IR-91

# Applying Inference Networks to Multiple Collection Searching

Zhihong Lu James P. Callan W. Bruce Croft
Computer Science Department, University of Massachusetts
Amherst, MA 01003-4610
{zlu,callan,croft}@cs.umass.edu

**Abstract**

The paper describes how to use inference networks to solve two problems in searching multiple collections: collection selection and result merging. The effectiveness of the approaches is demonstrated with the INQUERY system and 3 gigabyte TREC collections.

## 1 Introduction

As hundreds or even thousands of collections are available on the Internet, the IR community must cope with the problem of searching multiple collection. The simplest approach is to build a single index for all collections. But this approach is practically prohibited by its obvious drawback: it is too slow, because searching such a gigantic index takes a long time to complete. Worse, this search may not complete due to network resource limits, such as limited bandwidth in the case of hundreds of collections. One way to get a quick response is to narrow the search to a portion of the index, which leads to a partition of the index. The natural way to partition is for each partition to correspond to one collection. Heuristics are needed to determine which partitions (collections) are the most useful. Consequently searching multiple collection becomes a three step process:

- select the most useful collections (*collection selection*),
- search each selected collection,
- merge the results from each selected collection (*result merging or data fusion*).

In this paper, we discuss how to use the inference network, a probabilistic approach to information retrieval, to solve the problems in searching multiple collections. In the next section, we briefly introduce the current INQUERY inference network, a traditional use of the model for document retrieval. In Section 3, we discuss how to solve the problem of collection selection using the collection retrieval inference network. In Section 4, we discuss how to merge retrieval results returned from multiple collections. In Section 5, we present the experimental results. In Section 6, we describe related work. In the final section, we summarize the results and discuss the future work.

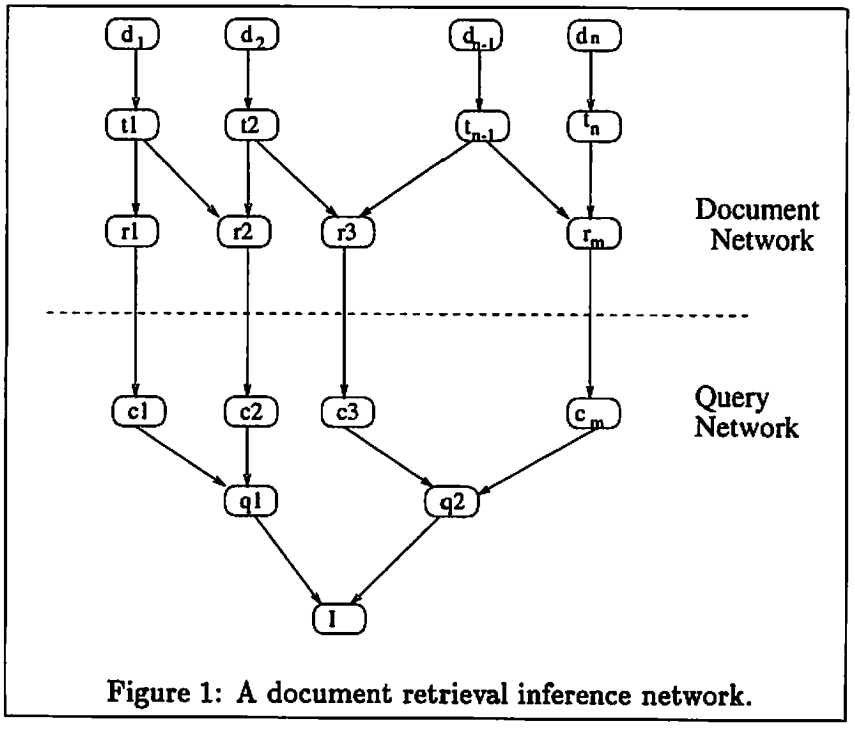


Figure 1: A document retrieval inference network.

# 2 The Current INQUERY Inference Network

## 2.1 Bayesian Inference Networks

A Bayesian inference network[1] is a directed, acyclic dependency graph(DAG) in which nodes represent propositional variables or constants and edges represent dependence relations between propositions. If a proposition represented by node $p$ "causes" or implies the proposition represented by node $q$, we draw a directed edge from $p$ to $q$. The node $q$ contains a ***link matrix*** that specifies $P(q|p)$ for all possible values of the two variables. When a node has multiple parents, the link matrix specifies the dependence on the set of parents and characterizes the dependence relationship between that node and all nodes representing its potential causes. Given a set of prior probabilities for the root of the DAG, the network can be used to compute the probability or degree of belief associated with all remaining nodes.

## 2.2 The current INQUERY Inference Network

The basic document retrieval inference network[1], shown in Figure 1, consists of two component networks: a document network which is built once for a given collection and a query network which is built at query processing time. The document network consists of document nodes($d_i$), text representation nodes($t_i$), and concept representation nodes($r_i$). A document node corresponds to a abstract document, and a text representation node corresponds to a specific text representation of a document. A document may have multiple representations such as figure,

audio , video, etc. But in traditional collections, only the text representation is considered and the relationship between document nodes and text representation nodes is one-to-one. A concept node corresponds to a index unit; it can be a term or a phrase. The link from a $t$ node to an $r$ node means that the document is "about" the particular concept.

The query network consists of the query concept nodes ($c_i$), query nodes ($q_i$) and an information need node ($I$). A query concept corresponds to a basic unit to construct a query. The query concept nodes define the mapping between the concepts used to represent the document collection and the concepts used in the query. In the simplest case, the query concepts are the same as the representation concepts so that each query concept has exactly one parent (this is the case of the current INQUERY). A query node represents an individual query. An information need can be represented by several queries. The weights stored in the $I$ node reflect the importance of each query.

The retrieval inference network is intended to capture all of the significant probabilistic dependencies among the variables represented by nodes in the document and query networks. The retrieval task is to calculate the belief that the information need is met given that a particular document is observed. Documents are ranked and presented to the user ordered by their belief scores. In INQUERY, only one document is considered at a time i.e only one parent of each representation node is *true* at a time so that the link matrices can be replaced by a closed-form expression $P(r_i|d_j)$ that can be calculated quickly.

The belief $P(r_i|d_j)$ is estimated by the following function:

$$ntf = \frac{tf_{ij}}{tf_{ij} + K}$$

$$idf = \frac{log(\frac{N+0.5}{n_i})}{log(N + 1.0)}$$

$$P(r_i|d_j) = \alpha + (1 - \alpha) \cdot ntf \cdot idf$$

where

| | |
|---|---|
| $tf_{ij}$ | is the term frequency of term $r_i$ in document $d_j$, |
| $K$ | $= k \cdot ((1 - b) + b \cdot \frac{doc\_len}{ave\_doc\_len})$, where $k$ and $b$ are constants ($k = 2, b = 0.75$), |
| $N$ | is the number of documents in the collection, |
| $n_i$ | is the number of documents that contain the term $r_i$, |
| $\alpha$ | is a default belief (currently 0.4). |

The INQUERY belief function consists of two components: the default belief and the $tf \cdot idf$ component. The default belief represents a belief estimate when term $t_i$ is not observed in document $d_j$. The $tf \cdot idf$ component gives the additive estimate when the term $t_i$ is observed in document $d_j$.

## 3 Applying the Inference Network to Collection Selection

The task of collection selection is to identify the collections containing the most documents about the information need. Its major part is collection ranking. After the collections are ranked, how many top collections are presented to users can be determined by users (designating a number

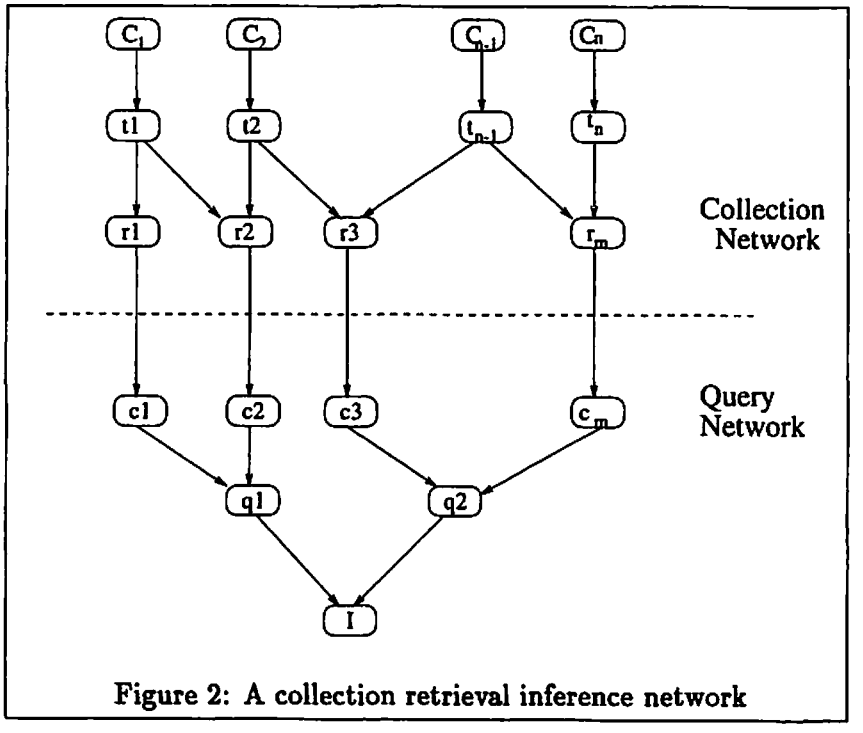


Figure 2: A collection retrieval inference network

before searching) or a clustering algorithm (clustering collections and returning the collections in the top clusters).

Collection ranking can be addressed by a *collection retrieval inference network* or *CORI net* for short (Figure 2) in which the $C$ nodes (replacing $d$ nodes in Figure 1) represent the abstract collections, $t$ nodes correspond to a specific representation of collections and $r$ nodes correspond to the concepts in the collections. The belief stored in the $r$ node should be directly proportional to the number of documents about the information need.

To determine what kind of information is stored in the $r$ nodes, we investigate how to estimate the number of documents about a particular term in a particular collection.

The number of documents about a particular term $r_i$ in the collection $C_m$ can be estimated by:

$$DF(r_i|C_m) = |\{d_j|d_j \in C_m \wedge P(r_i|d_j) > l\}|,$$

where

$l$ is a threshold, if $P(r_i|d_j) > l$, the term $r_i$ is assigned to the document $d_j$ .

Although it is possible to get a better threshold $l$ by learning from the query sets and collections whose relevant information is available, it is reasonable and convenient to set $l$ to the default belief, which is equivalent to the assumption that a term is assigned to a document when it is observed at least once in that document. Then

$$DF(r_i|C_m) = df_{im},$$

where $df_{im}$ is the number of documents that the term $t_i$ is observed in the collection $C_m$. The biggest benefit of setting $l$ as the default belief is that it can be obtained without knowledge about the $tf$ and $idf$ of each term in each document so that expensive computing and storage requirements are avoided.

Let $P(r_i|C_m)$ denote the belief that estimates the importance of a particular collection ($C_m$) for a particular term ($r_i$) based on the number of documents *about* $r_i$ in $C_m$. The collection ranking function is constructed in a way analogous to the ranking function for document retrieval in INQUERY:

$$ndf = \frac{DF(r_i|C_m)}{DF(r_i|C_m) + K} = \frac{df_{im}}{df_{im} + K}$$

$$icf = \frac{log(\frac{|C|}{cf} + 0.5)}{log(|C| + 1.0)}$$

$$P(r_i|C_m) = \alpha + (1 - \alpha) \cdot ndf \cdot icf$$

where

| | | |
|---|---|---|
| $\|C\|$ | is the number of collections, | |
| $cf$ | is the number of collections that contain $r_i$, | |
| $\alpha$ | is a default belief, and | |
| $K$ | $= k \cdot ((1 - b) + b \cdot \frac{cw}{ave\_cw})$, | where $k$ and $b$ are constants, |
| | | $cw$ is the number of words in the collection $C_m$, |
| | | $ave\_cw$ is the mean $cw$. |

The query processing in the CORI net is the same as that in the normal document inference network except that the proximity operators are replaced by Boolean AND, because the location information is not stored in the CORI net due to its high storage and computing costs.

Since the CORI net only keeps partial information on the contents of each collection, it has moderate storage requirements and scales with the growing number of available collections.

# 4 Result Merging

Document retrieval in multiple collections can be modeled as an inference network illustrated as Figure 3: the network is divided into several subnetworks, each of which corresponds to an individual collection. Each collection may have its own stemmer, stopword list and belief function. Belief scores returned from each individual collection need to be merged to form an overall document ranking. Generally these belief scores are collection-specific and incomparable. *Result merging* can not be done by simply sort the belief scores. The incomparability is caused by differences in:

- stemming,
- representation(e.g., word-based, distributed/LSI),
- algorithms used to compute scores, and

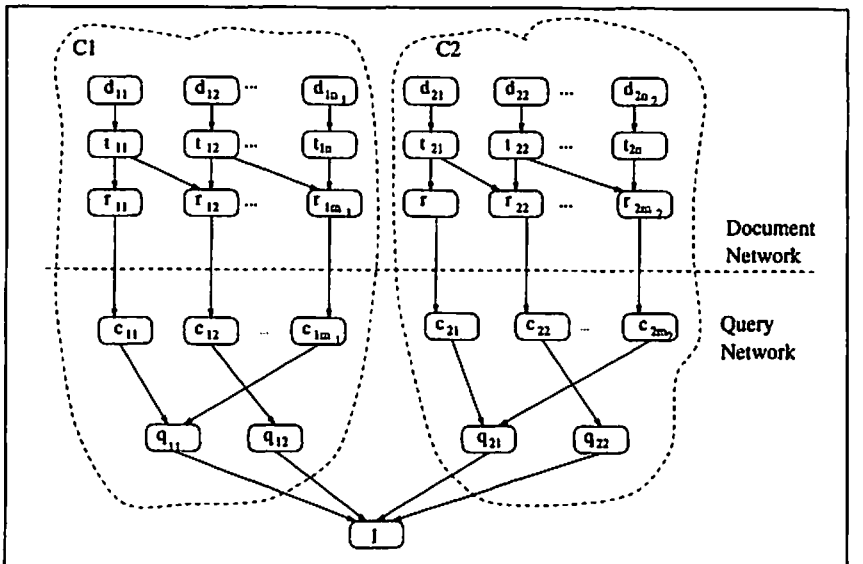


Figure 3: An inference network for searching multiple collection.

- idf and other collection-wide statistics.

The problem of result merging can be solved by two basic approaches:

- normalizing the belief score of each query unit (term, phrase, etc), and
- weighting the overall belief score of a query.

## 4.1 Normalizing the belief score of each query unit

This approach works only when each individual collection uses the same representation, the same queries and the same query processing – the same stemming, the same stopword list and no collection-specific query expansion allowed. Theoretically each collection can use different belief function. But it is really hard to find mapping between two belief functions because the functions are not linear.

Assuming that each individual collection uses the same representation, the same stemming and the same belief function ($P(r_i|d_j)$ in Section 2), normalizing the belief score of each query unit is addressed by the document network in Figure 3: calculating $P(r_{ni}|d_{nj})$ using global document frequency and average document length.

If there are no proximity operators, this normalization can be easily done with the help of the CORI net. The CORI net introduced in Section 3 keeps the collection-wide information about each collection, such as *df* (the number of documents that contain a particular term in a collection) and *cw* (the number of words in a collection). The global idf and average document length can easily be computed using these two statistics. Since the first step of multiple collection searching is collection selection, the global *df* and average document length can be dispersed

with the query to each individual collection. But when there are proximity operators, trouble arises because the number of documents that a proximity operator matches is calculated at the query processing time instead of being precomputed. The global statistics of the proximity operators can only be obtained by querying all collections, which is very expensive in a wide-area network.

## 4.2 Weighting the overall belief score of a query

Weighting the overall belief score of a query is addressed by the query netowrk in Figure 3: using the weights stored in the $I$ node to adjust the belief scores returned from different collections. In principle, this approach can deal with all differences that cause the problem of result merging. The problem is how to determine the weights. Here we only discuss methods which appliy to the case that each individual collection uses the same representation, the same stemming and the same belief function.

By analyzing the belief functions for document ranking ($P(r_i|d_j)$ in Section 2), we find that if two collections have comparable sizes, the scores in the collection containing more relevant documents tends to be smaller, because $\frac{log(\frac{N}{n}+0.5)}{log(N+1.0)}$ decrease as $n$ (the number of documents that contain the term) increases. In addition, the scores of documents in the collection with large average document length tend to be larger than the scores obtained when all collections are treated as a *single* collection. By analyzing the belief function for collection ranking($P(r_i|C_m)$ in Section 3), we find that the collection ranking scores are directly proportional to the number of documents that contains the term, and the $cw$ in the collection ranking function penalizes the collections with large average document length. Therefore the collection ranking scores can be used as complements of document scores such that the combined scores are close to the scores in the paradigm of one large single collection.

Two weighting functions are developed to calculate the weights:

- MF1: $W = 1 + b \cdot log(|C|) \cdot (s_n - \bar{s})/\bar{s}$
  where:
  - $b$ is a constant,
  - $|C|$ is the number of collections,
  - $s_n$ is the score of the collection $C_n$, and
  - $\bar{s}$ is the mean of the collection scores.

- MF2: $W = 1 + b' \cdot log(|C|) \cdot (s_n - min\_s)/(max\_s - min\_s)$
  where:
  - $b'$ is a constant,
  - $|C|$ is the number of collections,
  - $s_n$ is the score of the collection $C_m$,
  - $max\_s$ is the potentially maximal collection score, obtained by assuming that all terms in the query occur in a particular collection, and
  - $min\_s$ is the potentially minimal collection score, obtained by assuming that all terms in the query are missing in a particular collection.

Table 1: The TREC document collections used for experiments. The TREC volume number is shown in parentheses.

| Name | Documents | Words | Megabytes |
|---|---|---|---|
| AP '88 (2) | 79,919 | 21,425,011 | 249 |
| AP '89 (1) | 84,678 | 22,407,342 | 267 |
| AP '90 (3) | 78,321 | 21,555,502 | 249 |
| DOE (1) | 226,087 | 17,201,000 | 193 |
| Fed. Reg. '88 (2) | 19,860 | 20,068,562 | 219 |
| Fed. Reg. '89 (1) | 25,960 | 23,444,637 | 272 |
| Patent (3) | 6,711 | 19,624,651 | 254 |
| SJM '91 (3) | 90,257 | 36,441,456 | 301 |
| WSJ '87 (1) | 46,448 | 11,562,767 | 132 |
| WSJ '88 (1) | 39,904 | 9,738,438 | 109 |
| WSJ '89 (1) | 12,380 | 3,307,151 | 38 |
| WSJ '90 (2) | 21,705 | 6,500,181 | 73 |
| WSJ '91 (2) | 52,652 | 12,418,568 | 146 |
| WSJ '92 (2) | 10,163 | 2,880,247 | 35 |
| Ziff 1 (1) | 75,180 | 20,374,002 | 254 |
| Ziff 2 (2) | 56,920 | 15,637,443 | 184 |
| Ziff 3 (3) | 161,021 | 44,120,132 | 362 |

# 5 Experiments

## 5.1 Test Collections and Queries

The 3 gigabyte TREC collections were used to test the effectiveness of our approaches. The TREC collections are heterogeneous, containing 17 collections from different sources and/or periods of time (Table 1). The 150 queries developed for TREC topics 50-150 and topics 201-250 were used in the experiments. [2] has reported the experimental results about TREC topics 50-150 against TREC volume 1 collections (7 collections), TREC volume 1 + 2 (13 collections) and TREC volume 1 + 2 + 3 (17 collections). Here we report two more sets of experiments:

- 50 queries (201-250 topics) on TREC volume 2+3 (2 gigabytes, 10 collections) which are queries and collections used in the database merging track of TREC4 [3], and
- 50 queries (201-250 topics) on 106 collections obtained by splitting the TREC volumes 2 & 3 according to the source and publication date (Table 2).

## 5.2 Experiments with collection selection

### 5.2.1 Metrics

We define the metrics based the concepts of precision and recall. For a given query and top $n$ collections,

$$R_n = \frac{\sum_{i=1}^{n} r_i}{Rel}$$

$$P_n = \frac{\sum_{i=1}^{n} r_i}{n}$$

where:

$r_i$ is the number of relevant documents in the collection $C_i$,
$C_i$ is the $i$th collection in the rank generated by a ranking algorithm. and
$Rel$ is the total number of relevant documents in the all collections.

$R_n$ indicates the fraction of the relevant documents that can be retrieved when searching the top $n$ collections. $P_n$ indicates the average number of the relevant documents that each collection contains when searching the top $n$ collections.

### 5.2.2 Tuning the parameters $k$ and $b$

Experiments were conducted with values of $k$ ranging from 1 to 300 and with values of $b$ ranging from 0 to 1 (Table 3 and Table 4). The best results were obtained when $k = 200$ and $b = 0.75$.

The effectiveness of our method is also compared with the optimal ranking, shown in Figure 4 and Figure 5. The optimal ranking is obtained by ranking collections according to the number of relevant documents they contain. In our method, the top 20% of collections cover 50% of the relevant documents and the top 40% of collections cover 80% of the relevant documents. In an optimal ranking, the top 10% of collections cover 50% of the relevant documents and the top 20% of collections cover 65% (10 collections) and 80% (106 collections) of the relevant documents. Clearly, our method has rooms for improvement. In addition, our method works a little worse in 106 collections than 10 collections.

## 5.3 Experiments with result merging

The metrics used in this part are recall and precision, which are calculated by the TREC trec2_eval program using the top 1000 documents from the merged ranking. The effectiveness of our two weighted merging approaches were compared with the other two approaches: *normalized scores* (merging based on the normalized document scores from each collection) and *raw scores* (merging based on raw document scores from each collection). The *normalized scores* approach is treated as the baseline, because it is equivalent to the "single collection" paradigm.

Four sets of experiments were conducted:

1. 10 collection merging using normal collection ranking scores.

2. 106 collection merging using normal collection ranking scores.

3. 10 collection merging using collection ranking scores without $icf$.

4. 106 collection merging using collection ranking scores without $icf$.

The experimental results are listed in the Tables 5 – 8. **MFn-all** indicate the experiment which merges the document scores from all collections with the merging function **MFn**. **MFn-sel-m** indicate the experiment which merges the document scores from the selected top collections ($m$ is the average number of collections selected for each query) with the merging function **MFn**. A single pass clustering algorithm [4] is used to select the top collections based on their collection ranking scores.

The experiments show that the raw scores approach is significantly worse than the normalized scores, especially in the case of 106 collections, it causes more than 30% losses in almost all levels of recall. This result confirms the previous research suggesting that incomparable document scores can mislead results.

Our two weighted merging approaches are almost as effective as the normalized scores in low recall and cause losses in high recall, but the losses are significantly less than those in the raw scores approach. $MF2$ produces better results than $MF1$ in small number of collections with larger sizes, but worse in large number of collections with smaller sizes. The differences of these two functions are not significant. The results on **MFn-sel-m** show that if we cutoff 50% of collections, the precision at low recall is at least relatively unaffected and even get improved in the case of 106 collections above the 30 documents. This suggests that collection selection can lead to efficient searching without losing too much effectiveness.

A very interesting phenomenon is that the results using no-icf collection ranking scores are much better than those using normal collection ranking scores. This suggests that the $icf$ part hurts the results when the number of collections is relatively small. It is not clear if $icf$ will hurt a large number of collections.

# 6 Related Work

The problem of collection selection is a specific instance of the more general resource discovery problem [5]. The approaches to solving the resource discovery problem fall into two groups: browsing and searching [6]. Browsing allows users to follow pre-defined links between data items to find resources. Because the links are maintained manually, they are always out of date or non-existent. In addition if you have no idea where to find information, finding information following links can be very frustrating. Searching allows users to query a collection of "meta-information" about available collections. This approach is used in increasingly many Internet resource discovery systems (e.g. [7], [8], [9], etc). The meta information typically provides some sort of summary of the contents of each collection. Some systems keep human-generated summaries of each collection[7][8]. Human-generated summaries are often out of date. Some systems only index a small portion of each document such as titles [9]. This approach sacrifices important information of the contents of each collection. Some systems index all terms that occur in the individual collections. GLOSS [10] and our approach belong to this category.

GLOSS has two versions. In its boolean version[11], GLOSS keeps the number of documents in a collection containing a particular term and estimates the number of potentially relevant

documents of a collection $C$ for a Boolean AND query $Q$ as $|C| \cdot \Pi_{t \epsilon Q} \frac{df_t}{|C|}$, where $t$ is a term in $Q$, $df_t$ is the number of documents in $C$ containing $t$, and $|C|$ is the number of documents in $C$. In its vector space version[12], GLOSS keeps the number of documents in a collection $C$ containing a particular term $t$ ($df_t$) and the sum of weights of the term $t$ over all documents in the collection $C(w_t)$. The collections are ranked by the estimators which are based the sum of the similarity of each query term. The similarity is calculated as: sort the query terms in the increasing order of $df_t$, then the similarity of the $p$-th term is equal to $\sum_{j=p,\ldots,n} q_j \cdot \frac{w_j}{df_j}$. In the high-correlation scenario, the estimator is equal to $\sum_{j=1,\ldots,n}(w_j - w_{j-1}) \cdot sim_j)$. In the disjoint scenario, the estimator is equal to $\sum_{j=1,\ldots,n}(w_j \cdot sim_j)$.

The differences between our approach and GLOSS lie in not only different models used for retrieval but also that we used a single model to cope with retrievals from the meta-collection and from the normal document collections, while GLOSS used different processing for these two kinds of retrievals.

Recently, a decision-theoretic approach is proposed to solve collection selection problem[13], which uses expected recall-precision curve, expected number of relevant documents and cost factors for query processing and document delivery to make decision. But no performance studies are reported. In addition some approaches incorporate AI technology such as semantic knowledge to locate collections[14].

Our approach to result merging assumes that the collection-wide information is accessible, i.e. we can use the particular statistics about collections in the merging function besides the ranked list of documents in response to a query. Voorhees, et al, proposes another approach which assume that the only information the merging algorithm can obtain from a collection is a ranked list of documents in response to a query[15].

# 7 Conclusion

This paper describes how to use inference networks to solve problems in searching multiple collection. The effectiveness of the approaches is demonstrated in experiments with the INQUERY information retrieval system and the TREC set of document collections. The experimental results are encouraging because simple methods were quite effective with different numbers of collections that varied widely in size and content.

But in our work so far, all of the collections used the same representation, stemming algorithm, stopword list, and query processing techniques. When these differ, as they will in collections on wide-area networks, the problems, especially the problem of result merging, become more difficult. Although learning weights stored in the information need node can be expected to "flatten" all kinds of differences, it is not clear how to do it. In addition, we have not addressed how to perform relevance feedback in the environment of multiple collections. Although we believe our approach is scalable with growing number of available collections, more experiments need to be done.

## Acknowledgements

This material is based on work supported in part by the National Science Foundation, Library of Congress and Department of Commerce under cooperative agreement number EEC-9209623.

## References

[1] Howard Turtle and W. Bruce Croft. Evaluation of an Inference Network-Based Retrieval. *ACM Transactions on Information Systems, Vol. 9, No. 3, July 1993, Pages 187-222.*

[2] James P. Callan, Zhihong Lu and W. Bruce Croft. Searching Distributed Collections with Inference Networks. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval,* pages 21-28, Seattle, July 1995. Association for Computing Machinery.

[3] James Allan, Lisa Ballesteros, James P. Callan, W. Bruce Croft and Zhihong Lu "Recent Experiments with INQUERY", *Proceedings of the 4th Text REtrieval Conference (TREC-4),* Nov. 1995. National Institute of Standards and Technology.

[4] Edie Rasmussen. Clustering Algorithms. In Frakes and Baeza-Yates, editors, *Information Retrieval: Data Structures and Algorithms.* Chapter 16, pages 419-442. Prentice Hall, 1992.

[5] Katia Obraczka, Peter B. Danzig, and Shih-Hao Li. Internet Resource Discovery Services. *IEEE Computer, September 1993.*

[6] Michael F. Schwartz. Searching as a Primary Internet Discovery Paradigm. *Internet Society News 2(2), Summer 1993.*

[7] WAIS 2.0: Technical Description *http://www.wais.com/newhomepages/techtalk.html*

[8] ALIWEB *http://www.nexor.co.uk/public/aliweb/doc/introduction.html*

[9] World-wide Web Worm *http://wwwmcb.cs.colorado.edu/home/mcbryan/WWWW.html*

[10] Glossary of Servers *http://gloss.stanford.edu*

[11] Luis Gravano and Hector Garcia-Molina. Precision and Recall of GLOSS Estimator for Database Discovery. *Stanford Univerdity Technical Note Number STAN-CS-TN-94-10.*

[12] Luis Gravano and Hector Garcia-Molina. Generalizing GLOSS to Vector-Space Databases and Broker Hierarchies. *Proceedings of the 21st VLDB Conference,* Zurich, Switchland 1995.

[13] Norbert Fuhr. A Decision-Theoretic Approach to Database Selection in Networked IR. *Workshop on Distributed IR, Germany, 1996.*

[14] NetSerf: Using Semantic Knowledge to Find Internet Information Archives. *Proceedings of the 18th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 4-11, Seattle, July 1995. Association for Computing Machinery.

[15] Ellen M. Voorhees. Siemens TREC-4 Report: Further Experiments with Database Merging. *Proceedings of the 4th Text REtrieval Conference (TREC-4)*, Nov. 1995. National Institute of Standards and Technology.

Table 2: The 106 small collections used for experiments

| Name | Documents | Words | Megabytes | Name | Documents | Words | Megabytes |
|---|---|---|---|---|---|---|---|
| AP '02/88 | 4198 | 1,131,354 | 24.02 | Patent 1 | 560 | 1,121,011 | 14.50 |
| AP '03/88 | 8196 | 2,185,743 | 20.98 | Patent 2 | 560 | 1,384,363 | 17.72 |
| AP '04/88 | 7790 | 2,089,128 | 21.79 | Patent 3 | 560 | 1,647,816 | 22.52 |
| AP '05/88 | 8302 | 2,196,009 | 13.29 | Patent 4 | 560 | 2,187,990 | 27.20 |
| AP '06/88 | 8081 | 2,147,904 | 25.61 | Patent 5 | 560 | 1,334,895 | 17.34 |
| AP '07/88 | 7301 | 1,944,938 | 24.45 | Patent 6 | 560 | 1,457,301 | 18.94 |
| AP '08/88 | 7496 | 1,965,891 | 25.74 | Patent 7 | 560 | 1,951,090 | 25.88 |
| AP '09/88 | 6822 | 1,860,132 | 25.29 | Patent 8 | 560 | 1,960,078 | 25.34 |
| AP '10/88 | 7794 | 2,047,714 | 22.84 | Patent 9 | 560 | 1,129,486 | 14.63 |
| AP '11/88 | 6928 | 1,789,060 | 23.02 | Patent 10 | 560 | 1,456,725 | 18.77 |
| AP '12/88 | 7011 | 1,862,026 | 21.75 | Patent 11 | 560 | 1,985,994 | 26.67 |
| AP '01/90 | 6955 | 1,896,444 | 22.09 | Patent 12 | 560 | 1,977,166 | 24.86 |
| AP '02/90 | 6521 | 1,790,725 | 21.60 | WSJ '04/90 | 3569 | 1,031,459 | 11.71 |
| AP '03/90 | 6867 | 1,895,808 | 19.77 | WSJ '05/90 | 3643 | 1,075,995 | 12.19 |
| AP '04/90 | 6594 | 1,801,585 | 19.36 | WSJ '06/90 | 3439 | 1,031,572 | 11.66 |
| AP '05/90 | 6972 | 1,879,613 | 20.97 | WSJ '07/90 | 3356 | 1,010,925 | 11.42 |
| AP '06/90 | 6673 | 1,795,132 | 22.15 | WSJ '08/90 | 3523 | 1,080,297 | 12.17 |
| AP '07/90 | 6645 | 1,783,995 | 21.02 | WSJ '09/90 | 3244 | 972,240 | 11.03 |
| AP '08/90 | 6218 | 1,714,105 | 21.96 | WSJ '12/90 | 931 | 263,581 | 3.02 |
| AP '09/90 | 5721 | 1,575,057 | 21.06 | WSJ '01/91 | 3594 | 1,075,323 | 12.24 |
| AP '10/90 | 6816 | 1,866,108 | 20.94 | WSJ '02/91 | 3143 | 934,107 | 10.64 |
| AP '11/90 | 6310 | 1,691,537 | 19.81 | WSJ '03/91 | 3525 | 1,006,483 | 11.51 |
| AP '12/90 | 6029 | 1,660,525 | 18.31 | WSJ '04/91 | 3969 | 1,126,537 | 13.20 |
| Fed. Reg. '01/88 | 2329 | 1,803,698 | 19.76 | WSJ '05/91 | 3663 | 1,062,965 | 12.80 |
| Fed. Reg. '02/88 | 2103 | 1,804,778 | 24.65 | WSJ '06/91 | 3500 | 972,884 | 11.65 |
| Fed. Reg. '05/88 | 1684 | 1,991,153 | 23.36 | WSJ '07/91 | 3709 | 1,073,732 | 12.83 |
| Fed. Reg. '06/88 | 1489 | 1,608,745 | 8.74 | WSJ '08/91 | 3309 | 998,711 | 11.96 |
| Fed. Reg. '07/88 | 2142 | 2,075,110 | 19.61 | WSJ '09/91 | 3306 | 957,309 | 11.44 |
| Fed. Reg. '08/88 | 2917 | 3,154,918 | 21.50 | WSJ '10/91 | 4055 | 1,185,133 | 14.25 |
| Fed. Reg. '09/88 | 2437 | 2,433,860 | 17.45 | WSJ '11/91 | 3481 | 1,009,305 | 12.15 |
| Fed. Reg. '10/88 | 1761 | 2,239,603 | 22.92 | WSJ '12/91 | 3198 | 938,938 | 11.26 |
| Fed. Reg. '11/88 | 2133 | 2,139,415 | 34.78 | WSJ '01/92 | 3788 | 1,095,584 | 13.19 |
| Fed. Reg. '12/88 | 865 | 799,735 | 26.51 | WSJ '02/92 | 3437 | 943,365 | 11.45 |
| SJM '01/91 | 7833 | 1,978,510 | 27.15 | WSJ '03/92 | 2938 | 827,919 | 9.89 |
| SJM '02/91 | 7080 | 1,749,601 | 24.24 | Ziff '07/89 | 5614 | 1,587,169 | 17.90 |
| SJM '03/91 | 7649 | 1,871,014 | 26.04 | Ziff '08/89 | 5579 | 1,632,490 | 18.24 |
| SJM '04/91 | 7601 | 1,813,333 | 25.18 | Ziff '09/89 | 5803 | 1,660,332 | 18.73 |
| SJM '05/91 | 7746 | 1,832,807 | 25.43 | Ziff '10/89 | 6363 | 1,877,615 | 21.08 |
| SJM '06/91 | 7478 | 1,796,817 | 24.82 | Ziff '11/89 | 4855 | 1,524,419 | 17.11 |
| SJM '07/91 | 7758 | 1,797,531 | 25.11 | Ziff '12/89 | 180 | 91,409 | 1.00 |
| SJM '08/91 | 7620 | 1,794,627 | 24.95 | Ziff '01/91 | 12083 | 2,099,223 | 27.45 |
| SJM '09/91 | 7574 | 1,861,253 | 25.50 | Ziff '02/91 | 11775 | 1,968,896 | 26.05 |
| SJM '10/91 | 7906 | 1,893,012 | 26.18 | Ziff '03/91 | 12350 | 2,102,378 | 27.60 |
| SJM '11/91 | 7284 | 1,739,419 | 23.99 | Ziff '04/91 | 12910 | 2,320,771 | 29.96 |
| SJM '12/91 | 6723 | 1,596,137 | 22.18 | Ziff '05/91 | 12515 | 2,282,758 | 29.48 |
| Ziff '01/89 | 5042 | 1,326,666 | 15.16 | Ziff '06/91 | 12485 | 2,264,248 | 29.21 |
| Ziff '02/89 | 4633 | 1,342,725 | 15.03 | Ziff '07/91 | 12795 | 2,255,295 | 29.26 |
| Ziff '03/89 | 4563 | 1,334,304 | 14.96 | Ziff '08/91 | 12050 | 2,257,338 | 28.80 |
| Ziff '04/89 | 5020 | 1,617,165 | 17.98 | Ziff '09/91 | 13116 | 2,412,764 | 31.14 |
| Ziff '05/89 | 5437 | 1,713,169 | 18.95 | Ziff '10/91 | 12819 | 2,479,634 | 31.31 |
| Ziff '06/89 | 5091 | 1,596,044 | 17.80 | Ziff '11/91 | 11379 | 1,918,375 | 25.32 |
| Ziff '88 | 94 | 8,198 | 0.12 | Ziff '12/91 | 20675 | 850,198 | 18.04 |
| Ziff '90 | 2674 | 1,624,844 | 17.71 | Ziff '01/92 | 41 | 4,358 | 0.06 |

Table 3: The effects of varying $K$ (10 collections in TREC volumes 2 & 3, topics 201-250). In (a) $b = 0$ and $k$ is varied. In (b) $k = 200$ and $b$ is varied.

(a) $b$ = 0 and $k$ is varied

$R_n$

| n | k=1 | k=10 | k=50 | k=100 | k=150 | k=200 | k=250 | k=300 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.299 | 0.309 (+3.2) | 0.323 (+7.9) | 0.318 (+6.3) | 0.303 (+1.2) | 0.305 (+1.9) | 0.305 (+2.0) | 0.290 (-3.0) |
| 2 | 0.494 | 0.527 (+6.8) | 0.541 (+9.5) | 0.544 (+10.2) | 0.546 (+10.7) | 0.554 (+12.2) | 0.555 (+12.4) | 0.553 (+12.0) |
| 3 | 0.702 | 0.744 (+6.1) | 0.754 (+7.5) | 0.752 (+7.1) | 0.749 (+6.8) | 0.753 (+7.4) | 0.745 (+6.1) | 0.745 (+6.1) |
| 4 | 0.804 | 0.842 (+4.7) | 0.841 (+4.5) | 0.838 (+4.2) | 0.838 (+4.2) | 0.841 (+4.5) | 0.841 (+4.6) | 0.841 (+4.6) |
| 5 | 0.860 | 0.892 (+3.6) | 0.888 (+3.2) | 0.884 (+2.7) | 0.878 (+2.0) | 0.878 (+2.1) | 0.878 (+2.0) | 0.875 (+1.7) |
| 6 | 0.915 | 0.925 (+1.1) | 0.917 (+0.3) | 0.909 (-0.6) | 0.911 (-0.5) | 0.913 (-0.2) | 0.912 (-0.3) | 0.908 (-0.7) |
| 7 | 0.942 | 0.953 (+1.1) | 0.945 (+0.3) | 0.946 (+0.4) | 0.946 (+0.4) | 0.945 (+0.3) | 0.945 (+0.3) | 0.944 (+0.2) |
| 8 | 0.967 | 0.976 (+0.9) | 0.969 (+0.2) | 0.969 (+0.2) | 0.965 (-0.3) | 0.965 (-0.2) | 0.964 (-0.4) | 0.957 (-1.0) |
| 9 | 0.993 | 0.992 (-0.0) | 0.991 (-0.1) | 0.989 (-0.4) | 0.989 (-0.4) | 0.986 (-0.6) | 0.986 (-0.6) | 0.986 (-0.6) |
| 10 | 1.000 | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) |

$P_n$

| n | k=1 | k=10 | k=50 | k=100 | k=150 | k=200 | k=250 | k=300 |
|---|---|---|---|---|---|---|---|---|
| 1 | 39.39 | 41.20 (+4.6) | 42.59 (+8.1) | 43.18 (+9.6) | 42.63 (+8.2) | 42.73 (+8.5) | 42.80 (+8.7) | 41.92 (+6.4) |
| 2 | 33.02 | 35.66 (+8.0) | 35.50 (+7.5) | 35.90 (+8.7) | 35.67 (+8.0) | 35.91 (+8.7) | 36.21 (+9.7) | 36.04 (+9.1) |
| 3 | 31.35 | 32.69 (+4.3) | 33.48 (+6.8) | 33.46 (+6.7) | 33.12 (+5.7) | 33.16 (+5.8) | 32.90 (+4.9) | 32.90 (+4.9) |
| 4 | 26.92 | 28.26 (+5.0) | 28.34 (+5.2) | 28.30 (+5.1) | 28.30 (+5.1) | 28.32 (+5.2) | 28.31 (+5.1) | 28.11 (+4.4) |
| 5 | 22.84 | 23.99 (+5.0) | 23.94 (+4.8) | 23.80 (+4.2) | 23.74 (+3.9) | 23.75 (+4.0) | 23.74 (+3.9) | 23.67 (+3.6) |
| 6 | 20.49 | 20.69 (+1.0) | 20.62 (+0.6) | 20.44 (-0.2) | 20.41 (-0.4) | 20.56 (+0.3) | 20.55 (+0.3) | 20.42 (-0.3) |
| 7 | 18.00 | 18.13 (+0.7) | 18.02 (+0.1) | 18.14 (+0.8) | 18.14 (+0.8) | 18.11 (+0.6) | 18.10 (+0.6) | 18.10 (+0.6) |
| 8 | 16.20 | 16.26 (+0.3) | 16.19 (-0.1) | 16.19 (-0.1) | 16.11 (-0.6) | 16.11 (-0.6) | 16.04 (-1.0) | 16.02 (-1.1) |
| 9 | 14.70 | 14.69 (-0.1) | 14.67 (-0.2) | 14.66 (-0.3) | 14.66 (-0.3) | 14.63 (-0.5) | 14.63 (-0.5) | 14.63 (-0.5) |
| 10 | 13.27 | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) |

(b) $k$ = 200 and $b$ is varied

$R_n$

| n | b=0 | b=0.25 | b=0.5 | b=0.75 | b=1.0 |
|---|---|---|---|---|---|
| 1 | 0.305 | 0.305 (+0.1) | 0.313 (+2.9) | 0.313 (+2.6) | 0.260 (-14.5) |
| 2 | 0.554 | 0.547 (-1.3) | 0.548 (-1.0) | 0.559 (+1.0) | 0.491 (-11.4) |
| 3 | 0.753 | 0.733 (-2.7) | 0.739 (-1.9) | 0.755 (+0.2) | 0.634 (-15.8) |
| 4 | 0.841 | 0.842 (+0.1) | 0.839 (-0.2) | 0.842 (+0.2) | 0.755 (-10.2) |
| 5 | 0.878 | 0.908 (+3.4) | 0.900 (+2.5) | 0.891 (+1.5) | 0.851 (-3.0) |
| 6 | 0.913 | 0.936 (+2.5) | 0.926 (+1.5) | 0.916 (+0.4) | 0.923 (+1.1) |
| 7 | 0.945 | 0.953 (+0.8) | 0.944 (-0.1) | 0.945 (-0.0) | 0.949 (+0.4) |
| 8 | 0.965 | 0.973 (+0.8) | 0.964 (-0.1) | 0.959 (-0.6) | 0.971 (+0.6) |
| 9 | 0.986 | 0.989 (+0.3) | 0.993 (+0.7) | 0.989 (+0.2) | 0.989 (+0.3) |
| 10 | 1.000 | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) | 1.000 (+0.0) |

$P_n$

| n | b=0 | b=0.25 | b=0.5 | b=0.75 | b=1.0 |
|---|---|---|---|---|---|
| 1 | 42.73 | 42.78 (+0.1) | 43.43 (+1.6) | 43.43 (+1.6) | 36.69 (-14.1) |
| 2 | 35.91 | 36.00 (+0.3) | 35.84 (-0.2) | 36.39 (+1.3) | 33.02 (-8.0) |
| 3 | 33.16 | 32.40 (-2.3) | 32.90 (-0.8) | 33.09 (-0.2) | 27.92 (-15.8) |
| 4 | 28.32 | 27.92 (-1.4) | 28.16 (-0.6) | 28.41 (+0.3) | 25.23 (-10.9) |
| 5 | 23.75 | 24.22 (+2.0) | 24.15 (+1.7) | 23.98 (+1.0) | 22.59 (-4.9) |
| 6 | 20.56 | 20.81 (+1.2) | 20.74 (+0.9) | 20.52 (-0.2) | 20.43 (-0.6) |
| 7 | 18.11 | 18.18 (+0.4) | 18.06 (-0.3) | 18.14 (+0.2) | 18.12 (+0.0) |
| 8 | 16.11 | 16.32 (+1.3) | 16.17 (+0.4) | 16.11 (-0.0) | 16.18 (+0.4) |
| 9 | 14.63 | 14.68 (+0.3) | 14.69 (+0.4) | 14.66 (+0.2) | 14.68 (+0.3) |
| 10 | 13.27 | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) | 13.27 (+0.0) |

Table 4: The effects of varying $K$ (106 small collections, topics 201-250). In a) $b = 0$ and $k$ is varied. In b) $k = 200$ and $b$ is varied.

(a) $b$ = 0 and $k$ is varied

$R_n$

| n | k=1 | k=10 | k=50 | k=100 | k=150 | k=200 | k=250 | k=300 |
|---|---|---|---|---|---|---|---|---|
| 1 | 0.0407 | 0.0377 (-7.3) | 0.0382 (-6.2) | 0.0366 (-10.1) | 0.0427 (+4.9) | 0.0442 (+8.4) | 0.0371 (-8.9) | 0.0380 (-6.6) |
| 2 | 0.0728 | 0.0707 (-2.8) | 0.0712 (-2.2) | 0.0696 (-4.3) | 0.0698 (-4.1) | 0.0670 (-7.9) | 0.0656 (-9.9) | 0.0694 (-4.6) |
| 3 | 0.0983 | 0.0990 (+0.7) | 0.0991 (+0.8) | 0.1015 (+3.3) | 0.1017 (+3.5) | 0.1046 (+6.5) | 0.1028 (+4.6) | 0.1027 (+4.5) |
| 4 | 0.1291 | 0.1322 (+2.4) | 0.1336 (+3.5) | 0.1369 (+6.0) | 0.1431 (+10.9) | 0.1438 (+11.4) | 0.1428 (+10.6) | 0.1397 (+8.2) |
| 5 | 0.1572 | 0.1639 (+4.3) | 0.1680 (+6.9) | 0.1698 (+8.1) | 0.1765 (+12.3) | 0.1748 (+11.2) | 0.1741 (+10.8) | 0.1728 (+9.9) |
| 10 | 0.2968 | 0.3088 (+4.1) | 0.3073 (+3.5) | 0.3069 (+3.4) | 0.3032 (+2.2) | 0.2982 (+0.5) | 0.2981 (+0.4) | 0.2962 (-0.2) |
| 20 | 0.5164 | 0.5344 (+3.5) | 0.5376 (+4.1) | 0.5341 (+3.4) | 0.5241 (+1.5) | 0.5236 (+1.4) | 0.5237 (+1.4) | 0.5248 (+1.6) |
| 30 | 0.6759 | 0.6992 (+3.5) | 0.7105 (+5.1) | 0.7073 (+4.6) | 0.7037 (+4.1) | 0.7050 (+4.3) | 0.7012 (+3.7) | 0.6995 (+3.5) |
| 40 | 0.8007 | 0.8050 (+0.5) | 0.8121 (+1.4) | 0.8108 (+1.3) | 0.8075 (+0.8) | 0.8071 (+0.8) | 0.8071 (+0.8) | 0.8064 (+0.7) |
| 60 | 0.9204 | 0.9350 (+1.6) | 0.9370 (+1.8) | 0.9387 (+2.0) | 0.9328 (+1.3) | 0.9315 (+1.2) | 0.9285 (+0.9) | 0.9281 (+0.8) |
| 80 | 0.9712 | 0.9838 (+1.3) | 0.9844 (+1.4) | 0.9835 (+1.3) | 0.9825 (+1.2) | 0.9821 (+1.1) | 0.9821 (+1.1) | 0.9815 (+1.1) |
| 100 | 0.9990 | 0.9998 (+0.1) | 0.9998 (+0.1) | 0.9997 (+0.1) | 0.9997 (+0.1) | 0.9996 (+0.1) | 0.9996 (+0.1) | 0.9996 (+0.1) |

$P_n$

| n | k=1 | k=10 | k=50 | k=100 | k=150 | k=200 | k=250 | k=300 |
|---|---|---|---|---|---|---|---|---|
| 1 | 5.286 | 4.857 (-8.1) | 4.551 (-13.9) | 4.469 (-15.4) | 5.082 (-3.9) | 5.082 (-3.9) | 4.673 (-11.6) | 4.735 (-10.4) |
| 2 | 4.806 | 4.571 (-4.9) | 4.520 (-5.9) | 4.643 (-3.4) | 4.673 (-2.8) | 4.408 (-8.3) | 4.357 (-9.3) | 4.459 (-7.2) |
| 3 | 4.510 | 4.429 (-1.8) | 4.415 (-2.1) | 4.571 (+1.4) | 4.599 (+2.0) | 4.707 (+4.4) | 4.497 (-0.3) | 4.497 (-0.3) |
| 4 | 4.342 | 4.342 (+0.0) | 4.546 (+4.7) | 4.612 (+6.2) | 4.709 (+8.5) | 4.776 (+10.0) | 4.709 (+8.5) | 4.714 (+8.6) |
| 5 | 4.196 | 4.224 (+0.7) | 4.490 (+7.0) | 4.482 (+6.8) | 4.624 (+10.2) | 4.559 (+8.7) | 4.563 (+8.8) | 4.543 (+8.3) |
| 10 | 3.935 | 4.116 (+4.6) | 4.190 (+6.5) | 4.259 (+8.2) | 4.220 (+7.3) | 4.080 (+3.7) | 4.092 (+4.0) | 4.114 (+4.6) |
| 20 | 3.405 | 3.537 (+3.9) | 3.541 (+4.0) | 3.551 (+4.3) | 3.520 (+3.4) | 3.502 (+2.8) | 3.492 (+2.5) | 3.504 (+2.9) |
| 30 | 2.963 | 3.063 (+3.4) | 3.109 (+4.9) | 3.099 (+4.6) | 3.086 (+4.2) | 3.100 (+4.6) | 3.087 (+4.2) | 3.082 (+4.0) |
| 40 | 2.630 | 2.673 (+1.6) | 2.696 (+2.5) | 2.686 (+2.1) | 2.683 (+2.0) | 2.680 (+1.9) | 2.680 (+1.9) | 2.680 (+1.9) |
| 60 | 2.062 | 2.086 (+1.2) | 2.086 (+1.2) | 2.087 (+1.2) | 2.083 (+1.0) | 2.080 (+0.9) | 2.074 (+0.6) | 2.073 (+0.5) |
| 80 | 1.627 | 1.637 (+0.6) | 1.640 (+0.8) | 1.639 (+0.7) | 1.636 (+0.6) | 1.635 (+0.5) | 1.635 (+0.5) | 1.635 (+0.5) |
| 100 | 1.326 | 1.327 (+0.0) | 1.327 (+0.0) | 1.326 (+0.0) | 1.326 (+0.0) | 1.326 (-0.0) | 1.326 (-0.0) | 1.326 (-0.0) |

(b) $k$ = 200 and $b$ is varied

$R_n$

| n | b=0 | b=0.25 | b=0.5 | b=0.75 | b=1.0 |
|---|---|---|---|---|---|
| 1 | 0.0442 | 0.0370 (-16.1) | 0.0381 (-13.8) | 0.0401 (-9.2) | 0.0363 (-17.8) |
| 2 | 0.0670 | 0.0618 (-7.7) | 0.0661 (-1.4) | 0.0660 (-1.5) | 0.0589 (-12.1) |
| 3 | 0.1046 | 0.0862 (-17.6) | 0.0964 (-7.9) | 0.1004 (-4.0) | 0.0827 (-21.0) |
| 4 | 0.1438 | 0.1252 (-12.9) | 0.1354 (-5.8) | 0.1369 (-4.8) | 0.1134 (-21.1) |
| 5 | 0.1748 | 0.1529 (-12.5) | 0.1679 (-3.9) | 0.1718 (-1.7) | 0.1441 (-17.5) |
| 10 | 0.2982 | 0.2977 (-0.2) | 0.3076 (+3.2) | 0.3129 (+4.9) | 0.2786 (-6.6) |
| 20 | 0.5236 | 0.5262 (+0.5) | 0.5292 (+1.1) | 0.5329 (+1.8) | 0.4944 (-5.6) |
| 30 | 0.7050 | 0.6933 (-1.7) | 0.7046 (-0.1) | 0.7034 (-0.2) | 0.6701 (-5.0) |
| 40 | 0.8071 | 0.8053 (-0.2) | 0.8104 (+0.4) | 0.8160 (+1.1) | 0.7858 (-2.6) |
| 60 | 0.9315 | 0.9420 (+1.1) | 0.9373 (+0.6) | 0.9357 (+0.4) | 0.9356 (+0.4) |
| 80 | 0.9821 | 0.9789 (-0.3) | 0.9817 (-0.0) | 0.9825 (+0.0) | 0.9767 (-0.5) |
| 100 | 0.9996 | 0.9996 (-0.0) | 0.9994 (-0.0) | 0.9996 (+0.0) | 0.9994 (-0.0) |

$P_n$

| n | b=0 | b=0.25 | b=0.5 | b=0.75 | b=1.0 |
|---|---|---|---|---|---|
| 1 | 5.082 | 4.694 (-7.6) | 4.796 (-5.6) | 5.018 (-1.2) | 4.653 (-8.4) |
| 2 | 4.408 | 4.469 (+1.4) | 4.408 (+0.0) | 4.439 (+0.7) | 3.949 (-10.4) |
| 3 | 4.707 | 4.197 (-10.8) | 4.687 (-0.4) | 4.571 (-2.9) | 3.837 (-18.5) |
| 4 | 4.776 | 4.306 (-9.8) | 4.668 (-2.2) | 4.617 (-3.3) | 3.903 (-18.3) |
| 5 | 4.559 | 4.237 (-7.1) | 4.551 (-0.2) | 4.522 (-0.8) | 3.873 (-15.0) |
| 10 | 4.080 | 4.076 (-0.1) | 4.288 (+5.1) | 4.273 (+4.8) | 3.718 (-8.9) |
| 20 | 3.502 | 3.462 (-1.1) | 3.528 (+0.7) | 3.560 (+1.7) | 3.247 (-7.3) |
| 30 | 3.100 | 2.997 (-3.3) | 3.070 (-1.0) | 3.101 (+0.0) | 2.885 (-6.9) |
| 40 | 2.680 | 2.626 (-2.0) | 2.673 (-0.2) | 2.699 (+0.7) | 2.539 (-5.3) |
| 60 | 2.080 | 2.087 (+0.3) | 2.084 (+0.2) | 2.087 (+0.3) | 2.070 (-0.5) |
| 80 | 1.635 | 1.635 (-0.0) | 1.637 (+0.1) | 1.636 (+0.1) | 1.631 (-0.3) |
| 100 | 1.326 | 1.326 (+0.0) | 1.326 (-0.0) | 1.326 (+0.0) | 1.326 (-0.0) |

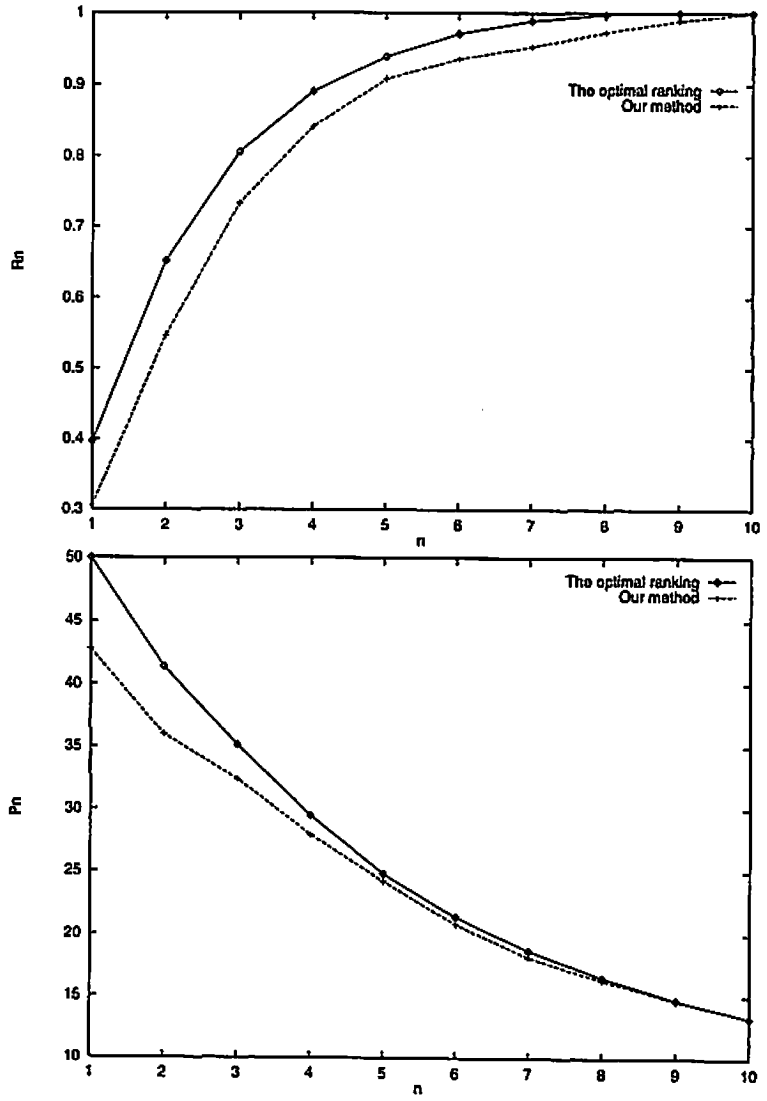


Figure 4: Comparison with the optimal ranking (10 collections).

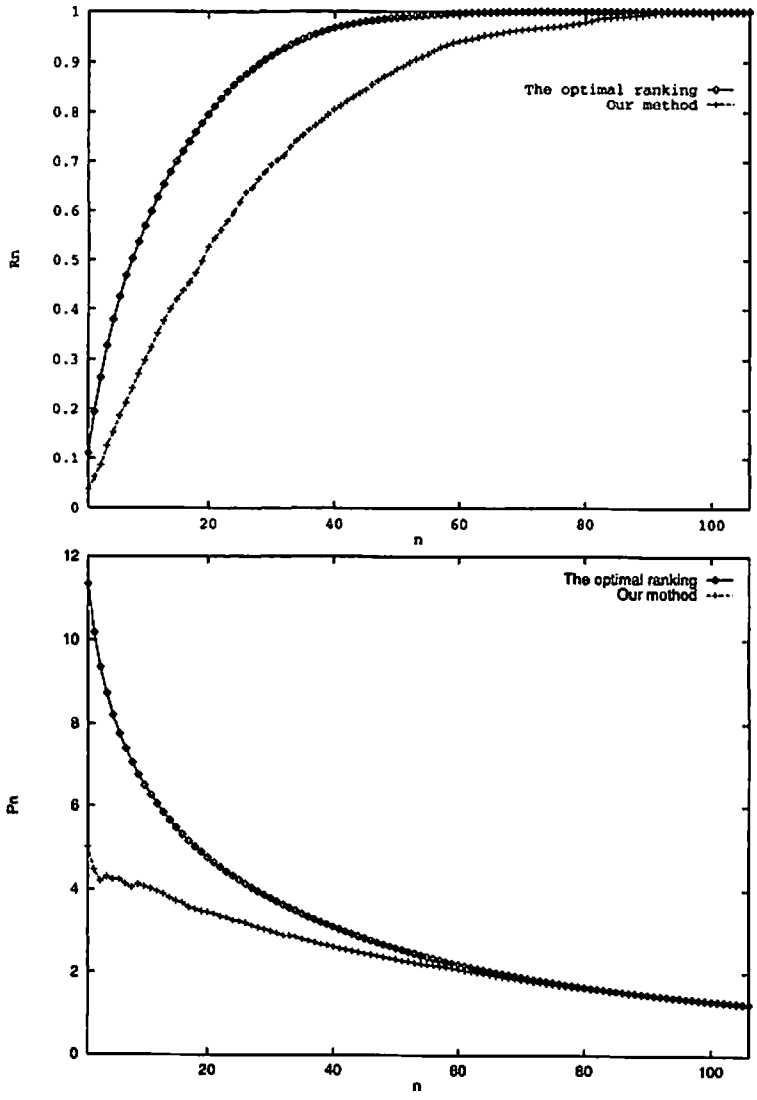


Figure 5: Comparison with the optimal ranking (106 collections).

Table 5: The effectiveness of weighted merging functions (10 collections in TREC volumes 2 & 3, topics 201-250).

| Interpolated Recall - Precision Averages (49 queries): | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Normalised | Raw | | MF1-all | | MF2-all | | MF1-sel-5 | | MF2-sel-5 |
| 0% | 70.27 | 62.81 | (-10.6) | 68.02 | ( -3.2) | 69.27 | ( -1.4) | 69.19 | ( -1.5) | 68.56 | ( -2.4) |
| 10% | 48.61 | 40.11 | (-17.5) | 46.87 | ( -3.6) | 45.31 | ( -6.8) | 46.53 | ( -4.3) | 45.99 | ( -5.4) |
| 20% | 40.51 | 33.80 | (-16.6) | 39.61 | ( -2.2) | 38.47 | ( -5.0) | 39.46 | ( -2.6) | 38.86 | ( -4.1) |
| 30% | 34.21 | 29.31 | (-14.3) | 33.76 | ( -1.3) | 33.61 | ( -1.8) | 30.56 | (-10.7) | 31.13 | ( -9.0) |
| 40% | 28.83 | 25.09 | (-13.0) | 27.65 | ( -4.1) | 28.02 | ( -2.8) | 23.99 | (-16.8) | 24.27 | (-15.8) |
| 50% | 23.61 | 19.96 | (-15.5) | 22.05 | ( -6.6) | 23.04 | ( -2.4) | 19.19 | (-18.7) | 19.75 | (-16.3) |
| 60% | 17.82 | 14.31 | (-19.7) | 15.33 | (-14.0) | 16.28 | ( -8.6) | 13.55 | (-24.0) | 14.24 | (-20.1) |
| 70% | 8.95 | 8.64 | ( -3.5) | 7.37 | (-17.7) | 8.91 | ( -0.4) | 5.90 | (-34.1) | 7.16 | (-20.0) |
| 80% | 5.45 | 4.71 | (-13.6) | 3.98 | (-27.0) | 4.52 | (-17.1) | 2.77 | (-49.2) | 3.40 | (-37.6) |
| 90% | 2.44 | 2.26 | ( -7.4) | 1.54 | (-36.9) | 2.13 | (-12.7) | 0.63 | (-74.2) | 1.39 | (-43.0) |
| 100% | 0.08 | 0.07 | (-12.5) | 0.06 | (-25.0) | 0.07 | (-12.5) | 0.00 | (-100.0) | 0.00 | (-100.0) |
| Average precision (non-interpolated) over all rel docs | | | | | | | | | | |
| | 23.82 | 19.94 | (-16.3) | 22.39 | ( -6.0) | 22.56 | ( -5.3) | 20.83 | (-12.6) | 21.06 | (-11.6) |
| Precision: | | | | | | | | | | |
| 5 docs: | 51.43 | 42.86 | (-16.7) | 51.02 | ( -0.8) | 48.98 | ( -4.8) | 50.61 | ( -1.6) | 48.57 | ( -5.6) |
| 10 docs: | 44.29 | 41.02 | ( -7.4) | 45.10 | ( +1.8) | 44.08 | ( -0.5) | 43.88 | ( -0.9) | 43.27 | ( -2.3) |
| 15 docs: | 41.36 | 37.28 | ( -9.9) | 41.63 | ( +0.7) | 41.09 | ( -0.7) | 40.27 | ( -2.6) | 40.00 | ( -3.3) |
| 20 docs: | 39.08 | 35.71 | ( -8.6) | 40.10 | ( +2.6) | 39.18 | ( +0.3) | 39.49 | ( +1.0) | 38.78 | ( -0.8) |
| 30 docs: | 35.78 | 31.97 | (-10.6) | 36.33 | ( +1.5) | 35.44 | ( -1.0) | 35.17 | ( -1.7) | 35.44 | ( -1.0) |
| 100 docs: | 24.96 | 21.82 | (-12.6) | 24.47 | ( -2.0) | 24.63 | ( -1.3) | 23.96 | ( -4.0) | 24.14 | ( -3.3) |
| 200 docs: | 19.06 | 17.27 | ( -9.4) | 18.33 | ( -3.8) | 18.71 | ( -1.8) | 18.06 | ( -5.2) | 18.49 | ( -3.0) |
| 500 docs: | 11.55 | 10.76 | ( -6.8) | 11.28 | ( -2.3) | 11.56 | ( +0.1) | 10.82 | ( -6.3) | 10.98 | ( -4.9) |
| 1000 docs: | 7.16 | 6.92 | ( -3.4) | 7.06 | ( -1.4) | 7.20 | ( +0.6) | 6.66 | ( -7.0) | 6.79 | ( -5.2) |
| R-Precision (precision after R (= num_rel for a query) docs retrieved): | | | | | | | | | | |
| Exact: | 28.74 | 24.95 | (-13.2) | 27.81 | ( -3.2) | 27.94 | ( -2.8) | 26.55 | ( -7.6) | 26.67 | ( -7.2) |

Table 6: The effectiveness of weighted merging functions (106 collections in TREC volumes 2 & 3, topics 201-250).

| Interpolated Recall - Precision Averages (49 queries): | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | Normalised | Raw | | MF1-all | | MF2-all | | MF1-sel-50 | | MF2-sel-50 |
| 0% | 70.27 | 45.99 | (-34.6) | 68.19 | ( -3.0) | 67.10 | ( -4.5) | 73.62 | ( +4.8) | 74.35 | ( +5.8) |
| 10% | 48.61 | 31.96 | (-34.3) | 43.40 | (-10.7) | 42.88 | (-11.8) | 43.80 | ( -9.9) | 44.22 | ( -9.0) |
| 20% | 40.51 | 25.09 | (-38.1) | 36.24 | (-10.5) | 33.80 | (-16.6) | 32.99 | (-18.5) | 31.71 | (-21.7) |
| 30% | 34.21 | 21.30 | (-37.7) | 30.02 | (-12.2) | 27.82 | (-18.7) | 25.31 | (-26.0) | 24.53 | (-28.3) |
| 40% | 28.83 | 17.77 | (-38.4) | 24.32 | (-15.6) | 22.65 | (-21.4) | 20.08 | (-30.4) | 19.33 | (-33.0) |
| 50% | 23.61 | 14.94 | (-36.7) | 18.77 | (-20.5) | 17.58 | (-25.5) | 15.48 | (-34.4) | 14.65 | (-38.0) |
| 60% | 17.82 | 10.53 | (-40.9) | 11.97 | (-32.8) | 11.11 | (-37.7) | 10.04 | (-43.7) | 9.16 | (-48.6) |
| 70% | 8.95 | 5.68 | (-36.5) | 5.58 | (-37.7) | 5.28 | (-41.0) | 4.03 | (-55.0) | 3.85 | (-57.0) |
| 80% | 5.45 | 2.93 | (-46.2) | 3.56 | (-34.7) | 3.24 | (-40.6) | 2.46 | (-54.9) | 2.27 | (-58.3) |
| 90% | 2.44 | 1.23 | (-49.6) | 1.37 | (-43.9) | 1.47 | (-39.8) | 0.58 | (-76.2) | 0.55 | (-77.5) |
| 100% | 0.08 | 0.08 | ( +0.0) | 0.08 | ( +0.0) | 0.07 | (-12.5) | 0.00 | (-100.0) | 0.00 | (-100.0) |
| Average precision (non-interpolated) over all rel docs | | | | | | | | | | |
| | 23.82 | 14.49 | (-39.2) | 20.08 | (-15.7) | 18.97 | (-20.4) | 16.31 | (-23.1) | 17.84 | (-25.1) |
| Precision: | | | | | | | | | | |
| 5 docs: | 51.43 | 28.98 | (-43.7) | 46.94 | ( -8.7) | 46.53 | ( -9.5) | 51.02 | ( -0.8) | 50.20 | ( -2.4) |
| 10 docs: | 44.29 | 28.37 | (-35.9) | 43.06 | ( -2.8) | 42.24 | ( -4.6) | 42.65 | ( -3.7) | 43.27 | ( -2.3) |
| 15 docs: | 41.36 | 26.80 | (-35.2) | 39.32 | ( -4.9) | 39.59 | ( -4.3) | 39.73 | ( -3.9) | 40.68 | ( -1.6) |
| 20 docs: | 39.08 | 25.51 | (-34.7) | 35.51 | ( -9.1) | 35.82 | ( -8.3) | 35.82 | ( -8.3) | 36.84 | ( -5.7) |
| 30 docs: | 35.78 | 24.06 | (-32.7) | 32.86 | ( -8.2) | 32.59 | ( -8.9) | 33.61 | ( -6.1) | 33.33 | ( -6.8) |
| 100 docs: | 24.96 | 18.59 | (-25.5) | 23.10 | ( -7.5) | 22.59 | ( -9.5) | 20.76 | (-16.8) | 20.88 | (-16.3) |
| 200 docs: | 19.06 | 15.02 | (-21.2) | 17.43 | ( -8.6) | 17.16 | (-10.0) | 15.41 | (-19.2) | 15.38 | (-19.3) |
| 500 docs: | 11.55 | 9.98 | (-13.6) | 10.70 | ( -7.4) | 10.44 | ( -9.6) | 9.27 | (-19.7) | 9.16 | (-20.7) |
| 1000 docs: | 7.16 | 6.40 | (-10.6) | 6.72 | ( -6.1) | 6.54 | ( -8.7) | 5.65 | (-21.1) | 5.60 | (-21.8) |
| R-Precision (precision after R (= num_rel for a query) docs retrieved): | | | | | | | | | | |
| Exact: | 28.74 | 20.50 | (-28.7) | 26.16 | ( -9.0) | 25.71 | (-10.5) | 24.26 | (-15.6) | 24.24 | (-15.7) |

Table 7: The effectiveness of weighted merging functions – using no-icf collection ranking scores (10 collections in TREC volumes 2 & 3, topics 201-250).

| Interpolated Recall - Precision Averages (49 queries): | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Normalised | MF1-all | | MF2-all | | MF1-sel-5 | | MF2-sel-5 | |
| 0% | 70.27 | 69.95 | ( -0.5) | 70.47 | ( +0.3) | 69.94 | ( -0.5) | 70.97 | ( +1.0) |
| 10% | 48.61 | 46.00 | ( -5.4) | 46.21 | ( -4.9) | 46.92 | ( -3.5) | 47.35 | ( -2.6) |
| 20% | 40.51 | 38.78 | ( -4.3) | 38.82 | ( -4.2) | 39.01 | ( -3.7) | 38.80 | ( -4.2) |
| 30% | 34.21 | 33.43 | ( -2.3) | 33.58 | ( -1.8) | 32.05 | ( -6.3) | 32.11 | ( -6.1) |
| 40% | 28.83 | 28.11 | ( -2.5) | 28.18 | ( -2.3) | 26.11 | ( -9.4) | 26.14 | ( -9.3) |
| 50% | 23.61 | 23.40 | ( -0.9) | 23.50 | ( -0.5) | 20.57 | (-12.9) | 20.65 | (-12.5) |
| 60% | 17.82 | 16.65 | ( -6.6) | 16.71 | ( -6.2) | 14.58 | (-18.2) | 14.56 | (-18.3) |
| 70% | 8.95 | 8.64 | ( -3.5) | 8.67 | ( -3.1) | 6.81 | (-23.9) | 6.82 | (-23.8) |
| 80% | 5.45 | 4.93 | ( -9.5) | 4.97 | ( -8.8) | 3.47 | (-36.3) | 3.34 | (-38.7) |
| 90% | 2.44 | 2.57 | ( +5.3) | 2.58 | ( +5.7) | 1.75 | (-28.3) | 1.71 | (-29.9) |
| 100% | 0.08 | 0.07 | (-12.5) | 0.07 | (-12.5) | 0.00 | (-100.0) | 0.00 | (-100.0) |
| Average precision (non-interpolated) over all rel docs | | | | | | | | | |
| | 23.82 | 22.95 | ( -3.7) | 23.06 | ( -3.2) | 21.87 | ( -8.2) | 21.93 | ( -7.9) |
| Precision: | | | | | | | | | |
| 5 docs: | 51.43 | 51.84 | ( +0.8) | 52.24 | ( +1.6) | 52.24 | ( +1.6) | 52.24 | ( +1.6) |
| 10 docs: | 44.29 | 44.69 | ( +0.9) | 45.10 | ( +1.8) | 45.71 | ( +3.2) | 46.53 | ( +5.1) |
| 15 docs: | 41.36 | 42.18 | ( +2.0) | 42.45 | ( +2.6) | 41.90 | ( +1.3) | 41.77 | ( +1.0) |
| 20 docs: | 39.08 | 40.10 | ( +2.6) | 40.10 | ( +2.6) | 40.31 | ( +3.1) | 40.00 | ( +2.4) |
| 30 docs: | 35.78 | 35.92 | ( +0.4) | 36.05 | ( +0.8) | 35.51 | ( -0.8) | 35.78 | ( +0.0) |
| 100 docs: | 24.96 | 25.04 | ( +0.3) | 25.10 | ( +0.6) | 24.96 | ( +0.0) | 24.86 | ( -0.4) |
| 200 docs: | 19.06 | 19.13 | ( +0.4) | 19.13 | ( +0.4) | 18.99 | ( -0.4) | 19.00 | ( -0.3) |
| 500 docs: | 11.55 | 11.70 | ( +1.3) | 11.72 | ( +1.5) | 11.18 | ( -3.2) | 11.23 | ( -2.8) |
| 1000 docs: | 7.16 | 7.23 | ( +1.0) | 7.24 | ( +1.1) | 6.87 | ( -4.1) | 6.86 | ( -4.2) |
| R-Precision (precision after R (= num_rel for a query) docs retrieved): | | | | | | | | | |
| Exact: | 28.74 | 28.21 | ( -1.8) | 28.17 | ( -2.0) | 27.41 | ( -4.6) | 27.30 | ( -5.0) |

Table 8: The effectiveness of weighted merging functions – using no-icf collection ranking scores (100 collections in TREC volumes 2 & 3, topics 201-250).

| Interpolated Recall - Precision Averages (49 queries): | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Normalised | MF1-all | | MF2-all | | MF1-sel-50 | | MF2-sel-50 | |
| 0% | 70.27 | 69.19 | ( -1.5) | 69.04 | ( -1.8) | 71.26 | ( +1.4) | 70.89 | ( +0.9) |
| 10% | 48.61 | 46.50 | ( -4.3) | 46.06 | ( -5.2) | 46.60 | ( -4.1) | 46.47 | ( -4.4) |
| 20% | 40.51 | 36.49 | ( -9.9) | 36.41 | (-10.1) | 35.82 | (-11.6) | 35.87 | (-11.5) |
| 30% | 34.21 | 30.11 | (-12.0) | 29.93 | (-12.5) | 26.65 | (-22.1) | 26.67 | (-22.0) |
| 40% | 28.83 | 23.92 | (-17.0) | 23.84 | (-17.3) | 19.38 | (-32.8) | 19.39 | (-32.7) |
| 50% | 23.61 | 18.87 | (-20.1) | 19.06 | (-19.3) | 14.70 | (-37.7) | 14.71 | (-37.7) |
| 60% | 17.82 | 11.87 | (-33.4) | 11.88 | (-33.3) | 7.80 | (-56.2) | 7.75 | (-56.5) |
| 70% | 8.95 | 5.26 | (-41.3) | 5.58 | (-37.7) | 4.05 | (-54.7) | 4.04 | (-54.9) |
| 80% | 5.45 | 3.00 | (-45.0) | 2.95 | (-45.9) | 2.25 | (-58.7) | 2.24 | (-58.9) |
| 90% | 2.44 | 1.10 | (-54.9) | 1.14 | (-53.3) | 0.75 | (-69.3) | 0.75 | (-69.3) |
| 100% | 0.08 | 0.09 | (+12.5) | 0.09 | (+12.5) | 0.00 | (-100.0) | 0.00 | (-100.0) |
| Average precision (non-interpolated) over all rel docs | | | | | | | | | |
| | 23.82 | 20.28 | (-14.9) | 20.27 | (-14.9) | 18.45 | (-22.5) | 18.42 | (-22.7) |
| Precision: | | | | | | | | | |
| 5 docs: | 51.43 | 51.43 | ( +0.0) | 51.43 | ( +0.0) | 50.61 | ( -1.6) | 50.61 | ( -1.6) |
| 10 docs: | 44.29 | 44.49 | ( +0.5) | 44.08 | ( -0.5) | 44.08 | ( -0.5) | 43.67 | ( -1.4) |
| 15 docs: | 41.36 | 40.95 | ( -1.0) | 40.54 | ( -2.0) | 40.00 | ( -3.3) | 40.00 | ( -3.3) |
| 20 docs: | 39.08 | 37.45 | ( -4.2) | 37.14 | ( -5.0) | 36.53 | ( -6.5) | 36.43 | ( -6.8) |
| 30 docs: | 35.78 | 34.29 | ( -4.2) | 34.29 | ( -4.2) | 34.35 | ( -4.0) | 34.35 | ( -4.0) |
| 100 docs: | 24.96 | 23.20 | ( -7.1) | 23.27 | ( -6.8) | 22.45 | (-10.1) | 22.29 | (-10.7) |
| 200 docs: | 19.06 | 17.36 | ( -8.9) | 17.46 | ( -8.4) | 16.09 | (-15.6) | 16.03 | (-15.9) |
| 500 docs: | 11.55 | 10.63 | ( -8.0) | 10.65 | ( -7.8) | 9.46 | (-18.1) | 9.45 | (-18.2) |
| 1000 docs: | 7.16 | 6.61 | ( -7.7) | 6.63 | ( -7.4) | 5.78 | (-19.3) | 5.75 | (-19.7) |
| R-Precision (precision after R (= num_rel for a query) docs retrieved): | | | | | | | | | |
| Exact: | 28.74 | 26.23 | ( -8.7) | 26.24 | ( -8.7) | 24.43 | (-15.0) | 24.36 | (-15.2) |