The Mirror DBMS at TREC-8

Arjen P. de Vries and Djoerd Hiemstra
{arjen,hiemstra}@ctit.utwente.nl
CTIT, University of Twente
The Netherlands

Abstract

The database group at University of Twente participates in TREC-8 using the Mirror DBMS, a prototype database system especially designed for multimedia and web retrieval. From a database perspective, the purpose has been to check whether we can get sufficient performance, and to prepare for the very large corpus track in which we plan to participate next year. From an IR perspective, the experiments have been designed to learn more about the effect of the global statistics on the ranking.

1 Introduction

The Mirror DBMS [dV99] combines content management and data management in a single system. The main advantage of such integration is the facility to combine IR with traditional data retrieval. Furthermore, IR researchers can experiment more easily with new retrieval models, using and combining various sources of information. This is an important benefit for advanced IR research; web retrieval, speech retrieval, and cross-language retrieval, each require the use of several representations of content, which is hard to handle in the traditional file-based approach, and becomes too slow in traditional database systems.

In the Mirror DBMS, the IR retrieval model is completely integrated in the database architecture, emphasizing efficient set-oriented query processing. The support for information retrieval in our system is presented in detail in [dV98] and [dVW99]. It supports other types of media as well, which has been demonstrated in the image retrieval system prototype described in [dVvDBA99]. The main goal of our participation in TREC is to test if our system can handle larger data sets without too many problems. Also, we wanted to find out the effect of global statistics on the ranking.

This paper is organized as follows. Sections 2 and 3 review the design of the Mirror DBMS and its support for IR, and discuss its use for TREC processing. Section 4 explains the experimental setup and interprets our results. Section 5 discusses our experience with using the Mirror DBMS for TREC, followed by conclusions.

2 Design

A complete overview and motivation of all aspects of the design of the Mirror DBMS is presented in [dV99]. Although following a traditional three-schema architecture, it uses different

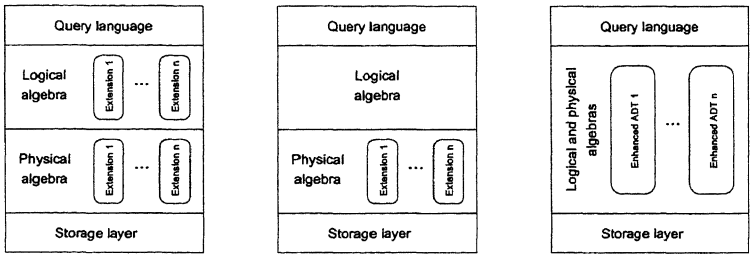

Figure 1: The multi-model DBMS architecture next to the extended relational and E-ADT DBMS architectures (from left to right).

data models at different levels: we therefore classify its design as **multi-model DBMS architecture**. The crucial architectural difference from other extensible database systems is that query processing at the logical layer uses *only* operators that are provided by the physical layer (see also Figure 1), and, domain-specific query processing (such as an IR extension) is defined *at the logical level primarily*. This choice enforces a system-wide physical data model and algebra spanning all extensions. Of course, the physical algebra can also be extended if necessary, i.e. when logical operations cannot be expressed efficiently in the physical algebra. The strict separation between the logical and physical levels allows using algebraic query optimization techniques, a key property of relational database management systems but hardly ever used in non-business application areas like content management.

The multi-model architecture provides the query processor with transparancy through the layers. Put informally, query evaluation can 'look down' from the original request through all layers of the architecture. This should enable set-oriented query evaluation for almost every request, and allow maximal exploitation of parallelization and pipelining. In contrast, the black-box ADTs of 'object-relational' database systems restrict the DBMS in the possible manipulations of the query plans. This makes it more complicated to distribute and parallelize the query plans, or change the buffer strategy for iterative query processing as proposed in [JFS98]. Another alternative, the enhanced ADTs proposed by Seshadri [Ses98], provides little opportunity for optimizations that cross the bounderies between different extensions. Figure 1 compares these three architectures schematically.

3 Implementation

The prototype implementation of the Mirror DBMS uses Moa at the logical level, and Monet at the physical level. Monet is a parallel main-memory database system under development at the CWI in Amsterdam [BK95, BMK99], that is targeted as a backend system for various (query-intensive) application domains, such as GIS and data mining.[1] Moa is an object algebra studied in the database group at University of Twente, that is extensible with domain-specific structures. The Moa tools transform expressions in this algebra into sequences of operations in MIL, an algebra for the binary relational data model supported by Monet.

For the support of IR, we extended Moa with new structures at the logical level to handle document representation, ranking, and the computation of co-occurrence statistics. In com-

[1]Monet is used succesfully on a commercial basis by Data Distilleries, a start-up specializing in data mining applications.

document collection

dj	ti	tfij
1	a	2
1	c	3
2	a	1
2	b	2
2	e	2

intermediate results

qdj	qti	qtfij	qntfij
1	a	2	0.796578
2	a	1	0.621442
2	b	2	0.900426

Table 1: Representation of content in BATs

bination with Moa's kernel support for collections and tuples, these structures can model a wide variety of IR retrieval models: the current prototype supports the well-known Okapi ranking scheme, InQuery's inference network retrieval model, as well as the linguistically motivated retrieval model (LMM, presented in Section 4.3). To illustrate, the following Moa expression ranks a collection of documents:

```
map[sum(THIS)](
  map[getBL(THIS, query, stats)]( docs )
);
```

The first `map` operation computes term probabilities for the query terms occurring in the document, using the global statistics specified in structure `stats`. The subsequent `map` combines these probabilities using a sum operation. Although this particular expression may not seem very interesting, the IR ranking operators can be combined with other operators such as select, resulting in a powerful query language.

The representation of the logical IR structures at the physical level is termed the **flattened representation** of the content. It consists of three binary tables (BATs), storing the frequency $tf(t_i, d_j)$ of term t_i in document d_j, for each term t_i occurring in document d_j. Table 1 illustrates this for a collection $\{d_1, d_2\}$ with documents $d_1 = [a, c, c, a, c]$ and $d_2 = [a, e, b, b, e]$. Computing the probability of relevance of the objects for query $q = [a, b]$ proceeds as follows. First, a table with the query terms is joined with the document terms in `ti` (the result is called `qti`). Next, (using additional joins) the document identifiers and the term frequencies are looked up (`qdj` and `qtfij`).[2] Finally, the retrieval status values are computed with some variant of the popular $tf \cdot idf$ ranking formula. To support these computations, Monet's physical algebra has to be extended with new operators, either in C or C++, or as a MIL procedure. The latter is preferrable for easy experimentation; for example, the following MIL procedure computes the term probabilities given normalized term frequency and inverse document frequency using the LMM model:

```
PROC bel( nidfi, ntfij ) := {
  RETURN log( 1.0 + nidfi * ntfij * C );
}
```

An evaluation run processes 50 topics in batch, but the client interfaces of the Mirror DBMS

[2]Note that these joins are executed very efficiently, because the Moa structures make sure that the BATs remain *synchronized* all the time.

have been designed for interactive sessions with an end-user. Also, transferring the data from Monet to the Moa client has not been implemented optimally. Furthermore, optimizations such as using materialized views are not performed in the current Moa rewriter. These minor flaws would have inferred an unfair performance penalty to the evaluation of the architecture, and made logging the results rather cumbersome. Therefore, as a (temporary) solution, the MIL program generated by the Moa rewriter has been manually edited to loop over the 50 topics, log the computed ranking for each topic, and use two additional tables, one with precomputed normalized inverse document frequencies (a materialized view), and one with the document-specific constants for normalizing the term frequencies.

4 Experimental setup and results

Collection fusion is the process of merging the results of retrieval runs on seperate, autonomous document collections into an effective combined result [VGJL95]. We have focused on this problem because large collections will be fragmented (horizontally) in several partitions, each managed by a separate server. Maintaining the exact global statistics induces an extra overhead, that may not be necessary if the fragments are sufficiently large.

Collection fusion is a trivial task for exact matching retrieval systems like systems using Boolean retrieval, but more complicated if a ranked retrieval system is used. In a number of publications on collection fusion it is argued that simply comparing similarity measures accross subcollections leads to unsatisfactory results because of differences in the collection-dependent frequency counts [Bau97, CLC95, VF95, VGJL95]. One of the objectives of the TREC-8 evaluation described in this paper is to question this hypothesis. We feel that similarity measures across subcollections might in fact be comparable, but show worse evaluation results because of the evaluation setup.

4.1 Evaluation using the TREC collection

Relevance assessments on the TREC test collections are assembled by the pooling method: a pool of possibly relevant documents is created by taking the a sample of documents retrieved by each participating system. This pool is then shown to the human assessors [VH99a]. The sampling method used in TREC takes the top 100 of the retrieved documents of each participating system.

Since the start of TREC in 1992, the test collections have been used in numerous evaluations outside the official TREC. For these evaluations, all documents that were not in the top 100 of any of the official participating systems are assumed to be not relevant. But, evaluations that did not contribute to the TREC pool probably have unjudged documents in the top 100 making these evaluations less reliable than the official TREC evaluation. This is especially true for new, previously unexplored approaches to retrieval. If a systems finds relevant documents that no system was able to find before, then these documents will probably not be judged in an old TREC collection. *The only way to check the relevance of these documents is by official TREC participation.*

4.2 Conditions for naive collection fusion

Let us define 'naive' collection fusion as the process of merging the search results on the subcollections based on the document similarities. The first condition for naive collection fusion is that each subcollection uses the same retrieval model or weighting algorithm for retrieval. Secondly, we assume that each subcollection uses the same indexing vocabulary [Bau97]. A third condition is that subcollections are sufficiently large to allow for the reliable local estimation of document frequencies. If the subcollections are too small, ineffective retrieval on the subcollections will affect the merged result.

An evaluation of Callan et al. [CLC95] under these conditions for TREC topics 51-150 showed that naive merging was significantly worse than ranking based on globally estimated document frequencies, causing losses from 10-20% in average precision. But, the results of naive merging reported by Callan et al. [CLC95] were not part of an official TREC participation. It is likely that their merged run has a worse coverage of judgements, because the TREC-2 and 3 pools were (almost) only created by systems that use a central index for retrieval. Maybe, their merged run was as good as the central index run after all. To check this hypothesis, we decided to put up a retrieval run using naive merging for judging.

4.3 Some theoretical back-up for naive merging

The Mirror DBMS uses the linguistically motivated probabilistic model of information retrieval [Hie99, HK99]. The model builds a simple statistical language model for each document in the collection. The probability that a query $T_1, T_2, \cdots, T_n$ of length n is generated by the language model of the document with identifier D is defined by the following equation:

$$P(T_1 = t_1, \cdots, T_n = t_n | D = d) = \prod_{i=1}^{n} (\alpha_1 \frac{df(t_i)}{\sum_t df(t)} + \alpha_2 \frac{tf(t_i, d)}{\sum_t tf(t, d)}) \quad (1)$$

Equation 1 can be rewritten to a vector product formula by first dividing it by $\prod_{i=1}^{n}(\alpha_1 df(t_i)/\sum_t df(t))$ [Hie99]. This will not affect the ranking within a subcollection, but it will affect the final ranking after merging the search results of the seperate subcollections, because we divided by collection specific document frequencies. It can be shown that the ranking of the vector product formula in table 2 approximates the ranking defined by the conditional probability $P(D|T_1, T_2, \cdots, T_n)$ of a document being relevant given a query.

vector product formula:	$\text{similarity}(Q, D) = \sum_{k=1}^{l} w_{qk} \cdot w_{dk}$
query term weight:	$w_{qk} = tf(t_k, q)$
document term weight:	$w_{dk} = \log(1 + \frac{tf(t_k, d)}{df(t_k) \sum_t tf(t, d)} \cdot \frac{\alpha_2 \sum_t df(t)}{\alpha_1})$

Table 2: $tf \cdot idf$ term weighting algorithm

From Bayes' rule we know that dividing equation 1 by $P(T_1, T_2, \cdots, T_n)$ and multiplying it by P(D) results in $P(D|T_1, T_2, \cdots, T_n)$. For a large collection and a query that has a small number of hits, $tf(t, d) = 0$ for most terms t and documents d. Therefore, $\prod_{i=1}^{n}(\alpha_1 df(t_i)/\sum_t df(t))$ approximates the marginal probability $P(T_1, T_2, \cdots, T_n)$ and the ranking defined by table 2

approximates the ranking defined by $P(D|T_1, T_2, \cdots, T_n)$. The a-priori probability $P(D=d)$ of a document d being relevant can be included by adding the logarithm of equation 2 to the similarities of table 2 as a final step.

$$P(D = d) = \frac{\sum_t tf(t,d)}{\sum_t \sum_d tf(t,d)} \tag{2}$$

We hypothesise that, if the approximation is not too far off, the result after merging is not significantly worse than what would have been possible with a central index.

4.4 Official results

Table 3 lists the official TREC runs. Global runs denote runs using the global collection statistics. Local runs denote the naive collection fusion runs, using local collection statistics on the four TREC subcollections: Federal Register, Foreign Broadcast Information Services, Los Angelas Times and Financial Times.

run name	description	avg. prec.
UT800	global run	0.260
UT803	global run; LCA	0.176
UT803b	global run; LCA from F.Times and LA Times	0.260
UT810	local run (judged)	0.043
UT813	local run; LCA from local	0.145

Table 3: official results

Unfortunately, our submitted official runs have been degraded by two bugs, that affected in particular the naive merging run that was judged by NIST. By our own mistake, the global runs have used the wrong (local) normalizing constant for the *idf*;[3] an error in Monet's `join` implementation resulted in random answers for three of the four local runs. After fixing these bugs, the results of the global run UT800 improved from 0.260 to 0.275 and the results of the local run UT810 improved from 0.043 to 0.260. Table 4 lists the results on the four subcollections. Except for the Federal Register, which has hits for only 19 topics anyway, the average precision on the subcollections do not differ much at all. Unofficial runs, with these bugs fixed, are indicated in this paper by a 'u' postfix (so 'UT500u' is the fixed 'UT500' run).

run name	Fed.Reg.	FBIS	LATimes	F.Times	merged
UT800u (global)	0.326	0.317	0.279	0.356	0.275
UT810u (local)	0.351	0.319	0.276	0.356	0.260
topics w. hits	19	43	45	49	50

Table 4: average precision per subcollection after bug-fix

The merged local run is about 6% worse than the global run. This might be a significant difference according to some significance test, like e.g. the t-test [Hul93]; but, if so, it is still not valid to draw the conclusion that the global approach is indeed better than the naive merging approach. This conclusion would only be valid if both evaluations were done under identical, controlled, conditions; which they are not, because both runs were not judged by NIST and we do not control the other systems that contributed to the pool. Almost all systems that contributed to the TREC-8 pool were systems using the global approach.

[3]Strange enough, this mistake improves average precision slightly on the TREC-6 topics.

Therefore, the pool favours central index approaches over distributed index approaches if it is used to evaluate runs that did not contribute to the pool. This can be shown by looking at the percentage of documents that are judged for different cut-off levels of the fixed UT800u and UT810u runs. The percentage of documents in a run that are judged, will be called the judged fraction.

run name	P at 10	P at 30	P at 100	P at R	avg. P
UT800u (global)	0.496	0.378	0.234	0.319	0.275
UT810u (local)	0.436	0.343	0.222	0.310	0.260

run name	J at 10	J at 30	J at 100	J at R
UT800u (global)	1.000	1.000	0.996	0.987
UT810u (local)	0.984	0.978	0.952	0.947

Table 5: merged results after bug-fix: a) precision; b) judged fraction

Table 5a and b show the precision and the judged fraction of the global and the local run at different cut-off levels. There is a major difference between the judged fractions of the global run and the local run. The global run misses 0.4% of the documents in its top 100. The local run misses 4.8% of the documents in the top 100, some of them are even missing in the top 10.

4.5 Local context analysis

Based on its success on InQuery at previous TREC conferences, we expected a significant improvement by using topics expanded with LCA [XC96]. Also, investigating the expansion terms, LCA seemed to do a good job. For example, on topic 311 (which is about industrial espionage), it finds terms like 'spy', 'intelligence', and 'counterintelligence', and from the financial times sub-collection it even identifies 'Opel', 'Volkswagen', and 'Lopez' as relevant terms. But, instead of improving the effectiveness of retrieval, the measured performance turned out to have degraded. Some tweaking of the parameters, reducing the weights of expansion terms and using fewer of them (N=30), the performance improved upon the baseline, but only slightly; on the runs submitted for TREC-8, it has degraded performance.

A possible explanation for these disappointing results is that the algorithm has been applied to documents instead of passages (as done in [XC96]), and the TREC collection itself was used to find expansion terms instead of another, larger collection. One result was that the varying length of documents had a large impact on the expansion terms chosen, which is undesirable. Another explanation is that LMM weighting provides such a high baseline, that it is very hard to improve upon. A comparison between the (impressive) baseline results of LMM on TREC-6 favours the latter explanation: because the performance of the Mirror DBMS with LMM weighting scheme, without LCA, was almost as good as InQuery's performance after using LCA. With the tweaked LCA, LMM weighting performed better on all reported precision and recall points, except for the precision at twenty retrieved documents, at which InQuery performed slightly better. On the TREC-8 topics it did not contribute positively to the results.

5 Discussion

Although more of anecdotal than scientific value, the story of our participation in TREC-8 with the Mirror DBMS illustrates the suitability of this architecture for experimental IR. Eight days before the deadline, it still seemed impossible to participate with this year's TREC, as Monet kept crashing while indexing the data; until, the seventh day, the new release suddenly made things work! We decided to try our luck and see how far we could get in a week; and we should admit, it has been a crazy week. It meant running the topics on TREC-6 first, to compare the results with the runs performed before; as well as changing the ranking formula to integrate document length normalization. In the weekend, we implemented the use of co-occurrence statistics (which has turned out to be not so useful as expected). So, in one week we managed to index the data, perform various experiments for calibration, run the best experiments on TREC-8, and submit five runs, just before the final deadline.

5.1 Efficiency

The machine on which the experiments have been performed is a Sun Ultra 4 with 1 Gb of main-memory, running SunOS 5.6. The machine is not a dedicated server, but shared with some other research groups as a 'compute server'. Monet effectively claims one processor completely while indexing the collection, or processing the fifty topics on each of the sub-collections. The division of the complete collection in five sub-collections (as it comes on different compact discs) is maintained. The topics are first run in each sub-collection, and the intermediate results are merged. Depending on the size of the sub-collection, estimating the top 1000 ranking takes between 20 seconds and two minutes per topic. How to further improve this execution performance is discussed below.

Preparation of the five sub-collections takes about six hours in total. Computing the table with document-specific term frequencies is performed using Monet's module for crosstables. But, using the grouping operation for all documents at once allocates all available memory, and eventually crashes the DBMS because it cannot get more, if it is run on the complete set of documents of any but the smallest sub-collection.[4] Therefore, the indexing scripts run on fragments of the sub-collections at a time, and frequently write intermediate results to disk, obviously slowing down the process more than necessary.

5.2 The road ahead

The execution performance of the Mirror DBMS on TREC is clearly better than a naive (nested-loop) implementation in any imperative programming language, but, the obtained efficiency is not fast enough to beat the better stand-alone IR systems that also participate in TREC. But, compared to the techniques used in systems like InQuery (see [Bro95]), the current mapping between the logical and physical level is too straightforward: it does not use inverted files, has not fragmented the terms using their document frequency, and it ranks all documents even if only the beliefs for the top 1000 are used. Also, Monet should make it relatively easy to take advantage of parallelism in modern SMP workstations.

The merits of some possible improvements can only be evaluated experimentally. For ex-

[4]Notice that such problems are not necessarily solved by using commercial systems; Sarawagi et al. report similar memory problems with DB2 when using normal SQL queries for mining for associations hidden in large data sets [STA98].

ample, it is not so clear beforehand whether inverted files are really the way to go. Query processing with inverted files requires merging the inverted lists before beliefs can be computed, which is hard to perform without trashing the memory caches frequently; which has been shown a significant performance bottleneck on modern system architectures (see e.g. [BMK99] for experiments demonstrating this for Monet).

Without experiments, much improvement can be expected from fragmentation of the document representation BATs based on the document frequency, in combination with the 'unsafe' techniques for ranking reported in [Bro95]. Some preliminary experiments indicate a 100 times improvement with only a small loss in precision. Such (domain-specific) optimization techniques are easy to integrate in the mapping from Moa structures to MIL, thanks to the declarative nature of the algebraic approach. A similar argument applies to extending the Mirror DBMS with the buffer management techniques discussed in [JFS98]. In MIL, buffer management is equivalent to directing Monet to load and unload its tables. By integrating such directives in the generated MIL programs, it is expected that these improvements can also be added without many complications.

6 Conclusions

Without any additional algorithms, LMM ranking produces reasonably good results. Unfortunately, due to the bug in our experiments, we cannot yet give conclusive answers about the difference between using local or global statistics; but, we may conclude that the difference is rather small. Our current use of co-occurrence statistics has not improved our results, but further research is necessary in this area.

Despite of the flaws in the current implementation, we believe that the Mirror DBMS has proven to be a useful platform for IR experiments on the TREC data. The true benefits of its design will only be exploited when the system is developed further, and the indexing task is more challenging. Next year, the Mirror DBMS should be ready to participate in the large WEB track.

Acknowledgements

Many thanks go to Peter Boncz and the other members of the CWI Monet team, for their great support. The work reported in this paper is funded in part by the Dutch Telematics Institute project DRUID.

References

[Bau97] C. Baumgarten. A probabilistic model of distributed information retrieval. In *Proceedings of the 20th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'97)*, pages 258–266, 1997.

[BK95] P.A. Boncz and M.L. Kersten. Monet: An impressionist sketch of an advanced database system. In *BIWIT'95: Basque international workshop on information technology*, July 1995.

[BMK99] P.A. Boncz, S. Manegold, and M.L. Kersten. Database architecture optimized for the new bottleneck: Memory access. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. To appear.

[Bro95] E.W. Brown. *Execution performance issues in full-text information retrieval.* PhD thesis, University of Massachusetts, Amherst, October 1995. Also appears as technical report 95-81.

[CLC95] J.P. Callan, Z. Lu, and W.B. Croft. Searching distributed collections with inference networks. In *Proceedings of the 18th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'95)*, pages 21–28, 1995.

[dV98] A.P. de Vries. Mirror: Multimedia query processing in extensible databases. In *Proceedings of the fourteenth Twente workshop on language technology (TWLT14): Language Technology in Multimedia Information Retrieval*, pages 37–48, Enschede, The Netherlands, December 1998.

[dV99] A.P. de Vries. *Content and multimedia database management systems.* PhD thesis, University of Twente, Enschede, The Netherlands, December 1999.

[dVvDBA99] A.P. de Vries, M.G.L.M. van Doorn, H.M. Blanken, and P.M.G. Apers. The Mirror MMDBMS architecture. In *Proceedings of 25th International Conference on Very Large Databases (VLDB '99)*, Edinburgh, Scotland, UK, September 1999. Technical demo.

[dVW99] A.P. de Vries and A.N. Wilschut. On the integration of IR and databases. In *Database issues in multimedia; short paper proceedings, international conference on database semantics (DS-8)*, Rotorua, New Zealand, January 1999.

[Hie99] D. Hiemstra. A probabilistic justification for using tfidf term weighting in information retrieval. *International Journal on Digital Libraries*, 1999. To appear.

[HK99] D. Hiemstra and W. Kraaij. Twenty-One at TREC-7: Ad-hoc and cross-language track. In Voorhees and Harman [VH99b].

[HT98] Laura M. Haas and Ashutosh Tiwary, editors. *SIGMOD 1998, Proceedings ACM SIGMOD International Conference on Management of Data, June 2-4, 1998, Seattle, Washington, USA.* ACM Press, 1998.

[Hul93] D. Hull. Using statistical testing in the evaluation of retrieval experiments. In *Proceedings of the 16th ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR'93)*, pages 329–338, 1993.

[JFS98] B.Th. Jónsson, M.J. Franklin, and D. Srivastava. Interaction of query evaluation and buffer management for information retrieval. In Haas and Tiwary [HT98], pages 118–129.

[Ses98] P. Seshadri. Enhanced abstract data types in object-relational databases. *The VLDB Journal*, 7(3):130–140, 1998.

[STA98] S. Sarawagi, S. Thomas, and R. Agrawal. Integrating mining with relational database systems: Alternatives and implications. In Haas and Tiwary [HT98], pages 343–354.

[VF95] C.L. Viles and J.C. French. Dissemination of collection wide information in a distributed information retrieval system. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval (SIGIR'95)*, pages 167–174, 1995.

[VGJL95] E.M. Voorhees, N.K. Gupta, and B. Johnson-Laird. The collection fusion problem. In D.K. Harman, editor, *Proceedings of the 3rd Text Retrieval Conference TREC-3*, number 500-225 in NIST Special Publications, pages 95–104, 1995.

[VH99a] E.M. Voorhees and D.K. Harman. Overview of the 7th text retrieval conference. In *Proceedings of the 7th Text Retrieval Conference TREC-7* [VH99b], pages 1–23.

[VH99b] E.M. Voorhees and D.K. Harman, editors. *Proceedings of the Seventh Text Retrieval Conference (TREC-7)*, number 500-242 in NIST Special publications, 1999.

[XC96] J. Xu and W.B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th International Conference on Research and Development in Information Retrieval (SIGIR '96)*, pages 4–11, Zürich, Switzerland, 1996.