A Machine Learning based Approach to Evaluating Retrieval Systems

Huyen-Trang Vu and **Patrick Gallinari**
Laboratory of Computer Science (LIP6)
University of Pierre and Marie Curie
8 rue du capitaine Scott, 75015 Paris, France
{vu,gallinar}@poleia.lip6.fr

Abstract

Test collections are essential to evaluate Information Retrieval (IR) systems. The relevance assessment set has been recognized as the key bottleneck in test collection building, especially on very large sized document collections. This paper addresses the problem of efficiently selecting documents to be included in the assessment set. We will show how machine learning techniques can fit this task. This leads to smaller pools than traditional round robin pooling, thus reduces significantly the manual assessment workload. Experimental results on TREC collections[1] consistently demonstrate the effectiveness of our approach according to different evaluation criteria.

1 Introduction

The effectiveness of retrieval systems is often justified by benchmark test collections. A standard test collection consists of lots of documents, a set of information needs, called topics and human judgment about the relevance status of each document for a topic. Nowadays, it is relatively easy to gather huge set of millions of documents and hundreds of topics. The key obstacle for forming large sized test collections lies therefore in the topic assessment procedure. Assessing the whole document sets is unfeasible, even for small sized collection of 800,000 documents (Voorhees and Harman, 1999). In order to keep the assessment process practical, one often selects a certain number of documents for judgment. This is called (document) pooling and the outcome the pool or the qrels (*q*uery *r*elevance *s*et). The collected documents are then judged by humans, documents outside the pool are assumed non relevant. A representative pool is therefore essential to the whole evaluation process.

This paper proposes a method to form the assessment set with the support of machine learning algorithms. Based on relevance judgments of relatively shallow pools, a ranking algorithm will attempt to give priority for relevant documents so that the assessment set can be fixed at a feasible size without skewing the system evaluation result. The judgment process is indeed kept as much subjective-free as possible: the first relevance feeback step is designed appropriately so that the assessor cannot give any bias towards any particular rank or any system, the learning process is completely transparent to the assessors and parameters of the ranking function are collection-tailored rather than exported from previous collections.

The method will then be evaluated on TREC ad-hoc collections. Results from our comprehensive experiment confirm that the qrels generated by our method are much more representative than those of the same size by the TREC method. The outcome qrels is substantially smaller, so much cheaper to produce than the official TREC qrels, yet their conclusions about system effectiveness are quite compatible.

The remaining of this paper is organized as follows. We review related work in Section 2. Section 3 presents the general framework of applying machine learning techniques to forming test collections. We also give a brief introduction

[1]http://trec.nist.gov

about RankBoost (Freund et al., 2003) and Ranking SVM (Joachims, 2002b), the two learning algorithms used in our experiment. Section 4 introduces data sets and experimental setup. Section 5 is dedicated to present experimental results according to different evaluation criteria. Precisely, Section 5.1 shows the capacity of small pools on identifying relevant documents and Section 5.2 illustrates their impact on system comparison; Section 5.3 presents statistical validation tests. We conclude and discuss perspectives in Section 6.

2 Related work

2.1 TREC methodology

Since the seminal work of test collection forming in 1975 (Sparck Jones and Van Rijsbergen, 1975), pooling has been outlined as the main approach to form the assessment set. The simple solution of round robin pooling from different systems proposed in that report has been adopted in most existing IR evaluation forums such as TREC, CLEF[2] or NTCIR[3]. For convenience, we will denote that strategy as TREC-style pooling. To have the assessment set, from submissions (restricted length $L = 1000$ for most TREC tracks), only n top documents per submission are pooled. Despite different technical tricks to control the final pool size such as gathering only principal runs or reducing the value of n, the assessment procedure is still quite time-consuming. In TREC 8 ad-hoc track, for example, despite limiting the pool depth n at 100 and gathering only 71 of 129 submissions, each assessor has to work with approximately 1737 documents per topic (precisely, between 1046 and 2992 documents). Assuming that it takes on average 30 seconds to read and judge a document, the whole judgment procedure for this topic set can therefore only terminate after a round-the-clock month. Meanwhile, a simple analysis on the ad-hoc collections from TREC-3 to TREC-8 revealed that there are on average 94% documents judged as non relevant. Since most of existing effectiveness measures do not take into account these non relevant documents, it would be bettter to not waste effort on judging non relevant documents provided that the quality of test collections is always conserved. Several advanced pool sampling methods have been proposed but due to some common drawbacks, none of them has been used in practice.

[2]http://www.clef-campaign.org/
[3]http://research.nii.ac.jp/ntcir/

2.2 Topic adaptive pooling

Zobel (Zobel, 1998) forms the shallow pools according to the TREC methodology. When there are enough judged documents (up to the set of 30 top documents per run in his experiment), an extrapolation function will then be estimated to predict the number of unpooled relevant documents. The idea is to judge more documents for topics that have high potential to have relevant documents else. Carterette and Allan (Carterette and Allan, 2005) have recently replaced that extrapolation function by statistical tests to distinguish runs. This method produced interesting empirical outcomes on TREC ad-hoc collections, lack however a sound theoretical basis and is clearly of very high complexity due to iterative statistical tests of every run pairs. Furthermore, this incremental/online pooling approach raises a major concern about the unbiasness requirement from the human judgment as the assessors know well that documents come later are of lower ranks, thus of lower relevance possibility.

2.3 System adaptive pooling

Cormack et al. (Cormack et al., 1998) propose the so-called Move-To-Front (MTF) heuristic to give priority for documents based on the corresponding system performance. In their experiment, the latter factor has been simply the number of non relevant documents this system has introduced to the pool since the last relevant document. Aslam et al. (Aslam et al., 2003) formulate this priority rule by adopting an online learning algorithm called Hedged (Freund and Schapire, 1997).

Our method relies on this idea of pushing ahead relevant documents by weighting retrieval systems. There are however two major differences. Whilst all aforementioned proposals favor *online* paradigm with a series of human interaction rounds, our method works in batch mode. We believe that the latter is more suitable for this task since it eliminates as much as possible the bias introduced by human assessor towards any document. Moreover, the batch mode enables us to exploit intuitively the inter-topic relationship what is not the case of online paradigm. The second difference lies in the way of estimating the ranking function. It is widely accepted that machine learning techniques can deliver more reliable model on previously unseen data given much less training instances than any classical statistical techniques or expert rules can.

2.4 Generate *pseudo* assessment set

Several evaluation methodologies, especially for web search engines, have been proposed to evaluate systems *without* relevance judgment. These proposals can be grouped into two main categories. The first (Soboroff et al., 2001; Wu and Crestani, 2003; Nuray and Can, 2006) exploits *internal* information of submissions. The second (Can et al., 2004; Joachims, 2002a; Beitzel et al., 2003) benefits *external* resources such as document and query content, or those of web environment. We skip the second category since these resources are not available in generic situations.

Soboroff et al. (Soboroff et al., 2001) sample documents of a shallow pool (top ten documents returned by retrieval systems) based on statistics from past qrels. Wu and Crestani (Wu and Crestani, 2003), Nuray and Can (Nuray and Can, 2006) adopt metasearch strategies on document position. A certain number of top outcome documents will then be considered as relevant without any human verification. Different voting schemes have been tried in the two aforementioned papers. Their empirical experiment illustrated how the quality of these pseudo-qrels is sensible to the chosen voting scheme and to other parameters such as the pool depth or the diversity of systems used for fusion. They also confirm that *pseudo*-qrels are often unable to identify best systems.

In sum, the thorough literature review confirmed the importance of relevance assessment sets in IR evaluation yet the lack of an appropriate solution to have a reliable set given a moderate amount of judgment resource.

3 Machine learning based Pooling

3.1 General framework

Let M denote the topic set size available for the training purpose, N the number of participating systems, k_1 the pool depth to get the training data from any participating system and K the final pool size.

The training process consists of two main steps. Firstly, for each training topic, k_1 first documents of all N systems are gathered and the assessors are asked to assess all of these documents. Let $\mathcal{T}$ denote the outcome of this assessing step on all M topics. From the information of $\mathcal{T}$, a function f will then be learned which assigns to each document a value corresponding to its relevance degree for a given query.

At the usage time, for each given topic, the whole retrieved list of N systems will be fused. These documents will then be sorted in the decreasing order of their values according to f and the K top documents will be sent to the assessor for judgment. This last set of judgements will be the qrels used for the system evaluation.

In the training framework, it is clear that the second step plays the major role. An effective scoring function can substantially save the workload at the last assessment step. We will now focus on methods for estimating such scoring function.

3.2 Document ranking principle

The scoring function f can be estimated in different ways as seen in the last section. In this study, we adopt the learning-to-rank paradigm for estimating this scoring function. The principle of document ranking will be sketched in this section. The next sub-section will introduce the two specific ranking algorithms used in our experiment.

A ranking algorithm aims at estimating a function which describes correctly all partial orders inside a set of elements. An ideal ranking in information retrieval must be able to place all relevant documents above non relevant ones for a given topic. The problem can be described as follows. For each topic, the document collection is decomposed into two disjoint sets $\mathcal{S}_+$ and $\mathcal{S}_-$ for relevant (non relevant respectively) documents, R and NR are their cardinality. A ranking function $H(d)$ assigns to each document d of the document collection a score value. We seek for a function $H(d)$ so that the document ranking generated from the scores respect the relevance relationship, that is any relevant document has a higher score than any non relevant one. Let "$d \rhd d'$" signify that d is ranked higher than d'. The learning objective can therefore be stated as follows.

$$d_+ \rhd d_- \Leftrightarrow H(d_+) > H(d_-), \forall (d_+, d_-) \in \mathcal{S}_+ \times \mathcal{S}_-$$

There are different ways to measure the ranking error of a scoring function h. The natural criterion might be the proportion of misordered pairs (a relevant document is below a non relevant one) over the total pair number $R.NR$. This criterion is an estimate of the probability of misordering a pair $P(d_- \rhd d_+)$.

$$\mathbf{RLoss}(H) = \sum_{\substack{d_+ \in \mathcal{S}_+ \\ d_- \in \mathcal{S}_-}} D(d_+, d_-) \big[\!\big[d_- \rhd d_+ \big]\!\big] \quad (1)$$

$$= \sum_{\forall (d_+, d_-)} D(d_+, d_-) \big[\!\big[H(d_-) > H(d_+) \big]\!\big] \quad (2)$$

where $[\![\phi]\!]$ is 1 if ϕ holds, 0 otherwise; $D(d_+, d_-)$ describes the importance of the pair in consideration, it will be uniform $\left(\frac{1}{R.NR}\right)$ if the information is unknown.

In practice, we have to average RLoss over the training topic set. This can be done by either *macro*-averaging at topic level or *micro*-averaging at document pair level. For presentation simplification, this operation has been implicit.

3.3 Discriminative ranking algorithms

Since RLoss is neither continuous nor differentiable, its direct use as a training criterion raises practical difficulties. Also, in order to provide reliable predictions on previously unseen data, the prediction error of the learning function has to be bounded with a significant confidence. For both practical and theoretical reasons, RLoss is then often approximated by a smooth error function.

In this study, we will explore the performance of two ranking algorithms, they are RankBoost (Freund et al., 2003) and Ranking SVM (Joachims, 2002b). As far as we know, these algorithms are actually among a few state-of-the-art ranking learning algorithms whose convergence and generalization properties have been theoretically proved (Freund et al., 2003; Joachims, 2002b; Clémençon et al., 2005).

3.3.1 RankBoost

RankBoost (aka RBoost) (Freund et al., 2003) returns a scoring function for each document d by minimizing the following *exponential* upper bound of the ranking error RLoss (Eq. (2)):

$$\mathbf{ELoss}(H) = \sum_{(d_+, d_-)} D(d_+, d_-) e^{H(d_-) - H(d_+)} \quad (3)$$

This is an iterative algorithm like all other boosting methods (Freund and Schapire, 1997). The global ranking function of a document d is a linear combination of all base functions $H(d) = \sum_{t=1}^{T} \alpha_t h_t(d)$. At each iteration t, a new training data sample is generated by putting more weight $D(.,.)$ on difficult pairs (d_+, d_-). A scoring function h_t is proposed (it can even be chosen among the features used to describe documents) and the weight α_t is estimated in order to minimize the ELoss at that iteration.

RBoost has virtues particularly fitting the pooling task. First, it can operate on relative values. Second, it does not impose any independence assumption between combined systems. Finally, in the case of binary relevance judgment which usually occurs in IR, there is an efficient implementation of RBoost whose complexity is linear in terms of the training instance number (cf: the original text (Freund et al., 2003)).

3.3.2 Ranking SVM

Ranking SVM (Joachims, 2002b), rSVM for short, is a straightforward adaptation of the max-margin principle (Vapnik, 2000) to pairwise object ranking. The score function is often assumed to be linear in some feature space, that is $H(d) = \mathbf{w}^T \mathbf{\Psi}(d)$ where $\mathbf{w}$ is the vector of weights to be estimated and $\mathbf{\Psi}$ is a feature mapping. The max-margin approach minimizes the following approximation of RLoss:

$$\mathbf{rSVMLoss}(H) = \max \left\{ 1 + \left(H(d_-) - H(d_+) \right), 0 \right\} \quad (4)$$

for all pairs (d_+, d_-) while at the same time controlling the complexity of function space described via the norm of vector $\mathbf{w}$ for generalization objective.

Notice that rSVM does not explicitly support rank values as does RBoost. Nevertheless, we will see later that the discriminative nature allows rSVM to work quite well on features merely deduced from rank values. Its behavior difference is in fact ignorable in comparison with RBoost.

4 Experimental setup

Our method is general enough to be applicable to any ad-hoc retrieval information task where pooling could be useful. In this paper, we will however focus on TREC traditional ad-hoc retrieval collections. Experiments have been performed on the three corpora TREC-6, TREC-7 and TREC-8. Statistics about the number of runs, of judgments, of relevant documents are shown in Tab. 1. Due to limit of space, we will detail results on the TREC-8 case and only mention the results on the two others.

	#runs	#judgments	#rel. docs	Depth-5
TREC 6	79	1445.4	92.2	144.3
TREC 7	103	1606.9	93.5	114.6
TREC 8	129	1736.6	94.6	143.4

Table 1: Information about three TREC ad-hoc collections. The three last columns are averaged over the topic set size (50 topics/collection).

Training data is gathered from the top five answers of each run. The pool depth of five has been arbitrarily chosen to have both sufficient training data and to eliminate potential bias from assessors towards a particular system or towards early identified answers while judging a shallow pool. Furthermore, this training data set is large enough for testing the ranking algorithm efficiency.

Each document is described by an N-dimensional feature vector where N is the number of participating systems. The j^{th} feature value for a document is a function of its position in the retrieved list, ties are arbitrary broken. A document at rank i is assigned a feature value of $(L + 1 - i)$ where L is the TREC limit of submission run (L is usually set up at 1000). Documents outside submission runs receive the zero feature value (i.e. it is assumed to be at rank $(L + 1)$). For implementation speed, the input for rSVM is further scaled down to the interval $[0, 1]$.

Due to the small topic set size, we use a *leave-one-out* training strategy: a model will be trained for each topic by using judgments of all other topics. The training data set size is presented in the last column of Tab. 1. The workload for training dataset does not exceed the effort for assessing 5 topics in the full pool of TREC.

We employ SVM^{light} package[4] for rSVM. We adopt the efficient RBoost version for binary feedback and binary base functions h_t (cf. (Freund et al., 2003)), boosting is iterated 100 times and we impose positive weighting for all coefficients α_t.

The non-interpolated average precision (MAP) has been chosen to measure system performance[5]. This metric has been shown to be highly stable and reliable with both small topic set size (Buckley and Voorhees, 2000) and very large document collections (Hawking and Robertson, 2003).

RBoost and rSVM pools will be compared to the TREC-style pools of the same size. We also include "local MTF" (Cormack et al., 1998) in the experiment. The "global MTF" has been shown to slightly outperform the local version in the aforementioned paper. However, we believe that the global mode is merely for demonstration but unlikely practical of online judgment since it insists that all queries are judged simultaneously with a strict synchronisation among all assessors. Hereafter, for simplicity, the TREC-style pool of the first n documents retrieved by each submission will be denoted by Depth-n, the equivalent pool (with the same average final pool size m over the topic set) produced by RBoost, rSVM or MTF will be RBoost-m, rSVM-m or MTF-m respectively. In all figures in the next section, the abscissa denotes the pool size m and values of n will be present along the Depth-n curve.

5 Experimental results

This section will examine small pools produced either by the TREC method or by RBoost/rSVM/MTF from two angles: their pooling performance and their influence on system comparison result.

5.1 Identify *relevant* documents

Fig. 1 shows the ratio of relevant documents retrieved by different pooling methods (i.e. the recall). The curves obtained by RBoost and rSVM are quite similar and much higher than that by TREC methodology. The curve of MTF is in the middle of RBoost/rSVM and Depth-n at the beginning and then catches that of RBoost at the pools of about 600 documents.

[4] `http://svmlight.joachims.org`

[5] `http://trec.nist.gov/trec_eval/`

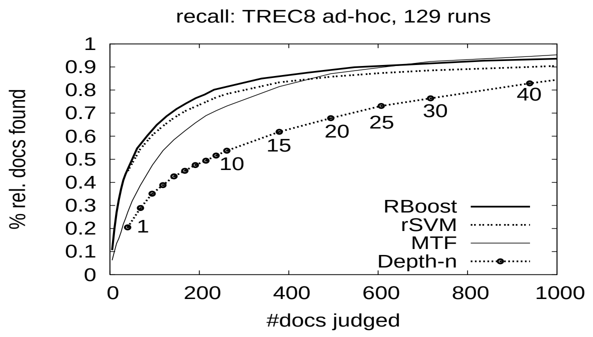


Figure 1: Along the incrementally enlarged pools: *relevant* documents identified in comparison with the full assessment set.

5.2 Correlation of system rankings

Once the pool is obtained by a given method, the assessor will give relevance judgment for all documents of that pool, called qrels for the outcome. This qrels will be used as the ground truth to measure effectiveness of a retrieval system.

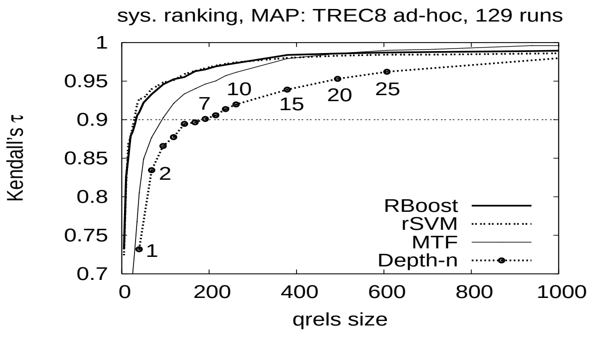


Figure 2: Kendall's τ correlation of system ranking according to different qrels methods in comparison with that produced by the full assessment set.

The simplest way to compare different systems is to sort them by the decreasing effectiveness values. The correlation of each two system rankings will then be quantified through a correlation statistic. In this study, we follow TREC convention (Buckley and Voorhees, 2004; Carterette and Allan, 2005), that is taking the 0.9 value of Kendall's τ as the sufficient threshold to conclude that the difference of two system rankings is ignorable. We compare here the system ranking obtained by the official TREC qrels with those by Depth-*n* where *n* varies from 1 to 100. We then replace Depth-*n* by RBoost-*m*, rSVM-*m* and MTF-*m*. The results are shown in Fig. 2 for TREC-8 and in Tab. 2 for the 7 first pool depths. We observe from the figure the similar order of pooling methods as seen in the previous section. The MTF curve meets those of RBoost and rSVM from qrels of more than 400 documents. The results obtained on the two collections of TREC-6 and TREC-7 are in line with those observed on TREC-8 (Tab. 2).

It is clear that system ranking correlation quantified by any rank correlation statistics provides necessary but not sufficient information about system comparison. Ranking systems by their sample means is indeed the simplest way with at least two implicit assumptions. First, runs have *similar variances*, this usually does not hold in practice even after discarding poorest runs. Second, all run swaps have the same importance without taking into account their *statistically significant difference* and their positions in the system ranking. In practice, swap of adjacent systems does not make much sense if they are not significantly different to each other according to statstical tests. The next section will be devoted to further statistical validations.

5.3 Statistical Validations

5.3.1 Significant difference detection

We register for a given qrels all system pairs which are significantly different on the topic set. The quality of a qrels can be measured by the similarity of this significant difference detection in comparison with that obtained by the official TREC qrels. We carry out the paired t-test for each pair of runs with 95% significance level. The recall and the false alarm rate of these detections are shown in Fig. 3. In terms of recall, RBoost and rSVM qrels are much more better than its TREC-style counterparts and MTF is in the middle. In terms of false alarm rate, there are some changes concerning rSVM and MTF. Precisely, rSVM at small qrels of less than 100 documents is the best whilst that is MTF qrels of more than 150 documents.

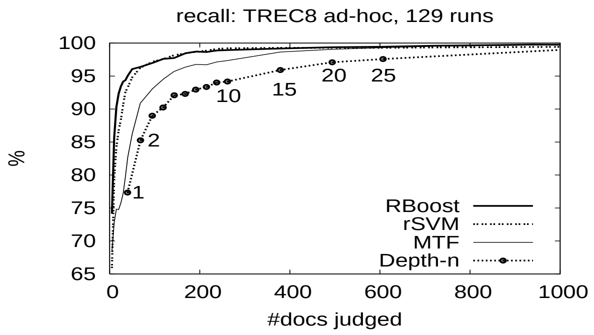
5.3.2 Tukey grouping

This multicomparison test[6] aims to group runs based on their statistical difference. We concentrate

[6]IR-STAT-PAK (Tague-Sutcliffe and Blustein, 1995)

	TREC-6 (79 sys.)					TREC-7 (103 sys.)					TREC-8 (129 sys.)				
n	m	D-*n*	MTF	SVM	RBst	m	D-*n*	MTF	SVM	RBst	m	D-*n*	MTF	SVM	RBst
1	37	.835	.843	.888	**.914**	32	.788	.809	.888	.891	40	.733	.805	**.927**	**.909**
2	66	.875	.899	**.925**	.934	56	.831	.890	**.920**	**.922**	68	.829	.877	.939	.933
3	93	.892	**.925**	.939	.956	76	.851	**.918**	.931	.935	95	.864	**.903**	.948	.946
4	118	**.903**	.940	.949	.967	95	.876	.926	.935	.947	119	.877	.921	.951	.953
5	144	.907	.949	.958	.972	115	.884	.936	.942	.954	143	.896	.933	.959	.955
6	170	.915	.953	.961	.974	133	.894	.942	.951	.957	168	.898	.940	.963	.963
7	195	.925	.959	.967	.977	152	**.903**	.950	.956	.962	191	**.901**	.946	.967	.966

Table 2: Kendall's τ obtained on small qrels. D-n: TREC-style Depth-n qrels, SVM: rSVM-m; RBst: RBoost-m.



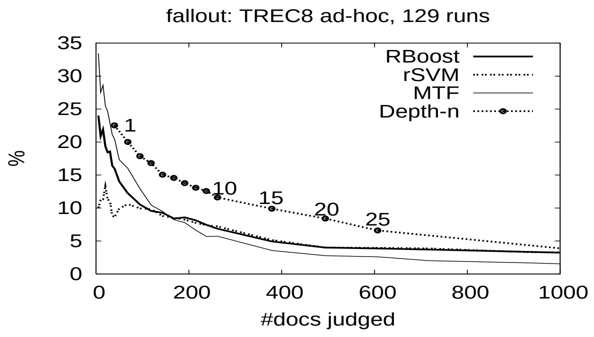


Figure 3: Comparing qrels of RBoost-*m*, rSVM-*m*, MTF-*m* and Depth-*n* in terms of pairs of significantly different systems: recall (top) and false alarm rate (bottom)

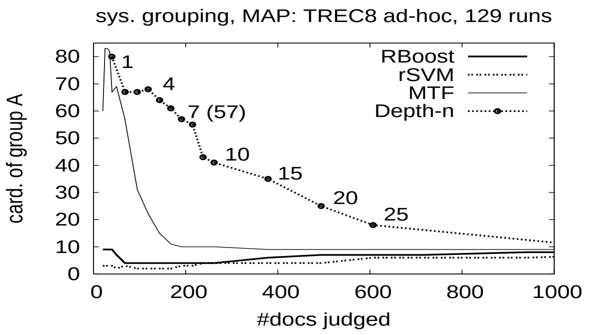


Figure 4: Cardinality of group A (95% confidence level) after the arcsine-root data transformation.

particularly on *the top group*, called group A which consists of runs on which there is *not enough evidence* to conclude that they are statistically significantly worse than *the top run*. In practice, this figure will be meaningful if it is around 10 (one often says about the top 10 runs). It will however become meaningless if the group A is too large, for example contains more than half of systems in consideration. Note that Tukey test relies on the assumption of Equality of Variances. This requirement can not be completely satisfied in practice, even after some data transformation such as arcsine-root or using rank values.

The size of group A on TREC-8 collection is shown in Fig. 4. We observe from that figure the stability of the two curves of RBoost and rSVM, this implies that the two qrels RBoost-35 and rSVM-35 which have both satisfied the 0.9 requirement of Kendall's τ can replace the official TREC qrels. The effort saving is therefore a factor of 50 (if ignoring the cost of training data set preparation) and of 10.5 otherwise. MTF needs qrels of at least 168 documents to produce comparable group A's with that of the official TREC qrels. The Depth-*n* pools however should not be recommended with less than 1000 documents in total (i.e. pooling more than 40 top documents per run).

6 Conclusions and Discussion

This study has well illustrated that two algorithms of RBoost and rSVM are quite suitable for qrels construction task. The final qrels are not only small enough to ask for human judgment but also result in reliable conclusion about system effectiveness in comparison with the counterpart of TREC methodology and that of MTF.

It is necessary to include other metasearch methods for further study. This will allow us to validate not only the impact of the metasearch training principle based on pairwise ranking error RLoss but also the capacity of automatic feature selection of the two ranking algorithms used in this paper.

This method needs to be further verified on challenging ad-hoc retrieval scenarios such as Terabyte, Web Topic Distillation or Robust Tracks in TREC context. The hardness of these scenarios involves two main issues. First, the number of document judged relevant varies largely across the whole topic set. Second, some topics might even have no relevant document in shallow pools. These matter any statistical inference on shallow pools.

Acknowledgement The authors thank M.-R. Amini, B. Piwowarski, J. Zobel and the anonymous reviewers for their thorough comments. We acknowledge NIST to make accessible the TREC submissions. This work was supported in part by the IST Programme of the European Community, under the PASCAL Network of Excellence, IST-2002-506778. The publication only reflects the authors' views.

References

[Aslam et al.2003] J.A. Aslam, V. Pavlu, and R. Savell. 2003. A unified model for metasearch, pooling, and system evaluation. In *Proc. CIKM'03*.

[Beitzel et al.2003] S. M. Beitzel, E. C. Jensen, A. Chowdhury, and D. Grossman. 2003. Using titles and category names from editor-driven taxonomies for automatic evaluation. In *Proc. CIKM'03*.

[Buckley and Voorhees2000] C. Buckley and E.M. Voorhees. 2000. Evaluating evaluation measure stability. In *Proc. SIGIR'00*.

[Buckley and Voorhees2004] C. Buckley and E.M. Voorhees. 2004. Retrieval evaluation with incomplete information. In *Proc. SIGIR'04*.

[Can et al.2004] F. Can, R. Nuray, and A. B. Sevdik. 2004. Automatic performance evaluation of web search engines. *Info. Process. Management*, 40(3):495–514, May.

[Carterette and Allan2005] B. Carterette and J. Allan. 2005. Incremental Test Collections. In *CIKM'05*.

[Clémençon et al.2005] S. Clémençon, G. Lugosi, and N. Vayatis. 2005. Ranking and scoring using empirical risk minimization. In *Proc. COLT'05*.

[Cormack et al.1998] G.V. Cormack, Christopher R. Palmer, and C.L.A. Clarke. 1998. Efficient construction of large test collections. In *Proc. SIGIR'98*.

[Freund and Schapire1997] Y. Freund and R.E. Schapire. 1997. A decision-theoretic generalization of on-line learning and an application to boosting. *J. Compt. Sys. Sci.*, 55(1):119–139, August.

[Freund et al.2003] Y. Freund, R. Iyer, R.E. Schapire, and Y. Singer. 2003. An efficient boosting algorithm for combining preferences. *J. Mach. Learning Res.*, 4:933–969, November.

[Hawking and Robertson2003] D. Hawking and S. Robertson. 2003. On collection size and retrieval effectiveness. *Information Retrieval*, 6(1):99–105.

[Joachims2002a] Th. Joachims. 2002a. Evaluating retrieval performance using clickthrough data. In *Proc. SIGIR wshop on Math./Formal Methods in IR*.

[Joachims2002b] Th. Joachims. 2002b. Optimizing search engines using clickthrough data. In *KDD'02*.

[Nuray and Can2006] R. Nuray and F. Can. 2006. Automatic ranking of information retrieval systems using data fusion. *Info. Process. Management*, 42(3):595–614, May.

[Soboroff et al.2001] I. Soboroff, Ch. Nicholas, and P. Cahan. 2001. Ranking Retrieval Systems without Relevance Judgments. In *Proc. SIGIR'01*.

[Sparck Jones and Van Rijsbergen1975] K. Sparck Jones and C. J. Van Rijsbergen. 1975. Report on the need for and provision of an ideal information retrieval test collection. Technical Report 5266, Computer Lab., Univ. Cambridge.

[Tague-Sutcliffe and Blustein1995] J. Tague-Sutcliffe and J. Blustein. 1995. A statistical analysis of the TREC-3 data. In *Proc. TREC-3*.

[Vapnik2000] N. V. Vapnik. 2000. *The Nature of Statistical Learning Theory*. Springer-Verlag.

[Voorhees and Harman1999] E.M. Voorhees and D. Harman. 1999. Overview of the Eighth Text REtrieval Conference (TREC-8). In *Proc. TREC 8*.

[Wu and Crestani2003] Sh. Wu and F. Crestani. 2003. Methods for Ranking Information Retrieval Systems Without Relevance Judgements. In *SAC'03*.

[Zobel1998] J. Zobel. 1998. How reliable are the results of large-scale information retrieval experiments? In *Proc. SIGIR'98*.