Routing and Ad-hoc Retrieval with the TREC-3 Collection in a Distributed Loosely Federated Environment

Nikolaus Walczuch*, Norbert Fuhr,
Michael Pollmann, Birgit Sievers
University of Dortmund, Germany

Abstract

In this paper, we investigate retrieval methods for loosely coupled IR systems. In such an environment, each IR system operates independently on its own document collection. For query processing, an agent takes the query and sends it to the different IR systems. From the answers received from theses servers, it forms a single ranking and sends it back to the user. In the work presented here, we examine different retrieval methods for performing routing and ad-hoc-queries in such an environment. For experiments, we use the TREC-3 collection and the SMART retrieval system.

1 Introduction

In the past, IR systems have been considered as running on a large, fairly static document collection held on a single machine. However, there is a growing need for the development of IR systems that operate in a distributed environment:

- Document collections for similar subject fields are set up at different locations with different systems and retrieval methods. In retrieval, a user would like to query all relevant databases at once, without bothering about the underlying differences.
- With the improvements of local computing power and network access, the coupling of local and non-local IR data bases becomes possible.

*On leave from: Universidad de los Andes, Merida, Venezuela

So far, little research has been performed that focuses on retrieval in a distributed environment. More precisely, we must distinguish between different types of distribution. We use the term "distributed IR systems" in the same way as in the field of databases: On the logical level, such a system behaves in the same way as a non-distributed systems, the differences are only at the physical level (e.g. distributed storage, replication and partitioning of data, special query processing methods and communication protocols). In contrast, loosely coupled systems also behave differently on the logical level. This approach offers the advantage that it is possible to couple existing systems, without the need to change the software. A simple example for loosely-coupled IR systems is the design of the WAIS system (see [Kahle et al. 93]). The underlying protocol Z39.50 allows a client to do parallel searches in IR systems.

In this paper, we only consider the latter type of system, although some of the methods investigated also may be feasible for distributed IR systems. We call the different IR systems "servers". A user submits a query to a client, which in turn connects to an "agent". The agent's task is to communicate with the different servers. It sends the query to the servers, receives the different answers and then forms a single ranking and sends it back to the user. Some of the servers contacted may in fact also be agents, which in turn know further servers or agents that help processing the query. An example structure of this type is depicted in figure 1.

An example for such a structure is the SFgate system described in [Pfeifer et al. 95] which implements a WAIS gateway for WWW clients and acts as an agent in that it can access several servers in parallel. In the original WAIS system, there are only clients and servers, where the client also must perform the agent's task.

For performing retrieval in such an envioronment, a number of problems have to be solved:

- How do agents and servers communicate?
- What information should the agent have about the different servers that it can access?
- How can the agent select servers that provide relevant documents for the current query?
- Which retrieval methods should be provided by the servers in order to give optimum support for retrieval in loosely coupled IR data bases?
- How should the information agent merge incoming ranking results from different sources (with possibly unknown quality)?

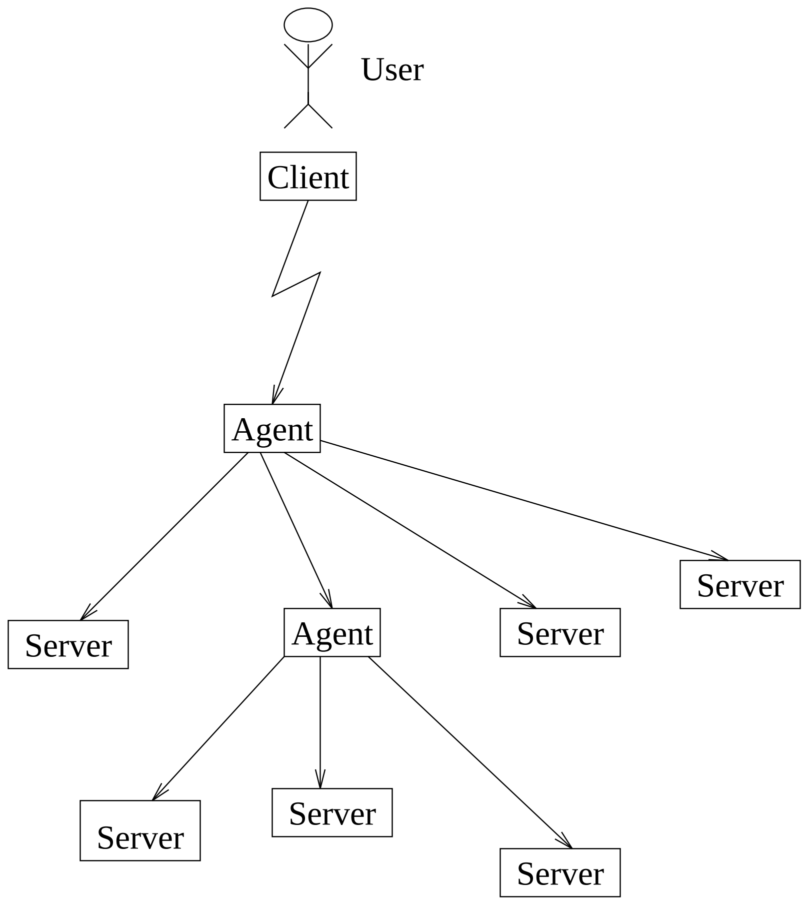

Figure 1: Retrieval with loosely coupled IR systems

- What retrieval quality can be archieved in such an environment, or what is the loss in retrieval quality in contrast to retrieval on a single large data base?

As the TREC collection originally consists of five different document collections, it is quite natural to use this collection for experiments with a distributed environment. In contrast to the very general case, we only consider a single agent contacting five servers, each running on one of the subcollections. As experimental system, the SMART retrieval system was used for simulating retrieval methods in this environment. So we did not really implement a distributed environment, since our focus here is on the investigation of retrieval methods for such an environment. Once good retrieval methods have been devised, an architecture for federated IR systems supporting these methods can be developed.

In the remainder of this paper, we first describe the basic concepts of the retrieval methods investigated, followed by the description of the experiments performed. The results are discussed in Section 4, and finally, we come to the conclusions and give an outlook on further work.

2 Basic concepts

2.1 Distributed routing retrieval

For the routing task, we have queries and a set of documents with relevance judgements. As new documents arrive, the system has to assign relevance status values (RSVs) to these documents, by considering these documents one by one.

Since there is relevance feedback data available, learning methods can be applied in order to improve the representation of queries. As a result, the weights of the query terms are modified. Furthermore, we can apply query expansion methods in order to add terms to the original query.

In a distributed environment, the agent collects relevance feedback data and performs the learning process, possibly in cooperation with the servers. Having improved the query representation this way, it sends it to the servers. Each server processes this query against its incoming documents, and sends the requested number of documents (or all documents with a RSV exceeding a predefined threshold) back to the agent. The agent combines the streams from the different servers into one single output stream and sends it to the client.

In our experiments, we assume that we can compute overall document frequencies, thus being able to compute query term weights in the same way as in a single collection environment. This can be achieved by additional communication overhead, where the agent collects the collection frequencies of the query terms from all servers in the learning phase. On the other hand, document indexing weights are independent of the overall document frequencies of terms.

2.2 A Concept for distributed ad-hoc retrieval

For ad-hoc queries, only the query formulation, but no relevance feedback data is given for the current query.

After receiving a new query from a client, the agent forwards the query to the servers (possibly after preselecting the servers relevant for this query). Each server processes this query on its database and sends the prespecified number of answer documents to the agent. Now the agent merges these results into one single ranked output list. For performing this task, there are at least three different possibilities:

1. The agent takes the RSVs computed by the servers as absolute values and forms the output by merging the input lists according to decreasing RSVs.
2. The agent collects all the documents received for a query and creates a new ranking for this set of documents, e.g. by treating this set as a small document collection.
3. If the server receives additional collection information from the servers, a better ranking can be produced: Assume that each server also sends its document frequencies for all query terms. Then the agent can compute the overall document frequencies and use them as improved query term weights for ranking the set of documents received.

In our experiments, we investigated the last two possibilities.

3 Experiments

For the description of the document and query indexing methods given below, we use the following notations:

q_k	query
d_m	document
t_i	term
tf_{im}	within-document-frequency (wdf) of t_i in d_m
$\max tf_m$	maximum wdf tf_{mi} of all terms in d_m.
idf_i	inverse document frequency of t_i

Based on these parameters, standard SMART indexing routines were applied ([Salton & Buckley 88]): The augmented term frequency atf_{im} is computed as

$$atf_{im} = 0.5\left(1 + \frac{tf_{im}}{\max tf_m}\right),$$

and the logarithmic term frequency ltf_{im} is defined as

$$ltf_{im} = 1 + \ln tf_{im}.$$

Furthermore, we always use cosine normalization for document indexing, i.e. the weights are normalized such that the sum of squares of the indexing weights in a document is 1.

3.1 Distributed Routing

As pointed out in 2.1, we can use the same method of query term weighting as in a single collection environment. On the other hand, document indexing cannot be based on overall document frequencies. For this reason, we choose to use no collection frequency information at all for document indexing. This method is also suited very well for highly dynamic collections.

For experimental setting, the following steps were performed:

1. The training data set D1 first was indexed with the "anc" method using only augmented term frequency and cosine normalization. Alternatively, we also applied the "ltc" method based on logarithmic term frequency in combination with cosine normalization.

2. The query set Q3 (queries 101 to 150) was weighted with the "atc" method using the product of augmented term frequency and inverted document frequency .

3. For internal verification of our methods, data set D2 was used by applying the same document indexing method as for D1.

4. Since the document indexing weights are independent of any specific collection, we were able to use a single database containing the whole TREC collection for our simulation experiments.

5. For learning from feedback data, we produced statistics of terms and phrases only from the relevant documents of D1 for the queries of Q3.

6. Using this statistics, we ran query expansion of Q3 using the standard Rocchio method (see e.g. [Salton & Buckley 90]). Let $\vec{q}$ the original query vector, $\vec{r}$ the centroid of the relevant document vectors and $\vec{n}$ the centroid of the nonrelevant ones, then we computed a improved document vector according to the formula

   $$\vec{q'} = 8 \cdot \vec{q} + 16 \cdot \vec{r} - 4 \cdot \vec{n}.$$

   For query expansion, only a certain number of terms finally was considered in the computation of $\vec{q'}$. For this purpose, we used "percent expansion" with different parameters for words and phrases. That is, from all terms occurring at least once within a relevant document, only a certain percentage was considered in the final query; for this purpose, terms were ranked according to the number of relevant documents in which they occurred.

7. By using only relevant documents of D1 for expansion statistics we tried to avoid influence of terms from non-retrieved but nonrelevant documents in the occurrence statistic and in the expansion process.

Table 1 shows the experimental results for query set Q3 and document set D2. Obviously, document indexing method "lnc" gives better results than "anc". We also varied the weighting factors for phrases in order to optimize retrieval quality. For query expansion, percent expansion with different parameters was tested and compared with the case of a fixed number of expansion terms.

The parameter combination of the last line in table 1 was used for the official run. For computing the query term weights, the combination of the document sets D1 and D2 was used. Running these queries on the test data set D3 produced the official run `dortR1`.

3.2 Distributed ad-hoc retrieval

For simulating distributed retrieval, we tested the cases 2 and 3 from above. For this purpose, document sets D1 and D2 were split according to the 5 different sources, thus forming 5 separate databases. The documents in each database were indexed with the method "ltc", i.e. the product of logistic term frequency and inverse document frequency, followed by cosine normalization. The idf weight was computed from the local collection frequency only. Then, for each query from the set Q4, the following steps were performed:

1. For each database, the query was indexed with the "ltc" method (using local collection frequency only). With this query, the top ranking 1000 documents were selected from each database.

2. With the 5 results, a new temporary document collection was formed. In a distributed environment, this document base would be constructed by the agent.

3. For testing the case 2 from above, the temporary collection was reindexed with the "lnc" method, and the query with Q4 with "ltc" (based on frequency information from the temporary collection only). Retrieval with this combination produced the results for the official run `dortD2`.

4. For case 3, we assumed that we had collection frequency information for each query term from all servers. Thus, we could apply the "ltc" indexing method for both documents and queries based on frequency information from the whole collection. This way, the official run `dortD1` was produced.

index. method	word exp.	phrase exp.	word wt.	phrase wt.	11p-average
anc	20%	8%	1.0	0.5	0.4051
anc	16%	8%	1.0	0.5	0.4098
anc	14%	7%	1.0	0.5	0.4121
anc	13%	6%	1.0	0.5	0.4125
anc	14%	6%	1.0	0.5	0.4127
anc	300	50	1.0	0.5	0.4130
anc	350	50	1.0	0.5	0.4141
anc	14%	7%	1.0	0.7	0.4147
anc	14%	6%	1.0	0.7	0.4149
anc	14%	5%	1.0	0.7	0.4150
anc	14%	4%	1.0	0.7	0.4155
anc	14%	3%	1.0	0.7	0.4155
anc	13%	6%	1.0	0.7	0.4157
anc	14%	6%	1.0	0.8	0.4146
anc	14%	7%	1.0	1.0	0.4120
lnc	14%	4%	1.0	0.5	0.4275
lnc	350	50	1.0	0.5	0.4275
lnc	13%	4%	1.0	0.5	0.4278
lnc	12%	4%	1.0	0.5	0.4279
lnc	11%	4%	1.0	0.5	0.4283
lnc	14%	7%	1.0	0.7	0.4259
lnc	14%	6%	1.0	0.7	0.4268
lnc	16%	3%	1.0	0.7	0.4274
lnc	15%	4%	1.0	0.7	0.4274
lnc	13%	6%	1.0	0.7	0.4274
lnc	350	50	1.0	0.7	0.4277
lnc	15%	3%	1.0	0.7	0.4278
lnc	14%	5%	1.0	0.7	0.4278
lnc	300	50	1.0	0.7	0.4280
lnc	14%	4%	1.0	0.7	0.4282
lnc	14%	3%	1.0	0.7	0.4284
lnc	13%	4%	1.0	0.7	0.4284
lnc	12%	4%	1.0	0.7	0.4291

Table 1: Results of learning runs for routing (learning with D1, testing with D2)

4 Results

4.1 Routing results

The results of the official runs show that our routing run is clearly above the average of all runs. Thus, we can conclude that our approach — although it is very simple — works and yields good results.

4.2 Ad-hoc results

Both ad-hoc runs produced results of average quality. Considering the constraints underlying our approach, and that no tuning or learning was performed, this is a positive result. Comparing the two approaches, there seems to be no significant difference. Thus, we can conclude that it does not matter whether or not the ranking performed in the agent can use global or local term frequency information only. So the overhead in transmitting this frequency information to the agent can be saved. It seems that the retrieval quality is affected mainly by the indexing functions used in the servers.

5 Conclusions and Outlook

The results of our experiments show that retrieval in distributed environments can be performed without loosing too much in terms of retrieval quality.

Here we only have considered very simple weighting schemes. By using more sophisticated schemes, and especially by applying appropriate learning methods, much better results can be expected.

With the increase of network connectivity and the growing number of document bases accessible, the development of retrieval methods for even large numbers of loosely coupled IR systems is becoming a major research issue.

References

Kahle, B.; Morris, H.; Goldman, J.; Erickson, T.; Curran, J. (1993). Interfaces for Distributed Systems of Information Servers. *Journal of the American Society for Information Science 44(8)*, pages 453–467.

Pfeifer, U.; Fuhr, N.; Huynh, T.T. (1995). Searching Structured Documents with the Enhanced Retrieval Functionality of freeWAIS-sf and SFgate. To appear in: *Proc. of the 3rd World Wide Web*

Conference '95. Also available from http://ls6-www.informatik.uni-dortmund.de/~pfeifer/fwsf.html.

Salton, G.; Buckley, C. (1988). Term Weighting Approaches in Automatic Text Retrieval. *Information Processing and Management 24(5)*, pages 513–523.

Salton, G.; Buckley, C. (1990). Improving Retrieval Performance by Relevance Feedback. *Journal of the American Society for Information Science 41(4)*, pages 288–297.