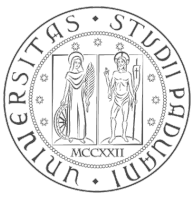

UNIVERSITÀ DEGLI STUDI DI PADOVA

DEPARTMENT OF INFORMATION ENGINEERING

MASTER COURSE IN COMPUTER ENGINEERING

REPERIMENTO DELL'INFORMAZIONE - P.ssa Maristella Agosti, Ing. Gianmaria Silvello

---

RANK FUSION STRATEGIES: FOX-SHAW AND CONDORCET

Luca Pietrogrande
Matricola: 1149640

January 31, 2018
Academical Year 2017-2018

1 Introduction

This report describes the implementation, and the related evaluation, of some rank fusion strategies. Namely, some basic strategies, described by Fox and Shaw in [1], and the Condorcet-fuse advanced strategy, described in [3].

In this section, the ground problem and the experiment guidelines are described. Then, in section 2, implementation details are given and, in section 3, the results of the evaluation are presented and commented. In the end, in section 4, the work meets its conclusions.

1.1 The ground problem

The rank fusion problem, or meta-search problem, consists in combining several runs of retrieved documents, to obtain a new run whose quality, in terms of IR measures, aims to be higher. Typically, the runs to be *fused* have documents in common. Otherwise, the rank fusion approaches here implemented lose in effectiveness and the problem itself may become trivial.

Additionally, we point out that, since the score scale for each run is different, a score normalisation method needs to be performed, in order to compare the different runs.

1.2 Outline of the experiment

The experiment has been conducted as follows:

Generation and normalisation of the runs: $n = 10$ different runs of retrieved document have been generated, resulting from n different retrieval systems (details in section 2.1), using Terrier [4]. Furthermore, the runs documents scores have been normalised (details in section 2.2).

Implementation of simple strategies: The basic strategies, described by Fox and Shaw in [1], have been implemented.

Evaluation of the basic strategies: The quality of the results obtained with the basic strategies have been evaluated, with trec_eval [6] C library.

Implementation of the advanced strategy: The Condorcet-fuse metasearch algorithm, presented by Montague and Aslam in [3], has been implemented.

Evaluation of the advanced strategy: As for the basic strategies, the results obtained have been evaluated using trec_eval [6].

2 Experiment phase

The whole project has been coded in Java, in this section some technical details are presented. The full code implementation is available at the GitHub link `https://github.com/palinao/metasearch-project.git`. Additional information in the repository README file.

2.1 Generation of the runs

As mentioned, the rank fusion strategies have been applied starting from $n = 10$ different runs. The runs have been generated with Terrier [4], retrieving documents from the TIPSTER disk 4&5 collection[1], used in TREC-7 campaign [5].

The runs have been obtained by using 10 different configurations, varying the model used and the presence or absence of a stemmer and a stop words list. This choice helped having

[1] Without Congressional Records

Model	Stop words list	Stemmer	Run filename
TF.IDF	✓	✓	`TD_IDF_0.res`
TF.IDF	✓	×	`TD_IDF_2.res`
TF.IDF	×	✓	`TD_IDF_5.res`
TF.IDF	×	×	`TD_IDF_8.res`
BM25	✓	✓	`BM25b0.75_1.res`
BM25	✓	×	`BM25b0.75_4.res`
BM25	×	✓	`BM25b0.75_6.res`
BM25	×	×	`BM25b0.75_9.res`
BB2	✓	×	`BB2c1.0_3.res`
BB2	×	✓	`BB2c1.0_7.res`

Table 1: Different runs configurations and related files.

more diverse runs, useful for the rank fusion purposes. The stemmer used is Terrier *PorterStemmer* implementation and the stop-list is the english default stop-list of Terrier. The different configurations are shown in table 1.

2.2 Runs normalisation

In order to compare the different runs, the documents scores have been normalised, according to the formula described by Lee in [2]:

$$\text{Min\_Max} = \frac{\text{old\_sim} - \text{minimum\_sim}}{\text{maximum\_sim} - \text{minimum\_sim}}$$

where, in each run, *old_sim* is the original relevance score, *minimum_sim* is the lowest relevance score among the documents for the considered topic and *maximum_sim* is, similarly, the highest one.

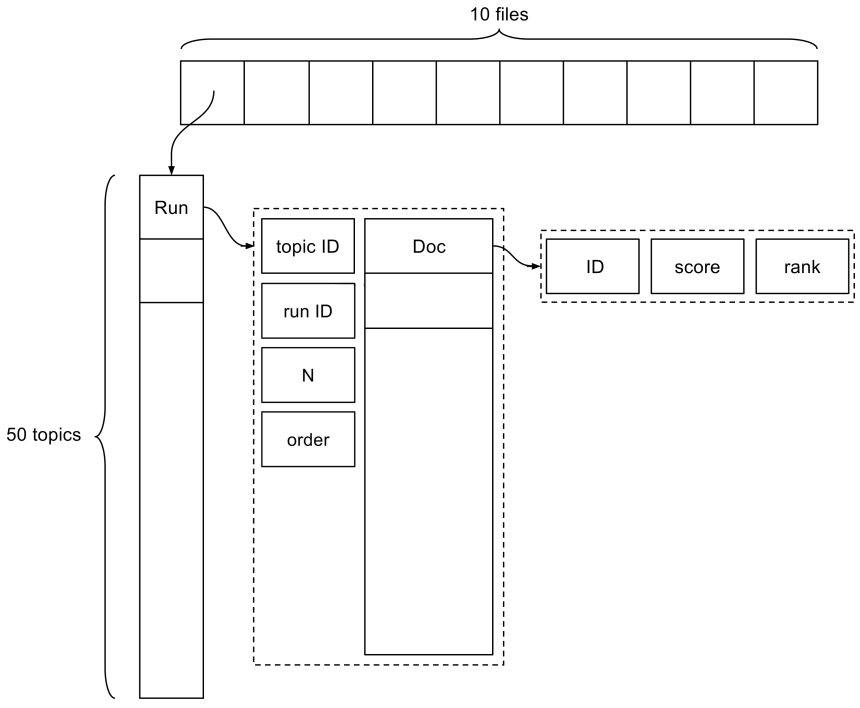

Figure 1: Data structure used for input runs.

Figure 2: Data structure used for scores sets.

2.3 Data structure

In order to proceed with the coding of the rank fusion strategies, the input run files had to be parsed. The parsed data have been stored in a matrix data structure, as depicted in figure 1.

The whole structure is a matrix of *Run* class. *Run* is a custom class that features five fields:

- *topicID*: *String* with the id of the topic which the run documents are associated to (e.g. topic 351)
- *runID*: *String* with the id of the run file which the run documents are associated to (e.g. BM25b0.75_1)
- *N*: integer with the number of the stored documents
- *order*: integer that encodes the current type of sorting (e.g. 1 = *by document id*)
- *docs*: an array of documents, whose class is *Doc*.

Where *Doc* class has then these fields:

- *ID*: the ID of the document
- *score*: the document score relevance in the run
- *rank*: the document rank in the run.

We underline that, inside *Run* class, the documents can be stored either by alphabetical order, according to their IDs, or by rank value, according to their relevance scores. This choice allows a binary search in the run, when the documents are sorted by ID, and smooths the output in the results files, when they are sorted by relevance score.

2.4 Basic strategies

Here the basic strategies, as originally presented in [1], are summarised. For each document, in the set of the relevance scores assigned by the n different systems:

CombMAX: select the maximum score

CombMIN: select the minimum score

CombMED: select the median score

CombSUM: sum all the scores

CombANZ: sum all the scores and divide by the number of non-zero entries

CombMNZ: sum all the scores and multiply for the number of non-zero entries.

To perform these strategies, the several runs have been parsed by topic and by document, to find a $n = 10$ elements sorted vector (one for each document), containing the relevance scores associated to a document by each run. The resulting array is depicted in figure 2. The first, middle and last elements represent the minimum, median and maximum scores, while the other three strategies are trivially computable. Note that for arrays with an even number of entries, the median value is the mean of the two central elements.

Algorithm 3 Condorcet-fuse.

1: Create a list L of all the documents
2: Sort(L) using Algorithm 1 as the comparison function
3: Output the sorted list of documents

Algorithm 1 Simple Majority Runoff.

1: $count = 0$
2: **for** each of the k search systems S_i **do**
3: If S_i ranks d_1 above d_2, $count{+}{+}$
4: If S_i ranks d_2 above d_1, $count{-}{-}$
5: If $count > 0$, rank d_1 better than d_2
6: Else rank d_2 better than d_1

Figure 3: Initial procedure and comparison procedure [3].

2.5 Condorcet-fuse strategy

We introduce the Condorcet-fuse strategy, designed, and first implemented, by Montague and Aslam [3]. In this strategy, the rank fusion problem is seen as a particular *election* problem. In an *election*, some *voters* express their preferences (or, like in this case, preference orders) to promote some *candidates*. In Condorcet-fuse, the *voters* are the runs to be merged and the *candidates* are the documents of each run. In this scenario, the *winners* will be the documents of the merged run and, to sort them, the Condorcet *voting* procedure is followed.

The Condorcet procedure defines the winner as the candidate(s) that beats or ties with all the other candidates in pair-wise comparisons. From this general idea, the authors proved that the final Condorcet ranking is both effective, in terms of IR measures, and efficiently computable, with an *"elegant"* algorithm.

Remanding to [3] for a complete description, the algorithm is implemented as a particular instance of the sorting algorithm quick sort, where the comparison function is the pair-wise competition between two candidates (i.e. documents). In figure 3, the pseudocode is given. Algorithm 3 is the main function, that implements quick sort, while Algorithm 1 is the comparison function, that defines the documents ranking.

The actual Java code implementation follows the guidelines established by the pseudocode.

3 How the strategies perform

Method	MAP
Condorcet-fuse	0.1886
CombMNZ	0.1883
CombSUM	0.1879
CombMax	0.1854
CombMed	0.1818
CombANZ	0.1781
CombMin	0.1703

Table 2: Standings of the different rank fusion methods, according to MAP value.

In this section, the results of the merged runs performance evaluation are presented. As for the Java code, the merged runs and the related IR measures are available at the same link https://github.com/palinao/metasearch-project.git.

The different runs have been evaluated with trec_eval [6]. For the scope of the project, Mean Average Precision (MAP) has been taken into account, as it is a general and good comparison

Method	R-Prec
Condorcet-fuse	0.2424
CombMax	0.2387
CombSUM	0.2374
CombMNZ	0.2362
CombANZ	0.2313
CombMed	0.2270
CombMin	0.2217

Table 3: Standings of the different rank fusion methods, according to R-Prec value.

measure. In table 2, the MAP value for each run is presented in descendant order. It can be stated that Condorcet-fuse outperforms the other methods, even though the difference with CombMNZ and CombSUM is not much remarkable. It is a good result, but we underline that it strongly depends on the original runs set and that every execution of the Condorcet-fuse algorithm leads to slightly different results, as it features a random component (i.e. the pivot element choice in the quick-sort algorithm).

Another relevant measure is the Precision at the Recall Base (R-Prec). In table 3, for each method, the mean of R-Prec among all topics is given. As for MAP, Condorcet-fuse shows better performance than the basic strategies and with an higher gap towards the others. However, we have to remark that this measure may be slightly less significant than MAP. In fact, it is an average value between R-Prec of different topics, each of which has a different Recall Base value.

4 Conclusions

In this report, the work behind the implementation of some rank fusion strategies has been described. First, the meta-search topic has been introduced and the experiments has been outlined. Afterwards, the more significant designing and implementing choices have been given out and, in the end, the results have been evaluated and discussed. We showed how, for the experiment data, Condorcet-fuse strategy performed better than the Fox and Shaw basic strategies, even though with not much remarkable difference. This result is in line with what we expected and, considering the low computational cost of the strategy, we can state that it is a good alternative to the other basic approaches that have been discussed.

References

[1] Fox E. A., Shaw J. A., *Combination of Multiple Searches.* TREC 1993: 243-252, 1993

[2] Lee J. H., *Combining Multiple Evidence from Different Properties of Weighting Schemes.* SIGIR 1995: 180-188

[3] Montague M. H., Aslam J. A., *Condorcet fusion for improved retrieval.* CIKM 2002: 538-548, 2002

[4] Terrier website, *terrier.org*, accessed December 20th, 2017

[5] TREC website, *http://trec.nist.gov/*, accessed December 21st, 2017

[6] TREC website, trec_eval download *http://trec.nist.gov/trec_eval/*, accessed January 19th, 2018