Defining an Optimal Configuration Set for Selective Search Strategy - A Risk-Sensitive Approach

Josiane Mothe, Md Zia Ullah

Defining an Optimal Configuration Set for Selective Search Strategy – A Risk-Sensitive Approach

Josiane Mothe
IRIT UMR5505 CNRS, INSPE, Univ. de Toulouse
Toulouse, France
josiane.mothe@irit.fr

Md Zia Ullah
IRIT UMR5505 CNRS
Toulouse, France
mdzia.ullah@irit.fr

ABSTRACT

A search engine generally applies a single search strategy to any user query. The search combines many component processes (e.g., indexing, query expansion, search-weighting model, document ranking) and their hyperparameters, whose values are optimized based on past queries and then applied to all future queries. Even an optimized system may perform poorly on some queries, however, whereas another system might perform better on those queries. Selective search strategy aims to select the most appropriate combination of components and hyperparameter values to apply for each individual query. The number of candidate combinations is huge. To adapt best to any query, the ideal system would use many combinations. In the real world it would be too costly to use and maintain thousands of configurations. A trade-off must therefore be found between performance and cost. In this paper, we describe a risk-sensitive approach to optimize the set of configurations that should be included in a selective search strategy. This approach solves the problem of which and how many configurations to include in the system. We show that the use of 20 configurations results in significantly greater effectiveness than current approaches when tested on three TREC reference collections, by about 23% when compared to L2R documents and about 10% when compared to other selective approaches, and that it offers an appropriate trade-off between system complexity and system effectiveness.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**; **Retrieval effectiveness**; **Information retrieval query processing**.

KEYWORDS

Adaptive information retrieval, Query driven parameterization, Learning to rank, Search engine parameters, Risk sensitive systems

ACM Reference Format:
Josiane Mothe and Md Zia Ullah. 2021. Defining an Optimal Configuration Set for Selective Search Strategy – A Risk-Sensitive Approach. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM '21), November 1–5, 2021, Virtual Event, QLD, Australia.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3459637.3482422

1 INTRODUCTION

According to the American Customer Satisfaction Index [26], users of World Wide Web search engines are generally satisfied with the results they obtain. Behind this general satisfaction, however, lie disparities in user satisfaction with individual searches. According to comparative analyses of the performances of various search engines on test sets carried out in evaluation forums such as TREC (Text Retrieval Conference, `trec.nist.gov`), even those that perform best on average over all the queries still fail on some queries, whereas others may perform better on these queries even if, on average, they perform less well [22, 37]. User satisfaction would increase further if the search engine being used performed equally well on all queries.

A search engine performs several distinct component processes. These include indexing to extract the document terms that will be used for query-document matching, automatic query reformulation, search weighting to decide which documents to retrieve, and ranking retrieved documents. A variety of models have been proposed to perform these component processes (e.g., Bo1, Bo2 [1], etc. for query reformulation). Each model has hyperparameters that influence system effectiveness and can take a variety of values (e.g., the number of terms to be added to the query in query expansion). When building a system, the model for each component must be chosen and its hyperparameters set optimally. Current practice is to optimize them experimentally; system effectiveness is maximized by a combination of component and hyperparameter values based on past searches or training queries [29]. This is done for each collection separately by using Grid search [36, 47], Line search [32], Bayesian optimization [29] or transfer learning [33]. Once optimized, the same system is then used for all future queries. This ensures the best performance on average for the training queries, but not for individual queries.

The first attempt to improve the performance of search engines for individual future queries was the introduction of a *selective query expansion* (SQE) [2, 9, 14, 24, 53, 55], which applies query expansion only to those queries that will benefit from it. SQE considers two search strategies, one with automatic query re-formulation and one without. SQE's use of only two search strategies did not improve search engine performance very much, however, thus it is no longer used. Subsequently, *selective search strategy* approaches have increased the number of candidate configurations (i.e., the set of chosen component processes and their hyperparameter values), the system can choose from to process a new query [3, 8, 16, 23, 54].

The problem of deciding which candidate configurations to include in a selective search strategy is difficult, because the number of parameters that can influence a search is very large [20, 40], hence

the number of possible configurations is also very large. For example, Deveaud *et al.* used 20,000 candidate configurations, which makes the approach impractical for real-world search engines [16]. The main problem of Deveaud et al. is an adhoc way of candidate selection, which may cause the risk of choosing non appropriate configurations. The second problem is to manage a very large pool of configurations while a smaller number would be enough. There are obvious advantages to reducing the configurations to a manageable number. If it were possible to use a considerable smaller set of candidate configurations that work well for different queries, the configuration selection approach would be usable in practice, especially for web search engines which need to be updated regularly. This paper focuses on this specific problem.

Here, we develop a new greedy approach that can select from a large pool of possible configurations those that should be integrated as candidate configurations for a future selective search strategy. During the training phase on training queries, our greedy approach iteratively adds one representative configuration at a time according to the principle that a configuration should be useful. Our risk-sensitive approach applies a risk-reward trade-off function to evaluate whether each added configuration increases system effectiveness on some queries and minimizes the risk of poor performance. Moreover, we incorporate this process into a selective search strategy that decides which of the candidate configurations should handle a new query. We evaluate our method on three TREC collections using different well-established measures of effectiveness and show that it is more effective than the best configuration of the pool and more effective than other strong baselines.

2 RELATED WORK

Selective search strategy. A selective search strategy aims to select the most appropriate search or system configuration among many to apply to each individual query [23]; it thus differs from *data fusion*, which fuses the retrieved document lists obtained by several system configurations [21], but still uses a single strategy to process all the queries [21, 25, 27, 41]. It also differs from *double fusion*, which fuses the retrieved document lists over query variations for a single system (query fusion) and over multiple systems for a single query (system fusion) [6]. Finally, it differs from *distributed selective search*, which avoids searching the entire corpus for each query [38]. This section focuses on selective search strategy approaches.

For each query, the *selective query expansion* approach selects one of two configurations for each query: either the configuration with query expansion or the original, unexpanded query; most previous studies on SQE focused on how this decision is made. Cronen-Townsend *et al.* and Amati *et al.* estimated how much the results of the expanded query strayed away from the sense of the original query to decide whether it should be expanded [2, 14]. The two configurations used in SQE had only limited effectiveness. Xu *et al.* [54] improved SQE such that the system chooses between different query expansion strategies according to the type of query. They defined three types of query: ambiguous queries, queries about a specific entity and broader queries, and proposed different methods to treat each type of query. Kevyn Collins-Thompson [13] introduced a constraint optimization framework for SQE by reducing the expansion risk, without hurting the average gain.

Previous attempts to use a selective search strategy have considered more than two configurations. Arslan and Dinçer [3] used eight configurations consisting of eight term-weighting models in which the hyperparameters were first optimized by using Grid search. In this approach, the selection of a configuration to process a new query is based on the frequency distributions of query terms on the document collection. By considering more configurations in selective search strategy the system performance improved [3, 54] when compared to SQE.

The most effective selective search strategy approaches make the configuration selection decision based on training by machine learning from queries [16, 23, 40, 53] rather than making the decision based on a query score as was done previously [3, 14, 54]. He and Ounis [23] introduced a model selection approach in which queries were clustered considering three pre-retrieval features and the best-performing retrieval model was attributed to each cluster. For a given query, the retrieval model attributed to the closest cluster was then selected to treat the query. The per-parameter learning (PPL) method of Mothe and Washha [40] predicts the best configuration to use by considering independently each of its component processes and hyperparameters. PPL trains a multi-class classifier for each component process and hyperparameter such that each class corresponds to a model (in the case of components) or a value (in the case of hyperparameters). Seven such classifiers were trained corresponding to more than 80,000 configurations in total. These classifiers were then used to predict the best component and hyperparameter configurations for an unseen query, considering its features. An independently trained classifier for each parameter may not effectively model system performance since the parameters might be mutually dependent [20].

Xu *et al.* [53] focused on the automatic query expansion component of selective search strategy. They adapt the standard *learning to rank* documents model whose purpose is to rank a sample of documents retrieved by a search system according to their supposed relevance by learning from examples of query-document preferences [30, 48]. The adaption was that the model does not learn document ranking but, rather, it learns to rank the candidate expansion terms based on query-term preferences. In Deveaud *et al.* the configuration selection for new queries was also cast as a problem of learning to rank [15, 16]. Their model learns to rank a set of configurations according to their potential ability to retrieve relevant documents for a given query based on examples of query-configuration preferences. They considered various components and hyperparameters and built more than 20,000 configurations. Although this approach has been shown to be effective, it is not applicable in practice because the huge number of configurations it considers is too costly in terms of computing time and maintenance.

None of the related work considers the problem of the choice of the candidate configurations in selective search strategy. In many of the previous studies just a few configurations were considered but their choice was not argued. When there were too many to be handled, they were selected randomly.

In this paper, we solve this problem with a risk-reward-based method that pre-selects a manageable number of the candidate configurations. A commercial search engine cannot maintain thousands of search configurations or thousands of query reformulations and

search components, specifically because they also need to be regularly tuned according to the evolution of queries. Our method requires that only a small, manageable number of configurations be maintained. Once the set of candidate configurations is obtained, then any selective search strategy can be applied.

Risk-sensitive criteria. In information retrieval, risk has been defined as "the risk of performing a given particular query less effectively than a given baseline system" [6, 7, 17, 18, 51]. Risk-sensitive functions have been studied in the context of *learning to rank* documents where the same search strategy is applied for any query and a single document ranking function is learned. Wang and Zhu [50] presented a risk-averse ranking algorithm by considering the mean-variance analysis of a ranked list, inspired by portfolio theory. Wang *et al.* [51] defined F_{risk}, which estimates the average reduction in effectiveness by using a given document ranking model rather than a reference model. They proposed a risk-reward trade-off function, U_{risk}, to directly optimize a *risk-sensitive learning to rank model* that lowers the risk and enhances the reward of the document ranking model performance. It is used to select the best unique ranking model to be applied to any query.

Other variants of risk-reward trade-off functions have been proposed in the literature based on F_{risk} [7, 17–19] to circumvent the problem of U_{risk} that it is unclear whether the estimated loss of a system over a baseline is a "real risk" statistically [17]. Dinçer *et al.* [17] introduced T_{risk}, an inferential version of U_{risk} that follows a Student's t-distribution. Later, Dinçer *et al.* [18] also proposed Z_{risk} and G_{risk} to compare the risk-reward trade-off of a system against multiple baselines.

De Sousa *et al.* [46] incorporated risk functions for feature selection in *learning to rank* documents. They compared the models obtained by considering different subsets of features in order to select the most important ones. Benham *et al.* [6] introduced the risk function to estimate the risk-reward trade-off in rank fusion. Later, Benham *et al.* tried an S-shaped weighting function instead of the linear weighting function in risk measures; however, they found no conclusive differences in risk sensitivity [7]. They also suggested a naming convention and a reversed signed version of the existing risk measures so that higher value corresponds to higher risk. Also, Benham *et al.* [5] studied the inferential behavior of risk measures and the stability of its confidence intervals. They found that the distribution of risk-adjusted scores is asymmetrical, undermining the normality assumption of the t-test statistic used in T_{risk}, for example.

In approach described below, we develop a risk-sensitive criterion that aims to determine which candidate configurations should be included in a pool for a selective search strategy. We develop the appropriate risk and reward functions, which are based on overall system effectiveness optimization.

3 RISK-SENSITIVE CRITERIA TO SELECT CANDIDATE CONFIGURATIONS

For the rest of this paper, *configuration* refers to a specific setting of an ensemble of components and their hyperparameters e.g., BM25 with Bo2 query expansion using 5 documents and 10 added query terms. Each ensemble defines a possible configuration for the pool. *Candidate configuration* is a configuration from the initial pool that has been selected to be included in the meta-system that uses this configuration pool and a selective search strategy. We use *selective search strategy* when referring to the search that selects a candidate configuration for each individual query. We use *parameter* as a generic term for component or hyperparameter.

3.1 Overview

The usual selective search strategy framework comprises two phases [2, 3, 15, 16, 23]:

- *Configuration pool*: the meta-system contains a pool of configurations it can choose among (first phase);
- *Selective search strategy*: the meta-system selects the configuration from the pool to be applied for the current query, using the "best fit" principle (final phase).

Either the configuration pool is limited to 2-8 configurations [2, 3, 23, 53] and hence needs no further limitation, or it contains 20,000-80,000 configurations [15, 16, 40].

The main contribution of this paper is to define a third phase in between the two mentioned above:

- *Candidate configuration selection*: a limited number of configurations are selected from the initial pool as candidate configurations for the selective search strategy (intermediate phase).

On the one hand, a large candidate configuration pool increases the potential of selective search strategy. On the other hand, too many candidate configurations makes the system inoperable in real life systems. It also increases the risk of overfitting when training the selective search strategy process if there are many more configurations than query examples. Although the selection step is crucial for applicability when there are too many configurations to choose among, previous studies have not identified a means to select from a pool. In the next sub-section we describe the risk-reward functions we have developed to meet this challenge.

3.2 Risk/reward function

The risk and reward functions are used to reduce the configuration pool to a limited number of candidate configurations for selective search strategy.

Not all configurations are equally good or poor. Some configurations may be poor in terms of effectiveness, whatever the query is, some configurations may be good for some queries but poor for others, some may be appropriate for just a very limited number of queries that other configurations process equally well, some configurations may be very similar to others and therefore redundant, the specific setting of certain parameters may also not be important in certain contexts (given the setting of other parameters), etc.

We hypothesize that the more candidate configurations are available for the meta-system to choose, the higher the overall model effectiveness (in case of appropriate selection), but the higher the risk (in case of bad selection). Moreover, the more configurations there are, the costlier the meta-system in terms of time complexity; the cost of maintaining and ranking more configurations. To solve the problem of the effectiveness-cost trade-off, we need an optimized configuration set i.e., a limited number of complementary configurations in the meta-system.

We developed a greedy approach by starting from an effective configuration and iteratively adding complementary effective configurations to the pool. Intuitively, a straightforward choice for selecting the first configuration is the configuration that maximizes the average effectiveness over the training queries. The second configuration should be chosen as complementary to the first: it should be more effective than the first configuration, at least for certain queries (it can receive a *reward* for this) and at the same time, it should not negatively affect queries (i.e., bring *risk* into the system) if the selective search strategy model chooses it by mistake for a given query. Hence, the process of selecting the candidate configurations can be modeled as a trade-off between minimizing the risk when the meta-system does not obtain the best candidate configuration and maximizing the reward when the meta-system selects the best candidate configuration for a given query.

Below, we define the risk-sensitive approach for selective search strategy. Wang *et al.* [51] defined risk in the context of learning to rank documents whose purpose is to decide on the document ranking for a given query as "the risk [for the system] of performing a given particular query less effectively than a given baseline system" [6, 7, 17, 18]. More formally, they defined F_{Risk} as follows:

$$F_{Risk}(Q_T, M) = \frac{1}{|T|} \sum_{q_i \in Q_T} \max(0, B(q_i) - M(q_i)) \quad (1)$$

where Q_T is the training query set, $B(q_i)$ is the baseline effectiveness for query $q_i \in Q_T$, and $M(q_i)$ is the effectiveness of the document ranking model for which the risk is estimated. F_{Risk} thus estimates the average reduction in system effectiveness by using a given document ranking model rather than a reference model. They use this principle to select the best unique ranking model which can be applied to any query.

Inspired by the F_{Risk} measure, we have defined a function for selecting candidate configurations for selective search strategy. This function measures the risk associated with selecting for a given query the configuration c_j rather than the reference configuration c_r. The risk relates to c_r being better than c_j.

The risk function we define is $Eff_{Risk}(c_j)$, which accumulates the risk relative to queries in terms of effectiveness for the training query set. $Eff_{Risk}(c_j)$ is defined in eq. (2) where Q_T is the training query set, $p(c_r, q)$ is the performance (effectiveness) of the reference configuration c_r for the query q, and c_j is a configuration from the initial pool. $Eff_{Risk}(c_j)$ hence accumulates the loss in effectiveness in relation to queries when the configuration c_j is selected, whereas the reference configuration c_r would have been the better choice: it corresponds to the maximum possible risk.

$$Eff_{Risk}(c_j) = \frac{1}{|T|} \sum_{q_i \in Q_T} \max(0, p(c_r, q_i) - p(c_j, q_i)) \quad (2)$$

In addition to the risk function, we define the corresponding reward function. This is based on the potential increase in overall effectiveness (Eq. 3) using configuration c_j. The reward function is defined as follows:

$$Eff_{Reward}(c_j) = \frac{1}{|T|} \sum_{q_i \in Q_T} \max(0, p(c_j, q_i) - p(c_r, q_i)) \quad (3)$$

and aggregates the improvement in effectiveness that would result if the system takes account of c_j for the queries, where c_j performs better than the reference configuration c_r.

Once the risk and reward have been defined, we have to adapt formulas of Eqs. 2 and 3 to fit the problem of selecting a set of candidate configurations that can be used in the selective search strategy meta-system.

Let $\mathcal{R}$ be the entire initial configuration pool and S_{k-1} be the set of configurations that have already been selected at step k with an initial $S_0 = \{c_r\}$, which is a point of reference or the first selected configuration. Q_T is the set of training queries and $p(c_k, q_i)$ denotes the retrieval effectiveness (e.g., nDCG@10) for the query $q_i \in Q_T$ processed by the configuration c_k.

Given $\mathcal{R}$ and S_{k-1}, we define the risk for selecting the new configuration $c_k \in \mathcal{R} \setminus S_{k-1}$ to be added to S_{k-1} at step k using Eq. 2 in terms of effectiveness as follows:

$$E_{RISK}(c_k, S_{k-1}) = \frac{1}{|T|} \sum_{q_i \in Q_T} \max(0, \max_{c_j \in S_{k-1}} (p(c_j, q_i)) - p(c_k, q_i)) \quad (4)$$

where $\max_{c_j \in S_{k-1}} p(c_j, q_i)$ is the maximum effectiveness for the query q_i in relation to the set of configurations that have already been selected in S_{k-1}. In Eq. 4, the risk of adding the configuration c_k is measured as the cumulative decrease in effectiveness which the meta-system can achieve if it chooses c_k rather than the best configuration in S_{k-1} for each of the training queries: it therefore adapts Eq. 2.

Likewise, we define the reward function using Eq. 3 in terms of effectiveness as follows:

$$E_{REWARD}(c_k, S_{k-1}) = \frac{1}{|T|} \sum_{q_i \in Q_T} \max(0, p(c_k, q_i) - \max_{c_j \in S_{k-1}} p(c_j, q_i)) \quad (5)$$

The overall gain for the configuration $c_k \in \mathcal{R} \setminus S_{k-1}$ in relation to the set of training queries and already selected configurations S_{k-1} is defined as:

$$Gain(c_k, S_{k-1}) = Reward(c_k, S_{k-1}) - (1 + \alpha) Risk(c_k, S_{k-1}) \quad (6)$$

where the functions $Reward(c_k, S_{k-1})$ and $Risk(c_k, S_{k-1})$ refer to the effectiveness-based Eqs. 4 and 5, respectively. The $\alpha \geq 0$ is a risk-sensitive parameter that controls the trade-off between risk and reward. In our case, we set α as 0 to weight risk and reward equally. We keep a statistical analysis of this risk-sensitive parameter for future work [5, 18].

Finally, at step k we select the configuration c_k^* which maximizes the overall gain according to the following equation:

$$c_k^* = \operatorname*{argmax}_{c_k \in \mathcal{R} \setminus S_{k-1}} \left(Gain(c_k, S_{k-1}) \right) \quad (7)$$

We then update S_{k-1} as follows:

$$S_k = S_{k-1} \cup \{c_k^*\} \quad (8)$$

where S_k is the set of k risk-sensitive configurations selected for a set of training queries Q_T.

The risk-sensitive criteria model we propose is generic enough to be applied to any selective search strategy approach. In the next section, we present the selective search strategy approach based on a learning to rank algorithm.

3.3 Best query-configuration fit

The aim of the risk-sensitive criteria we have defined is to optimize the set of configurations to be used in the selective search strategy meta-system. When the meta-system has an optimized set of configurations to use, it must then decide which one to use for a given query.

For the selective search strategy part, we use learning to rank (L2R) algorithms to rank the configurations as suggested in Deveaud *et al.* [16]. Usually, L2R is used to rank documents on a per-query basis [10]: after training on past queries, the system is able to rank the retrieved documents for any new individual query. In L2R documents, a single configuration system is used to retrieve the initial set of documents and training is based on query-document pairs along with document relevance for these pairs or preferences. Deveaud *et al.* [16] adapted this principle for selective search strategy as follows. In the standard L2R documents model, training is composed of query-document pairs, along with their relevance or preferences, whereas in the L2R configurations model, the training data is composed of query-configuration pairs, along with the associated retrieval effectiveness. After training on past queries, the system is able to rank the configurations for any new individual query.

The principle is thus to train a ranking model $r(q_i, c_j) = r(f_{i,j})$ to assign a score to a given query-configuration pair (q_i, c_j) i.e., a given feature vector $f_{i,j}$. More generally, the ranking model can rank all the configurations for a given query q_i. In this case, the ranking model is $R(q_i, \mathcal{S}) = R(\mathbf{f_i})$, where $\mathcal{S}$ is the set of configurations and $\mathbf{f}_i = (f_{i,1}, f_{i,2}, \cdots, f_{i,|\mathcal{S}|})$ is the set of feature vectors for the query-configuration pairs. Like the L2R documents model, the ranking model $R(q_i, \mathcal{S})$ is learned from the training data by minimizing the loss function $\mathcal{L}(r; \mathbf{f}, \mathcal{S})$.

We adopted the model of Deveaud *et al.* in the final, selective search strategy phase. More precisely, given the set of training queries, we considered the only configurations selected using our risk-sensitive criteria. Then, for each individual training query, we trained the model to rank the configurations according to their effectiveness and kept only the top-ranked configuration. When the model was trained and given a new (test) query q_{t+1} for which we wanted to know the configuration to use, we built the feature vectors $\mathbf{f}_{t+1}$ for the candidate configurations only, used the trained model that predicts the ranking scores of the candidate configurations and obtained the one that gave the highest score. As an alternative to the L2R model of Deveaud *et al.*, we also formulated the query-configuration matching problem as a multi-class classification problem but this worked less well than L2R for this task, probably because many configurations were similar to each other, hence it is easier for a model to rank them rather than to use a one-vs-all classification approach. Thus, we report here only on the L2R approach.

4 EVALUATION

4.1 Data collection and evaluation measures

To evaluate our contributions, we considered three standard TREC collections from the Adhoc tasks. TREC78 which consists in approximately 500K newspaper articles, WT10G composed of approximately 1.6 million Web/blog page documents, and GOV2 that includes 25 million web pages. The TREC test collections also include topics which "standard" format comprises a topic ID, a title, a description and a narrative. The title contains two or three words on average that represent the keywords a user could have used to send a query.

In our experiments, a query is composed of the topic title only. There are 100 topics in TREC78 (merging topics from TREC7 and TREC8), 100 topics in WT10G, and 150 topics in GOV2. Finally, the collections provide *qrels* i.e., judged documents for each topic, which is used by the evaluation program *trec_eval*[1] to calculate the effectiveness for Adhoc.

We used common evaluation measures to estimate the retrieval effectiveness for a query with a configuration from the above mentioned three collections; these measures are the same as those used to label the query-configuration examples: AP (Average Precision), nDCG@10 (normalized discounted cumulative gain at the cut-off rank 10) and P@10 (precision at the 10 cut-off documents).

To evaluate the models we used 2-fold cross-validation on query sets in all the experiments as follows: half of the queries (let us call this query set Q_A), were used for training while the other half, $Q_{\overline{A}}$, was used for testing. Then, conversely, $Q_{\overline{A}}$ was used for training while Q_A was used for testing. We used 3 draws to randomly split queries into Q_A and $Q_{\overline{A}}$ and averaged the results. To prepare the folds for three trials, we randomly shuffled the queries using R's sample function (random seed=42) and divide the query set into two subsets for each trial. We also indicate the standard deviation across the three trials in order to show how robust the results are. The same splits are applied no matter what method we use which needs a training phase i.e., for our methods and some of the baselines (see subsection 4.3). We follow this 2-fold cross-validation for multiple trials approach because of the limited number of queries available in each collection [43, 45].

4.2 Generation of a pool of configurations

Configurations are built by varying the IR components and some of their hyperparameters. We have limited ourselves to the study of 2 main IR components: the term weighting model and automatic query expansion components from which we consider different variants from the literature. We also consider different values of the hyperparameter related to query expansion. The model we developed is generic enough to be applied also when considering other search components and/or other values of component hyperparameters. In this study, we thus did not include the indexing component because we think it would be too costly to maintain different indexes in practice. We did not include the learning to rank document component and keep this for future research.

For the *retrieval component*, we selected the 22 different models (e.g., LM, BM25) that are implemented in Terrier[2], the tool we used in our experiments (See Table 1). Each of the models come with a series of hyperparameters. We used their defaults values that have been optimized in Terrier on TREC collections and did not apply an extra -costly- grid search to optimize them. We could have also considered the hyperparameters for each retrieval model e.g., k_1

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

[2] https://github.com/terrier-org/terrier-core/blob/5.x/doc/configure_retrieval.md

Table 1: Components and models & query expansion hyperparameter values for defining candidate configurations.

Search weighting model
BB2, BM25, DFRee, DirichletLM, HiemstraLM,InB2, InL2, JsKLs, PL2, DFI0, XSqrAM, DLH13, DLH, DPH, IFB2, TFIDF, InexpB2, DFRBM25, LGD, LemurTFIDF, InexpC2
Query expansion model
No, KL, Bo1, Bo2, KLCorrect, Information, KLComplete
QE Hyperparameter and Values
of Exp. doc.: 2, 5, 10, 20, 50, 100
of Exp. terms: 2, 5, 10, 15, 20; Min. # of doc.: 2, 5, 10, 20, 50

and b for the BM25 model, and μ for the LM model, but we will retain this more in-depth analysis and related experimental work for a future study.

For the query expansion component, we selected the 7 different models. Automatic query reformulation comes with crucial hyperparameters (the number of: documents, terms used in query expansion, and documents in which the added terms should occur) for which we considered different values (See Table 1).

Combining all these retrieval and query reformulation models and hyperparameters from Table 1 results in more than $20,000$ configurations. This configuration pool is the one used as input to our candidate configuration selection model. The risk-sensitive function is thus used to reduce the set of candidate configurations by selecting the best ones in relation to the set of training queries. In this way, the parameters can influence each other within the same configuration and we do not need to address the problem explicitly. We consider various query-configuration examples to both select the most interesting candidate configurations and train the query-configuration selective search strategy model.

Query-configuration training examples follow a vector-based representation: the features $f_{i,j}$ depend on the query (q_i), the configuration (c_j), and a label. Any type of features could have been used for the query features. In past studies, Xu *et al.* [54] chose linguistic-based considerations to categorize queries into types such as ambiguous queries. Deveaud *et al.* [16] represent queries by both linguistic and post-retrieval features -calculated after a first search using the current query, which entail additional cost. For the evaluation, we have opted for LETOR features that have been used successfully for document ranking models [10, 31]. We calculate LETOR features directly from an initial search that we perform using a reference system (BM25).

With L2R documents, each LETOR feature is associated with a query-document pair: these correspond mainly to document scores for the query using different scoring functions [42]. In our case, which is different from L2R documents, we also need to define query-configuration features, but if the LETOR features were to be calculated for each configuration c_i, it would be too costly. Instead, we have chosen a unique fair configuration (c_{BM25} is this reference configuration: BM25 without query expansion in our experiments) to calculate the LETOR query-configuration features. This is a reasonable trade-off between computing costs and accurately representing a query-configuration pair. L2R documents based on an initial BM25 ranking will therefore be a fair baseline for comparing our model and is included in Table 2.

As LETOR features are associated with query-document pairs rather than query-configuration pairs, we need to aggregate those features in relation to the documents that have been retrieved for a given query. If we let $d_{i.} = (d_1^i, d_2^i, \cdots, d_n^i)$ be the top retrieved documents for the given query q_i based on the reference configuration, where n is the number of retrieved documents, we can estimate a set of scores $C_i = \{c_{ij}\}$, where $c_{ij} = \mho(\Psi(d_i^k, q_i))$ with $\mho$ as a set of aggregation functions and Ψ as a set of scoring functions, for example, TF-IDF. A similar principle was used in [4, 12] for query performance prediction.

With regard to $\mho$, any kind of statistical aggregation functions can be used. We focus on the mean, standard deviation and maximum functions, as [12] showed that these are complementary, at least for the query performance prediction task. With regard to scoring functions Ψ, we use those functions that are implemented in the Terrier FAT framework[3] [34]. The Terrier FAT framework uses the DAAT [49] retrieval strategy to identify the initial sample of n documents and keeps the assignment of these documents in its memory. This enables fast computation of any scoring function i.e., the weighting function, without resorting to the expensive inverted index in real time. The initial sample of n documents could be retrieved faster by taking advantage of dynamic pruning [35].

Finally, the query-configuration vectors are labeled. The label for a training example is the effectiveness of the configuration when treating the query. We successively consider different labels in the evaluation of the adhoc-oriented task: average precision (AP), normalized discounted cumulative gain at the cut-off rank 10 (nDCG@10), and precision for 10 cut-off documents (P@10). This representation is also used for all the selective search strategy baselines for comparison purposes. To rank the configurations using learning-to-rank models for selective search strategy, we use RankLib[4] and the SVM^r[5] toolkits. As for the label which needs to be an integer in RankLib, we discretize the metric by $\times$10,000 to make it an integer.

To train the selective search strategy model, each training query has to be evaluated for each of the configurations so as to obtain the respective effectiveness measures that serve as the ground truth. The cost of preparing the training data thus depends on the number of candidate configurations in the meta-system. The configuration selection step is therefore key and is handled by the risk-sensitive function for selecting the candidate configurations that we proposed in section 3.

4.3 Baselines

The baselines are as follows:

* ***BM25*** is obtained using Terrier BM25 with default parameters b=0.75 and k=1.2 [44] whatever the collection is.

* ***L2R-D SVM^r*** is the standard learning to rank documents model where the initial ranking is obtained using BM25. L2R is based on $SVM - rank$ for which we obtained the best results compared to other L2R algorithms (we tried Random Forest, SVM-rank, λ-MART, and LISTNet). Given a query, the documents with a relevance label greater than 0 are considered as relevant examples, whereas

[3] http://www.terrier.org/docs/v4.0/javadoc/index.html?org/terrier/matching/models/WeightingModel.html
[4] sourceforge.net/p/lemur/wiki/RankLib/
[5] www.cs.cornell.edu/people/tj/svm_light/svm_rank.html

documents with a label equal to 0 are considered as non-relevant examples. If there is no non-relevant document but only relevant documents for a query in the relevance judgment, we retrieve 1,000 ranked documents for that query using the BM25 model and consider the bottom-ranked documents as non-relevant documents. During testing, 1,000 documents were retrieved and the top 100 documents were re-ranked using SVM-rank model. We tried several numbers of documents and found that 100 gives the best performance. We kept this number whatever the collection is.

* ***GS (Grid Search)*** determines the best value for each parameter to be able to maximize the effectiveness for a set of queries [47].
* ***Best trained*** is the best configuration among the entire configuration set: it is chosen for Q_A and applied to $Q_{\overline{A}}$, and vice versa.

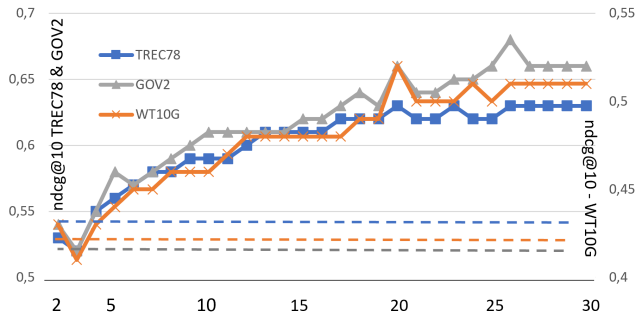

Figure 1: Best for 20 candidate configurations. Performance for E_{RISK} function on the three collections while varying the number of candidate configurations. The dotted dash horizontal lines are the single best configuration.

Then, we used two additional baselines which uses selective search strategy for each individual query. **Trained SQE** considers two configurations: one without query expansion and one with. We took the best configuration ("Best conf.") and its query expansion/non query expansion counterpart. In cases where the best configuration did not include query expansion, we chose the best configuration with expansion from the pool as its counterpart. Then we trained the meta-system for selective query expansion. **Deveaud *et al.*** where a random selection of 100 configurations per query is applied as in [16]. Compared to our risk-based model, we increased the number of configurations from 20 to 100 for Deveaud to give the latter more chances (results with 20 were lower).

Of all the baselines, only BM25 does not need training; they are applied to the entire query set. BM25 is deterministic. All the other baselines rely on using 2-fold cross-validation on the collection query set for three random trials. All the methods take into account the same three splits, the results of which are averaged (the standard deviation is also calculated).

We also indicate the Oracle results, none of the Oracle methods is deterministic because it is impossible to select it automatically. These methods would need a perfect system that chooses the most appropriate configuration. In practice, these configurations cannot be obtained through automatic training.

* ***Best conf.*** is the best configuration in the pool of initial 20,000 configurations in average over the queries of the considered collection i.e., it is the one that maximizes the evaluation measure under consideration and thus may be different according to the measure.
* ***Oracle***: for each of the queries, we selected the best configuration among the 20, 000 possible configurations;
* ***Oracle20SS***: for each of the queries, we selected the best configuration among the 20 possible configurations selected using the corresponding Risk function.

5 RESULTS

5.1 Impact of k on effectiveness and cost

We first measured the effectiveness of the E_{RISK} function by considering different numbers of candidate configurations (k) in the candidate configuration selection, intermediate phase. We wanted to analyze the impact of k on effectiveness, given the fact that as the number of candidate configurations increases, the meta-system's effectiveness also increases, but it becomes more complex and expensive. Whatever k in the intermediate phase, we then use two top-ranked configurations on a per-query basis in the final phase.

The cost of considering an additional configuration in the candidate set is as follows: at the training stage the system has to evaluate this configuration for each training query to be able to build the examples related to this configuration; it has to compare the effectiveness with all the other existing examples in the intermediate phase i.e., (c_k, q_i) examples (E_{RISK} function); and integrate with the example generation in the learning to rank configurations in the final phase. The testing (running) part is not significantly affected.

In the training, the model learns to choose the most appropriate k candidate configurations from the entire configuration pool and to choose the 2 top-ranked best configuration from k that fits each of the queries. After learning on the training queries, the model is applied to the test queries. We tried several L2R algorithms for the "best fit" part. L2R algorithms require examples to better learn the objective function. We only considered positive examples, which are defined on a per-query basis and correspond to the 2 top-ranked configurations from the examples generated using the k configurations as selected by the risk function and based on the maximum gain (see definition in Section 3). We did not use any negative examples. We tried different numbers of positive examples and found that 2 is appropriate.

We present the results for L2R based on Random Forest. The effectiveness increases with k, the number of candidate configurations (see Figure 1, ndcg@10). WT10G -left-side scale- has lower results than the other collections; however, the results for the three collections have the same shape. We cannot yet explain the small outlier peaks, but we can see that there are several stages in the effectiveness as shown by the curves in Figure 1. The first stage is around $k \in \{5-7\}$, another for $k \in \{12-16\}$, then $k = 20$ and finally $k \in \{26-30\}$. We did not report for $k > 30$ since our goal is to limit the number of configurations.

We chose 20 configurations for further experiments in the intermediate phase and two top-ranked configurations on a per-query basis from the 20 configurations in the final phase, as it appears to be an appropriate trade-off between effectiveness and a reasonable number of configurations for the different collections. This number could also be certainly optimized during the training phase, but we keep this point for future work. Twenty configurations is affordable

in terms of computing and time resources even if the training has to be renewed frequently.

5.2 Effectiveness

We compared the results obtained from our E_{RISK} function with various baselines. Furthermore, we compared the variants of the L2R model in the selective search strategy with a fixed number of configuration candidates ($k = 20$).

For the L2R models for learning configurations, we used 4 different models from the literature, including *point-wise* Linear Regression (LR) and Random Forest (RF), *pair-wise* SVM-Rank (SVM-^{r}) and *list-wise* LambdaMart [52] approaches. We report Random Forest only since it achieved the best results, although linear regression and SVM-rank were close to RF. For our context, in the Random Forest point-wise approach each instance is a vector of features x_i, which represents a query-configuration pair. The ground truth is the example label i.e., the effectiveness of the configuration for that specific query which is either considered as continuous or made to be discrete. In the case of continuous values, learning to rank can be solved as a regression problem, whereas in the case of discrete values, learning to rank is considered as a classification problem or as an ordinal regression problem, depending on whether there is an ordinal relationship between the classes of effectiveness [28].

Table 2 presents successively the results for three reference TREC collections. In each sub-table, horizontally, the first block shows the four baselines as described above; the second block shows our E_{RISK} L2R configurations (third row) with 20 configurations, as well as other trained selective search strategy methods: the trained selective query expansion (Trained SQE row) and the state-of-the-art Deveaud *et al.* [16]. The last block shows the Oracle methods. The table presents the values of the measures when averaged over three different random splits for 2-fold cross validation; we used exactly the same three random splits as we did for the baselines that need training. We also show the standard deviation for these three trials in square brackets.

From the results, we can observe that trained selective search strategy approaches (2nd horizontal block, Table 2) outperform all the baselines that do not select configurations (1st horizontal block apart for MAP on TREC78). Among these models Trained SQE is closed to the non-deterministic Best Conf. model for all the collections and measures; in its principle Trained SQE is also close to our model when two configurations only are kept while it in addition uses the constraint that one configuration should use query expansion and the other should not. With regard to our E_{RISK} method (ERisk-RF, Table 2), the improved differences are statistically significant after family-wise correction with Bonferroni [11] when compared to both Deveaud *et al.* [16] (↑) on TREC78, and WT10G for MAP, and L2R-D SVM^r (△) across collections and metrics. For Deveaud *et al.*, the results reported in Table2 are not directly comparable with the one reported in [16] where the authors used 5-fold cross-validation with one trial while in this paper, we used 2-fold cross-validation with three trials. Another major difference is the way the queries are represented. For a fair comparison, we re-implemented Deveaud *et al.* and used LETOR features to represent the queries as in our method while Deveaud *et al.* initially used pre-and-post retrieval features. We can also observe that the

Table 2: Risk-RF outperforms any baseline on all measures on collections. In a sub-table, the first block shows baselines that use a single configuration for all queries; the second block shows trained selective search strategy (SelSS) including ours ($k = 20$ configurations); the latest shows oracles. Effectiveness in absolute value, averaged on 3 draws plus standard deviation in square brackets. The best values (excluding Oracle) are in bold font. △ (resp. ↑) indicates statistically significant improvement compared to the L2R documents (resp. Deveaud *et al.*), two-tailed paired t-test ($p < 0.05$).

		TREC78		
	Methods	MAP	nDCG@10	P@10
Baselines	BM25	.21	.47	.43
	L2R-D SVM^r	.22 [.000]	.48 [.001]	.46 [.004]
	GS	.24 [.003]	.51 [.019]	.47 [.003]
	Best trained	.25 [.010]	.52 [.008]	.47 [.009]
SelSS	Trained SQE	.24 [.002]	.53 [.007]	.49 [.006]
	Deveaud *et al.* [16]	.24 [.002]	.56 [.003]	.52 [.004]
	ERisk-RF	**.28**$^{\triangle\uparrow}$ [.007]	**.63**$^{\triangle\uparrow}$ [.005]	**.60**$^{\triangle\uparrow}$ [.012]
	Best conf.	.26	.54	.51
	Oracle	.39	.83	.80
	Oracle20SS	.29	.63	.61
		WT10G		
Baselines	BM25	.22	.39	.38
	L2R-D SVM^r	.20 [.002]	.34 [.002]	.32 [.005]
	GS	.25 [.001]	.40 [.001]	.38 [.020]
	Best trained	.22 [.006]	.40 [.002]	.40 [.012]
SelSS	Trained SQE	.24 [.003]	.39 [.019]	.39 [.021]
	Deveaud *et al.* [16]	.29 [.006]	.49 [.002]	.47 [.006]
	ERisk-RF	**.32**$^{\triangle\uparrow}$ [.006]	**.52**$^{\triangle}$ [.014]	**.50**$^{\triangle}$ [.010]
	Best conf.	.25	.42	.41
	Oracle	.45	.72	.69
	Oracle20SS	.33	.53	.52
		GOV2		
Baselines	BM25	.27	.46	.54
	L2R-D SVM^r	.28 [.001]	.49 [.002]	.57 [.003]
	GS	.35 [.005]	.52 [.003]	.62 [.008]
	Best trained	.35 [.005]	.49 [.012]	.59 [.010]
SelSS	Trained SQE	.35 [.009]	.52 [.002]	.63 [.005]
	Deveaud *et al.* [16]	.40 [.003]	.66 [.001]	.77 [.005]
	ERisk-RF	**.41**$^{\triangle}$ [.002]	**.67**$^{\triangle}$ [.002]	**.79**$^{\triangle}$ [.010]
	Best conf.	.36	.52	.63
	Oracle	.50	.85	.94
	Oracle20SS	.42	.68	.80

results are robust in relation to the standard deviation across the three random train/test splits.

By delving deeper into the analysis of Table 2, we can see that the best configuration (First line last block) corresponds to the configuration which maximizes the reported measure when considering the large set of possible configurations. Among the more than 20,000 different configurations, for TREC78, the configuration that maximizes P@10 obtains 0.51 (IFB2 model without query expansion), whereas the configuration that maximizes nDCG@10 obtains

0.54 (InexpC2 model). By definition, the best configuration for a given measure cannot be surpassed by any other single configuration. Nonetheless, it is not possible to decide in advance what the best configuration will be for a given collection. Best trained (last row in the baseline block) uses exactly the same train/test splits as our L2R configuration risk-sensitive models, but selects a single configuration to handle the training queries. This selected configuration will be used for all the test queries; it is hence a fair baseline since it can be trained. For TREC78, Best trained obtains a P@10 of 0.47 (standard deviation 0.009 for the three trials); which is close to the best configuration ($P@10 = 0.51$). It is important to mention that none of our initial configurations uses document re-ranking. This is what L2R-D SVM^r (L2R documents, $P@10 = 0.46$) does (2nd line in the baseline block); it surpasses BM25 ($P@10 = 0.43$), on which it is based, for the initial document ranking but the results are similar to grid search GS ($P@10 = 0.47$). We can observe that certain other single configurations (including the best one) surpass L2R-D. Trained SQE is the closest to the best configuration. The same comments also hold for MAP and nDCG@10. Best trained and grid search are the best baselines; L2R-D is relatively close to them. These three methods involve training and use a single configuration for all the queries. We will see on the other collections that best trained is consistently the best among the methods that use a single configuration without any kind of oracle decision.

Oracle and Oracle20SS highlight the maximum level of effectiveness we could reach with a perfect match between the query and the best configuration for that query. It is 0.80 for TREC78 for P@10 when all the 20,000 configurations are considered, and 0.61 with the ERisk 20 selected configurations. Although our models can still be improved in order to achieve these maximum levels, we can observe that the training works reasonably well and that our models are close to the Oracle20SS. Similarly to the TREC78 collection, ERisk-RF is the most effective for the WT10G and GOV2 collections. All the baselines are surpassed. Again, GS is close to the best configuration; L2R-D generally surpasses the initial ranking (BM25); Best trained and L2R-D are close to each other, although Best trained is most often slightly better. None of the baselines that does not use any sort of Oracle are able to reach the "Best configuration" Oracle when taking into account a single configuration for all the queries. Risk-RF surpasses all the methods and the difference is statistically significant (apart from in two cases).

6 DISCUSSION AND CONCLUSION

Information retrieval approaches generally look for the best parameter settings that will then be used for all future queries. Here, we consider an alternative approach that aims to automatically select the best parameter setting to be applied to each individual query. Such a selective search strategy cannot handle the huge number of possible parameters, whether they are components (such as the weighting search model), or hyperparameters (such as the number of terms to add in the automatic query reformulation). Instead, it needs a limited set of candidate configurations that are, nonetheless, effective at returning highly appropriate results. The risk-sensitive method we developed in this paper provides a means for identifying this limited set of configurations. Moreover, we built this method into a meta-system that learns to select the best of these candidate configurations to fit a new query.

To some extent, our work can be seen as extending the selective query expansion approach [2, 14]. In SQE, however, the candidate configurations are limited to two - one configuration without expansion and one with - which limits the overall system effectiveness. In SQE, a single search weighting component is used for both the initial query and the expanded one. Also, it uses a single query expansion model and a set of hyperparameters for all queries. Arslan and Dinçer [3] used eight term weighting models with set hyperparameters which also limited the possible gains in effectiveness.

The use of a vast number of configurations - up to 20,000 as proposed by Deveaud *et al.* [16] improved effectiveness compared to previous methods. The cost of the system is high, however; it must handle and maintain too many configurations to be applicable in a real-world search engine. The method we described here limits the number of candidate configurations to one that should be applicable in practice. Our method outperforms that of Deveaud *et al.* also in that it avoids 'overfit' (when with too many candidate configurations compared to the number of training queries, the model learns very well on the training data but fails on the foresight task). Deveaud et al. represented queries by many pre-and post-retrieval features, which entails additional cost, whereas we represent queries using LETOR features in our proposed method. The extra cost of our approach is then small compared to using linguistic features [16, 54]. Although the results may be dependent of this choice, our choice paves the way for including the ranking document component when building the candidate configuration pool, which we will evaluate further in future studies.

Both the work presented here and our previous work in Deveaud et al. [16] are based on Terrier, making the results dependent on the underlying technological choice. Terrier however implements a large variety of models and can be considered as representative enough of IR state of the art components.

The risk-sensitive method we have developed to select candidate configurations enables the overall system to increase its performance substantially compared to the best configuration (ndcg@10 up to 0.52 on WT10G collection *versus* 0.42 for the best configuration), that demonstrates the power of selecting the appropriate configuration for each individual query. This method can be used in any search engine as long as it gathers information from the past queries to train the model.

This work could also be applied in future to the diversity task using Clueweb collections in which a query contains various aspects that might determine the relevance of the documents retrieved. Some configurations may cover more query aspects than others. Thus, selective search strategy with an optimized set of candidate configurations could outperform other document diversification strategies. Using these collections has the extra advantage, this would also demonstrate the applicability of the proposed approach in the web context we mentioned in the introduction.

ACKNOWLEDGMENTS

We would like to thank Toulouse Transfer Technology for their valuable help and funding for the patent publication linked to this research [39].

REFERENCES

[1] Giambattista Amati. 2003. *Probabilistic Models for Information Retrieval based on Divergence from Randomness. University of Glasgow, UK.* Ph.D. Dissertation. PhD Thesis.

[2] Giambattista Amati, Claudio Carpineto, and Giovanni Romano. 2004. Query Difficulty, Robustness, and Selective Application of Query Expansion. In *Advances in Information Retrieval*. Springer Berlin Heidelberg, Berlin, Heidelberg, 127–137.

[3] Ahmet Arslan and Bekir Taner Dinçer. 2019. A selective approach to index term weighting for robust information retrieval based on the frequency distributions of query terms. *Information Retrieval Journal* 22, 6 (2019), 543–569.

[4] Niranjan Balasubramanian, Giridhar Kumaran, and Vitor R. Carvalho. 2010. Predicting Query Performance on the Web. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Geneva, Switzerland) *(SIGIR '10)*. ACM, New York, NY, USA, 785–786.

[5] Rodger Benham, Ben Carterette, Alistair Moffat, and J. Shane Culpepper. 2019. Taking Risks with Confidence. In *Proceedings of the 24th Australasian Document Computing Symposium* (Sydney, NSW, Australia) *(ADCS '19)*. Association for Computing Machinery, New York, NY, USA, Article 1, 4 pages. https://doi.org/10.1145/3372124.3372125

[6] Rodger Benham and J. Shane Culpepper. 2017. Risk-Reward Trade-offs in Rank Fusion. In *Proceedings of the 22Nd Australasian Document Computing Symposium* (Brisbane, QLD, Australia) *(ADCS 2017)*. ACM, New York, NY, USA, Article 1, 8 pages.

[7] Rodger Benham, Alistair Moffat, and J. Shane Culpepper. 2020. On the Pluses and Minuses of Risk. In *Information Retrieval Technology*. Springer International Publishing, Cham, 81–93.

[8] Anthony Bigot, Sébastien Déjean, and Josiane Mothe. 2015. Learning to Choose the Best System Configuration in Information Retrieval: the Case of Repeated Queries. *Journal of Universal Computer Science (J. UCS)* 21, 13 (2015), 1726–1745.

[9] Guihong Cao, Jian-Yun Nie, Jianfeng Gao, and Stephen Robertson. 2008. Selecting Good Expansion Terms for Pseudo-relevance Feedback. In *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Singapore, Singapore) *(SIGIR '08)*. ACM, New York, NY, USA, 243–250.

[10] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. 2007. Learning to Rank: From Pairwise Approach to Listwise Approach. In *Proceedings of the 24th International Conference on Machine Learning* (Corvalis, Oregon, USA) *(ICML '07)*. ACM, New York, NY, USA, 129–136.

[11] Benjamin A Carterette. 2012. Multiple testing in statistical analysis of systems-based information retrieval experiments. *ACM Transactions on Information Systems (TOIS)* 30, 1 (2012), 1–34.

[12] Adrian-Gabriel Chifu, Léa Laporte, Josiane Mothe, and Md Zia Ullah. 2018. Query Performance Prediction Focused on Summarized Letor Features. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. ACM, New York, NY, USA, 1177–1180.

[13] Kevyn Collins-Thompson. 2009. Reducing the Risk of Query Expansion via Robust Constrained Optimization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management* (Hong Kong, China) *(CIKM '09)*. Association for Computing Machinery, New York, NY, USA, 837–846. https://doi.org/10.1145/1645953.1646059

[14] Steve Cronen-Townsend, Yun Zhou, and W. Bruce Croft. 2004. A Framework for Selective Query Expansion. In *Proceedings of the Thirteenth ACM International Conference on Information and Knowledge Management* (Washington, D.C., USA) *(CIKM '04)*. ACM, New York, NY, USA, 236–237.

[15] Romain Deveaud, Josiane Mothe, and Jian-Yun Nia. 2016. Learning to Rank System Configurations. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. ACM, New York, NY, USA, 2001–2004.

[16] Romain Deveaud, Josiane Mothe, Md Zia Ullah, and Jian-Yun Nie. 2018. Learning to Adaptively Rank Document Retrieval System Configurations. *ACM Transactions on Information Systems (TOIS)* 37, 1 (2018), 3.

[17] B. Taner Dinçer, Craig Macdonald, and Iadh Ounis. 2014. Hypothesis Testing for the Risk-sensitive Evaluation of Retrieval Systems. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Gold Coast, Queensland, Australia) *(SIGIR '14)*. ACM, New York, NY, USA, 23–32.

[18] B. Taner Dinçer, Craig Macdonald, and Iadh Ounis. 2016. Risk-Sensitive Evaluation and Learning to Rank Using Multiple Baselines. In *Proceedings of the 39th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Pisa, Italy) *(SIGIR '16)*. ACM, New York, NY, USA, 483–492.

[19] B. Taner Dinçer, Iadh Ounis, and Craig Macdonald. 2014. Tackling Biased Baselines in the Risk-Sensitive Evaluation of Retrieval Systems. In *Advances in Information Retrieval*. Springer International Publishing, Cham, 26–38.

[20] Nicola Ferro and Gianmaria Silvello. 2016. A General Linear Mixed Models Approach to Study System Component Effects. In *Proceedings of the 39th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Pisa, Italy) *(SIGIR '16)*. ACM, New York, NY, USA, 25–34.

[21] Edward A Fox and Joseph A Shaw. 1994. Combination of multiple searches. *NIST special publication SP* 243 (1994), 127–137.

[22] Donna Harman and Chris Buckley. 2009. Overview of the reliable information access workshop. *Information Retrieval* 12, 6 (2009), 615.

[23] Ben He and Iadh Ounis. 2004. A Query-Based Pre-Retrieval Model Selection Approach to Information Retrieval. In *Coupling Approaches, Coupling Media and Coupling Languages for Information Retrieval* (Vaucluse, France) *(RIAO '04)*. CID, Paris, FRA, 706–719.

[24] Ben He and Iadh Ounis. 2007. Combining fields for query expansion and adaptive query expansion. *Information processing & management* 43, 5 (2007), 1294–1307.

[25] D Frank Hsu and Isak Taksa. 2005. Comparing rank and score combination methods for data fusion in information retrieval. *Information retrieval* 8, 3 (2005), 449–480.

[26] Joseph Johnson. Aug 10, 2021. ACSI - U.S. customer satisfaction with Google 2002-2021. https://www.statista.com/statistics/185966/us-customer-satisfaction-with-google/

[27] Oren Kurland and J. Shane Culpepper. 2018. Fusion in Information Retrieval: SIGIR 2018 Half-Day Tutorial. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 1383–1386.

[28] Léa Laporte, Rémi Flamary, Stéphane Canu, Sébastien Déjean, and Josiane Mothe. 2014. Nonconvex regularizations for feature selection in ranking with sparse svm. *IEEE Transactions on Neural Networks and Learning Systems* 25, 6 (2014), 1118–1130.

[29] Dan Li and Evangelos Kanoulas. 2018. Bayesian Optimization for Optimizing Retrieval Systems. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. ACM, New York, NY, USA, 360–368.

[30] Hang Li. 2011. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems* 94, 10 (2011), 1854–1862.

[31] Yuan Lin, Jiajin Wu, Bo Xu, Kan Xu, and Hongfei Lin. 2017. Learning to rank using multiple loss functions. *International Journal of Machine Learning and Cybernetics* 10, 10 (2017), 485–494. Issue 3.

[32] David G Luenberger, Yinyu Ye, et al. 1984. *Linear and nonlinear programming*. Vol. 2. Springer, Switzerland.

[33] Craig Macdonald, B. Taner Dinçer, and Iadh Ounis. 2015. Transferring Learning To Rank Models for Web Search. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval* (Northampton, Massachusetts, USA) *(ICTIR '15)*. Association for Computing Machinery, New York, NY, USA, 41–50. https://doi.org/10.1145/2808194.2809463

[34] Craig Macdonald, Rodrygo LT Santos, Iadh Ounis, and Ben He. 2013. About learning models with multiple query-dependent features. *ACM Transactions on Information Systems (TOIS)* 31, 3 (2013), 11.

[35] Joel Mackenzie, J. Shane Culpepper, Roi Blanco, Matt Crane, Charles L. A. Clarke, and Jimmy Lin. 2018. Query Driven Algorithm Selection in Early Stage Retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining* (Marina Del Rey, CA, USA) *(WSDM '18)*. Association for Computing Machinery, New York, NY, USA, 396–404. https://doi.org/10.1145/3159652.3159676

[36] Donald Metzler and W Bruce Croft. 2007. Linear feature-based models for information retrieval. *Information Retrieval* 10, 3 (2007), 257–274.

[37] Stefano Mizzaro and Stephen Robertson. 2007. Hits Hits TREC: Exploring IR Evaluation Results with Network Analysis. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (Amsterdam, The Netherlands) *(SIGIR '07)*. Association for Computing Machinery, New York, NY, USA, 479–486. https://doi.org/10.1145/1277741.1277824

[38] Hafeezul Rahman Mohammad, Keyang Xu, Jamie Callan, and J. Shane Culpepper. 2018. Dynamic Shard Cutoff Prediction for Selective Search. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval* (Ann Arbor, MI, USA) *(SIGIR '18)*. Association for Computing Machinery, New York, NY, USA, 85–94. https://doi.org/10.1145/3209978.3210005

[39] Josiane Mothe and Md Zia Ullah. Application on 2019 July 29, published on 2021 Feb. 03. Apparatus and method for information retrieval using a set of pre-selected search configurations using efficiency and risk functions - Dispositif et procédé de récupération d'informations utilisant un ensemble de configurations de recherche pré-sélectionnées à l'aide de fonctions d'efficacité et de risque. https://worldwide.espacenet.com/patent/search/family/067956648/publication/EP3771996A1?q=19305984.7

[40] Josiane Mothe and Mahdi Washha. 2017. Predicting the Best System Parameter Configuration: the (Per Parameter Learning) PPL method. *Procedia Computer Science* 112 (2017), 1308 – 1317. Knowledge-Based and Intelligent Information and Engineering Systems: Proceedings of the 21st International Conference, KES-20176-8 September 2017, Marseille, France.

[41] Rabia Nuray and Fazli Can. 2006. Automatic ranking of information retrieval systems using data fusion. *Information processing & management* 42, 3 (2006), 595–614.

[42] Tao Qin, Tie-Yan Liu, Jun Xu, and Hang Li. 2010. LETOR: A benchmark collection for research on learning to rank for information retrieval. *Information Retrieval* 13, 4 (2010), 346–374.

[43] Fiana Raiber and Oren Kurland. 2014. Query-performance Prediction: Setting the Expectations Straight. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval* (Gold Coast, Queensland, Australia) *(SIGIR '14)*. ACM, New York, NY, USA, 13–22.

[44] Stephen Robertson, Hugo Zaragoza, et al. 2009. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends® in Information Retrieval* 3, 4 (2009), 333–389.

[45] Anna Shtok, Oren Kurland, David Carmel, Fiana Raiber, and Gad Markovits. 2012. Predicting query performance by query-drift estimation. *ACM Transactions on Information Systems (TOIS)* 30, 2 (2012), 11.

[46] Daniel Xavier De Sousa, Sérgio Daniel Canuto, Thierson Couto Rosa, Wellington Santos Martins, and Marcos André Gonçalves. 2016. Incorporating Risk-Sensitiveness into Feature Selection for Learning to Rank. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (Indianapolis, Indiana, USA) *(CIKM '16)*. ACM, New York, NY, USA, 257–266.

[47] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. 2006. Optimisation Methods for Ranking Functions with Multiple Parameters. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management* (Arlington, Virginia, USA) *(CIKM '06)*. ACM, New York, NY, USA, 585–593.

[48] Andrew Trotman. 2005. Learning to rank. *Information Retrieval* 8, 3 (2005), 359–381.

[49] Howard Turtle and James Flood. 1995. Query evaluation: strategies and optimizations. *Information Processing & Management* 31, 6 (1995), 831–850.

[50] Jun Wang and Jianhan Zhu. 2009. Portfolio Theory of Information Retrieval. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 115–122. https://doi.org/10.1145/1571941.1571963

[51] Lidan Wang, Paul N. Bennett, and Kevyn Collins-Thompson. 2012. Robust Ranking Models via Risk-sensitive Optimization. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. ACM, New York, NY, USA, 761–770.

[52] Qiang Wu, Christopher J. Burges, Krysta M. Svore, and Jianfeng Gao. 2010. Adapting Boosting for Information Retrieval Measures. *Information Retrieval* 13, 3 (2010), 254–270.

[53] Bo Xu, Hongfei Lin, and Yuan Lin. 2016. Assessment of learning to rank methods for query expansion. *Journal of the Association for Information Science & Technology* 67, 6 (2016), 1345–1357.

[54] Yang Xu, Gareth J.F. Jones, and Bin Wang. 2009. Query Dependent Pseudo-Relevance Feedback Based on Wikipedia. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval* (Boston, MA, USA) *(SIGIR '09)*. Association for Computing Machinery, New York, NY, USA, 59–66.

[55] Le Zhao and Jamie Callan. 2012. Automatic Term Mismatch Diagnosis for Selective Query Expansion. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval* (Portland, Oregon, USA) *(SIGIR '12)*. ACM, New York, NY, USA, 515–524.