Experiments with TREC using the Open Text Livelink Engine

Larry Fitzpatrick, Mei Dent, Gary Promhouse
Open Text Corporation

Abstract

In **TREC-5** we baselined the Open Text Livelink Search Engine 6.1 and tested the use of a new automatic feedback technique against both the baseline and automatic top-document feedback. Baseline queries were created in a manner consistent with real users: small queries (average 5 word), created without benefit of query execution, manual feedback or external sources. The interesting results were that other similar queries used as a source of new evidence for automatic query augmentation (*feed-forward* returned a 38% average precision improvement over the baseline, a 12% average precision improvement over automatic top-document feedback, a 6% improvement in top-document feedback (at the 5 and 10 document levels), and was amenable to thresholding for optimal application of the technique. Automatic top-document feedback yielded nominal improvements and hurt top-document precision, which is consistent with the literature. Attempts to use the embedded document structure to improve search results showed no improvements, despite subjective judgments in other domains that this can be worthwhile.

1 Introduction

This is Open Text's first participation in a **TREC** conference, even though the history of the Livelink technology extends back into the middle 1980's. We are participating as a Category C system in the *ad hoc* query track using the *off-the-shelf* Open Text Livelink Search Engine *Release* 6.1.

Our objectives in this participation were twofold. First, we wanted to test various subjective observations that the use of document structures in relevance scoring can improve retrieval effectiveness. Second, we evaluated several automatic query expansion techniques.

Early explorations in this **TREC** effort attempted to characterize the suitability of the **TREC** data to several structure weighting techniques. The text algebra that is the query language for the Livelink engine [8] integrates low-level relevance score composition functions with rich structure handling primitives. We have employed these features in a number of installations to customize scoring functions to the data and application yielding noticeable improvements in relevance scoring in limited subjective evaluations.

The experience in online environments, of others (e.g., [10] with Okapi) and ourselves (e.g., the *Open Text Index*), shows that users generally make little investment in searching (e.g., supply few query words, provide limited feedback) yet do notice the effectiveness (or ineffectiveness) of their searches. To address this, we have explored techniques that automatically augment user queries. We compared two separate automatic augmentation techniques using the **TREC** data: top-document feedback and feed-forward.

To facilitate useful comparisons, first we manually constructed queries with nominal intellectual input and generated runs from them to serve as a baseline (**OTC-1**). Then, as a basis for comparison against the baseline and against other augmentation techniques we measured automatic augmentation by the top ranked documents (*top-document feedback*). This technique is described by [2]. The run describing it is **OTC-2**.

Third, we explored the idea that a query is really an event that should not be decoupled from the user issuing it. We simulated the existence of a *user context* and measured augmentation by this private history. This is similar to the query augmentation described in [4]. The benefits to this technique are seen in a client-server environment where user context is readily (and privately) available and the processing burden imposed by augmentation is distributed to the client. The run describing *feed-forward* is **OTC-3**.

2 System Description

2.1 Search software

The roots of the Livelink engine are in the *New Oxford English Dictionary Project* undertaken by the University of Waterloo [1]. The original engine was built to handle the demanding query requirements of structured data as found on this project. Salient to this discussion is that the Livelink query language is based on a *text algebra*, PAT, [8] and that the data model imposes no fixed notion of a *retrieval unit* (or *document*) at index time. In fact, the notion of document is a flexible in that the retrieval unit is defined at query time. Instead, the indexing process defines *regions* over the text which may correspond to words, sentences, text fragments, fields, documents, aggregations of documents, etc. Regions may be generated from the data, externally supplied metadata, or may be added at runtime by query or by extension.

The statistical scoring capabilities of the Livelink engine exist as primitives in the *text algebra* and incorporate the age-old probabilistic components: global term weights, document term frequency, document length, externally supplied term weights, and a full complement of score combinators. These values and operators can be computed and applied over any regions defined in a database, not just what one normally thinks of as *the document*, or *retrieval unit*. Also unlike most systems, score generation for user queries is flexibly defined by the mapping of the user query language onto the text algebra primitives.

The upshot is that by custom combination of the region algebra with the ranking expressions, great flexibility can be obtained in defining how a retrieved item is to be scored. Because of these factors, we frequently encode application and data specific knowledge into scoring functions.

2.2 Test environment

The system description of the hardware used for index building and query evaluation is as follows:

Processor: DEC ALPHA server 2100

CPU Clock: 200 MHz

RAM: 128M available

Hard Disk: 74 GB hard disk

The memory requirements of the Livelink engine are configured by the administrator. To build the **TREC** database, 60 Mbytes of memory was used. This machine was both a development machine and a file server, and was rarely dedicated solely to **TREC** work.

3 Description of Experiments

Our efforts were directed at evaluating techniques for improving retrieval effectiveness. The first was the use of document structure. The second was the use of two automatic query augmentation techniques. We generated 3 runs that we will discuss, named **OTC-1**, **OTC-2**, and **OTC-3**. Here we provide a brief overview of the approach, and in the following sections describe each set of experiments in more detail. However, we first need to point out an embarrassing error.

The runs that we completed before the submission deadline were executed against two-thirds of the database, and resulted in loss of access to 519 of the 5524 total relevant documents. We discovered this when we received relevance judgments on the 16^{th} of October and recognized that the recall-precision numbers were approximately 25% below the test runs we had done internally against the **TREC-3** collection. The affected runs are **colm1** and **colm4** and are replaced by **OTC-1** and **OTC-3**. Henceforth, we refer to runs **OTC-1** and **OTC-3** for all comparisons, as **colm1** and **colm4** are useless. Figure 8 and Figure 10 are replacement recall-precision graphs for the official ones in the appendix run using the full data set.

It is well known in operational environments that users do not provide very many query terms. On the Open Text Index (a publicly available database created by crawling the WWW), accumulated statistics show the average number of query terms to be around 2. As the volume of data made available by search engines increases, we have to ask: *How will* 1 *or* 2 *query words effectively partition 10's of GB of data?* To counteract this effect, we have been exploring models of query augmentation that rely on implicitly derived evidence, as opposed to explicit feedback. [1] First, we evaluate the use of automatic top-document feedback. While not new, it is already part of the product and it provides a basis for comparison with other augmentation techniques and with the past literature on this method.

Second, we explore automatic query building by *feed-forward* augmentation. The premise of *feed-forward* is the notion that queries are not isolated one-shot events, but are often framed by the longer-term information needs of a user or community of users. We simulate a long-term user need and use this as the source of data for query augmentation. A number of related efforts over the last several years have looked at using past user behavior to improve information discovery activities, mostly in filtering applications ([6], [5]), but also in searching ([4]).

Each of our runs is described in Figure 1, and then explained in more detail in the following sections.

3.1 Baseline - OTC-1

The generation of the baseline run consisted of index building, query creation and query execution.

[1]Not that we're opposed to explicit feedback methods. In fact we support those as well, we just do not describe them here.

3.1.1 Index building

The index contained a total of $1,371,876$ documents. The documents were not reformatted during index building. The Livelink engine supports the logical addition of *metadata* to each document. We added metadata regions to identify which sub-collection each document came from, and which **TREC** CD it came from. This allowed us to divide the data into sub-collections at query time without having to build separate sub-databases.

Each SGML tag in the data was used to create a region for that tag name. In addition, certain SGML tags with common meaning were grouped under one region name. For example, the SGML tags in the data that identify a headline were **<HEAD>**, **<HL>**, **<HEADLINE>**, depending on the collection, and during index building stage we grouped these tags into the region **Headline**. [2] This enabled us to search independently for **<HEAD>** or **<HL>**, or alternatively any of the headline forms using **Headline**.

The **TREC** database was built in 4 hours.

3.1.2 Query creation

TREC-5 queries were manually created. All topic fields were used to create the query, although emphasis was placed on the **Description** field. We followed some rough guidelines to approximate real users' queries: (a) limit query operators to

[2]The documents were not changed in any way to do this.

stem expansion, either of the *or* operators or the *near* operator (which we used sparingly), (b) do not execute the queries or use **TREC** data in any way, (c) do not look at external resources for term generation, e.g., dictionaries or thesauri, and (d) keep queries reasonably small. On average about five terms per query resulted.

The average time to create a query for a topic was on the order of several minutes, but varied greatly. The total resource used was around one person-day.

3.1.3 Query execution

The baseline queries were run on the same machine as the indexing machine. Absolute CPU time consumed was under 60 seconds for all queries. Total elapsed time per query is under 1 minute, affected by factors such as network load, other development activities, etc. [3]

3.2 Using structure

A region in Livelink is a span of text in the database, defined by start and end offsets. Regions generally are used to represent portions of the text that have something in common. For example, a newspaper may be composed of many *story* regions, each of which contains a *title* region, a *headline* region, and a *body* region. The Livelink engine offers great flexibility in both the creation of these regions at build time or at run time, as well as the use of these regions to affect the rankings of documents.

We have used the region structure in a number of applications to improve perceived retrieval effectiveness. For example, on the Open Text Index, query terms matching text from the automatically generated summary region were weighted more than matches from the raw text, and query terms that appear to be site names were weighted more heavily when matched out of the URL region.

We ran a number of tests on structure weighting with the **TREC-3** collection and had intended to test this capability against **TREC-5**. However, the results of the experiments were not significant enough to warrant any **TREC-5**-time. But, here is a brief overview of what we tried and observed and we will not discuss it further.

Experiments were run to see if weighting regions differently during ranking of the documents had a measurable effect. Nothing more than nominal differences were observed. We attribute this to several factors. First, the **TREC** record structure is very limited in semantics, with *headline* the only widely used and truly notable feature. We posit that using this structure had no effect because many of the documents were newspaper articles and their construction of headlines is often obtuse. Second, the different **TREC** subcollections actually have different structure, which makes the problem more difficult. We did not try to compensate for this, outside of the mention of normalizing those things that were obvious.

[3] The machine was actively used as a departmental server during this effort, though we would have enjoyed having an Alpha 2100 all to ourselves!

We also partitioned long documents into sections to see if it improved retrieval effectiveness. We tried various partition sizes and various score combination functions. The partition size ranged from 50 to 200 words and the overlap ranged over 0, 10, 20, and 50. We noticed that varying the size of the artificial regions that fragmented a larger retrieval unit into smaller ones did not change the overall performance significantly, as observed by [11]. Also, we tried several score combination methods for use with fragments including maximum, average, median, and total. The best performer was the maximum score of the individual fragments.

3.3 Top document feedback as augmentation - OTC-2

We tested automatic top-document feedback using the *find similar pages* capability of the Livelink engine. This provided a measurement of the effectiveness of this technique with the Livelink engine as compared with research reported elsewhere ([2]). Also, it allowed comparison with the other automatic augmentation technique we tested, *feed-forward*.

This method was virtually identical to what [2] described. The baseline query was executed against the collection and the result list obtained. The top several documents were selected and terms were extracted from them. The top terms were ordered by importance using a function that took into account global and local term weights in addition to lexical similarity with other selected words. The top several terms were added to the original query and the query was re-executed. The parameters to this method were number of top documents to use and the number of top terms to select from the top documents and were tuned previously using **TREC-3** data.

Top-document feedback imposed an additional query processing load due to key term extraction and then re-execution with the augmented query. The augmented query took slightly longer than the base query to evaluate due to the additional terms (about 2x the original). The cost of key term extraction was nominal.

3.4 User context as feed-forward augmentation - OTC-3

We make the hypothesis that in many situations, search can be improved if treated more like a filtering activity. Filtering satisfies chronic information needs and so IR efforts on this problem use past performance to improve later activity. A search, on the other hand, has almost always been treated as an isolated event that satisfies an acute need. If we accept that a user's motivation to search is frequently consistent with longer term information needs, then his past information consuming habits should be predictive of relevance in a given search.

The idea that a user's past interests can be used to modulate a query's ranking order was explored in the Okapi project [4]. In this effort, a user's past documents provided *context* used to break ties in relevance scores of the query result list. Many other efforts have explored the use of user's past efforts to facilitate future discovery efforts ([5], [6] to name only two). Here, we explored this notion with the Livelink engine and a query's background context.

To test this hypothesis, we wanted a history for a *user*,. **TREC** does not provide this directly. However, we simulated a user's past interests, in the context of a single query, and used this information as the basis for *feed-forward* query augmentation. The source of this *past context* was other **TREC** queries. The basic *feed-forward* approach is as follows:

- Simulate a background context for each query, consisting of several hundred documents. We call this the *affinity corpus* for the query.
- Execute the query against this affinity corpus and select the top n documents.
- If the top documents exceed a similarity threshold to the query, select the m most representative terms from these documents to augment the query.
- Execute the augmented query against the full corpus and keep this result list.

The only information available to create an affinity corpus was the complete body of **TREC** queries. We used a complement of all 50 **TREC-3**, 10 **TREC-4**, and all 50 **TREC-5** queries executed against the **TREC-5** data as candidates for similar queries. To create an affinity corpus for each query, we selected the top 100 documents from each of the 3 most related queries. We compared queries by result-list ([7]) using a weighted Jaccard score that counted the overlap between the top 100 documents of each result list. [4]

The assumption was that a user with a given query need may have explored that need before and would have seen several documents related to his present need and these documents would be in his affinity corpus. We experimented with several different numbers for the top documents and top terms selected from the affinity corpus to augment the baseline query.

Since some queries had no queries like them, we applied a threshold to the augmentation. If the affinity corpus had low similarity to the query, as measured by a low query-query similarity score, we did not augment the query with terms from the affinity corpus. We experimented with several threshold values and settled on one that allowed about half of the queries to be augmented.

It took about a minute of elapsed time to create the affinity corpus for each query from the 109 other query results. To select the background documents and create a *feed-forward* augmented query added less than a second to each query. Remem-

[4] Initially, we tried to find related queries by manually grouping topics based on subjective judgments of relatedness. We found this to be a painful classification exercise, fraught with ambiguity. Fundamentally, it didn't take into account the underlying data. We abandoned this effort reasonably quickly.

ber, the affinity corpus was only 300 documents and this effort could take place at a client machine, thus adding no server burden for creating the augmentation. The augmented queries took roughly a factor of two longer to run than the base queries due to more terms. Finally, for queries that did not have a sufficiently strong affinity corpus, i.e., did not exceed the threshold, we just used the base query.

4 Discussion of Results

We executed three separate runs: a baseline run (**OTC-1**), an automatic top-document feedback run (**OTC-2**), and the automatic feed-forward using past queries as evidence (**OTC-3**).

4.1 Result analysis for baseline run - OTC-1

The baseline run measured the effectiveness of the manually constructed queries with the default relevance ranking algorithm. This run was very close to the median run for **TREC-5**, on average. The total number of relevant documents retrieved was 2741, and the average precision was 21.33%.

Figure 2 shows the number of topics that performed better or worse than **TREC-5** median, and the percentage of increase or decrease in average precision. The median difference is around +5%. Note that when a baseline query did worse than the **TREC-5** median, it was likely to do very much worse. Out of the 16 topics with an average precision worse than the median by more than 10%, we were later able to improve half of them to outperform the median by adding only one or two query words, by applying simple query refinement using only a few minutes per query. Clearly, we were hurt by not using manual feedback or extra-data resources when building these queries.

4.2 Result analysis for top-document feedback run - OTC-2

This run tested query augmentation by automatic top-document feedback. The engine's *find similar pages* feature generated the augmented query terms.

Both average precision and the total number of documents retrieved increased over the baseline. The total number of relevant documents retrieved was 3112, 13.5% above our baseline, OTC-1. The average precision was 22.91%, about 7.5% above OTC-1. The results are compared in Table 1.

Figure 3 shows the number of topics that performed better or worse than our baseline run (**OTC-1**), and the percentage change in average precision. Interestingly, comparing on average precision, when top-document feedback helped, it tended to help a lot and when it hurt, it only hurt a little (with one exception). Close examination of the results showed that when the top documents used for feedback were all relevant, the greatest improvement of average precision was achieved. When none of the top documents used for feedback were relevant, moderate improvement of

precision	top-document		feed-forward		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 1: Average and exact precision for baseline, top-document feedback and feed-forward methods.

average precision was achieved, partially because the top document precision in the baseline run was so poor. When the top documents used for feedback were a mixture of relevant and non-relevant documents, results were mixed.

4.3 Result analysis for feed-forward run - OTC-3

The result of the OTC-3 run showed an increase over the baseline run in both average precision and the total number of documents retrieved. The total number of relevant documents retrieved was 2877, about 5% over our baseline, OTC-1. The average precision was 24.65%, just over 15% above OTC-1.

Figure 6 shows the number of topics that performed better or worse than our baseline run, and the percentage change in average precision. The median difference was over +15%. When a proper threshold was applied, great improvement in average precision was achieved.

The application of feed-forward augmentation for a query was subject to a threshold. Of the 50 total queries, only 23 queries exceeded the threshold. For the 27 queries that could not be augmented, the baseline run (OTC-1) was used.

When we compared the feed-forward method against the baseline and against the top-document feedback method, the improvement is much more marked. Table 1 lists the average precision and recall for just the *augmented* as well.

Comparing just augmented queries, feed-forward showed a 38% improvement in average precision over the baseline which constituted a 12% improvement over top-document feedback. Figure 4 shows the recall-precision curves for all queries and Figure 5 shows the recall-precision curves for the augmented-only queries. The separation for all recall-precision points (except the 0th) is more pronounced for augmented-only queries. The ability to change the threshold to trade greater or lower levels of improvement against lesser or greater numbers of queries against which to apply it seems powerful.

Conventional wisdom holds that automatic feedback hurts top-document precision. Figure 7 shows that while top-document feedback did hurt top-document precision, the feed-forward method actually improved it.

5 Conclusion

As a first time participant, we certainly learned a lot about how to work effectively within the **TREC** evaluation framework. Except for the successful evaluation of a new automatic feedback technique, unfortunately, much of that knowledge will have to be applied to the next **TREC** and is not evident here.

After correcting for the error in which we failed to index a substantial portion of the relevance judgments for our official runs, our baseline performance was slightly better than the **TREC** median for the manual ad hoc category. Our misjudgment was in the method we used for building queries from **TREC** topics. It did not provide a favorable base for comparison against other systems, as it was neither fully manual nor fully automatic.

To create queries from topics, we read the topics and in a few minutes composed, on average, five word queries and used this as a baseline. This is consistent with what real users do. We did not consult external sources, the data itself, or use feedback in any way. However, we think we would have been better served to either fully automate the query creation step and participate in the automatic ad hoc category, or to rely on feedback and consultation of knowledge bases to refine the query before settling on baseline queries for the manual run - live and learn. Most systems in the manual category appear to have invested substantial effort in the latter. That notwithstanding, we demonstrated what we believe to be important advances in the application of automatic feedback methods. But, why automatic feedback at all when the results in the literature have been ho hum?

Our past experience indicates that end users in production search environments do not provide queries sufficiently rich to be effective for very large data collections. As collection sizes continue to increase, and as more users engage in searching, this situation promises to get even worse. One approach to improve search effectiveness for end users is to engage them in activities that tease out more information to help them formulate better queries. A complementary, not incompatible, approach is for the search environment to assist automatically, using whatever it can learn about past user, group, and system behavior to improve search effectiveness. We believe there exists much untapped evidence in end-to-end query systems.

In this **TREC-5** effort, we've shown some very positive results using implicitly derived information to help search effectiveness. Automatic feed-forward, using past queries as additional evidence, was superior to automatic top document feedback and actually improved top document precision.

When we compared only the 23 queries for which feed-forward augmentation was performed, the results were surprisingly good. Average precision for feed-forward was 38% over the baseline and 12% better than top-document feedback. Moreover, feed-forward augmentation improved the top 5 and top 10 document precision levels by about 6%. Consistent with reports elsewhere ([2]), automatic top-document feedback yielded nominal improvements over the baseline of 7.5% in average precision and 13.5% in recall and hurt top-document precision by about 5%.

The use of an empirically derived threshold for determining when to apply feed-forward definitely helped. Performance of feed-forward applied indiscriminately was nominally the same as the top-document feedback method. The fact that the feed-forward method using past queries as additional evidence could be easily subjected to a threshold is substantially in its favor. We do not think, however, that we have exhausted the possibilities.

We think that for many cases, a search is not an isolated event, yet this is how virtually all systems treat them. Of necessity, our efforts here took a system-centric view of looking for additional evidence [5] to help augment queries, because **TREC** does not provide any notion of a user. We most certainly have not exhausted system-centric sources of additional evidence. For example, we compared queries to past queries by result-set overlap. In this comparison, we derived judgments about which parts of the result list were likely to contain relevant documents, and so were weighted more heavily. We completely ignored in this evaluation (because we had to), which documents from past queries may have been viewed, selected, saved or printed by the original querier, thereby indicating higher probability of relevance ([6] studied several such factors applied in a non-searching context).

If system-side evidence can help automatically improve queries, and we think we have shown that it can, then surely user-specific evidence collection, located client-side so as not to impose a load on the server, appears even more promising. By simple observation of user behavior we note that user-generated queries often overlap past queries issued by the very same user. If a single query can be helped by other users' past query efforts, wouldn't the same user's past queries perform at least as well? Even more likely as a source of evidence than repeat queries is the probability that the user has read an electronic document related to the subject he is querying, whether it was email, web-browser, or other. What of the documents from those similar past queries that that the same user viewed, saved, printed, etc.? Why not what he has written? Can we identify evidence in a user's past information consuming habits to help future information seeking? We think so.

[5]other users' past queries

References

[1] The New Oxford English Dictionary Project at the University of Waterloo, D.L. Berg, G.H. Gonnet, F.Wm. Tompa. Technical Report OED-88-01.

[2] C. Buckley, G. Salton, J. Allan, A. Singhal, Automatic Query Expansion Using SMART: **TREC 3**, Proceedings of Third Text Retrieval conference (**TREC-3**), NIST Special Publication 500-225, 1995.

[3] An Algebra for Structured Text Search and a Framework for Its Implementation, Charles Clarke, G. Cormack, and F. Burkowski, The Computer Journal, V38.No.1 1995.

[4] Goker, A. and McCluskey, T. L. Incremental learning in a probabilistic information retrieval system. In: L.A. Birnbaum and G.C. Collins and (eds). Machine Learning. Proceedings of the Eighth International Workshop on Machine Learning (ML91), Evanston, IL, USA, June 1991. San Mateo, CA: Morgan Kaufmann, 1991. 255-9.

[5] Pattie Maes, Agents that Reduce Work and Information Overload, Communications of the ACM, Vol. 37, No.7, July 1994.

[6] M. Morita and Y. Shinoda: Information Filtering Based on User Behavior Monitoring and Best Match Text Retrieval, Proc. of the Seventeenth International ACM-SIGIR Conference on Research and Development on Information Retrieval, 272-281, Dublin, Ireland, July 1994.

[7] Vijay V. Raghavan, Hayri Sever, On the Reuse of Past Optimal Queries, Proc. of the Eighteenth International ACM-SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, July 1995.

[8] Pat expressions - An algebra for text search, Airi Salminen, Frank Tompa. Technical Report OED-92-02, 1992.

[9] An Overview of Waterloo's Database Software for the OED, Frank Wm. Tompa. Technical Report OED-92-01, 1992.

[10] Walker, S. and de Vere, R. Improving subject retrieval in online catalogues: 2. Relevance feedback and query expansion. London: British Library, 1990. (British Library Research Paper 72).

[11] Ross Wilkinson, Effective Retrieval of Structured Documents, In the ACM-SIGIR Conference on Research and Development in Information Retrieval, 1994.

A Appendix

Run	Description
colm1	Baseline run, manually created queries, partial database. Submitted, garbage run. Ignored.
colm4	Automatic expansion of query. Submitted, garbage run. Ignored.
OTC-1	Baseline run, manually created queries. Used as benchmark to compare other runs.
OTC-2	Automatic expansion of query by top ranked document feedback.
OTC-3	Feed-forward augmentation. Automatic expansion using simulated user context-documents that a user may have seen in the past.

Figure 1: Name and description of each run.

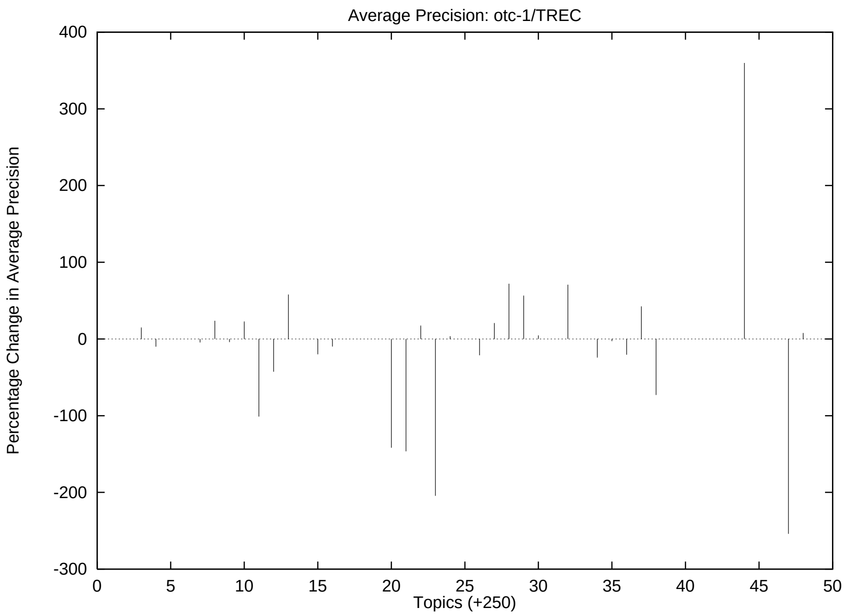

Figure 2: Percentage change in average precision for each topic: OTC-1 vs. TREC-5 median.

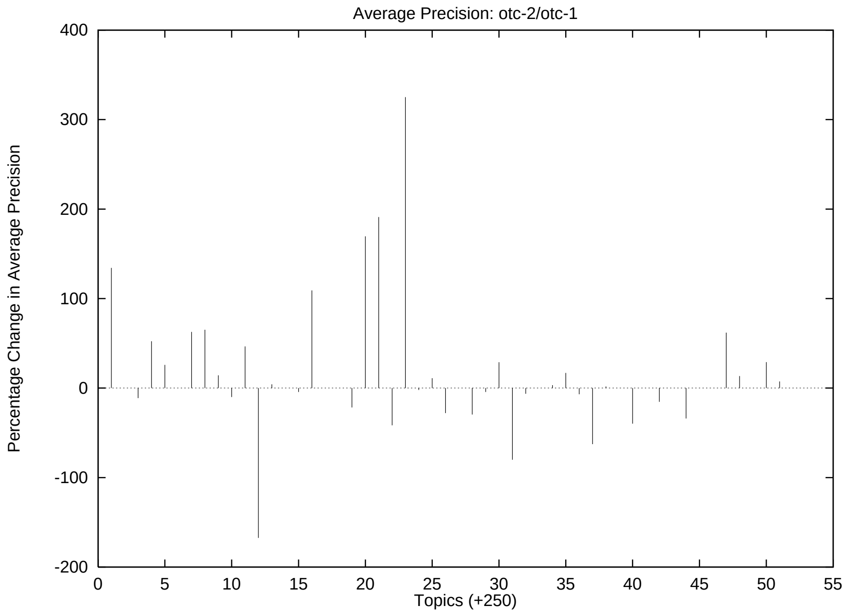

Figure 3: Percentage change in average precision for each topic: OTC-2 vs. OTC-1.

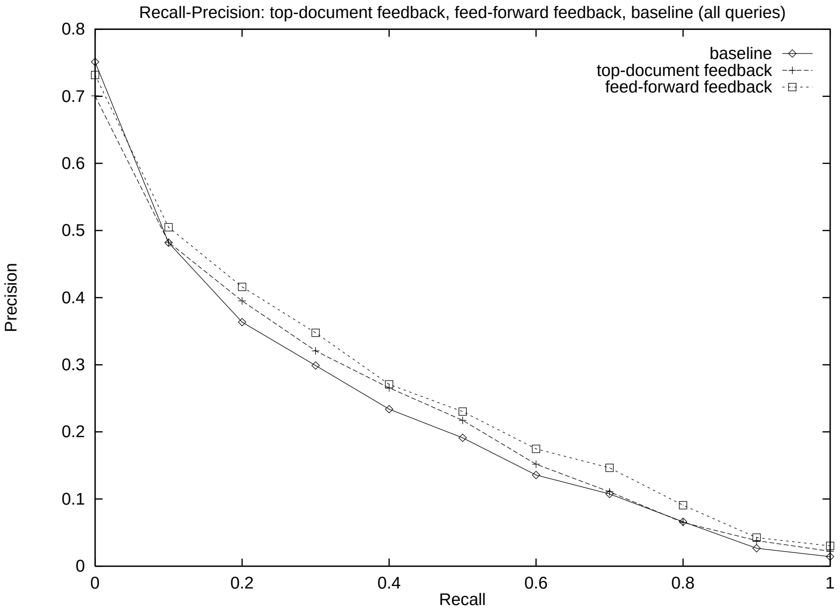

Figure 4: Recall-Precision for baseline, top-document and feed-forward feedback methods for all queries.

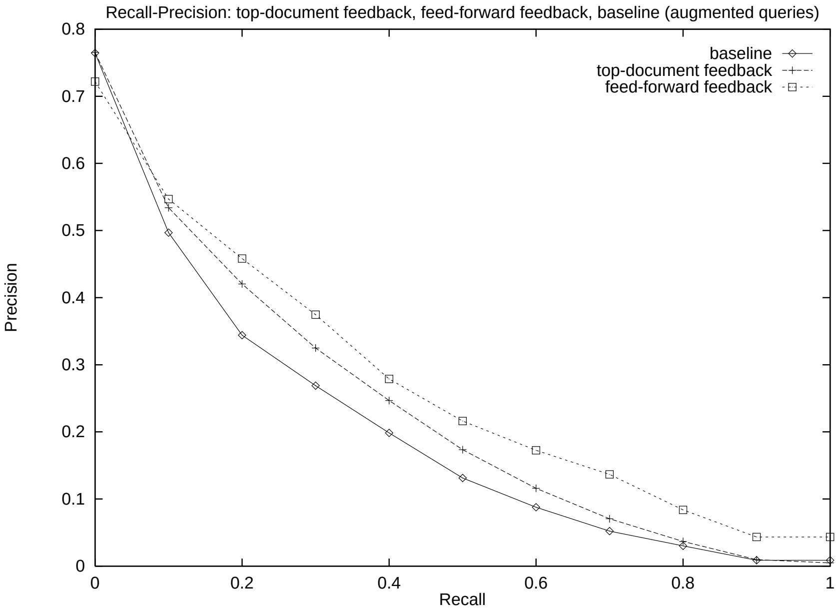

Figure 5: Recall-Precision for baseline, top-document and feed-forward feedback methods for augmented queries only.

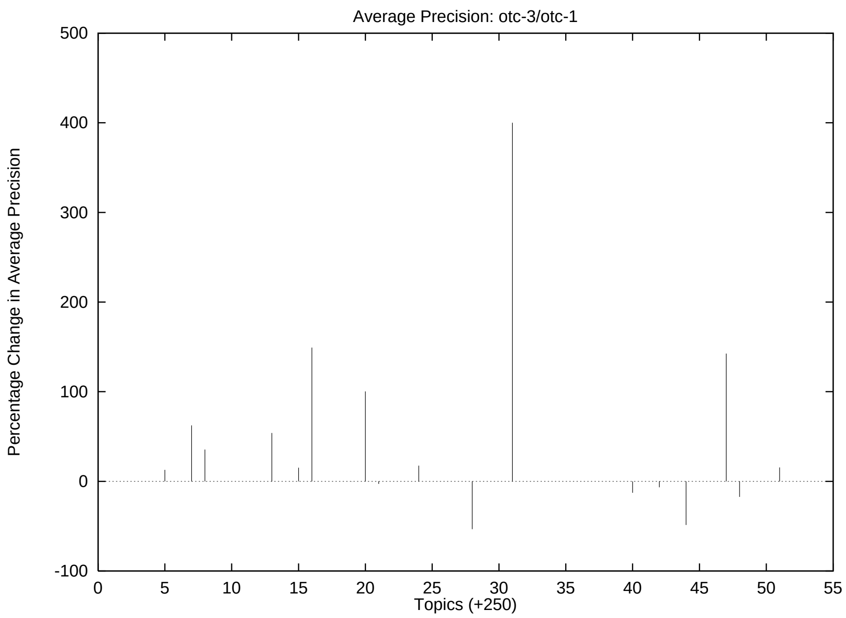

Figure 6: Percentage change in average precision for each topic: OTC-3 vs. OTC-1.

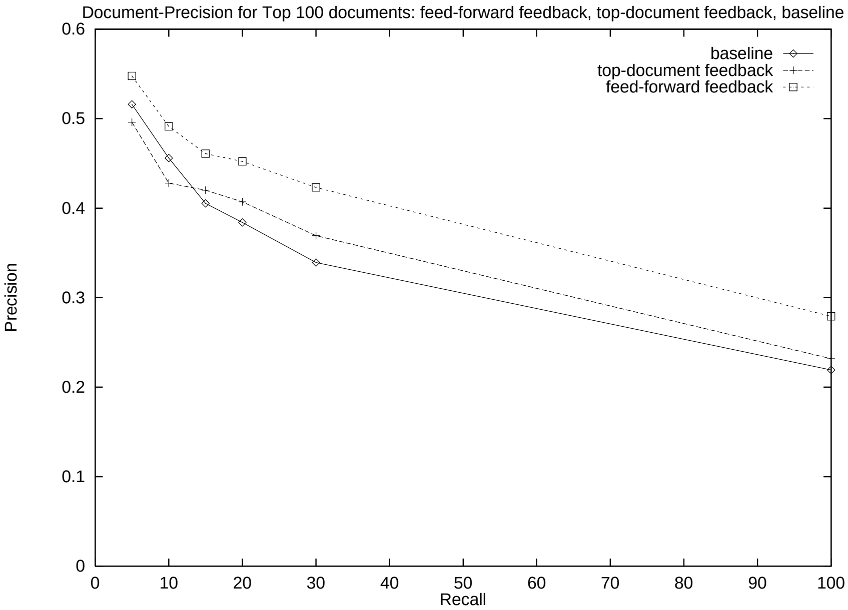

Figure 7: Document-Precision at top 100 documents for baseline, top-document feedback, and feed-forward methods.

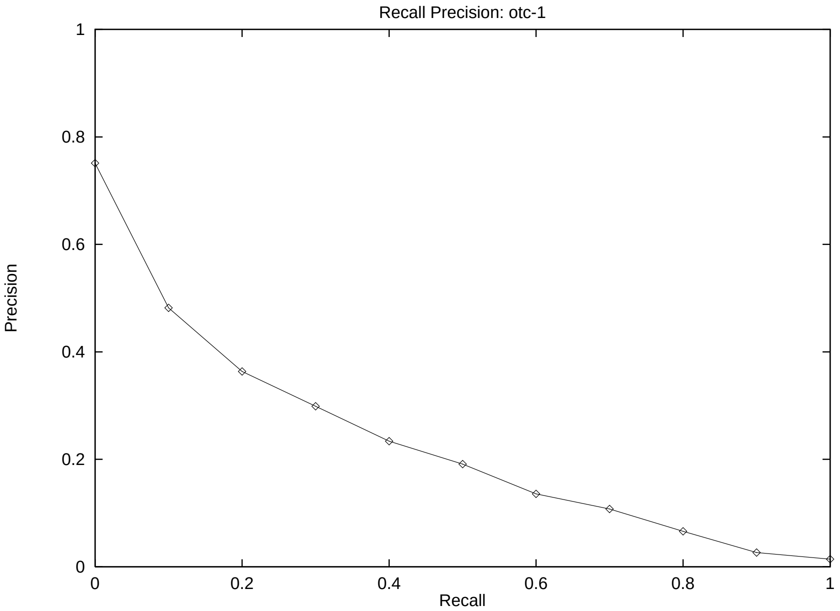

Figure 8: Recall precision for OTC-1.

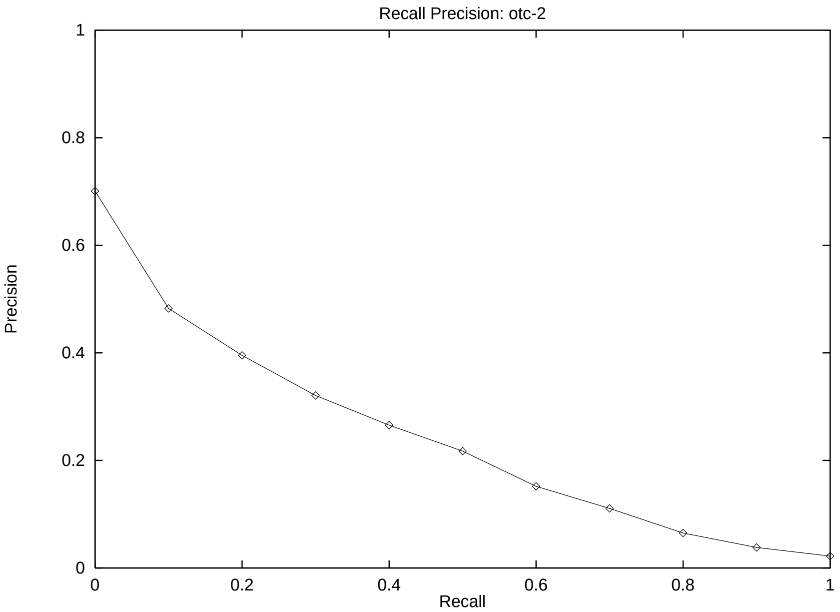

Figure 9: Recall precision for OTC-2.

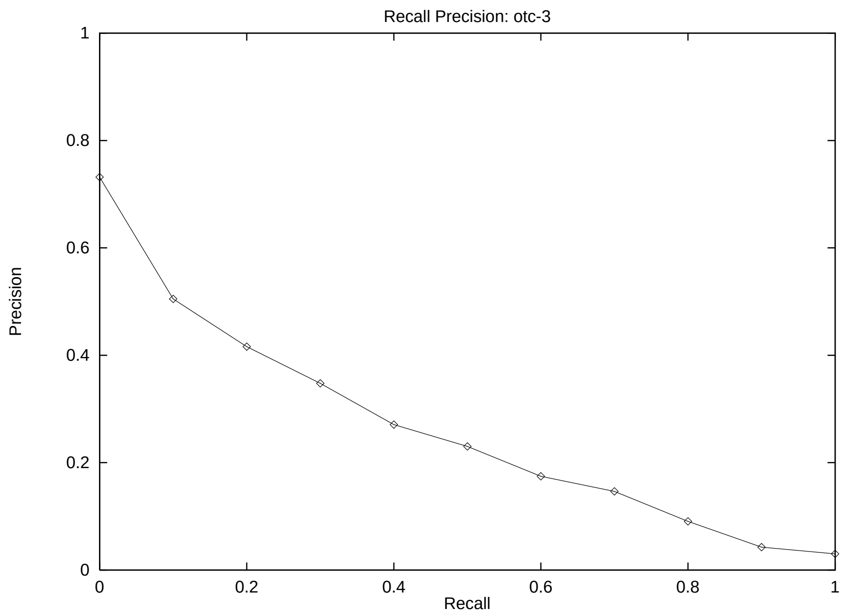

Figure 10: Recall precision for OTC-3.