Comparison of different Collection Fusion Models in Distributed Information Retrieval

Alexander Steidinger
Department of Computer Science
Free University of Berlin

Abstract

Distributed information retrieval comes into play when a user wants to get information from different sources in parallel. One of the challenges of this topic is the Collection Fusion problem: The distinct result lists of the underlying information retrieval systems (IR) have to be fused to give a global relevance-ranked result list according to the user's information need.

In this paper several Collection Fusion models were scrutinized which obey certain restrictions. They should not use more parameters than are given by common collections of distributed digital libraries so that they could be deployed in a real-life distributed retrieval environment.

The selected models are evaluated in a test system with an interface to the IR systems of Managing Gigabytes (MG) and Oracle. The quantitative analysis of the models was performed using test queries, test documents and relevance evaluations from TREC.

Two test runs were executed, one with all four collections stored in MG, another with three collections in MG and one in Oracle. The results of these test runs are discussed with respect to the test setup.

1 Introduction

Digital libraries are becoming a more and more important source of information. Users want to retrieve documents from these libraries ranked by relevance concerning the users' information needs. The general ranking problem of information retrieval is enforced by the distribution of queries to different document collections. In this so-called Collection Fusion problem the local result lists retrieved from different IR systems should be merged to a global one upholding an optimal ranking concerning relevance to the user's information need.
The main problems with Collection Fusion in Distributed Information Retrieval are:

- The sources use different ranking algorithms.
- The ranking algorithms used by the sources are unknown.
- The parameters used with these algorithms (e.g., inverse document frequencies or term frequencies) cannot be obtained from the sources.

The topic of this paper is the evaluation of different Collection Fusion models. Some of them have been described in literature, others were adapted from existing ones. They were selected from a greater set of models with the constraint to be deployable in real-life environments where only some of the model parameters are accessible.

Two test runs show which of the models are suitable to give good global result lists so that they can be integrated into distributed information retrieval of documents of distributed digital libraries.

A more detailed presentation of all the models, the experimental setup and the results can be found in [6]. The restrictions for selecting Collection Fusion models for further consideration come from the properties of real-world IR systems of publishers like Springer, Wiley, Elsevier or ACS (American Chemical Society). They deliver relevance-ranked result lists of known length with a similarity score, no more.

It is possible to gain more information from the IR systems if single-term queries are posed. In that case the document frequencies correspond to the length of the result lists. Of course, not all document frequencies of all terms could be fetched this way, but querying the most important terms would suffice.

It is impossible to get the term frequencies out of the result lists because they contain only bibliographical data of documents like author, article title, journal title, and so on, but not the full text.

2 The Models

As explained in the introduction only those models will be tested here that use no more than the following parameters:

- Relevance-ranked result lists. This is a prerequisite we will not talk about any further.
- Length of the result lists. This parameter is given by the publishers' IR systems mentioned above.
- Document frequencies (DF).
- Similarity scores for every delivered document in the result lists.
- Collection frequencies (CF). This is the number of non-empty result lists.

Models using one or more of the following parameters are discarded:

- Term frequencies like the common TF-IDF-models
- Collection size like in [3]
- Evaluation of the delivered relevance ranked lists for subsequent queries like in [1, 2]
- Number of documents known to be retrieved in advance like in [1].

Three of our six tested models were adapted or directly taken from [4]. The following subsections give a brief overview of the tested models together with the parameters they use. The common prerequisite is that the result lists are ordered by relevance rank. Is is assumed that the models are given result lists from different IR systems.

2.1 Round Robin (RR)

Prerequisite: -
The Round Robin model, also called *Uniform model* or *Interleaving model* simply removes the first elements of the result lists in a round robin fashion and puts them into the global result list. The next round starts with the now first elements of the result lists.

2.2 Round Robin Random (RRR)

Prerequisite: Length of the result lists
We build an array of the length $\sum_{k=1}^{|C|} l_k$ where the sum goes over all the lengths l_k of the result lists. Now we toss a die with $\sum_{k=1}^{|C|} l_k$ faces. An ideal die or an ideal random number generator will hit an index belonging to collection j with the probability

$$P_j = \frac{l_j}{\sum_{k=1}^{|C|} l_k}$$

Now the head of the list hit is removed and the procedure starts again now with the current result lists.

2.3 Round Robin Block (RRB)

Prerequisite: Length of the result lists
In this model we divide the length of all result lists by the length of the shortest non-empty list. The rounded values are the *block lengths* of the lists. Now we remove from each result list a number of elements due to the corresponding block lengths in a round robin fashion and put these blocks into the global result list. The next round starts with the now shorter result lists, but with the same block lengths. When a block length is bigger than the length of an actual list at the end, the whole list is taken into the global result list.

2.4 Raw Scores (RS)

Prerequisite: Similarity scores for each delivered document
This model is also called *Merge Sort model*. It simply takes the elements into the global result list corresponding to their similarity scores. The scores have to be normalized first to be comparable. Later on we will see what problems can occur with such a normalization.

2.5 Normalized Inverse Document Frequency (NIDF)

Prerequisites: Similarity scores and document frequencies
This models uses the inverse document frequency (IDF). Since we don't know if the IR systems of digital libraries use the IDF in their models (and if they do so what concrete expression they take for it) this model uses a very simple notion of inverse document frequency: It's just the reciprocal of the document frequency.

The IDF depends on the very collection of the used IR system. Our task is to weaken the influence of each collection by averaging over the different IDF values. Strictly speaking, we build the following normalized inverse document frequency for query term j:

$$\overline{IDF}_j = \frac{1}{|C|} \sum_{k=1}^{|C|} \frac{1}{DF_j^k}$$

The sum goes over the number of collections, and DF_j^k is the document frequency of term j in collection k. Now a collection weight for collection k is computed for the current query by summing over all the query terms the quotients of the normalized IDF and the IDF of collection k:

$$f_k = \sum_{j=1}^{|Q|} \frac{\overline{IDF}_j}{IDF_j^k} = \sum_{j=1}^{|Q|} \overline{IDF}_j \cdot DF_j^k$$

The factor f_k is now multiplied by the similiarity scores of collection k. The product is then used as a new score after which the documents of the global result list are ordered.

2.6 Collection Weight (CW)

Prerequisites: Similarity scores, document frequencies, and collection frequencies
This model is a combination of two models described in [4]: DF-ICF-collection weighting and weighted scores. This combination has the advantage that the collection scores needed by the second model are delivered by the first one. The first model is as follows: The conditional probability of the occurrence of term r_j in collection c_k is given by

$$P(r_j|c_k) = d_b + (1 - d_b) \cdot T \cdot I$$

where

$$T = d_t + (1 - d_t) \cdot \frac{\log(DF_j^k + 0.5)}{\log(maxDF_k + 1.0)}$$

$$I = \frac{\log(\frac{|C|+0.5}{CF_j})}{\log(|C| + 1.0)}$$

where

DF_j^k	=	number of documents in collection c_k containing term r_j
$maxDF_k$	=	maximum document frequency in collection c_k
CF_j	=	collection frequency = number of collections containing term r_j
d_t	=	minimum value for term frequency of r_j in collection c_k
d_b	=	minimum value for the probability of the occurrence of r_j in collection c_k

The factors T and I stand for *document frequency* and *inverse collection frequency*. The later tests assume a value of 0.4 for d_t and d_b as taken from literature.
Remark: The values of T and I always lie between 0 and 1.

Now the P-values for a query term for the collections are averaged over the number of collections. This average probability is further used as a collection score in the second model:

The weight of a term j of collection k is computed due to the model as

$$w_j^k = 1 + |C| \cdot \frac{s_j^k - \overline{s}_j}{\overline{s}_j}$$

where

$$\begin{aligned} |C| &= \text{number of searched collections} \\ s_j^k &= \text{score of collection } k \text{ for term } j \\ \overline{s}_j &= \text{averaged score for term } j \text{ over all searched collections} \end{aligned}$$

We can now build a new score for document i from collection k by multiplying the similarity score with the sum of the weights w_j^k over all query terms. These new scores are used to build the global result list. *Remark*: w_j^k could become negative. For example, if there are 4 collections then: $w_j^k < 0 \Leftrightarrow s_j < \frac{3}{4}\overline{s_i}$. If this is the case then documents ranked relevant will get at the end of the global list.

3 Experimental Setup

The experimental setup is based on the IRTUM/VIRUM test environment for Distributed Information Retrieval [5].

The TREC collections of the first two TREC disks were divided into four collections of different size. The relevance judgements of TREC were used to adjust the number of documents to be retrieved from each collection to mirror the number of relevant documents in each collection. 450 documents were to be retrieved from collection 1, 150 from collection 2, 220 from collection 3, and 180 documents from collection 4. Further a *single collection* was built containing all the 1000 documents. The four collections were deployed in two IR-Systems: Managing Gigabytes (MG) and Oracle InterMedia Text.

The test queries were the TREC topics No 51-200. The *narrative* part of the topics was used as a list of query terms. Stop words were eliminated and multiple terms were reduced to a single occurrence. No stemming was performed because Oracle InterMedia stems the query termes already and multiple stemming would do no good.

For the latter three Collection Fusion models the similarity scores of the delivered documents have to be normalized. The Oracle scores are normalized a priori and lie in the range from 0 to 100. MG has no normalized scores. This is due to a simplification in the vector space model. We decided to normalize both scores because Oracle never delivered documents with score 100. This has the advantage of both IR systems delivering scores in the whole range up to 100 and no collection is preferred. This linear transformation can of course lead to a distortion of the scores because we do not know how the scores of the IR systems are computed.

Two tests are run: Run 1 only using MG, run 2 using collection 1 in Oracle and the other three in MG. The *Single Collection* serves as a reference collection.

Problem with Oracle: Oracle allows to set a threshold in order to deliver only documents with similarity score above this boundary. We set this value to 30 to get documents in tolerable time. As a consequence less than the 450 documents of collection 1 are retrieved.

For the evaluation of the two test runs we used the TREC relevance judgements. From these a graphical presentation was generated using the precision and recall measures with values taken after every 5 documents retrieved.

4 Results

The evaluation showed that test run 1 yielded better results than test run 2. This was expected because of the different IR systems used in test run 2. Other tests showed that Oracle InterMedia itself generates worse relevance judgements with respect to the TREC evaluations and therefore worse result lists.

In Fig. 1 the precision and recall values of the documents in the global result list for the Round Robin Random model are shown. They are averaged over the 150 topics. For comparison there are also shown the values of the Single Collection. The error bars are the standard deviations of the mean values.

The recall curve of the RRR model lies far below the Single Collection curve, and the intersection with the ordinate after 1000 documents does so, too. The reason for this lies in the above mentioned Oracle problem: Less than the 450 documents are retrieved so there are also less relevant documents retrieved and the recall sinks.

In Fig. 2 you see the data for the NIDF model. The characteristics of the mean precision curve at the first 200 delivered documents is due to the favourization of the Oracle collection by the model: The

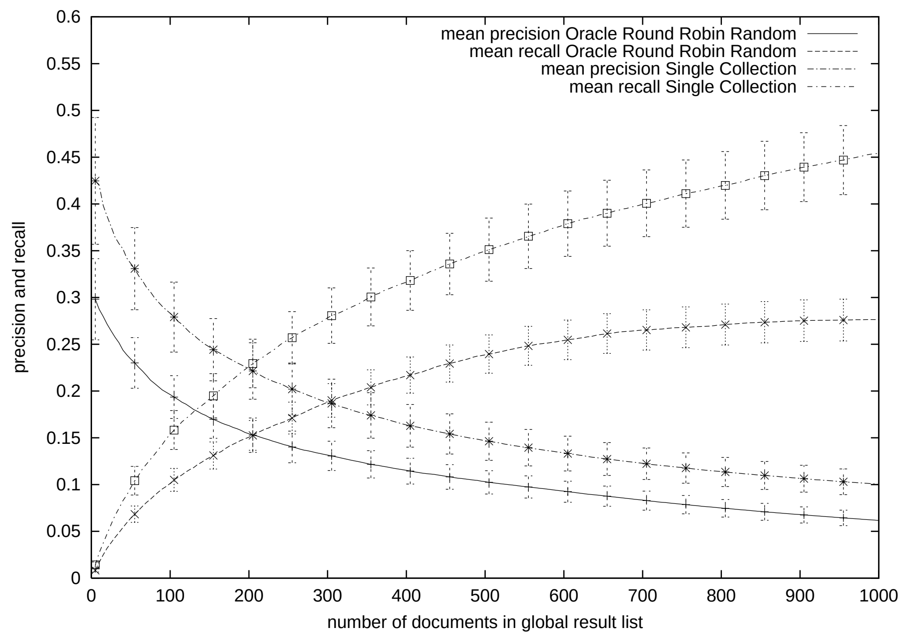

Fig. 1: Round Robin Random: precision and recall averaged over the topics in contrast to the single collection

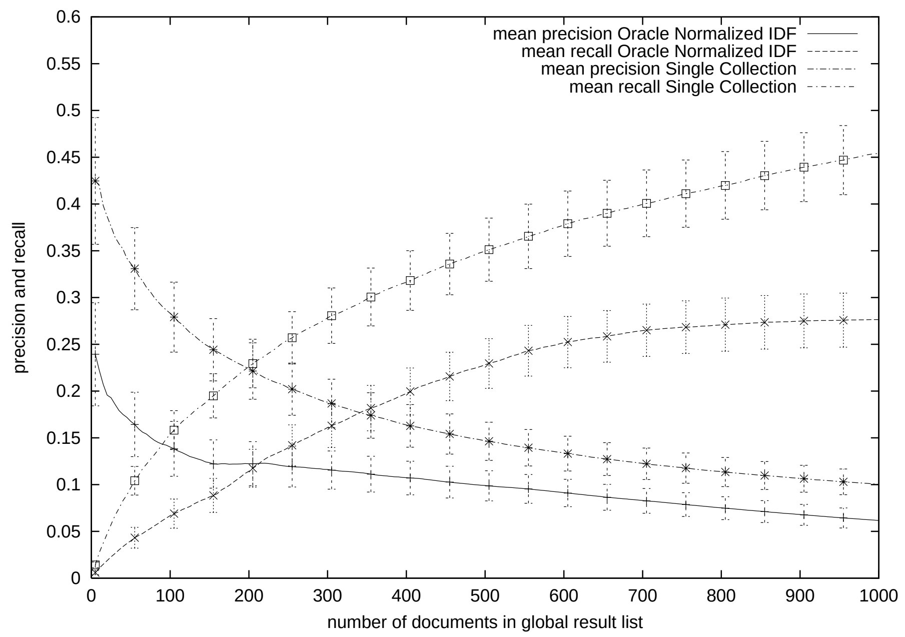

Fig. 2: Normalized IDF: precision and recall averaged over the topics in contrast to the single collection

collection weight for the Oracle collection is much higher than the weights of the other collections so at first most of the Oracle documents are put to the global result list.

The curve also shows that the Oracle ranking is poor, a lot of non-relevant documents are retrieved first and relevant documents came at the end of the result list or are not retrieved at all. When documents of the MG collections start in the global result list the precision even goes up a little bit.

The main results not shown here are:
The Round Robin Random and Round Robin Block models yield the best results in both test runs. Not far behind lies the Raw Score model. The collection weight model shows the same results as the Raw Score model. The reason for this behaviour lies in the parameter settings of the model. The collection weights differ to less to change the to change the order of the retrieved documents in the global result list compared to the order of the Raw Score model.

The by far worst model is the NIDF model. The reason for this is that the differences in the collection weights are big, sometimes they differ by a factor of two. This could be due to the simple use of the reciprocal of the document frequencies. Thus there is a demixing of the collections with the Oracle collection coming first in the global result list.

5 Discussion

Why are the simplest models best in our test environment? They do not depend on quantitative parameters like the document score, they only use the length of the ordered result lists. This makes them insensitive to collection-sensitive parameters. Also the number of documents to be retrieved from each collection was optimized, see the Experimental Setup.

The quality of the results also depends on the quality of the IR systems. When long result lists are retrieved from IR systems with a good ranking algorithm then the RRR and RRB models yield good results, because from the beginning more documents were taken from the long good result lists than from the shorter ones either in bigger blocks in RRB or randomly in RRR.

The quality of RRR should be even better than RRB because there is no fixed order like round robin after which the result lists are processed. The RRB always starts fetching documents from the same result list every round, so in the worst case a bad result list is processed first.

The bad thing is, when there are poor long result lists these models would yield worse results, as is the case with Oracle. But even then the test runs showed they perform better than the Round Robin model.

The results of the more complex algorithms like NIDF or CW could possibly be optimized by tuning of the model parameters. So a difference between the results of the Raw Score model and the Collection Weight model could be reached. One can think of using a more sophisticated expression for the inverse document frequency in NIDF like a logarithm than was applied here.

6 Outlook

The results show that the Round Robin Random and Round Robin Block models can be used in real query interfaces to establish distributed information retrieval in publishing houses even if not very good result lists are retrieved. They outperform the simple Round Robin model.

If heterogeneous collecions were used, a collection selection should be performed for the results presented here work under the assumption of more or less homogeneous collecions.

Implementations of the above models are currently deployed in a prototypical distributed retrieval interface in the Darwin system [7], a digital library for electronic journals at the Free University of Berlin. It allows the users to query different publishers' IR systems in parallel.

References

[1] E.M. Voorhees, N.K. Gupta, B. Johnson-Laird. Learning collection fusion strategies. ACM SIGIR, 172-179, 1995

[2] L. Gravano, H. Garcia-Molina. Merging ranks from heterogeneous internet sources, VLDB Conf., 196-205, 1997

[3] C. Baumgarten. A probabilistic solution to the selection and fusion problem in distributed information retrieval. ACM SIGIR, 246-253, 1999

[4] J.P. Callan, Z. Lu, W.B. Croft. Searching distributed collections with inference networks. ACM SIGIR, 21-28, 1995

[5] A. Grieger: SUVIR - Ein System zur Untersuchung von Verteiltem Information Retrieval, Master Thesis, FU Berlin, 2000, http://www.inf.fu-berlin.de/int/ag-db/index.html

[6] A. Steidinger: Das Collection Fusion Problem bei verteiltem Information Retrieval auf Verlagsservern, Master Thesis, FU Berlin, 2000, http://www.inf.fu-berlin.de/inst/ag-db/index.html

[7] Darwin - Digitale Naturwissenschaftliche Bibliothek der FU Berlin. http://darwin.inf.fu-berlin.de