International Journal of Digital Information and Wireless Communications (IJDIWC) 6(2): 78-86
The Society of Digital Information and Wireless Communications, 2016 ISSN: 2225-658X (Online); ISSN 2412-6551 (Print)

Enhancement of the Fusion of Incompatible Lists of Results

Guezouli Larbi and Azzouz Imane
LaSTIC, University of Batna 2,
Algeria
{larbi.guezouli,imane.azzouz}@univ-batna.dz

ABSTRACT

This work is located in the domain of distributed information retrieval (DIR). A simplified view of the DIR requires a multi-search in a set of collections, which forces the system to analyze results found in these collections, and merge results back before sending them to the user in a single list.
Our work is to find a fusion method based on the relevance score of each result received from collections and the relevance of the local search engine of each collection, which is the main issue of our work.

KEYWORDS

Text mining; distributed information retrieval; content-based retrieval; merging results.

1 INTRODUCTION

The growth of the number of servers on the worldwide network makes the management of a huge quantity of information an obligation. For this, the use of information retrieval systems became indispensable especially in the distributed world. To organize this huge quantity of information, web search engines find their origins in the information retrieval systems developed to find, from a database of documents, relevant documents to a user request. [1], [2], [3], [4], [5], [6]
Information retrieval systems can be categorized into classes based on their architectures: centralized and distributed. Centralized systems require that all documents were located on the same site. But distributed systems were considered as a set of independent information retrieval systems allowing the simultaneous access to distributed collections of documents on local or wide network. [7], [8], [9]

1.1 Problematic

One of essential problems of distributed information retrieval systems is the fusion of results obtained from a set of local information retrieval systems. The user must receive one list as response to his request. This list is the fruit of the fusion of different result's lists of different local information retrieval systems.
For example, user presents a query to the distributed information retrieval system. The system forwards the query to selected servers. Each server calls its local information retrieval system to search for the query. Each server provides a list of results to the distributed system. Finally, the distributed system merges all the lists of results in a single list and returns it to the user.
Therefore, the main problem is to define a set of rules for merging these lists.

1.2 Background

The field of information retrieval touches the distributed world in order to try to find relevant information contained in distributed servers. A user can't search the information in all these distributed servers. He needs a system which can do an automatic search for him.
A distributed information retrieval system [10] can do this work for the user, where the system receives the query of the user, then, and according to the query, it selects the pertinent servers and tries to find the needed information in these servers. Each server sends back a list of results.
The user can't process all the lists of results by himself, the system has to do this task for him.

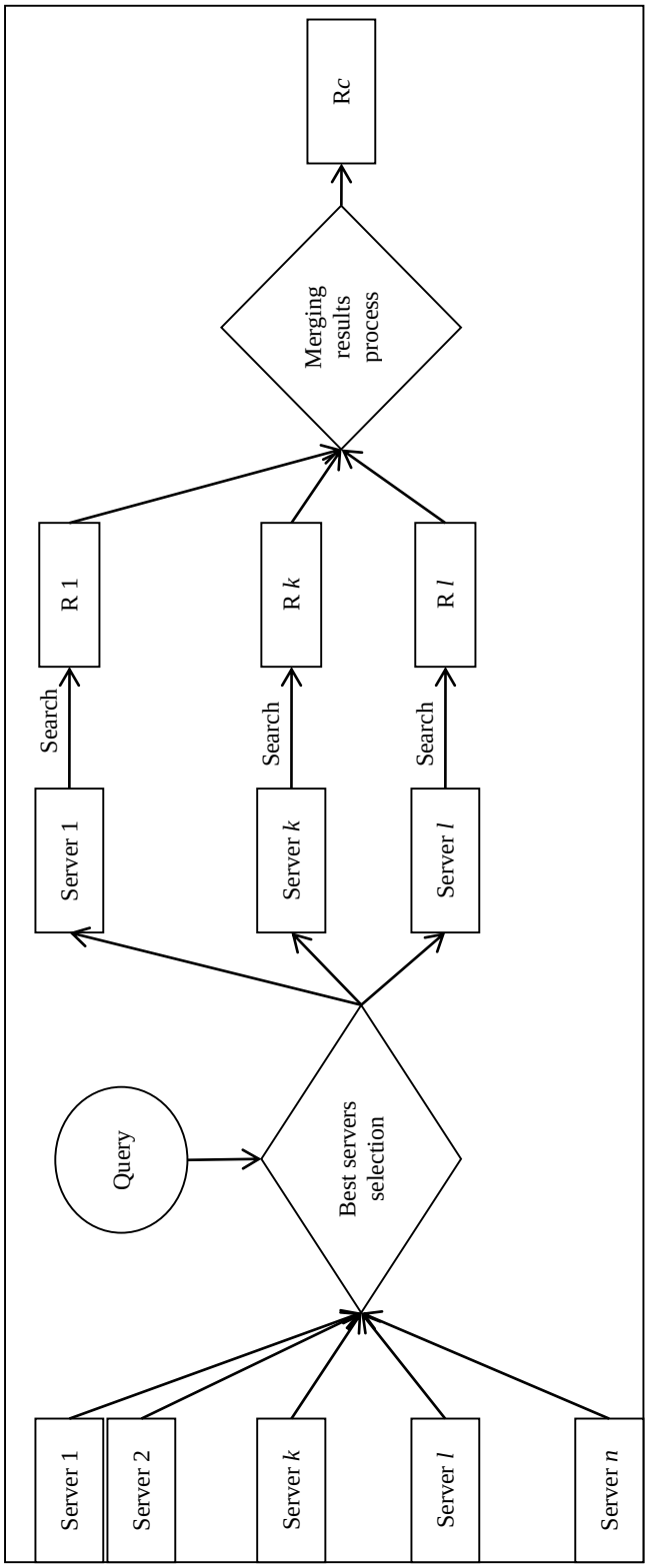

Figure 1. Information search process in distributed information retrieval system

Figure 1 shows the details of information search process in distributed information retrieval. In the first part, there is an important process called "Best servers selection". In this part, according to the query's content, the relevant servers will be selected.
The query will be sent to the selected servers in order to find relevant documents.
An important component of the system was the merging (or combining) results process. This module receives all the lists of results from selected servers and generates a combined list.
The system returns this combined list to the user who can use it easily.
In this work, we address issues pertaining to the merging results process in distributed information retrieval system.

1.3 Challenges

Merging results is challenging in particular in the context of a heterogeneous lists of results for several reasons.
The challenges are (1) make lists of results compatible, (2) sort merged results and (3) have a good relevance.
As we have said in the last section, the received lists of results from selected servers are not compatible because each server uses his own method of information retrieval. Therefore, merging and sorting results in one list is a problem.
We have to resolve this problem without losing the relevance.
The rest of the article is organized as follow. Section 2 discuses related work. We describe our proposals in section 3 and we report our experimental results in section 4. Finally, section 5 concludes the article.

2 RELATED WORK

Recently, there are a lot of researches realized in the domain of distributed information retrieval [3], [9], [11], [12], [13], [14], [15], [16], [17].
We focus our search on the last step of the process of distributed information retrieval systems which is the fusion of results.
The architecture of fusion process is shown in Figure 2.

Firstly, Roa et al.[11] have presented the results of the research carried out on the field of ranking strategies.
The analysis reported in their survey aims at providing a starting point towards future developments on benchmarking and empirical evaluation of ranking solutions for the Web of Linked Data.

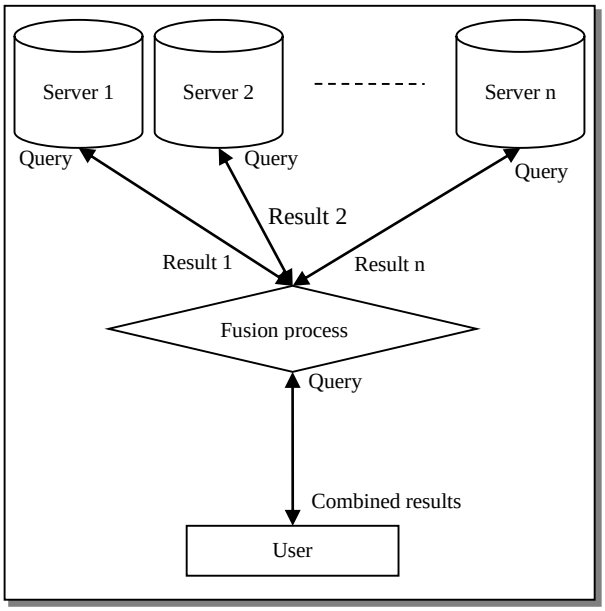

Figure 2. Architecture of fusion process

Dragut et al. [13] describe a system that integrates spatial objects from multiple local search engines.
They find the set of neighborhoods in a search engine that best approximates the area of a neighborhood in a different search engine.
Huo et al. [14] propose a fusion model to improve the tail queries by introducing new results from their reformulations. In our case, we have one query and several collections.
Savoy et al. [3] propose a method called RSM (Raw Score Merging) where they use local scores to sort results of different servers. They suppose that all servers use the same method to sort results and scores are compatible.
In our case, servers don't use the same method to sort results.
Other methods exist exploiting the heterogeneity of the sources of the documents. We can cite the CombSUM method and CombMNZ method.

Hubert et al. [18] propose CombSUM method which is based on the local scores obtained by different servers.
The score of the document D_j is calculated as follow:

$$ScoreCombSUM_j = \sum_{i=1}^{nbr_{servers}} score_{ij} \quad (1)$$

Hubert et al. [18] propose also CombMNZ method which is based on the local scores and the number of servers which have returned the query document in their results.
The score of the document D_j is calculated as follow:

$$ScoreCombMNZ_j = \left(\sum_{i=1}^{nbr\_servers} score_{ij}\right) \times Count_j \quad (2)$$

where $Count_j$ is the number of servers which have returned the document D_j.
This normalization isn't very important for our goal.
There is another kind of methods [3] based on the length of results of different servers. In this kind of methods, the local score of returned documents is weighted by the coefficient W_i calculated as follow:

$$W_i = 1 + \left(\frac{(S_i - S_m)}{S_m}\right) \quad (3)$$

where S_i is the score of the i^{th} server calculated as follow:

$$S_i = ln\left(1 + \left(\frac{L_i \cdot k}{\sum_{j=1}^{db} L_j}\right)\right) \quad (4)$$

where L_i is the number of documents returned pas the i^{th} server
and *db* is the number of interrogated servers.
These methods aren't pertinent for our case.
Rasolofo proposes in its thesis [19] the Round Robin approach where the merged list of results is constructed by taking alternately one document of each list returned by servers.
The disadvantage is that this method doesn't take into account the difference between the scores of the documents.
The CORI [20] algorithm is one of best used algorithms. The score of a document is calculated as follows:

$$D = \frac{D' + 0.4 \times D' \times C'_i}{1.4} \quad (5)$$

where:

$$D' = \frac{S - S_{min}}{S_{max} - S_{min}} \quad (6)$$

and S is the local score of a document;

S_{max} and S_{min} are the maximal and the minimal values of local scores of documents.

$$C'_i = \frac{C_i - C_{min}}{C_{max} - C_{min}} \quad (7)$$

and C_i is the relevance of the algorithm used by the i[th] server.

The final CORI score *D* is normalized between 0 and 1. According to the studies, the computational time of this method increases rapidly.

Yuwono and Lee propose a method called "Rank_based" [21] based on the conversion of ranks of documents in the list of results into a value of similarity.

This conversion isn't important for our case.

Rasolofo speaks about a method called Normalized Raw Score [19]. In this method the local scores are normalized by using maximal score of returned documents of all servers.

This normalization also isn't important for our case.

3 PROPOSED APPROACHES

In the distributed information retrieval we need to merge incompatible lists of results returned by several servers.

The problem of merging results is difficult because the lists of results are incompatible. They come from different information retrieval systems. Therefore, we can't gather all received lists of results in a single list without pretreatment.

To resolve this problem, we have to mathematically model the merge process which is the last step in the process of a distributed information retrieval system.

3.1 Problem Formulation

In the rest of this article we represent the merge process as a system defined as follows:

$$MP = <SR, CR, wf>$$

where:

MP is the Merge Process system;

SR is the Set of lists of Results returned by consulted servers (called: local results);

CR is the list of Combined Results (called: global results);

wf is the weight function used to make local results compatible.

In order to improve the relevance of the merging process and minimizing the calculation time, we had two ideas:

3.2 Approach Based on the Relevance of Local Information Retrieval Systems

The first idea is an approach based on the relevance of local information retrieval systems used by servers and local scores. We call it "M1" for (Method 1).

We define the weight function as the difference between the relevance of local information retrieval system and the maximal value of relevance of all used information retrieval systems. It is defined as follows:

$$wf(i) = (S_{max} - S_i) \times 100 \quad (8)$$

where:

wf(i) represents the weight of the i[th] server;

S_i is the relevance of the information retrieval system used by the i[th] server. It is restricted between 0 and 1;

S_{max} is the maximal relevance of information retrieval systems used in all known servers.

The constant 100 is used to make wf_i as a percentage since S_i is restricted between 0 and 1.

The algorithm 1 gives the details of calculation of the weight function.

Algorithm 1
Calculate $wf(i) = (S_{max} - S_i) \times 100$

Require: i
Ensure: $wf(i)$
Find S_{max}
return $(S_{max} - S_i) \times 100$

And then, the local results (*SR*) of documents returned by the i[th] server are weighted by *wf(i)*.

$$SR_{ij} = SR_{ij} \times wf(i) \quad (9)$$

Finally, documents selected by servers are sorted by their weights to build the *CR* list (global results).

The algorithm 2 shows how to combine the lists of local results in a single list.

Algorithm 2 Combine results

```
Require: SR, S
Ensure: CR
k ← 0
for I = 0 to nbServers do
for j = 0 to sizeof(SR[i]) do
CR[k] ← SR[i][j]*wf(i)
k++
end for
end for
Sort the table CR
return CR
```

3.3 Approach Based on the Relevance of Local Information Retrieval Systems and the Ranks of Selected Documents

The second idea is based on the relevance of local information retrieval systems used by servers and the ranks of selected documents in local results. We call it "M2" for (Method 2).

We define the weight function as the ratio between the relevance of local information retrieval system and the maximal value of relevance of all used information retrieval systems.

To make this weight dependent on the rank of documents, the ratio is multiplied by the complement of the rank of a document in its list of results.

The weight function is defined as follows:

$$wf(rank_i, i) = \left(rank_{max} - (rank - 1)\right) \times \frac{S_i}{S_{max}} \quad (10)$$

where:

$rank_i$ is the rank of a document in the local list of results *SR* of the ith server (ie. the rank in $SR[i] \geq 1$);

$wf(rank_i,i)$ is the new score of the $rank^{th}$ document of the ith server;

$rank_{max}$ is the maximal value of rank;

S_i is the relevance of the information retrieval system used by the ith server;

S_{max} is the maximal value of the relevance of all consulted servers $S_{max}=MAX(S_1,S_2,...,S_n)$.

We note that the new score $wf(rank_i,i)$ depends, at the same time, on the local ranks of documents in the local lists of results and the relevance of the information retrieval systems used by consulted servers.

The algorithm 3 gives the details of calculation of the weight function.

Algorithm 3 Calculate $wf(rank_i, rank_{max}, i, S)$

```
Require: rank_i, i, S, rank_max
Ensure: wf(rank_i, i)
Find S_max
return (rank_max − (rank − 1)) × S_i / S_max
```

Finally, documents with good scores are the closest to the query.

Using the new score $wf(rank_i,i)$, we try to make results provided by different servers compatible.

The algorithm 4 shows how to combine the lists of local results in a single list.

Algorithm 4 Combine results

```
Require: SR, S
Ensure: CR
k ← 0
for i = 0 to nbServers do
rank_max ← sizeof(SR[i])
for rank_i = 1 to sizeof(SR[i]) do
CR[k] ← wf(rank_i, rank_max, i, S)
k++
end for
end for
Sort the table CR
return CR
```

4. EXPERIMENTAL RESULTS

In this section we present results of performed experiments in order to validate the performances of our approaches.

We need to evaluate our approaches in real conditions by taking into account the factor of computational time and the sizes of the corpora.

4.1 Evaluation of Execution Time

In order to compare our propositions with existing approaches of fusion of results, we begin by calculating the execution time using the same corpora with these methods.

The following graphs show the evolution of execution time of developed methods.

The graphs of Figure 3 show that the fusion time of each method rises almost linearly depending on the number of servers. (Except the CORI method which rises exponentially).

We observe that our first proposition called "M1" presents a good execution time comparing to the other methods.
Our second proposition "M2" is situated between "CORI" and "Ranked based" methods which is a good result.

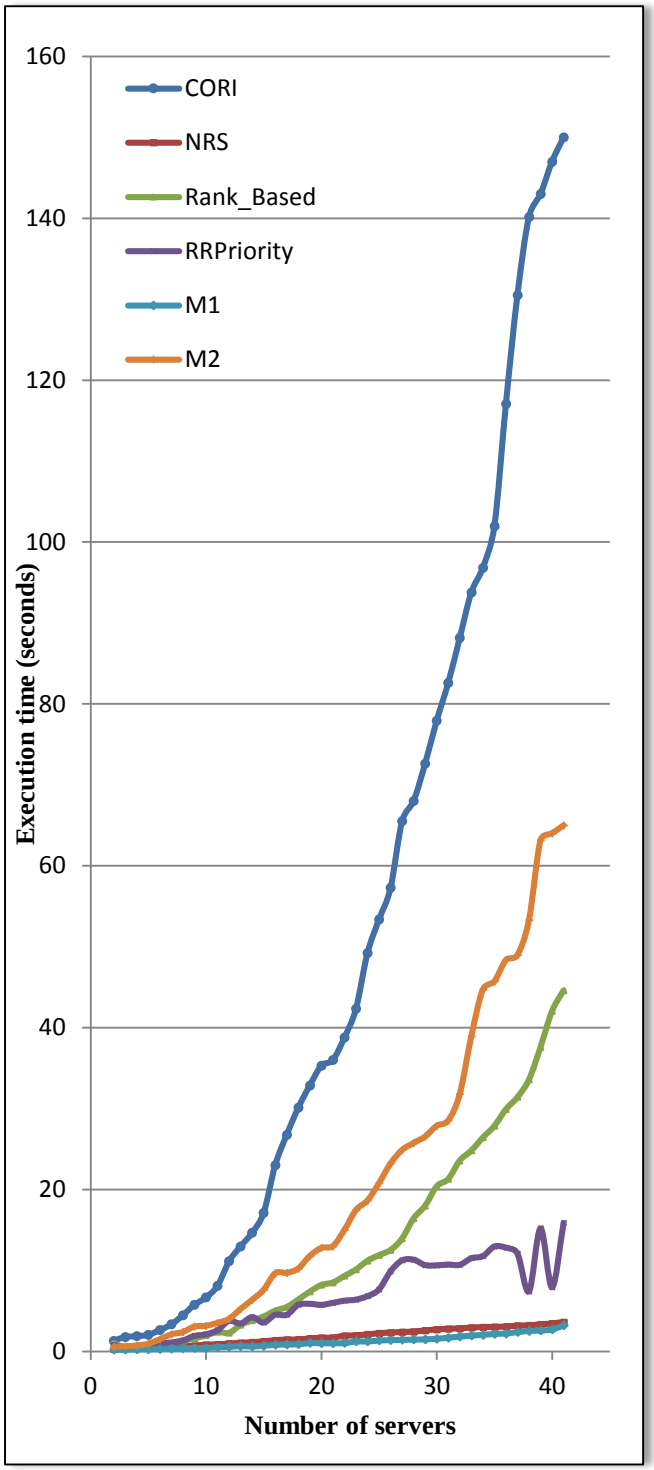

Figure 3 Evolution of computing time of fusion process according to the number of servers

4.2 Evaluation of the Relevance of Merge Process

The relevance of a method isn't based only on the computing time. For this, we propose to use another metric called the "Mean-Squared Error" (MSE).
Considerate that S is the number of documents sorted by a method of fusion. We propose to calculate the error rate of fusion process by the following formula:

$$MSE = \frac{1}{|S|} \times \sum_{i \in S} (Id_i - O_i)^2 \qquad (11)$$

In Eq. (11), Id_i is the rank of the ith document in the ideal sorted list of results; and O_i is the rank assigned to the ith document by the fusion method to evaluate.
In order to evaluate the relevance of implemented methods of fusion, we use the set of servers showed in Table 1, where sp_i is the relevance of the ith server.

Table 1. Results obtained by different servers

Server 1		**Server 2**		**Server 3**	
sp_1 = 90		**sp_2 = 70**		**sp_3 = 40**	
doc	**score**	**doc**	**score**	**doc**	**score**
LA123	65.5	FR453	87.54	FT567	87.54
LA673	57.8	FR012	75.5	FT195	51.64
LA946	35.7	FR673	11.84	FT548	40.9
LA765	19.81				
LA546	10.74				

Server 4		**Server 5**	
sp_1 = 35		**sp_2 = 60**	
doc	**score**	**doc**	**score**
DTR318	42.9	MHT217	90.43
DTR707	24.95	MHT232	15.54
DTR850	29.15	MHT305	22.56
DTR964	44.54	MHT471	13.07
DTR123	83.64		

After several tests with changing the number of servers, we obtained the results shown on the Table 2. Each column represents the sorted list of results obtained by one method.

Table 2. Comparison table of fusion results obtained by several methods

CORI Modified	NRS	RRPriority	Rank_Based
MHT217	LA123	LA123	LA123
DTR123	FR453	FR453	LA673
FT567	LA673	MHT217	LA946
FR453	MHT217	FT567	MHT217
LA123	FR012	DTR123	FR453
LA673	FT567	LA673	MHT305
FR012	DTR123	FR012	LA765
FT195	LA946	MHT305	DTR123
LA946	DTR964	FT195	DTR964
DTR964	DTR318	DTR964	FR012
DTR318	FT195	LA946	MHT232
FT548	LA765	FR673	FT567
DTR850	MHT305	MHT232	DTR318
LA765	DTR850	FT548	LA546
DTR707	MHT232	DTR318	FT195
MHT305	MHT471	LA765	DTR850
MHT232	DTR707	MHT471	FR673
LA546	FT548	DTR850	MHT471
MHT471	FR673	LA546	FT548
FR673	LA546	DTR707	DTR707

CORI Modified	M1	M2
MHT217	FR453	FR453
DTR123	LA123	LA123
FT567	MHT217	MHT217
FR453	FR012	FR012
LA123	LA673	LA673
LA673	FT567	FT567
FR012	LA946	DTR123
FT195	DTR123	LA946
LA946	FT195	FT195
DTR964	LA765	FT548
DTR318	FT548	DTR964
FT548	DTR964	LA765
DTR850	DTR318	DTR318
LA765	MHT305	MHT305
DTR707	DTR850	DTR850
MHT305	LA546	DTR707
MHT232	MHT232	MHT232
LA546	DTR707	LA546
MHT471	FR673	FR673
FR673	MHT471	MHT471

According to the prepared query, we find that results of "CORI modified" method are the best comparing to other methods. For this, we consider that results of "CORI modified" method represent the ideal list of results, and we compare the results of all other methods with those of "CORI modified".

On Table 2 we can see that the 8 first answers of our methods are almost equivalent to answers of "CORI modified" method. And we can see clearly that our two approaches are better than the other methods (NRS, RRPriority, Rank_Based).

After these tests, we can apply the Eq. (11) to calculate the error rate of each method. We found results shown in Table 3.

Table 3. Error Rates of Results Returned by Several Methods

Methods	MSE
CORI Modified	00.00
M1	05.80
M2	05.80
NRS	10.40
RRPriority	10.60
Rank_Based	13.50

We observe on Table 3 that the error rates of proposed methods "M1" and "M2" are better than those of NRS, RRPriority and Rank_Based methods.

Finally, according to these tests, we conclude that our approaches are good in computing time and in terms of relevance comparing to existing methods.

4.3 Discussions

Results obtained by performed experiments show that the calculation time of merge process rises almost linearly depending on the number of servers, except the "CORI modified" method which rises exponentially.

Graphs of Table 3 show that our first proposition called "M1" presents a good execution time comparing to the other methods. Our second proposition "M2" is situated between "CORI modified" and "Ranked based" methods which is a good result.

Results of the relevance tests show that the error rates of proposed methods "M1" and "M2" are

better than those of NRS, RRPriority and Rank_Based methods. The MSE of our proposed approaches is 5.8, but the MSE of NRS method is 10.4, the MSE of RRPriority method is 10.6 and that of Rank_Based method is 13.5.

Based on these tests, we conclude that our proposed approaches are good in computing time and in terms of relevance comparing to existing methods.

5 CONCLUSION

In our work, we were interested to the distributed information retrieval systems and especially to the last step in the process of these systems which is result's merging.

After studying different existing approaches of result's merging, and in order to ameliorate the performances of these methods, we proposed to add the relevance of local information retrieval systems of different servers.

By using the relevance of local information retrieval systems, the merging process provides good performance when we merge different lists of results obtained by different servers.

The highlight of this proposition is to taking into account the heterogeneity of local information retrieval systems.

To evaluate our propositions and to compare them with existing approaches, we used the computing time and the error rate of each method.

Our tests showed that proposed approaches "M1" and "M2" are better than existing methods: NRS, RRPriority and rank_based in terms of computing time - relevance ratio.

REFERENCES

1. Craswell, Nicholas Eric: Methods for Distributed Information Retrieval. 2000, The Australian National University.
2. Hassina, Aliane, Alimazighi Zaia, Boughacha Rime, and Djeliout Toufik. Un système de reformulation de requêtes pour la recherche d'information. Revue d'Information Scientifique et Technique (RIST), 2003. **13**(1): p. 25-33.
3. Savoy, Jacques, Yves Rasolofo, and Faiza Abbaci: Recherche d'informations dans des sources distribuées. In: *XIXème Congrés Informatique des Organisations et Systèmes d'Information et de Décision (INFORSID)*. 2001: Martigny, Suisse. p. 237–252.
4. Si, Luo, Jamie Callan, Suleyman Cetintas, and Hao Yuan. An effective and efficient results merging strategy for multilingual information retrieval in federated search environments. Information Retrieval journal, 2008. **11**(1): p. 1-24.
5. Ming-Feng, Tsai, Wang Yu-Ting, and Chen Hsin-Hsi: A Study of Learning a Merge Model for Multilingual Information Retrieval. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval. 2008. Singapore, Singapore: ACM.
6. Paltoglou, Georgios, Michail Salampasis, and Maria Satratzemi. A results merging algorithm for distributed information retrieval environments that combines regression methodologies with a selective download phase. Information Processing and Management, 2008. **44**(4): p. 1580-1599.
7. Abbaci, Faiza: Méthodes de sélection de collections dans un environnement de recherche d'information distribuée. 2003, Neuchâtel University.
8. Paltoglou, Georgios: Algorithms and strategies for source selection and results merging (collection fusion algorithms) in distributed information retrieval systems. In: *Department of applied informatics*. 2009, University of macedonia Thessaloniki.
9. Meng, Weiyi, Clement Yu, and M. Tamer Ozsu: Advanced Metasearch Engine Technology. 2010: Morgan & Claypool.
10. Zygmunt, Mazur. On a Model of Distributed Information Retrieval Systems Based on Thesauri. Information Processing Management journal, 1984. **20**(4): p. 499-505.
11. Antonio, J. Roa-Valverde and Sicilia Miguel-Angel. A Survey of Approaches for Ranking on the Web of Data. Information Retrieval journal, 2014. **17**(4): p. 295-325.
12. Alonso, Inostrosa-Psijas, Wainer Gabriel, Gil-Costa Veronica, and Marin Mauricio: DEVs Modeling of Large Scale Web Search Engines. In: Proceedings of the 2014 Winter Simulation Conference (WSC '14). 2014. Savannah, Georgia: IEEE Press.
13. Eduard, Constantin Dragut, Dasgupta Bhaskar, Beirne Brian P., Neyestani Ali, Atassi Badr, Yu Clement, and Meng Weiyi. Merging Query Results From Local Search Engines for Georeferenced Objects. ACM Trans. Web, 2014. **8**(4): p. 20:1-20:29.
14. Shuai, Huo, Zhang Min, Liu Yiqun, and Ma Shaoping: Improving Tail Query Performance by Fusion Model. In: Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management (CIKM '14). 2014. Shanghai, China: ACM.
15. Horatiu, Bota, Zhou Ke, Jose Joemon M., and Lalmas Mounia: Composite Retrieval of Heterogeneous Web Search. In: Proceedings of the 23rd International Conference on World Wide Web (WWW '14). 2014. Seoul, Korea: ACM.
16. Savoy, Jacques and Le Calvé: Recherche d'informations dans un environnement distribué. Cahier de recherche en informatique. 1996: Université de Neuchâtel, Faculté de

droit et des sciences économiques, Division économique et sociale.

17. Wu, Shengli and Jieyu Li: Merging Results from Overlapping Databases in Distributed Information Retrieval. In: 21st Euromicro International Conference on Parallel, Distributed and Network-Based Processing (PDP). 2013.
18. Hubert, Gilles, Yannick Loiseau, and Josiane Mothe: Etude de diffrentes fonctions de fusion de systmes de recherche d'information. In: *Le document numérique dans le monde de la science et de la recherche (CIDE'10)*. 2007: Nancy, France. p. 199–207.
19. Yves, Rasolofo Omega: Recherche d'information distribuée: approches pour la sélection de collections et la fusion de listes de résultats. 2002.
20. SI, LUO and JAMIE CALLAN. A semisupervised learning method to merge search engine results. ACM Trans. Inf. Syst., 2003. **21**(4): p. 457–491.
21. Yuwono, Budi and Dik Lun Lee: Server ranking for distributed text retrieval systems on the internet. In: Proceedings of the Fifth International Conference on Database Systems for Advanced Applications. 1997.