high quality" but reduce space requirements? Is it adequate to archive a few select judgments, or can more canonical (so more concise) information be saved instead?
3. Can relevance feedback techniques cope with "query drift," a slow shift in the focus of the user's interest over time? (Query drift should not be confused with "concept drift," where the meaning of an indexing concept changes over time.[BR93] The two are related in that a query concept could drift and cause the query proper to drift, but we are interested in more general cases where the idea of "relevance" changes.)

This work investigates each of those questions. We focus on relevance feedback when the judgments are presented a few at a time rather than all at once, a process we call *incremental* feedback. The intent is that successive applications of feedback will incrementally converge on a query as good as that which could be achieved when all the judgments are applied simultaneously.

Although relevance feedback and the TREC routing task have been researched extensively, there has been little work that addresses the questions above in the context of incremental feedback in a filtering environment.

Section 2 starts by describing the information filtering environment in more detail and how we constructed experiments to answer the questions raised above. Section 3 briefly answers the question of whether relevance feedback works incrementally, and Section 4 addresses storage issues by exploring two approaches to incremental application of feedback that remember only selected information from past cycles. Section 5 demonstrates that the incremental approach can successfully cope with query drift and finally, Section 6 summarizes the results and presents some open questions.

2 What is incremental feedback?

In information filters, documents arrive continuously, though the number of documents arriving at any time may vary. A retrieval system monitors the stream of documents and when it finds one that matches a user's query, the document

Appears in Proceedings of SIGIR '96.
August 1996, Zurich, Switzerland

is saved. This part of the retrieval problem is handled by various existing filtering engines, both in the research community (*e.g.*, SIFT[YGM95] and InRoute[Cal96]) and the commercial setting (*e.g.*, Logicon[Yoc85]).

When documents have been selected, the user reads them and has the opportunity to mark them as relevant or not. These judgments are combined with the query to generate a new query—perhaps after several judgments have been made, perhaps after every one. The new query is then used to monitor the flow of incoming documents. This cycle continues until the query is no longer needed.

On the surface, the feedback in this process appears to be straightforward relevance feedback. However, unlike in other settings, older relevance judgments must be retained so that unusual information in newly judged documents does not accidentally bias the feedback in the wrong direction. This situation arises in interactive situations, but the longevity of a filtering query means that it may not be practical to archive all past relevance judgments because of limited storage space.

Work on iterative query formulation using feedback is fairly well known, but has only been applied to an interactive setting where the database remains constant.[Sal71, Boo88, IJA92] Some work has been done on deciding how many documents are needed to help feedback generate a good query and how many expansion terms should be used.[BSA94]. In the area of text classification, efforts have been made to reduce the amount of training needed to build a reasonable classifier.[LG94]

Aalbersberg's work on incremental feedback[IJA92] is very similar to one of the cases of this study (saving all context from past judgments), though he is concerned with a static collection and the interactive setting. The problems of archival space and query drift are usually of little consequence in such situations.

2.1 Simulating incremental feedback

A complete evaluation of incremental feedback in an information filtering context requires either extensive user studies or complex simulations of the process using a carefully constructed ordering of documents. To simplify the problem, we ignore the actual selection of documents: we are only interested in the stream of relevance judgments that results from the selection and presentation to some user. This approach limits the conclusions we can draw from this study, but creates a simpler platform for initial experiments.

For these experiments, our initial queries are the description sections of TREC queries 51 through 100. These queries are short (4 to 30 words), so are a reasonable model of a genuine user's initial query.[CCW95] The queries are improved by feedback on TREC disks 1 and 2; the resulting query is tested on TREC disk 3.

The relevance judgments are used in the order they are listed in the files as distributed. For full feedback runs, the order is actually unimportant. For incremental feedback runs, the judgments are fed back a fraction at a time until all judgments for a query are exhausted. That is, the first feedback cycle uses the first $\frac{1}{n}$ of the judgments for each query, the second cycle uses the second nth, and so on.

Table 1 presents some statistics about the set of relevance judgments for disks 1 and 2 (those used for feedback). The immense number of judgments—an average of almost 1800 per query!—would require substantial time to process, so we chose a random subset for all of our experiments by selecting about 10% of the full set of judgments, biased in favor of documents judged relevant (to ensure a reasonable likelihood of successful feedback). Judgments were selected from the relevant set with a 30% probability and from the non-relevant with a 5% probability.

	Judgments in set	Per Query Min	Max	Avg
full	89179	1047	2890	1784
full, only rel	16386	40	894	328
random subset	8497	99	332	170
subset, only rel	4916	7	276	98

Table 1: Number of relevance judgments in the entire training set (tip12) and in the randomly selected subset. A breakdown including only the positive judgments is also included.

2.2 Relevance feedback algorithm

In all experiments discussed below, feedback starts from the *original* query, possibly includes some information from past feedback cycles, includes some new set of relevance judgments, and generates a new query by adding up to 100 new terms. Note that the original query is used in every cycle. This anchors the results at the user's initial query, an approach that works well in general but will turn out to be inappropriate for drifting queries.

Any term that occurs in a relevant document is a candidate for appearing in the top 100. The candidate terms are first ordered by *rtf*, the number of times the term occurs in the relevant documents. The top 500 terms in that ranking are re-ranked according to a Rocchio formula:

$$Rocchio_i = w_{query} + 2w_{rel} - \frac{1}{2}w_{non-rel}$$

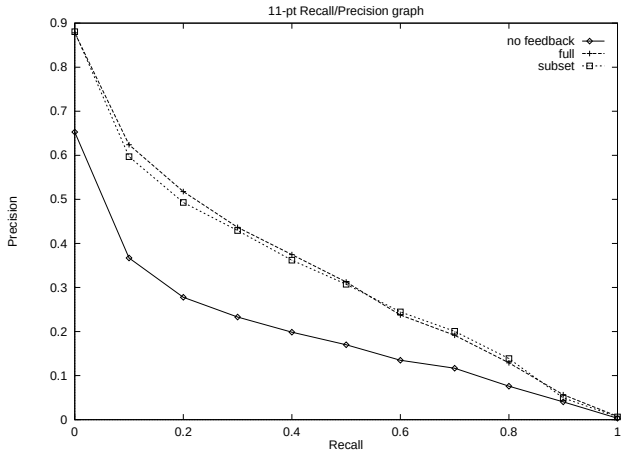
Where w_x is the weight of the term in the query, relevant documents, or non-relevant documents. The weight of term t in the relevant set is calculated as follows:

$$\begin{aligned} w_{relset}(t) &= \frac{1}{|relset|}\sum_{d\in relset} bel_{t,d} \\ bel_{t,d} &= 0.4 + 0.6 \cdot tfbel_{t,d} \cdot idf_t \\ tfbel_{t,d} &= \frac{tf_{t,d}}{tf_{t,d} + 0.5 + 1.5\frac{len_d}{avgdoclen}} \\ idf_t &= \log(\frac{N+0.5}{docf_t})/\log(N+1) \end{aligned} \quad (1)$$

where $tf_{t,d}$ is the number of times term t occurs in document d, len_d is the length of document d, $avgdoclen$ is the average length of documents in the collection,[1] N is the number of documents in the collection, and $docf_t$ is the number of documents that contain term t. This formula is the belief function currently used by Inquery.[Tur90, ABC+95]

The weights in the non-relevant documents and in the query are calculated similarly. The first 100 terms ranked by the Rocchio weight are added to the query; the original

[1] In an information filtering setting, the notion of "collection" is awkward. For these experiments we use the training database as the collection. In practice, collection data would have to be built up over time.[Cal96]



Run	AvgPrec	
Orig	0.1798	
Subset	0.3166	+76.1%
Full	0.3253	+80.9%

Figure 1: Base cases for incremental feedback, showing that feedback helps substantially over the original queries. It also shows that the 10% subset of judgments ("subset") works quite well compared to the "full" set.

query terms are *always* included in the new query. Because the user-supplied query terms are generally more reliable than the automatically-generated ones, the new terms are down-weighted slightly by multiplying them by 0.3.

Note that this feedback scheme is not the best known approach. In particular, all query structure is ignored (the queries are a weighted sum of words), no dynamic feedback optimization[BS95, ABC+95] is done, and only terms are added. However, this simpler approach is fast, reasonably effective, and easy to understand, so it provides an excellent approximation for these experiments.

When average precision is reported in this study, it is the average, non-interpolated precision for the top 1000 documents retrieved, assuming all other relevant documents were retrieved at rank infinity. (This measure is that calculated by the "TREC evaluation" programs.) Note that the query is being evaluated against the entire test set (TREC disk 3) at once rather than as an information filter. This choice may seem odd since we are evaluating information filters, but it nonetheless gives a good measure of the query's quality: a query that ranks a collection well can also be used to filter it with high accuracy, provided an appropriate threshold can be found.

2.3 Baseline measures

Figure 1 shows the effect of applying relevance feedback to the original queries, both using the full set of judgments and the random subset. The "subset" run is non-incremental feedback on the sampled set of 8,497 judgments. The "full" run is the result of including *all* 89,179 judgments and shows that the sampling provides a reasonable approximation of the total set. The "subset" run is used as the baseline in most experiments of this study, since it is the best we could expect in an incremental approach (using that

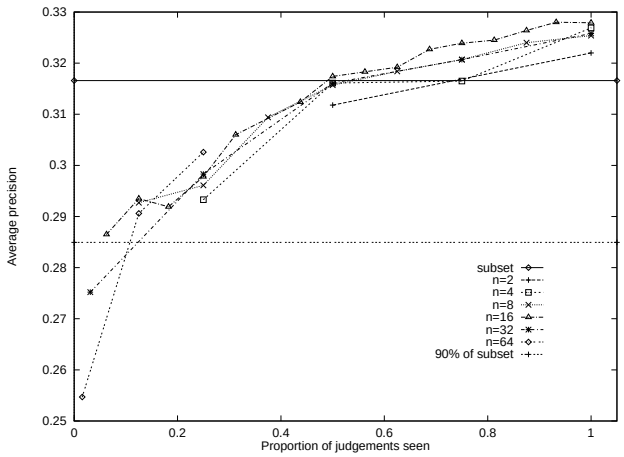


Figure 2: Rapid rise in effectiveness. Not all runs were made at every possible data point; however, the trend is clear. The horizontal lines show the baseline and 90% of the baseline. Note that the y-axis starts at 25% average precision.

set of judgments).

Note that the random subset achieves almost the same precision and recall as the full set, even though it includes only 10% as many judgments. This result indicates that we have lost very little important information by sampling and also suggests that when documents are fed back incrementally, "ideal" effectiveness will be achieved by the time only 10% of the documents have been considered—and perhaps even earlier.[2]

3 Does incremental feedback work?

The first question raised in the introduction was whether standard feedback techniques can be successful when judgments are applied incrementally. The only modest degradation caused by using a random sample already suggests that applying judgments incrementally will be successful. However, that 10% sample is scattered throughout the full set of judgments. What happens to effectiveness as the 10% sample is applied incrementally?

Figure 2 shows how average precision rises as a greater proportion of the subset's judgments are applied. A set of runs were made that applied $\frac{1}{n}$ of the judgments, then $\frac{2}{n}$, and so on. The graph shows the effectiveness reaching the baseline ("subset" feedback from Figure 1) when roughly half of the training documents have been presented, no matter how small the number of judgments fed back at a time.[3] With roughly 10% of the judgments applied, average precision is within 10% of the baseline. Note that if no feedback is done, the effectiveness is 43% lower than the baseline (not

[2] Similar results were also obtained when the sampled subset was varied for a small number of the tests in this study. However, no systematic study was undertaken to verify that there is no subtler bias in the sample used throughout. This omission means that the conclusions may not generalize when small numbers of judgments have been fed back.

[3] The apparently large improvement suggested by the graph around the halfway point is an artifact of the scales; the y-axis is not based at zero. It is only a few percent improvement.

shown in the figure; but see Figure 1), so relatively few documents results in substantial improvement. In fact, 10% of the relevance judgments is actually 1% of the full judgment set, meaning that very high effectiveness was obtained with a miniscule sampling of the full set! Admittedly, 1% is still 10–30 judgments per query (see Table 1) which is a lot for interactive settings. But for long-standing queries, it is comforting to know that reasonable performance can be achieved quickly. The answer to the first question is clearly that incremental application of feedback works, and that it achieves "high quality" very quickly.

The slight improvement over the baseline—which should be the best achievable effectiveness—at the right of the graph is the result of a few queries' being improved by the loss of statistical information from earlier cycles of $\frac{1}{n}$ judgments. Candidate terms are selected only from relevant documents, so statistics about relevant terms that do not occur until later cycles will be missing information about earlier non-relevant documents. Omitting that information helps a few queries which brings the average up slightly. If statistics are kept on terms that occur in all judged documents, the results are identical to full feedback.

4 Archiving judgments

The second introductory question raised was whether the same retrieval effectiveness could be achieved if less information were archived. The previous section showed that as the number of judgments grows, the quality of retrieval improves—but in those cases, all past judgments were also available. What about situations where it is not practical to archive all past judgments because the storage space is not available?

To examine that question in more depth, we first show that it is important to archive *some* information and then consider two types of archiving: by judged document, and by important concept. The results will show that surprisingly little data from earlier runs needs to be archived.

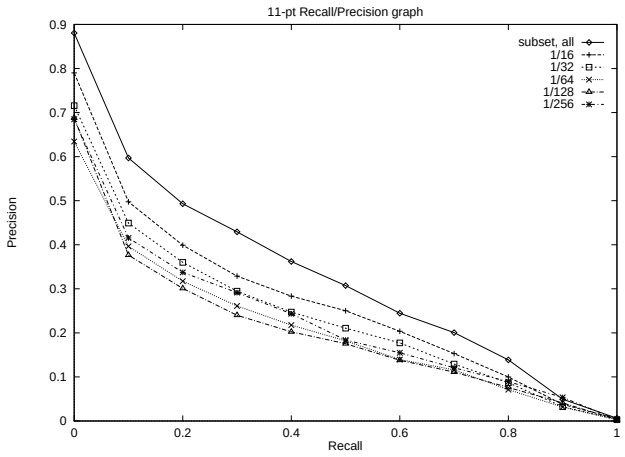
4.1 Saving nothing

A first approach to incremental feedback is to ignore all previous feedback cycles and start afresh each time new judgments arrive. (This is an extreme method since it would be likely that information from previous cycles would be implicit in the new query.) The effect this has on average precision varies substantially but is uniformly bad. If the judgments are provided $\frac{1}{16}$ at a time (roughly 10 per query per cycle), this approach results in a 9–22% drop in effectiveness (depending on which 16th of the judgments are used; on average the drop is about 15%) compared to feeding back all documents in one cycle. Figure 3 shows the result of feeding back one nth of the judgments for several values of n; the particular nth used is chosen randomly and is representative. Not surprisingly, smaller values of n result in better effectiveness: full feedback of the subset's judgements is equivalent to $n = 1$.

A 20–40% drop in effectiveness will be unacceptable in most settings, so it is clear that some amount of archiving is necessary.

4.2 Save top documents

The set of documents used for feedback includes 8,497 documents, occupying just over 110Mb of space (the 4,916 relevant documents account for only about 35Mb). Extrapolating to the full training set, there is over a gigabyte worth of training documents for 50 queries. We have just shown that we need to remember past relevance judgments to do well, but in a large-scale filtering environment, it may not be feasible to allocate 20Mb (and growing) per existing query. What happens, then, if we remember a smaller set of documents?



nth	AvgPrec	
Subset	0.3166	
16th	0.2546	-19.6%
32nd	0.2235	-29.4%
64th	0.1955	-38.3%
128th	0.1885	-40.4%
256th	0.2124	-32.8%

Figure 3: Saving no information, feeding back a sample $\frac{1}{n}$ of the training set, versus feeding it all back.

For this experiment, at the end of each cycle, all but n relevant and n non-relevant documents (a total of $2n$) are "forgotten." The n were chosen in two ways:

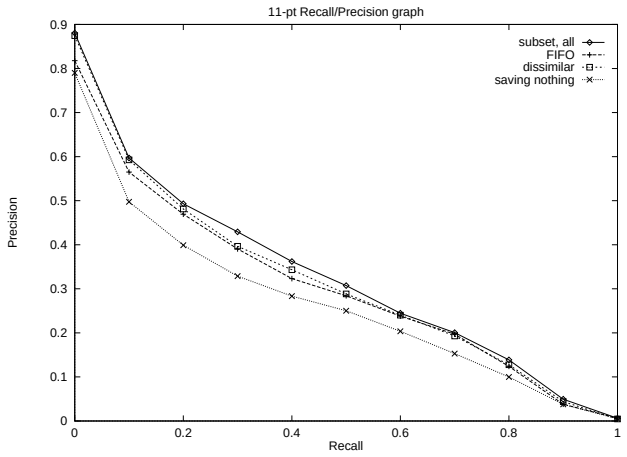
1. using a FIFO schedule, so the most recently seen n relevant (plus n non-relevant) documents—accumulated over multiple cycles, perhaps—were retained.

2. by keeping the n most "dissimilar" relevant documents (and another n "dissimilar" non-relevant documents). All documents were pairwise compared and as long as there were more than n documents remaining, the oldest document from the most-similar pair was discarded. Comparison was made by collapsing documents into vectors of term weights—by Equation (1)—and using an inner product as the similarity.

The judgments were presented $\frac{1}{16}$ at a time.

Figure 4 shows that both approaches are superior to saving no information (*i.e.*, the $\frac{1}{16}$ run from Section 4.1). FIFO accomplishes the improvement by increasing the number of documents that are being fed back. Indeed, as n increases, FIFO improves—when n is 50, over half of the relevant documents are being remembered and precision drops less than 2% off the baseline.

Keeping dissimilar documents outperforms FIFO with the same value of n. This result is not surprising since a set



nth	AvgPrec	
Subset	0.3166	
fifo	0.2965	-6.3%
dissim	0.3076	-2.8%
no save	0.2546	-19.6%

Figure 4: Keeping 20 documents (10 relevant, 10 non-relevant)

of dissimilar documents is more likely to cover the important topics that are relevant to the query. This recall-oriented explanation is supported by the "dissim" run of Figure 4, that retrieved 6,607 relevant documents over 50 queries; the FIFO run retrieved only 6,040 (the baseline got 6,728 and with nothing saved, only 5,797 were found). Interestingly, the "dissim" run actually out-performed the baseline run slightly in the first 15 documents retrieved.

A loss of 2-3% in effectiveness may be an acceptable price for archiving only 10% of the relevance judgments. However, if our goal is to save space, storing entire documents may be a mistake: documents are sometimes quite large and can vary dramatically in size.

4.3 Save top concepts

A different approach to saving context from past cycles is to remember statistics about the most "significant" concepts found so far. To calculate the weights as described in Section 2.2, the context needs to include the following: the term, whether it was an original query term, and also the document frequency, term frequency, and cumulative belief value of the term in both relevant and non-relevant documents.

If we archive that information, the statistics for the saved terms can be "pre-set" from the context at the start of a feedback cycle. Because only a subset of terms is stored in the context, the statistics for all other terms will be inaccurate—they will be based upon fewer relevance judgments. The question is how many terms must be kept in the context to achieve acceptable effectiveness.

It is certainly possible to keep statistics on all candidate terms seen to date, though doing so is probably nearly as space-consuming as—and less flexible than—keeping the full documents. That approach results in almost exactly the same effectiveness as feeding back all judgments at once.

Keep top n terms	AvgPrec	%Change
full	0.3166	–
50	0.3150	−0.5
100	0.3132	−1.1
250	0.3223	+1.8
500	0.3254	+2.8
750	0.3254	+2.8
900	0.3249	+2.6
1000	0.3254	+2.8
1250	0.3266	+3.2
1500	0.3251	+2.7
2000	0.3233	+2.1
3000	0.3238	+2.3
4000	0.3243	+2.5
5000	0.3206	+1.3
10000	0.3193	+0.9

Table 2: Top n terms saved in context

Ranking of top n	top 500		top 1000	
	AvgPrec	%Change	AvgPrec	%Change
full	0.3166		0.3166	
prop_df	0.3254	+2.8	0.3254	+2.8
rdf	0.3144	-0.7	0.3171	+0.2
rtf	0.3121	-1.4	0.3166	−0.0

Table 3: Varying selection of "top" terms

Feeding back $\frac{1}{8}$ of the judgments[4] at a time and varying the number of terms kept in context, we found that keeping a small number of terms can actually improve performance over full feedback (the slight inaccuracy of statistics noted at the end of Section 3 increases when only some terms are remembered from cycle to cycle). Table 2 shows that almost any number of terms works well; the differences are minor. Unless otherwise noted, we are letting n be 1000 terms for all runs that save context.

An important question is how to determine which of the candidate terms is in the "top n." We tried three different approaches:

1. Rank by the proportion of relevant documents containing the term minus the proportion of non-relevant documents containing the term (prop_df). A term that occurs in only relevant documents will be ranked at 1.0, a term that occurs in every training document will be ranked with a 0.0 weight, and terms occurring primarily in non-relevant documents will have negative weights. (This is the ranking method used above to show the effect of archiving varying numbers of terms.)

2. Rank by the total number of relevant documents containing the term (rdf).

[4] An eighth was used for these experiments rather than the sixteenth used for others. This is a historical accident and has no other significance. Where runs using both fractions were made, the effects are consistent.

org→new	Rel in Common	% of rel for org	new
53 → 55	9	1	1
55 → 87	10	1	3
57 → 97	31	3	5
61 → 99	142	52	24
62 → 67	17	2	2
63 → 66	84	29	29
64 → 71	9	1	1
72 → 73	25	12	5
74 → 85	13	1	1
85 → 94	10	1	2
88 → 89	7	4	4
89 → 90	6	3	2
94 → 95	78	12	12
97 → 98	265	39	19
98 →100	12	1	2

Table 4: Overlap in relevant documents for queries

3. Rank by the total number of occurrences of the term in the relevant documents (rtf).

Table 3 shows that the differences in effectiveness are not large, but that it is preferable to select terms based upon both positive and negative information.

It is clear that "high quality" effectiveness is achievable when an extremely small amount of information—as few as 250 words—is kept from past judgments. Archiving term information is less flexible than storing entire judged documents (*e.g.*, term adjacency information is lost), but the storage space is small and very predictable.

5 Query drift

Question (3) in the introduction raised the issue of query drift. If a request is long-lived enough, the intent of the query may change: a particular subtopic of the query might turn out to be more interesting in the long run, or a vaguely related topic might arouse some curiosity and shift "relevance" to a tangential heading.

Techniques such as those presented in the previous section seem ideal for handling query drift. Since incomplete information is being saved, "antiquated" (no longer appropriate) feedback data should drop away, leaving a query more suited to the new interest.

This section discusses an experimental setting that simulates such a situation and then presents the results of several experiments.

5.1 Simulating query drift

A query that has drifted is essentially two queries—the *original*, and the *new*—presumably with documents that are relevant to both for some transitional period. Some of the TREC queries used in earlier experiments are quite similar in both their statement and in their relevant document set. That suggests a method for approximating drift.

We consider two queries overlapping if they have several relevant documents in common (the number of judged non-relevant documents in common is not considered). The overlap among judged documents ranges from about 40% of the judged documents down to none. Table 4 shows 15 pairs of overlapping queries selected such that no query appeared twice in either the original or new set (we wanted a one-to-one mapping). For example, consider queries 97 and 98:

> **Q97:** fiber optics technology actually in use
>
> **Q98:** individuals or organizations which produce fiber optics equipment

Those queries have 265 relevant documents in common. That accounts for 39% of query 97's relevant documents and 19% of query 98's. Other pairs are not as highly overlapping, but we wanted enough queries that the results would be meaningful. We also selected an additional fifteen pairs where exactly one document was in common so that we could evaluate how well these approaches handled a more abrupt query drift. Those queries are discussed in Section 5.3.

The judgements for both relevant and non-relevant documents were combined by breaking them into three groups: (1) documents judged only for the original query, (2) documents judged for *both* queries, and (3) documents judged only for the new query. The judgements were applied in that order, though incrementally.

5.2 Basic drifting

The following types of runs were made and evaluated for the queries in Table 4. As was done in Section 3, feedback was on TREC disks 1 and 2; evaluation was using disk 3. The difference here is the mixing of relevance judgements that are applied.

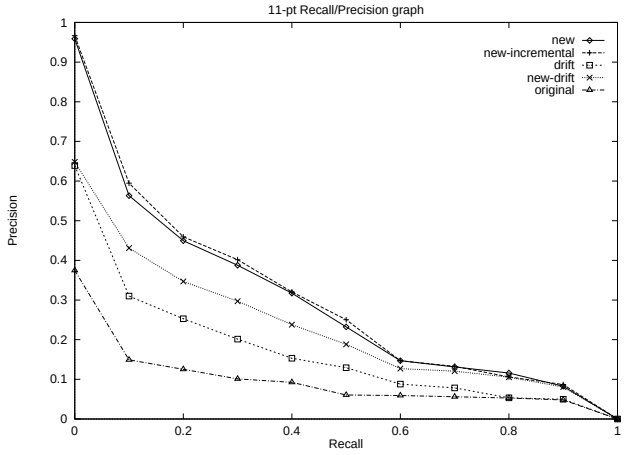
1. **new**: The new queries modified with only their own relevance judgements. This should be the best performance possible, but turns out not to be: the statistical effect noted at the end of Section 3 is more pronounced here because only the top 1000 terms are being saved when the the incremental judgements are used (in *new-incremental*).

2. **new-incremental**: Same as *new* but with the relevance judgements fed back incrementally, $\frac{1}{16}$ at a time, keeping 1000 words of context. This is the baseline for most runs, the best any drifted query could expect to do *incrementally*.

3. **drift**: The *original* queries modified with the mixed relevance judgements fed back $\frac{1}{16}$ at a time, keeping 1000 terms of context. The final cycle of feedback should result in queries that approximate the *new* queries.

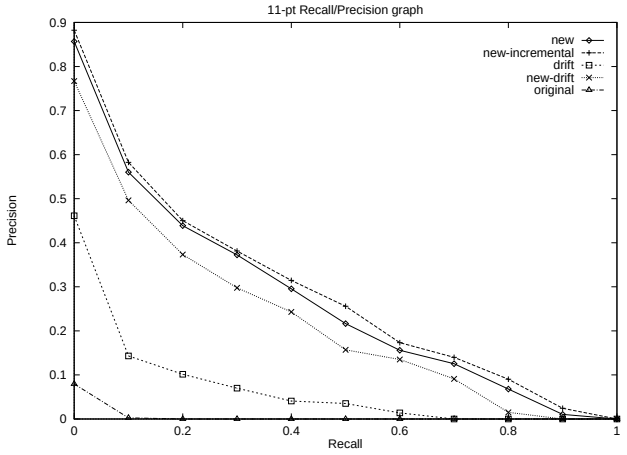
4. **new-drift**: The *new* queries modified with the mixed relevance judgements fed back $\frac{1}{16}$ at a time, keeping 1000 words of context. This run is very similar to the *drift* run, but is made to compensate for the "stickiness" of the *original* query that is part of the incremental feedback approach used—recall that every cycle of feedback includes the starting query in its result. We expect this query will not do as well as *new-incremental* since it will be modified by judgments for the *original* query.

5. **original**: The *original* queries modified with only their own relevance judgements (not incrementally), but then



Run	AvgPrec	
new	0.2858	-2.6%
new-inc	0.2934	
original	0.0870	-70.3%
drift	0.1560	-46.8%
new-drift	0.2190	-25.3%

Figure 5: Query drift with good overlap between the original and new queries.



Run	AvgPrec	
new	0.2621	-6.3%
new-inc	0.2799	
original	0.0006	-99.8%
drift	0.0486	-82.6%
new-drift	0.2149	-23.2%

Figure 6: Query drift, rip-current

Org→New	Org→New	Org→New
51→ 54	68→ 83	77→ 95
54→ 79	69→ 71	78→ 85
58→ 72	70→ 76	79→ 100
60→ 80	72→ 75	84→ 90
65→ 97	76→ 93	87→ 89

Table 5: Query pairs, abrupt shift in topic

evaluated as if they were the *new* queries. The purpose of this run is to demonstrate that the drifting has an impact on the queries, making them more like the new queries. That is, this run is expected to perform poorly.

Figure 5 shows that the original query is a bad surrogate for the new query and that the drifting helps substantially. The difference between the *drift* and *new-drift* runs makes it clear that any approach that hopes to handle query drift cannot anchor the feedback to the original query too firmly.

5.3 Basic drifting, abrupt shift

When the new query is substantially different from the old, holding fast to the original query is more likely to be a problem. To examine that hypothesis, we chose 15 query pairs that had exactly one overlapping relevant document. An example of the shift in query is:

> **from Q78**: activity by Greenpeace to carry out their environmental protection goals
>
> **to Q85**: allegations, or measures being taken against, corrupt public officials of any governmental jurisdiction worldwide

(The relevant document is WSJ910501-0020; it mentions how Greenpeace has published a list of the "alleged misdeeds" of Waste Management, Inc.) The 15 query pairs selected are listed in Table 5.

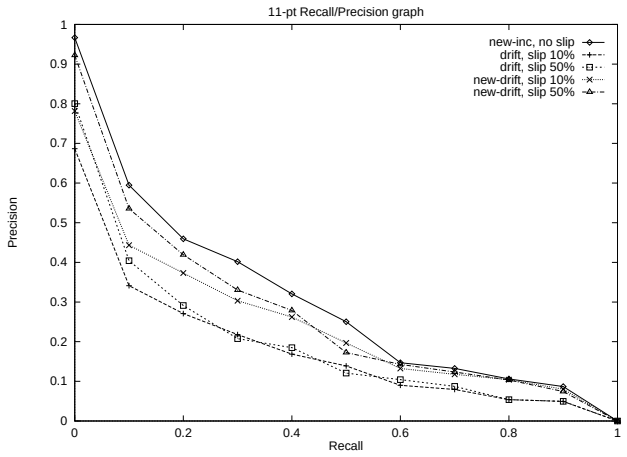
Figure 6 makes it very clear that in this setting the original query with its relevance judgements is a miserable substitute for the new query. Further, the increased necessity of making the original query fade from importance is evident by the differences between *drift* and *new-drift*.

5.4 Drift and slip

The previous two sections show that making the original query "sticky" hurts effectiveness substantially when a query drifts. The statistics maintained and archived about the top 1000 terms are also "sticky" in that the only way information about a term disappears is if the term eventually drops out of the list. We experimented with increasing the speed at which antiquated terms drop off the top 1000 by introducing a "slip" (aging) factor.

Whenever context is restored from an earlier cycle, all statistics are reduced by some fraction. If a term continues to occur in judged documents, its statistics will stay high. If, however, a term becomes useless (or counter-productive) we expect that it will lose weight more quickly than it would otherwise and will eventually drop from the top 1000 that are saved.

Figure 7 shows the result of running parallels of the *drift* and *new-drift* runs with slip factors of 10% and 50% on the slow-drift (not the abrupt shift) queries. The slip factor clearly helps in all cases. For these queries, a fairly aggressive reduction in past context is called for. In practice, the rate of fading will probably have to be determined by notic-



Run	Slip	AvgPrec	
new-inc		0.2934	
drift	none	0.1560	-46.8%
	10%	0.1675	-42.9%
	50%	0.1869	-36.3%
new-drift	none	0.2190	-25.3%
	10%	0.2318	-21.0%
	50%	0.2539	-13.5%

Figure 7: Query drift with slip factor

ing a shift in the type of documents judged relevant, perhaps based around some form of document clustering.

6 Conclusion

Relevance feedback is an excellent technique for improving the effectiveness of queries against a database. We have demonstrated that feedback can be applied incrementally to achieve similar effectiveness—provided that some context is maintained between feedback cycles.

In an environment where it is not feasible to archive large numbers of documents or large amounts of statistical information, incremental feedback works extremely well if a small number of past judgements is maintained. Its performance is improved further if statistical information about some of the important terms is kept instead.

If a user's notion of "relevance" is likely to drift over time, keeping around limited context prevents the query from locking in on the original notion of relevance. We implemented a novel simulation of this setting to demonstrate that claim, as well as to show that query drift is handled even more readily by phasing out old context.

There are still many questions that we are addressing in our on-going work:

- How much does the order of presented judgements affect the final results? Randomly ordering the judgements might be a good approach to addressing this question. Our intuition is that except for unusual cases, the order is unimportant.
- In particular, what happens if the relevance judgements are actually those made on top-ranked documents by the most recent query—that is, if we incorporate more of the retrieve, evaluate, feedback cycle? (Recall that for this work the judgements were presented in the TREC collection order which has no relation to the order that an actual filtering system would select them.) This requires substantially more processing since the training collection will need to be re-ranked after every cycle of feedback, but needs to be addressed.
- How well does this work extend to more complex query structures? We do not believe there will be any difficulty including phrases or other groupings of words, but the impact of more complex query structure (*e.g.*, nested fuzzy boolean operators) on feedback is not known. This problem has not been well-studied in even the more traditional IR setting.
- Since very few judgements are needed to achieve high effectiveness, and since remembering dissimilar documents was more useful than just remembering the most recent n, we believe that careful use of the stream of judgements may be extremely valuable. Can we select the judgements appropriately both to increase effectiveness and decrease computational expense beyond the levels mentioned in this work?

Acknowledgements

The question of how to do incremental feedback arose out of conversations with Bruce Croft and James Callan; I am grateful for their comments and suggestions. Thanks to Jody Daniels for her insightful comments several drafts of this work. Finally, my appreciation to the anonymous reviewers who pointed out some structural problems in the presentation of this material.

This material is based on work supported by the National Science Foundation, Library of Congress, and Department of Commerce under cooperative agreement number EEC-9209623. It is also based on work supported in part by NRaD Contract Number N66001-94-D-6054.

References

[ABC+95] James Allan, Lisa Ballesteros, James P. Callan, W. Bruce Croft, and Zhihong Lu. Recent experiments with INQUERY. In *Fourth Text REtrieval Conference (TREC-4)*, 1995. Forthcoming.

[Boo88] A. Bookstein. Set oriented retrieval. In *Proceedings of the 11th international conference on research and development in information retrieval*, pages 583–596, Grenoble, France, June 1988. Presses Universitaires de Grenoble.

[BR93] Carla E. Brodley and Edwina L. Rissland. Measuring concept change. In *AAAI Spring Symposium: Training issues in incremental learning*, pages 98–107, 1993.

[BS95] Chris Buckley and Gerard Salton. Optimization of relevance feedback weights. In Edward A. Fox, Peter Ingwersen, and Raya Fidel, editors, *Proceedings of the 18th annual international ACM SIGIR conference on research and development*

in information retrieval, pages 351–357, Seattle, Washington, July 1995. ACM.

[BSA94] Chris Buckley, Gerard Salton, and James Allan. The effect of adding relevance information in a relevance feedback environment. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the seventeenth annual international ACM-SIGIR conference on research and development in information retrieval*, pages 293–300, Dublin, Ireland, June 1994. Springer-Verlag.

[Cal96] James P. Callan. Document filtering with inference networks. In *Proceedings of the 19th annual international ACM SIGIR conference on research and development in information retrieval*, August 1996.

[CCW95] W.B. Croft, R. Cook, and D. Wilder. Providing government information on the internet: Experiences with THOMAS. In *Digital Libraries Conference*, Austin, Texas, June 1995.

[Har96] Donna Harman. Overview of the fourth Text REtrieval Conference (TREC-4). In *Fourth Text REtrieval Conference (TREC-4)*, 1996. Forthcoming.

[IJA92] IJsbrand Jan Aalbersberg. Incremental relevance feedback. In *Proceedings of the fifteenth annual international ACM SIGIR conference on research and development in information retrieval*, pages 11–22, 1992.

[LG94] David D. Lewis and William A. Gale. A sequential algorithm for training text classifiers. In W. Bruce Croft and C. J. van Rijsbergen, editors, *Proceedings of the seventeenth annual international ACM-SIGIR conference on research and development in information retrieval*, pages 3–12, Dublin, Ireland, June 1994. Springer-Verlag.

[Sal71] Gerard Salton, editor. *The SMART retrieval system: experiments in automatic document processing*. Prentice-Hall Series in Automatic Computation, Englewood Cliffs, New Jersey, 1971. Chapters 14-17.

[Tur90] Howard R. Turtle. *Inference networks for document retrieval*. PhD thesis, University of Massachusetts, Amherst, October 1990.

[YGM95] 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.

[Yoc85] 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.