AT&T at TREC-9

Amit Singhal Marcin Kaszkiel

AT&T Labs–Research
{singhal,martink}@research.att.com

Abstract

This year we come to TREC with a new retrieval system *Tivra* that we have implemented over the last year. Tivra is based on the vector space model, and is mainly designed to do large-scale web search with limited resources. We run Tivra on a cheap Linux box. It currently indexes around 14-15 gigabytes of web data per hour, and allows sub-second web searches for 2-3 word queries on a 700 MHz Pentium box. At the time of submissions Tivra was in its early development stages, and was not fully tested. However, we still submitted runs for both the web tracks – 10 gigabytes and 100 gigabytes. The results look quite reasonable for an untested version of the system. For the 10 gigabytes ad-hoc task, our results are above median for majority of the queries. This is specially notable given that we use only the title portion of the queries whereas the results pool contains results based on both long and short queries. Our results are among the top results in the 100 gigabytes task.

1 Tivra

Over the last year, we have implemented a new retrieval system called Tivra. Tivra is designed to be a large-scale web search engine which runs on relatively inexpensive Linux PCs. This year at TREC, we submitted several runs for the web tracks using an early development version of Tivra.

Tivra maintains a full positional inverted index on the title and the body of a web page. In our last measurements, Tivra indexed about 15 gigabytes of web data in an hour on a 700 MHz Pentium PC using about 1 gigabyte of RAM. We use a short stop-list of 118 words/numbers. At the time of our TREC runs, we did not use any stemming in Tivra. Since then, we have incorporated a plural stemmer into Tivra. We store raw term frequencies, and the corresponding byte-offsets for words. All term weighting is done at query time.

Tivra also builds indices for the anchor texts that point to a web page. In essence, each web page is indexed as three different documents: the page itself, the off-site anchor texts for the page, and the in-site anchor texts for the page. We maintain a distinction between the off-site anchor text (anchor text for in-links coming from outside the web page's site) and the in-site anchor text (anchor text for in-links coming from within the web page's site) to allow more emphasis on in-links from an outside host. The assumption here is that if a page from a different site points to some page, than this in-link is a stronger recommendation for the pointed-to page, as compared to an in-site in-link. The anchor index does not have positional information. This was motivated by the fact that anchor texts are typically short and positional information, which is mainly needed to enforce proximity, is not that important in this case. The total index size is roughly 15% of the raw web data indexed.

At retrieval time, Tivra processes all the inverted lists in document-order. [1] The inverted list are stored sorted by the document-ids. Tivra reads all the inverted list in one go and runs an efficient merge-sort maintaining a heap of top documents. All term weighting is done at this time. Tivra can compute several scores for a document, for example, a score based on off-site anchor texts, a score based on in-site anchor texts, a global tf$\times$idf score, proximity based scores, proximity in title, and so on. These scores can be combined to get the final document score/rank.

2 10 Gigabytes Web Task

We submitted six runs for the 10 gigabytes ad-hoc task. Four of them—`att0010gb`, `att0010gbl`, `att0010gbt`, and `att0010gbe`—do not use any linkage information for the documents. The other two runs—`att0010lf` and `att0010glv`—do use anchor text in their ranking. **All runs use only the title portion of the queries.** We strongly believe that very short queries are in the true spirit of current web search engines.

We use *dnb.dtn* scoring scheme developed in our previous TREC work (see [3] for details). Here is a description of our runs:

- `att0010gb`: This run places documents with all query terms ahead of documents which don't have all query terms. If this strict boolean `AND` doesn't get us 1,000 documents, we add high scoring documents that contain at least one of the two most uncommon (as measured by idf) query terms.
- `att0010gbt`: This run is similar to `att0010gb` but it prefers documents with all query terms in the title field ahead of all other documents.
- `att0010gbl`: This run is similar to `att0010gb` but it assigns an extra credit for locality of query terms in the document body. Here locality is implemented as a window of query length (in bytes) + 50 bytes.
- `att0010gbe`: This run is our two pass query expansion run. This is an overly simplified version of our query expansion run described in [3]. Here are the steps:
  - **Pass-1**: Using *dtn* queries and *dnb* documents, a first-pass retrieval is done.
  - **Expansion:** Top ten documents retrieved in the first pass are *assumed* to be relevant to the query. Rocchio's method (with parameters $\alpha = 3$, $\beta = 2$, $\gamma = 0$, γ is immaterial since we do not have any non-relevance here) is used to expand the query by adding twenty new words with highest Rocchio weights. [2] To include the *idf*-factor in the expansion process, documents are *dtb* weighted.
  - **Pass-2:** The expanded query is used with *dnb* documents to generate the final ranking of 1,000 documents.
- `att0010glf`: This run incorporates anchor texts for a page into the scoring function. The final document score is:

1.00	×	off-site anchor text based score	+
0.25	×	in-site anchor text based score	+
1.00	×	title based score	+
1.00	×	locality based score	+
1.00	×	global body score	

  We did not have a chance to train these parameters on any data and these are our best guess parameters.
- `att0010glv`: This is a variant of `att0010glf` and in this run the contribution on the anchor text is reduced as the query gets longer. The thought is that short queries benefit from page recommendations by others whereas the long one's don't. Something that the results don't support strongly.

The results are shown in Table 1. As we had expected given the early stages of development of our system, these results are not spectacular but they are definitely reasonable. These results indicate that linkage analysis (in form of anchor text based indexing) doesn't help the retrieval effectiveness. We would not make this claim with certainty in the general web search environment. In all the testing we have done in-house, linkage analysis improves the search precision notably on short queries. It is possible that the results we obtain the TREC environment are in fact an artifact of the environment.

Run	Average Precision	Best	>= Median	< Median
att0010gb	0.1341	0	27	23
att0010gbt	0.1182	1	24	25
att0010gbl	0.1380	0	32	18
att0010gbe	0.1464	1	27	22
att0010glf	0.1250	2	26	22
att0010glv	0.1288	0	29	21

Table 1: Results for 10 gigabytes task (title only queries)

Run	P@1	P@5	P@10
att0010gb	0.5357	0.5190	0.4964
att0010glf	0.5476	0.5048	0.4738

Table 2: Results for 100 gigabytes task (88 queries)

3 100 Gigabytes Web Task

We submitted two runs for the large web task—`att00100gb` and `att00100glf`. These runs are just the 100 gigabytes counterpart of the corresponding 10 gigabytes runs. The results in Table 2 are quite impressive given that one of the runs is a plain tf×idf based run. It is quite promising that for over half the queries, the very first document retrieved was judged relevant. Once again we notice that linkage analysis hasn't improved effectiveness. We are still skeptical of this result and are doing a more elaborate test internally.

4 Conclusions

We are pleased by the reasonably good performance of our untested development version of Tivra, our new search engine. Since the official submission we have removed several shortcomings of Tivra and we expect its performance to improve as we test it further. Even though results show that linkage analysis doesn't improve retrieval effectiveness, we are approaching this result with considerable caution. This result can very well be an artifact of the TREC environment. We are currently running a more elaborate experiment in-house to verify this result.

References

[1] M. Kaszkiel, J. Zobel, and R. Sacks-Davis. Efficient passage ranking for document databases. *ACM Transaction on Information Systems*, 17(4):406–439, October 1999.

[2] J.J. Rocchio. Relevance feedback in information retrieval. In *The SMART Retrieval System—Experiments in Automatic Document Processing*, pages 313–323, Englewood Cliffs, NJ, 1971. Prentice Hall, Inc.

[3] Amit Singhal, John Choi, Donald Hindle, David Lewis, and Fernando Pereira. AT&T at TREC-7. In E. M. Voorhees and D. K. Harman, editors, *Proceedings of the Seventh Text REtrieval Conference (TREC-6)*, pages 239–252, 1999.