Learning to Choose
Automatic Selection of the Information Retrieval Parameters

Anthony Bigot[1], Sébastien Déjean[2], and Josiane Mothe[1,3]

[1] Institut de Recherche en Informatique de Toulouse, UMR 5505 CNRS
{anthony.bigot,josiane.mothe}@irit.fr
[2] Institut de Mathmatique de Toulouse, UMR CNRS
sebastien.dejean@math.univ-tlse.fr
[3] École Suprieure du Professorat et de l'Éducation

Abstract. In this paper we promote a selective information retrieval process to be applied in the context of repeated queries. The method is based on a training phase in which the meta search system learns the best parameters to use on a per query basis. The training phase uses a sample of annotated documents for which document relevance is known. When an equal-query is submitted to the system, it automatically knows which parameters it should use to treat the query. This *Learning to choose* method is evaluated using simulated data from TREC campaigns. We show that system performance highly increases in terms of precision (MAP), specifically for the queries that are difficult to answer, when compared to any unique system configuration applied to all the queries.

Keywords: Information retrieval, Meta search, Evaluation, Learning in IR, System combination, Repeated queries

1 Introduction

In information retrieval (IR), a ranking function is used to decide the document to retrieve according to a query as well as their order. Document and query similarity is a key component of the ranking function. Many parameters are involved in this process: the way documents are indexed, the possible query expansion, the similarity function itself, ... Systems vary because they use different parameters throughout retrieval.

Over queries, systems do not behave homogeneously and a system S_1 that answers well a query Q_1 can answer query Q_2 poorly, while another system S_2 does oppositely. As a result, it is impossible to define the set of parameters that will be used for any queries. Rather, most of the approaches optimize the parameters so the results are best in average over queries. Alternatively we propose to chose among a set of systems or system configurations the one that fits the best, on a per query basis.
This principle implies to learn a link between queries and systems. The aim of

the learning can be to associate query characteristics to systems. For example it is possible to represent queries by features, learn which system is best for each training query or query cluster. A new query can then be compared to learn queries considering the same features to select the system to be applied. [1] use this principal to associate weigthing schema to queries. We consider an alternative process in which learning is applied to individual query: in that case the training consists in associating the best system to each query. In practice, this latest principle is possible if the queries are known in advance only. This is indeed the case for repeated queries. In this paper we focus on repeated queries.

Several studies have shown repeated queries occur frequently in real applications. Teevan et al. have shown that 33% of queries are exact repeats of a query issued by the same user (equal-query queries) and that one query out of four leads to the same single user click [2]. Sanderson and Dumais studied 3.3 million queries and showed more than a half of them were repeated queries [3].

In this paper, we consider a method that could be used in real systems to treat repeated queries using a selective approach. We call the method "learning to choose". The learning to choose method implements a meta system that makes use of systems variability. It aims at deciding which system configuration should be used to treat a given query. To do so, the learning to choose method learns the best query-system association: it learns the system configuration to be used for a given query in order to optimize the results. This training phase uses a subset of documents for which the document relevance is known. After training, whenever equal-query queries occur, the learnt system configuration will treat the query over the entire document collection.

The paper is organized as follows. In section 2, we report some related works. Section 3 presents the learning to choose method.Section 4 describes the evaluation framework. Section 5 details a preliminary analysis that aims at selecting a subset of system configurations. In section 6, we consider the only systems selected as depicted in section 5 and analyse the results of Learning to chose according to topic difficulty. We show that learning to choose improves the results on the hardest topics. The last section concludes the paper.

2 Related works

There are various ways to take advantage of system varaibility. Without being exhaustive, this section presents the different types of methods.

Data fusion. Data fusion takes advantage of system variability by combining the ranked lists retrieved by different systems for a given information need [4]. The issues of data fusion are to decide both the ranked lists to fuse and the fusing function. Some works have shown that the systems to fuse should be independent [5], [6]. The same fusing function is generally applied whatever the queries are. However, the fusing model can also be learnt on a per query basis; in that case, the fusing function is query dependent [7]. Unlike data fusion which fuse various system results, learning to choose selects the best system configuration that should treat a query among several.

Learning to rank. Variability and learning are also core elements of learning to rank methods [8]. In the training phase of learning to rank, the examples to learn consist of ranked lists of relevant documents associated with the corresponding information needs. The ranking function is trained on these examples. The testing phase uses the single learned ranking function on new queries [9]. Unlike learning to rank, learning to choose does not learn a general function that would work whatever the query is. Rather, the principle of the method is to select the best system configuration on a per information need basis by analyzing the performance on a subset of documents; the learnt system configuration is then used on the entire set of documents.
Repeated queries. Some approaches process repeated queries by storing documents that should be retrieved. This is done for frequently asked queries for which the results are cached [10], mainly to answer time response issue. In that case, one assumes that the relevant documents remain the same. In addition, because new documents are continuously added or deleted, index updates lead to cache invalidation: cached results correspond to entries whose results have changed [11]. This cache method can also be used to answer queries that are close to previous queries [10]. Alternatively, some specific treatments can be done on a per query basis [12]. Since learning to choose does not store the results but rather the best way to obtain them, it can be applied in the context of dynamic collections.
Selective approaches. Finally, the closest related work is selective approaches. For example, He and Ounis [13] suggest a method to select among several term-weighting models depending on the query. Queries are described by features that are used to cluster them. Training associates the best term-weighting to each query cluster. When a new query is submitted to the system, it decides which cluster the query belongs to and process it by the corresponding system. In the same way Bigot et al. [14] suggest three methods that aim at deciding which system(s) would be the best to process a given query considering TREC participant systems as system candidates. All the three methods are based on a training stage in which for each query the decision is learnt using a sub-set of the document collection and the maximization of a performance measure (e.g. precision). Once trained the meta-system knows which system should process the query when the query will occur again; this system will then process the query over the entire document collection. The first method they called OneT2OneS (for One Topic to One System) selects the best retrieval system for each query. The two other methods are based on a first stage that clusters the systems prior to the training phase, then choose an ambassador for each cluster. They finally select the best cluster by selecting the ambassador that maximizes the performance measure. The two approaches that use system clustering are OneT2ClusterS (for One Topic to System Cluster) and ClusterT2ClusterS (for Cluster Topic to Cluster Systems). They differ in that ClusterT2ClusterS clusters topics (queries) according to their difficulty. Compared to [13] that learns some queries over the entire collection, [14] learns all the queries on a sample of documents. Our method is closer to the second approach since it targets re-

peated queries. In that context, queries are known and the problem is to chose the best system configuration to apply. Compared to [13] and [14] we do not use various systems but rather various retrieval configurations that do not imply various indexing. We think that our method is easier to use in real systems because of this. Moreover, we evaluate the method and show that a limited number of configurations are needed to improve results, specifically on difficult queries.

3 Learning to choose method for repeated queries

3.1 Description of the method

In our method, we make the hypothesis that system parameters lead to different system performances depending on the queries. Moreover, the method is based on the fact that for a given query, some system parameters will optimize the results. Thus our method aims at learning which system parameters should be used for a given query. However, IR parameters are not equally resource demanding. For example, making indexing parameters vary implies to index several times the collection which can be costly in terms of time and storage. For this reason, we rather make retrieval parameters vary. A given set of parameter values is called a system configuration. Our method implements a meta-system that choses among pre-defined system configurations on a per-query basis.

The next sections describe the steps of the method.

Training phase. The training phase consists in choosing which configuration should process each (repeated) query. Notice that the training query set consists on all the queries that have been identified as repeated and for which the meta-system should learn the best configuration. During the training phase we consider the various system configurations and the results they obtained on the query set using a sample of annotated documents. For these documents, we know if they are relevant to the query or not. Each system configuration treats the training document set over the query set. From the retrieved document lists we compute the evaluation measure for each query and each system configuration. The best configuration is then chosen for each query. The set of system configurations is an important parameter of our method which is analysed in detail in this paper.

Testing phase. After training, the meta-system knows which system configuration should process each query. Whenever an equal-query is submitted to the meta-system, it processes it using the learnt system configuration.

3.2 Defining the set of system configurations

Principle. The set of system configurations should be composed of configurations that perform well on average over queries. However, as we said previously, system variability can be important. For this reason, we add an additional constraint: to be kept a system configuration should obtain better results than the average on a subset of queries. Moreover, it is not mandatory that a given query is well performed by several system configurations; one being enough to ensure users' satisfaction. As a matter of fact, it is not needed to consider all possible system configurations; what is needed is to ensure that each query is well performed by at least one available system configuration. Considering the example given in table 1, we would like to select either system configuration S_1 or S_2 to ensure query Q_1 is well processed by at least one system configuration. Same goes for system configuration S_3 or S_4 for query Q_2.

Table 1: Illustration- Performances of 4 system configurations over two queries.

	S_1	S_2	S_3	S_4
Q_1	$\boxed{0.8}$	$\boxed{0.9}$	0.2	0.1
Q_2	0.2	0.1	$\boxed{0.9}$	$\boxed{0.8}$

System configuration selection. To start with, we dispose of various system configurations and the corresponding performance over the query set when using the document set. As we wanted to perform a quite large analysis, in our experiments, we consider 100 initial system configurations. However, the idea is to decide which system configurations are the most important ones. For defining the configurations to keep, we iteratively select the most important configuration as follows:

1. For each query, we compute the value of the performance measure that corresponds to the highest centile; the system configuration that gets this value becomes a candidate. For example, if we consider the mean average precision (MAP) as the performance measure, we calculate the MAP value MAP_{001} for which 1% of the MAP values will be higher than MAP_{001} and 99% of the MAP values will be lower than MAP_{001} for this query, whatever the system configuration is. Let us consider that the processed query is an easy one and that over 100 system configurations, the best system S_b got a MAP of 0.85, MAP_{001} for this query is set to 0.85; S_b becomes a candidate.
2. System configurations that have been selected in step 1 constitute the set of candidates for this first iteration;
3. For each configuration candidate, we compute the number of times it has been selected; that is to say the number of queries that selected each configuration. We keep the system configuration which is the most frequent. An additional condition to keep the system configuration is that its frequency is higher than a threshold. For example, S_b has been selected in step 1 by 10 queries and is the most selected configuration and given a threshold of 6 queries minimum, S_b is definitively kept;
4. The queries that selected this configuration are removed from the query set. Considering our example, the 10 queries that choose configuration S_b are removed.

5. We go back to step 1 considering all system configurations and the remaining queries (in our example using $Number\ of\ queries - 10$).

If there is no system configuration selected the process is iterated considering the second centile (the performance value for which 98% of the values are higher than it), third centile, ... If less than T queries remain unmatched, they have to all be selected at once (the centile criteria is relaxed). This rule is used to avoid selecting hardest topics one by one while centile is decreasing.

4 Evaluation framework

Evaluation collection. Evaluating our method in a real context implies to have access to real data (repeated queries, documents, document relevance); such a collection is not freely available. For this reason, we simulated the environment by evaluating our method on data from TREC (Text Retrieval Conference[4]). The collection we used (TREC7 and TREC8 adhoc sets) is composed of approximately 528155 documents (2 gigabytes), a set of 100 natural language topic statements and the associated relevance judgments. In our experiment, any TREC topics are considered as repeated information needs. Topics are composed of a title, a description and a narrative part and correspond to information needs. Notice that, since the methods are evaluated on TREC, we distinguish topics and queries in the rest of the paper: topic is used to refer to a TREC topic; a query is the internal representation used by the search system and is built from a topic. QRel consists of document relevance judgements for each topic. By confronting them to the list of retrieved documents, it is possible to compute performance measures to evaluate system configuration performance. We ran different runs with the Terrier platform [1] by making parameters varying. To be more realistic in usability, we considered a single indexing (which is the most resource demanding); and thus indexing parameters are set once. Those runs correspond to system configurations.
Evaluation Measures. We use the TREC evaluation software trec_eval[5], version 9.0 that computes many measures. Evaluation measures have been shown to be highly correlated [15]. In this work, we only consider average precision (AP) for each topic and mean average precision (MAP) over topics. AP is the result of averaging precision (number of relevant retrieved documents over retrieved documents) each time a relevant document is retrieved. MAP characterises a system configuration by averaging AP over topics.
Training. The training phase of the learning to choose method is applied to a subset of documents (training set). On this document subset, the various system configurations are used to process the 100 topics which are all considered as repeated queries to be learnt. To decide the best configuration for each topic, we used the QRel from which we removed the documents that do not belong to

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

the sample of training documents. In our experiments, we partition the document collection according to a training part of ⅔ and a testing part of ⅓. The repetition of queries is simulated by treating the same topic on test documents. Evaluation uses the QRel (after removing the training documents from it).
Cross Validation. To evaluate the learning to choose method, we applied cross validation. Indeed, a single partitioning of the data as presented in the previous subsection is not enough to make solid conclusions. For that reason, we performed 10 different partitions of the collection each composed of a training part of ⅔ and a testing part of ⅓ randomly selected [16]. Then we average the results over the 10 collections.
Baselines. The baseline is the best system configuration over the queries. The mean of all ten baselines is the general baseline (it was mandatory since the set of documents varies and thus the MAP varies as well depending on the draw).

5 System configuration selection

Preliminary study to select system configurations. Terrier allows to choose modules from the three steps of the IR process:

– Indexing (stemmer, size of blocks, empty documents ignored or not);
– Matching (retrieving model, topic fields to use, low idf terms ignored or not);
– Query expansion (QE) (QE model, number of terms to add to the query, number of top documents to consider for QE, number of top documents a term should appear in to be used during QE).

Indeed, indexing is costly in terms of time processing, disk space and updates when creating an IR stream. For that reason, we consider a single index and the only parameters that we make varying are matching and query expansion parameters. The indexing phase uses the default Terrier stopword list, Porter stemmer algorithm, a block size of 1 and ignores empty documents. Indeed using different stopword lists does not have significant impact on the results but using one (rather than none) does significantly improve the results [**?**]). Fuller et al. [17] showed Porter stemmer is the most accurate. Also Compaore et al. [18] showed that block size different from 1 and empty documents does not change the results significantly.

Concretely, we considered many configurations but kept only the 100 ones that get a MAP over 0.2 for at least 50% of the topics. This threshold of 0.2 is set to ensure that the performance of the selected system configurations are good enough.
Impact of the number of system configurations. In real applications, using 100 system configurations can be costly. For that reason, we study the impact of the number of system configurations to use. We consider various selection thresholds (10, 20, 30...100 configurations). One solution could be to randomly select the systems. However this solution would not take advantage of system variability. For this reason, we prefer to select system configurations that behave differently on the topics. To do so, an hybrid clustering (hierarchical clustering

combined to K-means) is applied so system configurations of each cluster lead to close performance in terms of AP for the topic set on the training document set. For each cluster the best configuration is selected to be used in the Learning to choose process. This method insures that the selected configurations are different enough. We then consider several thresholds: we analyse the results when 100% of the configurations are used (no clustering), when 90% of the intial configurations are used (meaning that we cluster the configurations into 90 clusters), 80% ... until 10% (meaning that only 10 configurations are used, corresponding to 10 different clusters or configuration profiles).

Figure 1 shows the results of this experiment. Grey lines corresponds to the training phase, and black lines to the testing phase. Dashed line is the baseline and plain line with dots correspond to the average over all ten sub-collections as explained in 4. Dotted lines reprensent the result of a random selection. Notice that a random selection leads to the expected value of AP and so to the averaged MAP over systems.

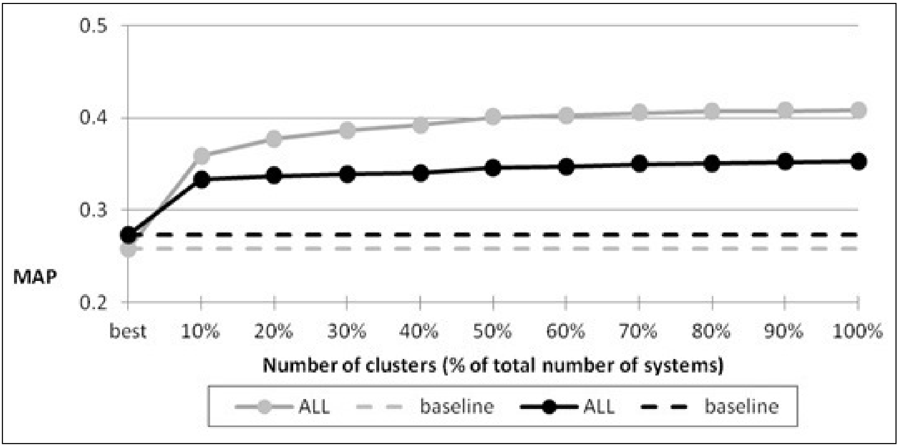

Fig. 1: Analysis of the number of different system configurations.

Results presented in figure 1 show that using only 10 different system configurations, improves the results by 22%. Using more than 10 sytem configuration candidates does not improve the results very much. From this preliminary study, we conclude that 10 configurations are enough to significantly improve the performance but less can also be enough.

In the following, we select a minimal set of system configurations (less than 10) to make further analysis on the behaviour of the selection technique on a small sample of configurations.

Minimal System Configuration Set. To select a minimal set of systems, we apply the method presented in 3.2 to the previous set of 100 systems.

In figure 2, the black line is the number of selected system configurations and the gray line is the average MAP; average MAP is considering the MAP of the systems selected for each topic during the selection. Figure 2 shows the lower the T is, the more specific the selection is, and the better the results are.

However, we do not want to be too specific because that leads to selecting too many system configurations; for that reason we consider 10 configurations or less.

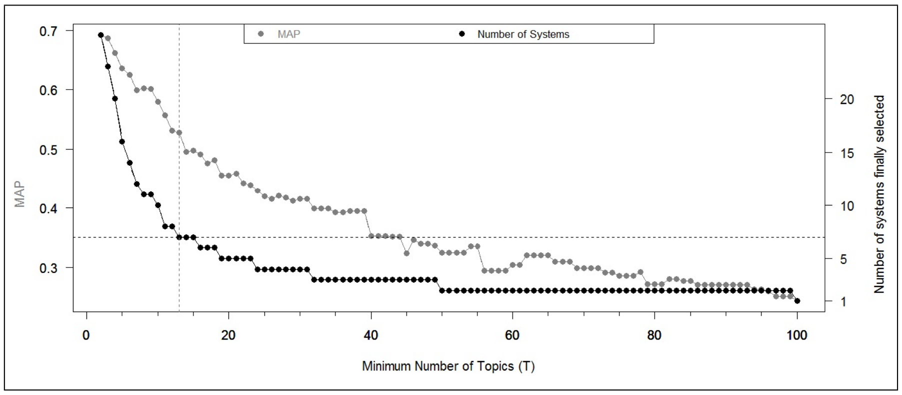

Fig. 2: Number of selected system configurations (right axis) & performance (left axis) according to T.

Figure 3 displays the average MAP in function of the number of system configurations; 10 being the maximum number of system configurations we allow to be selected by the algorithm. As a compromise between effectiveness and the number of selected system configurations, we choose 7 possible configurations.

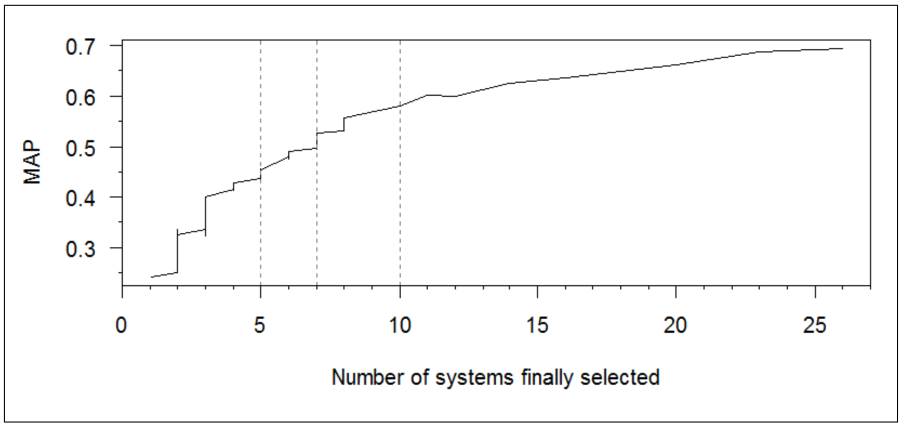

Fig. 3: Average performance according to number of selected system configurations.

This corresponds to 3 different possibilities of T (see the dots hovered by horizontal dashed line on figure 2) and the best MAP (0.528) is obtained for T=13 (vertical dashed line). The set of these 7 configurations is used for evaluating the learning to choose method in section 6.

6 Evaluation of Learning To Choose in a Close-to-Reality Context

In this section we evaluate learning to choose on a small set of systems. Topics are seen as new during the training phase and they are considered repeated information needs during the testing phase. To avoid randomness effect that could lead to one lucky or unlucky experiment, we proceed a cross validation on ten different splits of the document collection as explained in 4.

Topic Difficulty. As we ran 10 experiments, the number of topic clusters may change from one experiment to another. To be able to merge the results, we need to keep the same number of clusters for all 10 experiments. Each cluster is characterised by a difficulty label according to AP systems obtained. Among the 10 experiments, we observed that 4 to 6 levels of topic difficulty would be optimal for the Learning to chose method. To fix the number of difficulty level to consider, we run the hierarchical clustering on data computed with the full collection of documents and conclude that 5 difficulty levels is an optimal choice.

Results of the Selection. As the number of topics in a difficulty level varies along the 10 experiments, the overall average MAP by cluster is weighted with the number of topics included in the considered level. This is done for both the metasystem and the baseline. Figure 4 displays the average mean of MAP by topic difficulty level during the test phase. Topics on which systems performed poorly are qualified "hard", those on which they performed less poorly are qualified "medium hard" and so on for "medium easy" level, "easy" level and finally the "easiest" level on which systems performed greatly.

Black dots correspond to the metasystem's map and grey dash show the baseline MAP and are read on the left axis (with black guidelines). Grey squares show the relative difference between the metasystem and the baseline and are read on the right axis (with grey guidelines).

We show that performances are poorly increased on easy topics; that can be explained by the fact it is hard to improve system performances when they already are really high. At least, performances are not degraded. Performances on medium easy topics are improved by 12%; according to the t-test, the increase is significant for 6 experiments out of 10.

The best improvment is observed for the medium hard topics with an increase of 44% (increase is significant for 9 experiments). Finally, the hardest topics are improved by 43%. However the absolute increase is little (0.04), it is still significant for 8 experiments.

7 Conclusion and Future Works

Using more than ten different systems for a fusion technique is not realistic. First, we showed a method easy to reproduce to select a subset of systems. Systems are selected on the base that whether or not their performances belong to the highest centile for a defined number of topics. Matched topics are successively deleted from the selection until no topics remain. The analysis of this selection

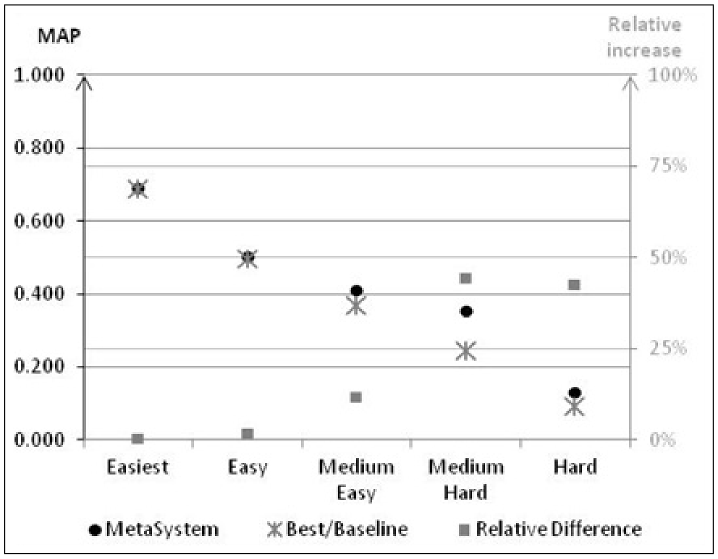

Fig. 4: Fusion Results according to Topic difficulty

method leads to select seven systems in order to balance performances and real conditions. Even if the last matched topics are poorly performed by the selected systems, due to the selection method itself, the evaluation is not degraded.

Learning to choose is then evaluated on the seven systems and cluster of topics are defined according to system performances: the better system performances are, the easier the topic are qualified. The evaluation shows that, based on a given set of systems, a fusion method is able to improve retrieval performances from 12% on medium difficulty topic to 44% on hard topics. Since results of the information retrieval process are not stored, the method is able to deal with dynamic collections.

In this study, query clusters are defined after queries have been processed at least once. This is possible in the context of repeated queries. However, such a method is not able to deal with queries submitted for the first time. Actually, future works should focus on applying difficulty prediction techniques. Such techniques can be based on linguistic predictors [12] or on different predictors such as query ambiguity [19].

Acknowledgments. We would like to acknowledge the French ANR agency for their support through the CAAS-Contextual Analysis and Adaptive Search project (ANR- 10-CORD-001 01)

References

1. Ounis, I., Amati, G., Plachouras, V., He, B., Macdonald, C., Lioma, C.: Terrier: A high performance and scalable information retrieval platform. In: Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006). (2006)
2. Teevan, J., Adar, E., Jones, R., Potts, M.A.S.: Information re-retrieval: repeat queries in yahoo's logs. In Kraaij, W., de Vries, A.P., Clarke, C.L.A., Fuhr, N., Kando, N., eds.: SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007, ACM (2007) 151–158
3. Sanderson, M., Dumais, S.T.: Examining repetition in user search behavior. In Amati, G., Carpineto, C., Romano, G., eds.: Advances in Information Retrieval, 29th European Conference on IR Research, ECIR 2007, Rome, Italy, April 2-5, 2007, Proceedings. Volume 4425 of Lecture Notes in Computer Science., Springer (2007) 597–604
4. Fox, E.A., Shaw, J.A.: Combination of multiple searches. In: Proc. TREC-2. (1994) 243–249
5. Croft, W.B.: 1. In: Combining approaches to information retrieval. Kluwer Academic Publishers (2000) 1–36
6. Wu, S., Bi, Y., Zeng, X., Han, L.: Assigning appropriate weights for the linear combination data fusion method in information retrieval. Inf. Process. Manage **45**(4) (2009) 413–426
7. Wilkins, P., Ferguson, P., Smeaton, A.F.: Using score distributions for query-time fusion in multimediaretrieval. In Wang, J.Z., Boujemaa, N., Chen, Y., eds.: Proceedings of the 8th ACM SIGMM International Workshop on Multimedia Information Retrieval, MIR 2006, October 26-27, 2006, Santa Barbara, California, USA, ACM (2006) 51–60
8. Liu, T.Y., Joachims, T., Li, H., Zhai, C.: Introduction to special issue on learning to rank for information retrieval. Inf. Retr **13**(3) (2010) 197–200
9. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In Ghahramani, Z., ed.: Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007. Volume 227 of ACM International Conference Proceeding Series., ACM (2007) 129–136
10. Baeza-Yates, R., Gionis, A., Junqueira, F., Murdock, V., Plachouras, V., Silvestri, F.: The impact of caching on search engines. SIGIR '07 Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval (July 2007) 183–190
11. Blanco, R., Bortnikov, E., Junqueira, F., Lempel, R., Telloli, L., Zaragoza, H.: Caching search engine results over incremental indices. In Rappa, M., Jones, P., Freire, J., Chakrabarti, S., eds.: Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010, ACM (2010) 1065–1066
12. Mothe, J., Tanguy, L.: Linguistic analysis of users' queries: towards an adaptive information retrieval system. In: SITIS 07 Third International IEEE Conference on Signal-Image Technologies and Internet-Based System, 2007, HAL - CCSD (June 17 2007)
13. He, B., Ounis, I.: Inferring query performance using pre-retrieval predictors. In: 11th International Conference, SPIRE 2004, Proceedings, Padova, Springer Berlin Heidelberg (2004) 43 – 54

14. Bigot, A., Chrisment, C., Dkaki, T., Hubert, G., Mothe, J.: Fusing different information retrieval systems according to query-topics: a study based on correlation in information retrieval systems and TREC topics. Information Retrieval Journal **14**(6) (2011) 617–648
15. Baccini, A., Djean, S., Mothe, J., Lafage, L.: How many performance measures to evaluate information retrieval systems ? Knowledge and Information Systems **30**(3) (2011) 693–713
16. Freund, Y., Schapire, R.E., et al.: Experiments with a new boosting algorithm. In: ICML. Volume 96. (1996) 148–156
17. Fuller, M., Zobel, J.: Conflation-based comparison of stemming algorithms. In: In Proceedings of the Third Australian Document Computing Symposium. (1998) 8–13
18. Compaoré, J., Déjean, S., Gueye, A.M., Mothe, J., Randriamparany, J.: Mining information retrieval results: Significant IR parameters (regular paper). In: Advances in Information Mining and Management, IARIA (2011)
19. Chifu, A.: Prédire la difficulté des requêtes : la combinaison de mesures statistiques et sémantiques (short paper). In: Conférence francophone en Recherche d'Information et Applications (CORIA), Neuchatel, Suisse, 03/04/2013-05/04/2013, www.unine.ch, Université de Neuchâtel (avril 2013) 191–200