Automatic Feedback Using Past Queries: Social Searching?

Larry Fitzpatrick
lef@opentext.com
Open Text Corporation
3 Bethesda Metro Ctr
Bethesda, MD, USA 20814

Mei Dent
mei@opentext.com
Open Text Corporation
185 Columbia Street West
Waterloo, Ontario, Canada N2L 5Z5

Abstract

The effect of using past queries to improve automatic query expansion was examined in the TREC environment. Automatic feedback of documents identified from similar past queries was compared with standard *top-document* feedback and with no feedback. A new query similarity metric was used based on comparing result lists and using probability of relevance. Our top-document feedback method showed small improvements over no feedback method consistent with past studies. On recall-precision and average precision measures, past query feedback yielded performance superior to that of top-document feedback. The past query feedback method also lends itself to tunable thresholds such that better performance can be obtained by automatically deciding when, and when not, to apply the expansion. Automatic past-query feedback actually improved top document precision in this experiment.

1 Introduction

Multi-gigabyte databases are *de regueur* today and the state of the art is pushing beyond terabyte sizes. Yet wide-spread experience with public and private search systems tells us that searchers are either unwilling or unable to invest in query construction. Over a wide range of operational environments, the average number of searcher-supplied query terms is frequently less than 2 and rarely more than 4. At the same time, the best performers in the TREC studies created queries with nearly 100 terms ([3], [1]). Clearly the developers of these systems believe that more terms lead to better effectiveness. It does seem intuitively obvious that as the data volumes increase, the number of search terms must also increase to provide the additional context necessary for effective retrieval – i.e., regardless of the engine ranking, how can 2 terms sift 100GBytes of data? [1]

One approach to this problem is to rely more heavily on automatic query expansion. Historically, automatic expansion has been viewed as operationally impractical because, while it has helped recall, it generally hurts precision at the top of the result list. We expect that as extraction techniques and our ability to estimate or infer relevance improve, automatic expansion methods should help offset the problems related to the paucity of user-supplied query terms and relevance judgments.

It is appealing to think that the results of past searches may be a source of additional evidence to help future searches. In operational environments, close examination of query logs clearly indicates the overlap in topic interests between searches. For example, the top 10 queries against the Open Text Web Index (http://www.opentext.com) often account for more than 5% of all queries. Additionally, external events create database *hot spots*. For example, during the heavy US East Coast snowstorm of January 1996, the Open Text Web Index experienced a sudden, profound, and sustained burst of activity due to the query *weather*, which prior to the storm had been rare. While these observations only consider queries that are identical, there is also great topical overlap between non-identical queries. A quick sample of daily query logs for the Open Text Index shows that between 5 and 9 of the top 10 most frequently occurring queries are all about one very specific and popular topic!

In this paper we describe experiments using past queries as a source of evidence for automatic term expansion. The hypothesis is that the top documents selected by a query from a pool of documents that are themselves the top documents from result lists of past *similar* queries are a good source for automatic term expansion. We compared the performance of this method against the unexpended baseline queries and against the baseline queries expanded with top-document feedback ([4]). We created a query similarity metric that used empirically derived probability of relevance information. We also evaluated the use of a threshold to control, on a per query basis, whether or not to apply automatic expansion. In the following sections we describe the methodology used, including the query similarity metric. Then we examine the results, review related efforts, summarize and discuss future directions.

2 Description of the Feedback Methods

The experiments were conducted with the TREC-5 data set [9]. The set of evaluation queries were the 50 ad hoc TREC-5 topics and their relevance judgments. For past queries feedback, we also used the 50 ad hoc TREC-3 [7] and 10 of the ad hoc TREC-4 queries [8] as the past queries. From the 50 ad hoc TREC-5 queries, baseline queries were created manually from the topics and were expressed in a simple natural

[1] Except, of course, for that class of high precision queries where a few words can identify exactly the need.

SIGIR 97 Philadelphia PA, USA

query language with stem expansion. No feedback, data or knowledge resources were used to generate the baseline queries. The average number of terms per baseline query was about 5. The small number of terms reflected the fact that we used no manual augmentation resources to discover additional terms – consistent with real users in the operational environments with which we are familiar. Manually created baseline queries presumably yielded a higher performance than automatically generated baseline queries would have, but this shouldn't effect the comparison of the two automatic feedback methods with each other.

The search system was the commercial off-the-shelf Open Text 6.0 Livelink OEM search engine. [2] A run of the baseline queries served as a benchmark against which the other methods were compared.

The complete TREC-5 data set was used for the experiment. It consists of 524929 documents from TREC disk-2 and disk-4, about 2 GBytes of text. Document length varies from collection to collection, with an average of 550 terms per document. We used the standard TREC performance analysis methods. Average precision, exact precision, recall-precision curve and document-precision curve are defined by Harman[9].

To provide a framework for the evaluation of different automatic query expansion methods, and in particular the effect of past queries, we posit a simple model for automatic query expansion. This model (see Figure 1) isolates the documents that are candidates for feedback in an *affinity pool*. Query expansion is done by evaluating the query against the affinity pool and extracting terms from the top documents retrieved there. The extracted terms are used to augment the query, then the query is executed against the main corpus.

Creation of the affinity pool seems overly complex for simple automatic feedback methods, such as top document feedback. However, it has allowed us to experiment with different methods for the creation of a query *context*. In these experiments, we also used the results of past queries to generate this pool, but in other configurations we've used external sources such as documents from web-browser history files. This is similar to creating context by watching information consuming habits of users, such as reported by Maes [10], Goker and McCluskey [6], and Morita and Shinoda [11].

The top-document and past queries feedback methods tested with this model are described below.

2.1 Top-document feedback

An automatic top-document feedback [3] run was created for the test queries. This method is virtually identical to the method described by Buckley, et al.[4] wherein the query is first executed against the corpus, the result set is ranked, some number of top documents are selected, key terms are extracted from the top documents, added to the query, and the query is re-executed. In terms of the feedback model described above, the affinity corpus is simply the top 200 documents from the relevance-ranked result list of the baseline query, so top documents extracted from it are the same as the baseline query's top documents.

The terms that expand the query are extracted from the top documents using the standard *abstract* operator in the search engine. [4] The number of documents selected for term expansion and the number of terms extracted from those documents can be selected at run-time. In this experiment, the top 2 documents were used to select the top 10 terms for expansion. The extracted terms were added to the baseline query and then the expanded query was run against the full corpus.

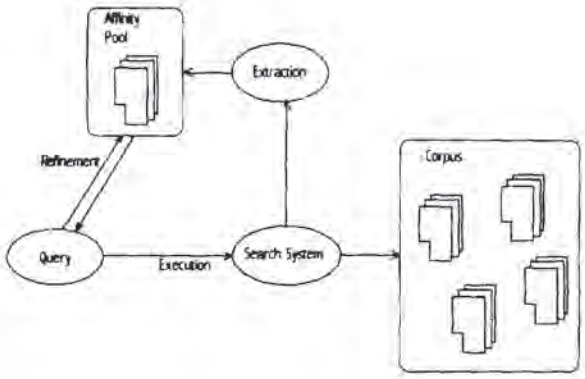

Figure 1: A simple model of automatic query expansion.

2.2 Past-query feedback

Past-queries feedback required a set of queries to use for generating the affinity pool for each query. The past queries consisted of the 50 ad hoc TREC-5 queries, 50 ad hoc TREC-3 queries, and 10 ad hoc TREC-4 queries. [5] These 110 queries were run against the TREC-5 data to generate result lists, relevance ranked, then truncated after 200 and saved. These were then used as input to form an affinity pool for each query.

To create an affinity pool for a query, we first computed the 3 queries most similar to it from the 109 other queries that exceeded a similarity threshold. If a query had 3 close matches then the affinity pool was created by pooling the top 100 documents from each of the matching past queries' result lists. A way to conceptualize this is: since the documents retrieved by similar past queries that end up in the affinity pool are from the same corpus as the final result documents, in effect, the past queries affinity pool is a mask that prevents baseline-query top documents that are not also past-query top documents from being used for feedback.

Feedback terms were the top ranked terms computed by the *abstract* operator when applied to the top documents from the result list of the baseline query executed against the

[2] The ranking algorithm was a straightforward probabilistic relevance ranking algorithm, one of the standard ranking methods available in the system.

[3] without the use of human-generated relevance judgments

[4] The *abstract* operator performs key term identification using a straightforward *tf* X *idf* term-scoring algorithm similar to that described in [4].

[5] The TREC-3 and TREC-4 queries were created manually using the same method as for the TREC-5 test queries. Only 10 of the TREC-4 topics were used because these are what we had available from another experiment.

$$S(L, L') = \frac{\sum_{i=1}^{n} wgt(Pos(L, D_i)) * wgt(Pos(L', D_i))}{\sum_{i=1}^{n} wgt^2(Pos(L, D_i))} \quad (1)$$

affinity pool. Once the baseline query was augmented with feedback terms, it was executed against the full corpus. The same constants for number of top documents and number of top terms were used as for the top-document feedback run.

The number of queries for which we could compute an affinity pool were, of course, dependent upon the threshold. For those queries that had no affinity pool, because there were less than 3 past queries similar enough to exceed the threshold, we used the baseline queries as part of the run. That is, no augmentation was done for some of the queries in the past-queries feedback run. We did this to make comparisons between the baseline and past queries feedback easier.

Two issues warrant further discussion. First the method of assessing inter-query similarity, and the threshold for determination of similar queries.

2.3 Measuring Inter-query Similarity

The query similarity method compared two queries by examining the overlap between their result lists. Essentially, it is a weighted overlap measure for the top 200 documents of each result list. This is somewhat similar to the method used by Raghavan, et al.[12], but without the use of relevance judgments and extended to take into account empirically derived weights based on the probability of relevance of each matching hit. The weighting gave more preference to matches that occurred near the beginning of the lists, because the probability of finding a relevant document in a ranked result list is greatest at the beginning and decreases towards the tail.

The similarity measure is described by Equation 1. L and L' are result lists to be compared. D_i is the i^{th} element in the result list. n is the number of elements in the result list. $Pos(L, D_i)$ is a function that maps an element in the result list to its *position* in the result list L. $wgt(P)$ is a weight function for different positions in the result list. The weight for a position in the result list is determined by the probability of finding a relevant document in that position. If a document does not appear in the result list, the weight is 0.

In this experiment, a large-grained partition of the result lists was used to produce the position mapping. The weight for each position was empirically derived from the relevance ranked result lists of 50 TREC-3 ad hoc queries against the TREC-3 data collection. Table 1 shows the mapping from an element in the result list to its *position* in the result list, and the probability of finding a relevant document in that position. For example, the 10^{th} document in the result list will be mapped to position 0, which has a probability of 33% to be relevant.

Closer examination of the result lists showed that much greater discrimination was possible by partitioning the result list more finely (Table 2). Though not evaluated, we expect this partition to produce a better similarity measure.

2.4 Query Similarity Thresholds

The past-queries feedback method is sensitive to whether or not there are any queries similar enough to the one being expanded to be used in generating the affinity pool. To avoid polluting the affinity pool with almost surely irrelevant documents in the low-similarity case, we used a threshold to eliminate poor matches. The threshold determined how many queries were expanded – the lower the threshold, the more queries are expanded.

Position	Range	Probability
0	0-29	0.33
1	30-99	0.17
2	100-199	0.10
3	not exist	0.00

Table 1: The mapping from ranges in the result list to positions and the probability of relevance used in the experiment (wide range).

Position	Range	Probability
0	0-1	0.59
1	2-9	0.42
3	10-99	0.19
4	100-199	0.10
5	not exist	0.00

Table 2: The mapping from ranges in the result list to positions and the probability of relevance (finer range).

For the 50 TREC-5 queries, when each was compared against 109 other queries (for a total of 5450 comparisons), we observed the similarity measure to range between 0 and 0.56. Theoretically it could range up to 1, but this happened only when comparing a query against itself. We ran test runs at two different thresholds, 0.025 and 0.012, chosen because of the number of queries that would be augmented (we didn't want too few, or too many). These thresholds yielded 23 and 35 queries (out of the 50 TREC-5 queries) respectively which have affinity pools. The effects of these thresholds on recall-precision and average precision are described in the following section.

3 Discussion of Results

Every query in top-document feedback had an affinity pool and so feedback was used for all 50 TREC-5 ad hoc queries. Top-document feedback showed an improvement of 7.5% over the baseline in average precision (Table 3) and an improvement of 13.5% in the total number of documents retrieved. Compared to the baseline, precision was higher for top-document feedback at all recall values except 0 (Figure 2). However, precision at top 5 and top 10 documents suffered, with a drop of 5% on average. Precision at all other points up to the 100 document precision level showed an improvement over the baseline (Figure 3).

Unlike in top-document feedback, some TREC-5 topics had no similar past queries to create affinity pools in past-queries feedback. For these topics, the baseline results were used. Though we saw improvement of precision at almost all recall levels for all queries (Figure 2), the actual effect of past-queries feedback is best revealed when only the results from queries which have affinity pools are compared (Figure 4). With only the queries that were augmented by past-queries feedback, precision improved at all recall levels.

As expected, top-document precision suffered with the

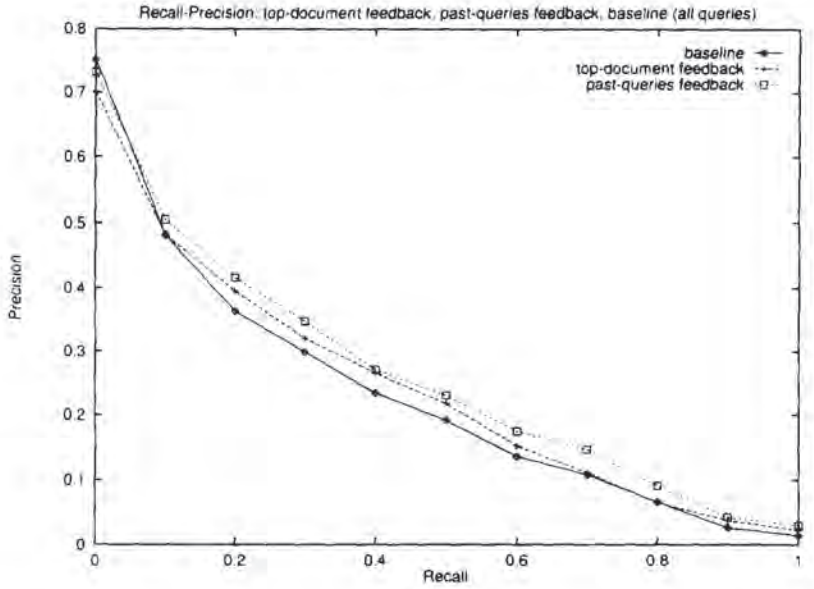

Figure 2. Recall-Precision for baseline, top-document and past-queries feedback methods for all TREC-5 queries.

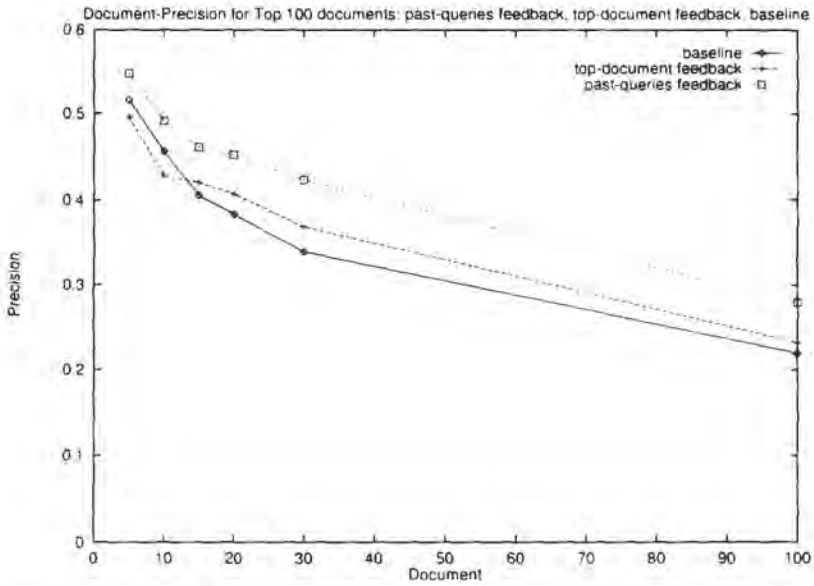

Figure 3. Document-Precision at top 100 documents for baseline, top-document and past-queries feedback methods for the 23 TREC-5 augmented queries only.

precision	top-document		past-queries		baseline	
	all queries	augmented	all queries	augmented	all queries	augmented
average	0.2291	0.2232	0.2465	0.2606	0.2133	0.1884
exact	0.2736	0.2629	0.3095	0.3112	0.2693	0.2238

Table 3: Average and exact precision for baseline, top-document and past-queries feedback methods.

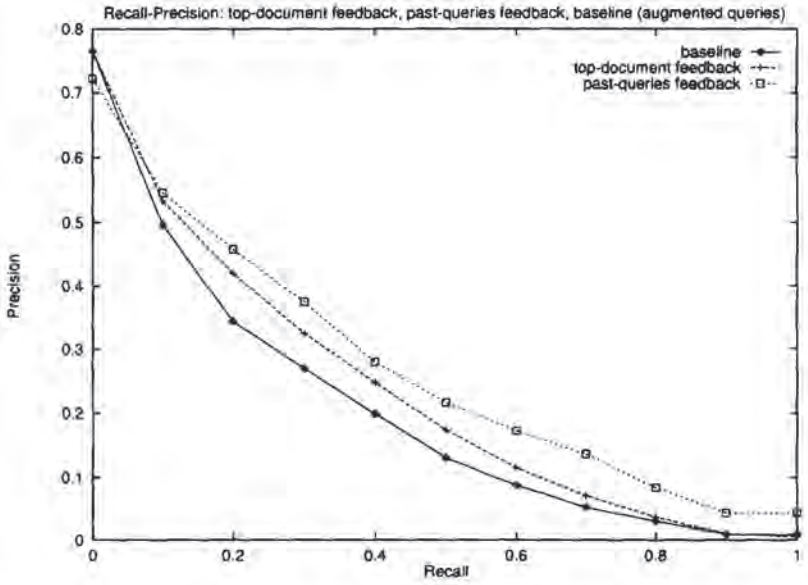

Figure 4: Recall-Precision for baseline, top-document and past-queries feedback methods for the 23 TREC-5 augmented queries only.

top-document feedback method. However, this was not the case for past-queries feedback. Precision at top 5 and top 10 documents in past-queries feedback improved 6% on average (Figure 3).

Table 3 shows the average precision and exact precision for the baseline, top-document feedback and past-queries feedback methods. For all 50 TREC-5 ad-hoc queries, past-queries feedback showed an average precision improvement of 15.5% over the baseline, an improvement of 7.6% over top-document feedback.

Comparing average precision for the expanded queries only (23 of the 50), past-queries feedback showed an improvement of 38.3% over the baseline and an improvement of 12.1% over top-document feedback. The difference between this and the measures for all queries is due to the fact that the baseline query result was used when an affinity pool could not be found for a query in the past-queries feedback method.

We experimented with four affinity pool thresholds. Two were used to demonstrate the effect of thresholds on the average precision and recall-precision curve. [6] The tighter the threshold, the fewer queries with affinity pools. When the threshold was 0.025, 23 topics had affinity pools. When the threshold was 0.012, 35 topics had affinity pools. Average precision and exact precision improves as the threshold tightens. For example, average precision improved 24.0% when the threshold was increased from 0.012 to 0.025.

Figure 5 shows the recall-precision curve for the two thresholds. Similar to average precision, precision was improved at all recall value when the threshold was increased.

[6] The results shown above are from the 0.025 threshold.

4 Related Work

Raghavan, et al.[12] reported on efforts to reuse past *optimal* queries for either of two purposes: to short-circuit query evaluation by matching a user query to one that had already been evaluated, and to jump-start optimal query formation using a *steepest descent* method by using the closest matching past query as the initial query vector. To frame this effort, they presented a detailed analysis of similarity measures between queries. They concluded that the best similarity metric between queries was obtained by comparing query result vectors rather than treating the queries as term-vectors. They presented distance and similarity algorithms that used the relevant and non-relevant document overlap and ordering relations between query results. Using the Cranfield 1400 collection they attempted to validate that the use of the best match past query as the initial query in a steepest descent query refinement method reduced the number of iterations needed to reach an optimal query.

Our effort differed from Raghavan's in several areas. First, the goal of the their experimental effort was to accelerate formation of optimal queries in the presence of relevance

DOCKET ALARM

Explore Litigation Insights

Docket Alarm provides insights to develop a more informed litigation strategy and the peace of mind of knowing you're on top of things.

Real-Time Litigation Alerts

Keep your litigation team up-to-date with **real-time alerts** and advanced team management tools built for the enterprise, all while greatly reducing PACER spend.

Our comprehensive service means we can handle Federal, State, and Administrative courts across the country.

Advanced Docket Research

With over 230 million records, Docket Alarm's cloud-native docket research platform finds what other services can't. Coverage includes Federal, State, plus PTAB, TTAB, ITC and NLRB decisions, all in one place.

Identify arguments that have been successful in the past with full text, pinpoint searching. Link to case law cited within any court document via Fastcase.

Analytics At Your Fingertips

Learn what happened the last time a particular judge, opposing counsel or company faced cases similar to yours.

Advanced out-of-the-box PTAB and TTAB analytics are always at your fingertips.

API

Docket Alarm offers a powerful API (application programming interface) to developers that want to integrate case filings into their apps.

LAW FIRMS
Build custom dashboards for your attorneys and clients with live data direct from the court.

Automate many repetitive legal tasks like conflict checks, document management, and marketing.

FINANCIAL INSTITUTIONS
Litigation and bankruptcy checks for companies and debtors.

E-DISCOVERY AND LEGAL VENDORS
Sync your system to PACER to automate legal marketing.