UNIVERSITY OF PADUA

School of Engineering

Master Degree in Computer Engineering

Course Name: Information Retrieval

Academic Year 2017-2018

Information Retrieval Project

Students:	*Student ID:*
Luca Rossi	1141231
Davide Storato	1153692
Roberto Tarantini	1134820
Andrea Ziggiotto	1140706

February 9, 2018

1 Introduction

The project consists in the development of some Rank Fusion algorithms and the subsequent evaluation of these algorithms. A TREC dataset that provides documents, topic and relevance judgements is used as collection and Information Retrieval Evaluation measures are applied to evaluate the performances. Rank Fusion algorithms combine a set of original runs to obtain a final run with the goal of having an higher ranking quality. In these experiments, six base methods proposed by E. Fox and J. Shaw [1] and one advanced method chosen from a list of proposals are used.

The base methods use a combination function that sets the new relevance score using the maximum (*CombMAX*), the minimum (*CombMIN*), the sum (*CombSUM*) and the median (*CombMED*) of individual relevance scores. Other metrics are derived by *CombSUM*: with *CombANZ* the new score is obtained by the division between *CombSUM* and the number of nonzero relevance scores; with *CombMNZ* the new score is obtained by the product between *CombSUM* and the number of nonzero relevance scores.

The advanced method that has been chosen is *ProbFuse*, proposed by D. Lillis, F. Toolan, R. Collier and J. Dunnion [3] and described in the Section 1.1. To make relevance scores comparable, the normalization formula proposed by J.H. Lee [2] is used:

$$Min\_Max = \frac{old\_sim - minimum\_sim}{maximum\_sim - minimum\_sim} \tag{1}$$

where *old_ sim* is the relevance score that has to be normalized and *minimum_ sim*, *maximum_ sim* are respectively the minimum and the maximum relevance score in a specific run. The normalization is applied independently on every run before the execution of base methods and ProbFuse method.

1.1 ProbFuse: A Probabilistic Approach to Data Fusion

The probabilistic approach described in [3] uses a probability calculated during a training phase to rank documents. In order to calculate these probabilities it is necessary to use different models and analyse their performance on a training set of Q queries. Each run, related to a different retrieval model, is divided into x segments. The probability that a document d belonging to a segment k is relevant, given a retrieval model m, can be calculated in two different ways.

The first method to calculate the probability is called *ProbFuseAll* and it is defined by eq. (2). $|R_{k,q}|$ is the cardinality of the set of documents in segment k that are judge to be relevant to query q, and $|k|$ is the cardinality of the set of document in segment k. In this computation it has been assumed that unjudged documents are nonrelevant. The second method is called *ProbFuseJudged* and it is developed by [3] in order to consider the case of incomplete relevance judgements. In this situation, some documents can be unjudged and so it is not possible to know if they are relevant or not to the given queries. The new eq. (3) ignores unjudged documents and so only take into account documents that have been judged to be either relevant or non relevant. $|N_{k,q}|$ is the cardinality of the set of documents in segment k that are judged to be non relevant to query q.

$$P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{R_{k,q}}{|k|}}{Q} \quad (2) \qquad P(d_k|m) = \frac{\sum_{q=1}^{Q} \frac{R_{k,q}}{|R_{k,q}|+|N_{k,q}|}}{Q} \quad (3)$$

$$S_d = \sum_{m=1}^{M} \frac{P(d_k|m)}{k} \quad (4)$$

The final ranking score S_d, computed for each document d, is given by eq. (4). Where M is the number of retrieval models being used and k is the segment number. If a model does not return a document in its results set, the probability is considered to be zero.

2 Description of the software

After importing selected runs, the software that implements Base Methods executes the relevance score normalization of each documents with the eq. (1) for every run. At this point, a list for each topic is created, containing all documents retrieved from each run for the specific topic. Moreover a group of *Record* instances is assigned to each document, where the class *Record* contains the rank and the relevance score for a specific run. All these information are gathered together using an *HashMap*, in which the key is the *Topic ID* while the value is the list of documents with their *Record*. In this way it is possible to compute more quickly the final fusion run. Cycling for every topic, on their documents list and so on *Record* instances, the final relevance score is computed applying methods mentioned in Section 1. After the computation of the new relevance score for every document, the software taking in consideration one topic at a time, sorts the documents and limits to 1000 the number of documents related to the topic. The last step provides the saving of the six new runs in six different text files in TREC format, in order to permit their evaluation.

The implementation of the *ProbFuse* algorithm begins with the import of the selected runs and the building of a data structure composed by an *ArrayList* of ten *HashMap*s (one of each run). Every *HashMap* is constituted by the *Topic ID* as key, and the list of the documents related to that topic as value. Afterwards, the *ProbFuse* algorithm receives the training set size as input, in order to compute the number of topic participating to the training phase. Topics are randomly extracted, added to a data structure similar to the one described above and removed from the initial data structure. In this way topics that are used in training phase are not used during the test phase.

In the training phase is necessary to know if a document is relevant or not relevant for a given topic, for this reason *Qrels* are imported. Qrels are saved in an *HashMap*, in order to guarantee a rapid access. The main task of the training phase is to build the matrix of the probability that a document in a given run and in a given segment is relevant. This matrix has dimensions *number of runs* per *number of segments* and every item of the matrix is computed with the eq. (3) in the case of *ProbFuseJudged* and eq. (2) in the case of *ProbFuseAll*. Runs are divided in segments and the matrix were populated computing the probability for each run.

In order to calculate the new relevance score of every documents, related to a specific topic, a new data structure is created. It is composed by an *HashMap*, where the key is the *Topic ID* and the value is another *HashMap*. The second *HashMap* has the *Document ID* as key and the list of Records with that *Document ID* as value. Considering that the class *Record* saves in its instances all the information related to each document included the belonging segment index, the computing of eq. (4) is efficient thanks to the new data structure.

Repeating this process to every topic, all new relevance score are calculated. Documents related to a topic can be sorted and only the first 1000 documents for every topic are saved. The last step provides the writing of the run in a text file respecting the TREC format.

3 Experiments and evaluation

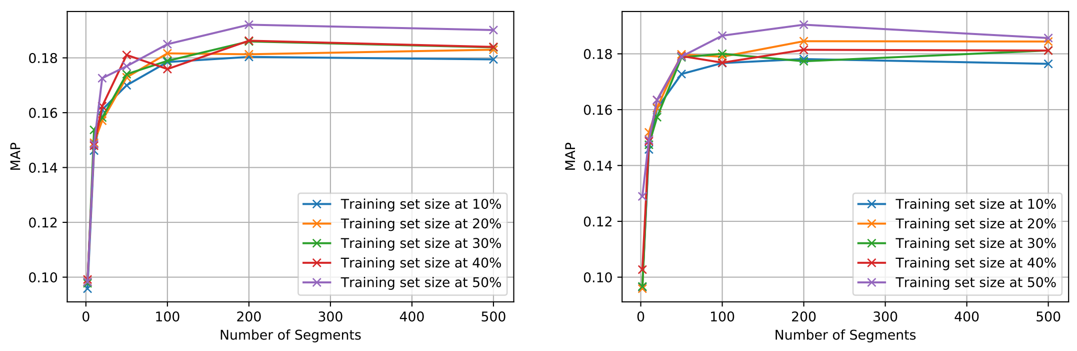
In this experiment documents are taken from TIPSTER disk 4&5, without congressional records, provided by TREC. Topics and relevance judgements are taken from TREC7 (topic number from 351 to 400).
As input 10 runs are used, instead of 10 different retrieval models. Runs are obtained by changing models (BM25, PL2, TF_IDF) and indexing parameters (using or not using the Porter Stemmer and StopWords removal) employing Terrier v4.1.
The generated runs are as follows:

- BM25: with stemming and stop word removal; with stemming; with stop word removal; without stopword removal and stemming;
- PL2: with stemming and stop word removal; with stemming; with stop word removal;
- TF_IDF: with stemming and stop word removal; with stemming; with stop word removal;

For some topics the returned documents are too few, hence it has been tried to set the Terrier parameter *ignore.low.idf.terms* to *false* (the default value for Terrier v4.1 is *true*), but without obtaining significant differences. To solve this problem, some tests have been done, checking the parameters for which the Information Retrieval System retrieved 1000 documents for each query. However these parameters are not found, so it is decided to accept that for some topic (not more than 5) there are few documents retrieved. To evaluate the performance of the developed system, the MAP measure is used to every run, executing the application *trec_ eval v8.1*.

ProbFuse methods need the estimation of optimal values for the training set size and the number of segments. In order to this, it was necessary to run ProbFuse algorithms many times with different combinations of parameters. In the first instance, ProbFuse algorithm has been evaluated changing the training set size value, belonging to $\{10, 20, 30, 40, 50\}$ %. For each of these values, the number of segments has been changed assuming values in the set $\{2, 10, 20, 50, 100, 200, 500\}$. For every pair ⟨*training_ set_ size, number_ of_ segment*⟩, ProbFuse algorithm has been executed 20 times and the pair's final MAP value is obtained doing the average between each single MAP value. With these results, it is possible to plot a graphic with five different trends, every trend is related to a training set size. This process has been done for both *ProbFuseAll* and *ProbFuseJudged* separately.

The graphics in fig. 1, obtained separately with *ProbFuseJudged* and *ProbFuseAll*, allow to see the best trend and choose it as optimal training set size. It is possible to observe that the curve related to the training set size at 50% lies above the others in most of the values on the x-axis, in particular at the point where the function assumes the maximum value.



(a) ProbFuseJudged *(b) ProbFuseAll*

Figure 1: MAP trend for different training set size for (average over 20 runs)

Once defined the optimal size it is necessary to proceed with the estimation of the best number of segments. To determine this value some tests have been done by varying the number of segments and keeping fixed the training set size to the optimal value. Fifty tests are performed and the overall measure of the MAP is computed averaging between 50 single MAP measure. The obtained results are represented in Figure 2, in order to see more clearly the trend of the values. The graphic shows as *ProbFuseJudged* performances are quite equal to *ProbFuseAll* probably because recovered documents not judged are too few.

The increasing of segments has the effect that calculated probabilities in the matrix *number of runs* per *number of segments* become more accurate, and so the MAP increases. In

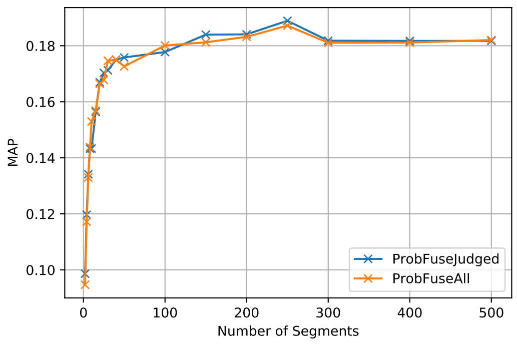


Figure 2: MAP at training set=50% (average over 50 runs)

fact with a small number of segments each one contains much documents and so the probabilities are not very significant. In the same way if the number of segments is too high, the documents are too few and the reliability of probabilities decreases. The peak that is possible to see in Figure 2, corresponding to 250 segments, is the ideal trade-off between the cases described above.

After completing these two phases, the optimal values for the algorithm parameters were determined. In this simulation the calibration of the parameters has provided as optimal value of training set size 50 and as optimal number of segment 250. Using the found parameters, it is possible to execute the ProbFuse software for 50 times and obtain an overall MAP measure, doing the average on the 50 single value of the MAP. These computed values are useful to compare the performance with base methods proposed by E. Fox and J. Shaw [1].

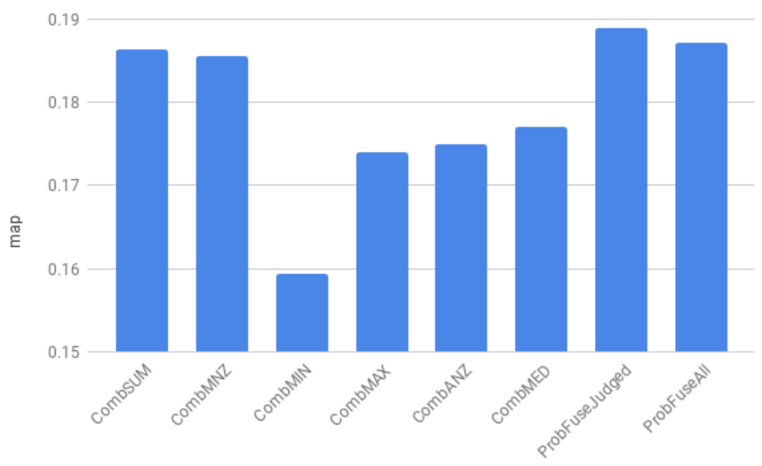


Figure 3: MAP value for the Comb Method run and ProbFuse run

The Figure 3 shows that only the base methods CombMNZ and CombSUM have good performance in terms of MAP. CombMAX have not very good performances because it tends to favour nonrelevant documents if they compare in some runs with high score. CombMIN is the worst method, probably because in the presence of a bad run, a relevant document is penalized if it has a low score in that run.

CombANZ and CombMED have a worse performance in comparison with CombSUM and CombMNZ, this happens probably because they do not consider if a document is present in all run or not. If an unrelevant document appears only in few runs, using CombANZ the score could be high because the average is made on its number of occurrences i.e if an unrelevant document has an high score and appears in only one run the average will be its initial score and it can be higher than scores of other relevant documents. In the same way, CombMED computes the median of the present score without penalty if a document compares too few times. Comparing the performance of the CombMNZ and CombSUM with the performance of the best single run, there is no significant improvements because runs used in this experiment are produced by only one *Information Retrieval System* and therefore there are no big differences between the considered runs.

4 Conclusions

The performances obtained by RankFusion methods are lower than those described in the considered papers. This happens because the runs are produced by the same Information Retrieval System and therefore there are no big differences between one and the others. This implies that if runs are only different permutations of the same documents, the resulting run will be only another ranking permutation of the initial documents, and so there are no significant improvements to the MAP because the number of relevant documents does not increase. In this case only the top-heavy behaviour of the measure creates the difference of the MAP. On the other hand, if the runs were made by many different documents there would be better results because the number of relevant documents would be greater and the value of the MAP would be much greater than that obtained from single runs.

It has been noted that the performance, in terms of MAP, of the ProbFuse in both the applications is very variable. This variability is attributed to the random choice of the topics used for the training, because they could be chosen not good and therefore do not accurately describe the collection. Despite the execution of a large number of tests in order to be able to find a reliable average value, the result continues to be not very stable.

References

[1] Edward A Fox and Joseph A Shaw. Combination of multiple searches. *NIST special publication SP*, 243, 1994.

[2] Joon Ho Lee. Combining multiple evidence from different properties of weighting schemes. In *Proceedings of the 18th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 180–188. ACM, 1995.

[3] David Lillis, Fergus Toolan, Rem Collier, and John Dunnion. Probfuse: a probabilistic approach to data fusion. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 139–146. ACM, 2006.