Bayesian Optimization for Optimizing Retrieval Systems

Li, D.; Kanoulas, E.

DOI
10.1145/3159652.3159665

Publication date
2018

Document Version
Final published version

Published in
WSDM'18

Bayesian Optimization for Optimizing Retrieval Systems

Dan Li
University of Amsterdam
Amsterdam, The Netherlands
d.li@uva.nl

Evangelos Kanoulas
University of Amsterdam
Amsterdam, The Netherlands
e.kanoulas@uva.nl

ABSTRACT

The effectiveness of information retrieval systems heavily depends on a large number of hyperparameters that need to be tuned. Hyperparameters range from the choice of different system components, e.g., stopword lists, stemming methods, or retrieval models, to model parameters, such as $k1$ and b in BM25, or the number of query expansion terms. Grid and random search, the dominant methods to search for the optimal system configuration, lack a search strategy that can guide them in the hyperparameter space. This makes them inefficient and ineffective. In this paper, we propose to use Bayesian Optimization to jointly search and optimize over the hyperparameter space. Bayesian Optimization, a sequential decision making method, suggests the next most promising configuration to be tested on the basis of the retrieval effectiveness of configurations that have been examined so far. To demonstrate the efficiency and effectiveness of Bayesian Optimization we conduct experiments on TREC collections, and show that Bayesian Optimization outperforms manual tuning, grid search and random search, both in terms of retrieval effectiveness of the configuration found, and in terms of efficiency in finding this configuration.

CCS CONCEPTS

• **Information systems → Retrieval models and ranking**;

KEYWORDS

Retrieval system; Hyperparameter optimisation; Bayesian Optimization; Covariance function

ACM Reference Format:
Dan Li and Evangelos Kanoulas. 2018. Bayesian Optimization for Optimizing Retrieval Systems. In *WSDM 2018: The Eleventh ACM International Conference on Web Search and Data Mining, February 5–9, 2018, Marina Del Rey, CA, USA.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3159652.3159665

1 INTRODUCTION

The effectiveness of information retrieval (IR) systems heavily depends on a large number of hyperparameters that need to be tuned. Hyperparameters range from the choice of different system components, e.g., stopword lists, stemming methods, retrieval models, to model parameters, such as $k1$ and b values in BM25, number of top-ranked documents to consider in pseudo-relevance feedback, and number of query expansion terms.

Retrieval performance is rather sensitive to parameter (or hyperparameter) tuning. Zhai and Lafferty [37] demonstrated this sensitivity for the smoothing parameters in language models, and Trotman et al. [31] for parameters in BM25. Automatic techniques for optimizing model parameters have attracted the attention of the research community in recent years [12–15, 20, 23, 24]. However, parameters are usually optimized in isolation, while their mutual dependencies are, to a great extend, unexplored [1, 19]. Ferro and Silvello [11] recently examined this mutual dependency between choices of stopword lists, stemmers and retrieval models, and concluded that parameter interactions have a strong effect on system performance.

Grid search has been the most widely used strategy for automatic joint optimization of hyperparameters in information retrieval [33]. Grid search is easy to implement, parallelization is trivial, and it is reliable in low dimensional spaces [3]. However, grid search suffers from the *curse of dimensionality* because the number of configurations grows exponentially with the number of hyperparameters [2]. To reduce the number of candidate configurations to be explored researchers often apply grid search on one dimension/hyperparameter at a time, while fixing the values of the rest hyperparameters until all of them have been traversed. However, this approach is not guaranteed to find a global optimum. Bergstra and Bengio [3] instead propose *random search* as a more efficient search algorithm. Random search generates candidate configurations drawn uniformly from the same configuration space as would be spanned by a regular grid. Bergstra and Bengio [3] show that random search is more efficient than grid search in high-dimensional spaces because objective functions of interest often have a low effective dimensionality, i.e. they are more sensitive to changes in some dimensions than others [7]. Nevertheless, random search remains agnostic to the effect that a δ-step in the hyperparameter space that would have to the effectiveness of an IR system. Quantifying this effect could lead to more efficient search strategies.

Bayesian Optimization has risen as a promising framework for efficient and effective search in the candidate configuration space [4, 5, 17, 25, 26, 29, 36]. Under the Bayesian Optimization framework, one does not need to explicitly specify the objective function; what is only necessary is the ability to query this function and get an observation. In the case of information retrieval, one does not need to analytically express system effectiveness as a function of the hyperparameters, but only observe the effectiveness of a system configuration in terms of an evaluation measure, e.g. average precision. Bayesian Optimization uses a surrogate model to approximate system effectiveness. Taking a prior belief over this surrogate model allows Bayesian Optimization to sequentially refine it. Further, the

WSDM 2018, February 5–9, 2018, Marina Del Rey, CA, USA

ACM ISBN 978-1-4503-5581-0/18/02…$15.00
https://doi.org/10.1145/3159652.3159665

surrogate model, which indicates a probability distribution over the possible system effectiveness functions, allows designing different strategies for selecting the next configuration to be tested, exploiting hyperparameter subspaces that have shown to contain high performance configurations, or exploring hyperparameter subspaces with high potential [25].

In this paper, we use Bayesian Optimization to automatically optimize the retrieval system hyperparameters. Information retrieval effectiveness, however, as a function of the hyperparameter space exhibits high irregularity. Furthermore, hyperparameters can be continuous, or categorical [10], with the latter contributing most of the irregularity of the objective function. To tackle this we model the effect of a δ-step in the hyperparameter space to the effectiveness of the IR system, by suggesting to use different similarity functions (covariance functions) for continuous and categorical hyperparameters, and examine their ability to effectively and efficiently guide the search in the hyperparameter space. We compare Bayesian Optimization to manual tuning, grid search, and random search using TREC collections, and demonstrate that Bayesian Optimization is able to find better configurations in terms of retrieval effectiveness when all methods are granted the same computational budget. To the best of our knowledge, this is the first work that uses Bayesian Optimization to find optimal configurations of retrieval systems.

Therefore, the main contributions of this work are the following:

(1) We propose the use of Bayesian Optimization for retrieval system configuration;
(2) We decompose the components of Bayesian Optimization that affect the effectiveness of the method and suggest an instantiation of it that fits the IR hyperparameter space; and
(3) We demonstrate the effectiveness of the method in building IR systems and explain the reason in terms of optimization behaviour.

2 RELATED WORK

We first discuss prior work on hyperparameter optimization in IR. Next, we give an overview of Bayesian Optimization methods and the domains they have been applied to.

2.1 Hyperparameter optimization in IR

Optimizing the hyperparameters of a retrieval system is inevitable in order to achieve optimal performance [33]. Different methods have been proposed in the literature for automatically optimizing individual components of a retrieval system [12–15, 20, 23, 24, 30, 38]. Taylor et al. [30] use gradient descent method to optimize the parameters of ranking functions like BM25F. Their approach however makes the assumption that the cost functions must be differentiable. Bigot et al. [6] propose a method for hyperparameter optimization on a per-query basis. However, this method is hard to generalise as it only works on queries already seen in the training set. Deveaud et al. [8] cast the problem of system configuration optimization as a ranking problem and use learning to rank approaches to select the best combination of hyperparameters for IR systems. The drawback is that all the configurations to be ranked must be run in advance for training and the element number of configurations (e.g. about 10,000 in the paper) grows exponentially with the number of space dimension. The work of [6] and [8] is orthogonal to the work in this paper. In both aforementioned works all candidate configurations need to be considered and the focus lies in finding the best one of them for each query. Instead, in this paper we propose a search strategy that avoids considering all candidate configurations and focuses search in the most promising sub-spaces of hyperparameter space.

2.2 Bayesian Optimization

The problem of hyperparameter optimization appears in many machine learning applications, and lately it has attracted a wide interest in that community [4, 5, 17, 25, 26, 29, 36]. Bayesian Optimization has emerged rapidly as a promising framework for efficiently identifying effective configurations. Popular Bayesian Optimization approaches include sequential model-based optimization (SMBO) [26], sequential model-based algorithm configuration (SMAC) [17], tree-based Parzen estimator (TPE) [5] and multi-task Bayesian Optimization [29] etc. Bayesian Optimization methods have been applied successfully in many tasks [28, 34, 36]. For example, Bayesian Optimization can find better hyperparameters for deep neural networks in MNIST digit recognition and CIFAR-10 object recognition [28]. Also, by applying TPE in feature selection, basic machine learning algorithms like logistic regression and SVM are proven to outperform state-of-art neural network models in topic classification and sentiment analysis task in NLP [36]. As an important surrogate model, Gaussian process as well as its covariance functions have been also studied extensively [18, 28]. These successful applications have inspired us to use Bayesian Optimization to optimize retrieval systems.

3 PRELIMINARIES

3.1 Bayesian Optimization

The Bayesian Optimization framework provides a mechanism to sequentially search for the global optimum x of an objective function $f(x) : \mathbb{X} \rightarrow \mathbb{R}$. There are two key components in Bayesian Optimization [25]. The first is a probabilistic *surrogate model* used to predict the objective function value y given a point x. For every x, there is a random variable y, whose distribution $p(y|x)$ is given by the surrogate model. One example of the predictive distribution $p(y|x)$ can be observed in the top panels of Figure 1. In this example the surrogate model is a Gaussian Process (described in Section 3.2). The black curve depicts the original objective function (for instance mean average precision), while the x-axis in the figure represents an 1-dimensional hyperparameter space, e.g. the parameter μ of a Language Model with Dirichlet smoothing. The predictive distribution of y can then be used to construct an *acquisition function*. An acquisition function is a policy for selecting the sequence of points $\{x_1, x_2, ..., x_i, ...\}$, i.e. a mechanism to select the next configuration x_{n+1} to test given $D_{1:n} := \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n)\}$. As the acquisition function x is usually a closed-form expression of hyperparameters it is easier to be optimized than the original objective function. The bottom panel of Figure 1 demonstrates an acquisition function. The acquisition function values for different configurations (i.e. along the x-axis) dictate the potential of a certain configuration to yield a high objective function value.

The second component is the *objective function* itself, a function of the target model requiring hyperparameter optimization. In

our case the objective function can be any retrieval effectiveness measure, such as average precision, normalized discount cumulative gain and so on. Computing the objective function for different system configurations is the most time-consumptive step [5].

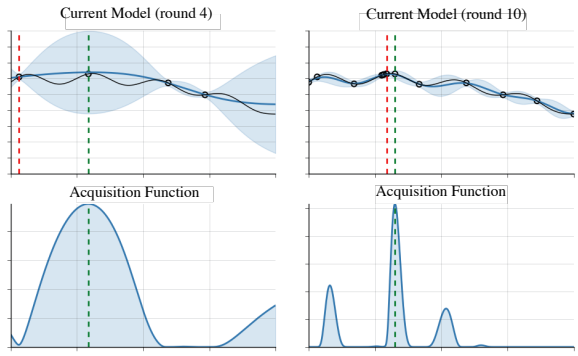


Figure 1: The optimization process of Bayesian Optimization. The upper panel describes the current model, and the bottom panel describes the acquisition function. The black curve is the actual objective function, the black dots are the observed values of the objective function. The red vertical line denotes the best value among the observed points, the green vertical line denotes the next point that Bayesian Optimization choose. The light blue line and shade are the estimated function values and variances.

The entire process of Bayesian Optimization is demonstrated in Algorithm 1. The algorithm stops when the computational budget is exhausted. Figure 1 uses an 1-dimensional continuous function to illustrate the optimization process, and demonstrates it at round 4 (i.e. after the first 3 configurations have been tests) at the left-most panels, and at round 10 at the right-most panels.

Algorithm 1 Bayesian Optimization

Require: surrogate model $\mathcal{M}$, acquisition function $\alpha_{\mathcal{M}}$, objective function f, input space X

1: Initially sample k points $\{x_1, x_2, \ldots, x_k\}$ from X, query f to get $\{y_1, y_2, \ldots, y_k\}$
2: Update $\mathcal{M}$
3: **for** $n = 1, 2, \ldots$ **do**
4: Select $x_{n+1} \leftarrow \underset{x}{\text{argmax}}\, \alpha_{\mathcal{M}}(x; D_n)$
5: Calculate $y_{n+1} \leftarrow f(x_{n+1})$
6: Augment $D_{n+1} = \{D_n, (x_{n+1}, y_{n+1})\}$
7: Update $\mathcal{M}$
8: **end for**
9: Select the best y from D

3.2 Gaussian Process

The Gaussian Process (GP) is the most commonly-used surrogate model in Bayesian Optimization [21]. The key hypothesis underlying GP is that any finite set of $\{y_i\}$ follows a Multivariate Gaussian Distribution. There is an analytic expression for the joint distribution $p(y_{1:n}|x_{1:n}) \sim N(0, K)$, with K being the covariance matrix,

$$K = \begin{bmatrix} k(x_1, x_1) & \cdots & k(x_1, x_n) \\ \vdots & \ddots & \vdots \\ k(x_n, x_1) & \cdots & k(x_n, x_n) \end{bmatrix}$$

and $k(x_i, x_j)$ being a covariance function. The conditional distribution of the objective function value given the sequence of past observations and a new point x_{n+1} is $p(y_{n+1}|x_{n+1}, D_{1:n}) \sim N(\mu_n(x_{n+1}), \sigma_n^2(x_{n+1}))$, where $\mu_n(x_{n+1})$, and $\sigma_n^2(x_{n+1})$ are the mean and the variance of the posterior distribution, respectively. They are defined as

$$\mu_n(x_{n+1}) = k^T K^{-1} y_{1:n}$$
$$\sigma_n^2(x_{n+1}) = k(x_{n+1}, x_{n+1}) - k^T K^{-1} k$$

and

$$k = \begin{bmatrix} k(x_{n+1}, x_1), & k(x_{n+1}, x_2), & \cdots, & k(x_{n+1}, x_n) \end{bmatrix}^T$$

The covariance function models the effect of taking a δ-step in the hyperparameter space to the objective function. In Section 4.2 we suggest covariance functions we believe to be appropriate for the continuous, and categorical hyperparameters of a retrieval system.

3.3 Acquisition Function

A good acquisition function is the one that finds an optimal trade-off between exploration and exploitation in the hyperparameter space on the basis of the application at hand. In practice, an acquisition function balances between configurations for which the predicted function value $f(x)$ is high (exploitation) and configurations for which the predicted variance $\sigma(x)$ is high (exploration) [25]. In this work, we consider three acquisition functions: probability of improvement (PI), expected improvement (EI) [9] and upper confidence bound (UCB), defined as follows:

$$\alpha_{PI}(x|D_n) := P(y > y^*) = \Phi(\frac{\mu_n(x) - y^*}{\sigma_n(x)})$$
$$\alpha_{EI}(x|D_n) := \mathbb{E}(max(y - y^*, 0))$$
$$= (\mu_n(x) - y^*)\Phi(\frac{\mu_n(x) - y^*}{\sigma_n(x)}) + \sigma_n(x)\phi(\frac{\mu_n(x) - y^*}{\sigma_n(x)})$$
$$\alpha_{UCB}(x|D_n) := \mu_n(x) + \beta\sigma_n(x)$$

where $\alpha(\cdot)$ is the acquisition function, y^* is a target value which is often set to $\max(\{y_i\}_{i=1}^n)$ [35], and Φ and ϕ are the Cumulative Distribution Function and Probability Density Function of the standard normal distribution, respectively. β is a hyperparameter that can be set according to some theoretically motivated guidelines.

A drawback of PI, intuitively, is that it is pure exploitative. Configurations that have a high probability their effectiveness being infinitesimally greater than y^* will be drawn over configurations that offer larger gains but with less certainty. EI on the other hand considers the magnitude of the improvement a configuration can potentially yield, instead of PI. In UCB it is the parameter β that controls the trade-off between exploration and exploitation. These three acquisition functions are point-wise, that is they only care about the improvement over y^* on each single point. There are also entropy-based acquisition functions that make use of what

has been observed about the objective function to pick up the most informative point, which will be left for the future study.

3.4 Initialization

The first posterior distribution of the surrogate model can be obtained by using the predefined prior distribution to draw the first hyperparameter point, i.e. the first configuration. This strategy runs the risk of getting stuck in a local optimum, since it heavily depends on the first point that will be used. A more general strategy, which we follow in this work is to sample a set of configurations randomly in the search space so that the surrogate model can observe the overall "landscape" of configurations. In this work we test three sampling methods: Latin hypercube sampling, random sampling, and sampling from Sobol sequences [16].

3.5 Selecting the Optimal Configuration

At the end of the optimization process one needs to decide which configuration is the optimal one. The natural choice is to select the configuration with the best observed effectiveness. However, this strategy runs the risk of overfitting the observed objective function values. A different strategy is to select the configuration with the highest predicted value, instead of the actual observed value, which can accommodate noisy outputs. The two strategies are called *incumbent* and *latent*, respectively [16], and we test both in this work.

4 METHODOLOGY

In this section we first elaborate the hyperparameter optimization process, then we focus on the particular characteristics of the retrieval systems hyperparameter space and discuss the covariance functions we consider appropriate for this space.

4.1 Hyperparameter Optimization Process

Following the Bayesian Optimization framework, we have two major modules in our algorithmic pipeline, the IR module and the BO module (see Figure 2). The IR module tackles the conditional hyperparameters, and computes the objective function value y_n, given a hyperparameter configuration x_n. The BO module adds (x_n, y_n) into the sample set, updates the posterior distribution of the surrogate models, selects the next hyperparameter configuration x_{n+1}, and passes it back to IR module.

4.2 Hyperparameter Structure and Covariance functions

Snoek et al. [26] suggest that it is the choice of the covariance functions that has the strongest effect on the performance of Bayesian Optimization . In this paper we only consider a simple case, *stationary covariance function*, which is defined as a function of $|\boldsymbol{x} - \boldsymbol{x'}|(= r)$. The *mean square differentiability* of a stationary covariance function around 0 determine the smoothness properties of the samples drawn from the corresponding Gaussian process [22], which essentially dictates the expected response in the objective function if a δ-step is taken in the hyperparameter space.

The *squared exponential covariance function* (SE) is a widely made choice for smooth objective functions; it is mean square

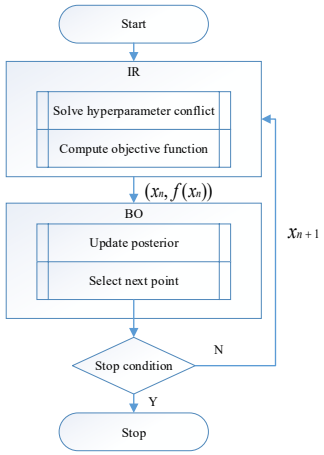


Figure 2: Hyperparameter optimization architecture.

differentiable at any order and thus very smooth. It is defined as:

$$K_{\text{SE}}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\frac{r^2}{2})$$

where $\boldsymbol{x}$ and $\boldsymbol{x'}$ are two points in the hyperparameter space, and r is the distance between $\boldsymbol{x}$ and $\boldsymbol{x'}$.

The *Matérn1 covariance function* (Matérn1) is appropriate to model rough objective functions, as it is only first-order mean square differentiable. Martén1 is defined as follows:

$$K_{\text{Matérn1}}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-r).$$

Selecting a covariance function requires examining the hyperparameter space of an IR system. Retrieval model parameters, e.g. the smoothness parameter λ in Jelinek-Mercer Smoothing of Language Models, or the parameter b in BM25, are typically continuous. On the other hand, the choice of the retrieval model itself can be expressed with a categorical hyperparameter; the use or not of relevance feedback algorithms is also categorical, as is the choice of a stopword list, or a stemmer. The default distance used in either *SE covariance function* or *Matérn1 covariance function* is Euclidean distance, defined as

$$r_E := \sqrt{\sum_{d=1}^{N} (x_d - x_d{}')^2}.$$

It would treat categorical hyperparameters as ordinal, whereas this should not be the case. Therefore, considering the heterogeneity of the hyperparameters of IR systems, Euclidean distance is not a good fit. Instead, inspired by [17], we use *Hamming distance* for categorical dimensions, while *Euclidean distance* for continuous or discrete dimensions. The *Hamming distance* is defined as

$$r_H := \sum_{d=1}^{N} (1 - \delta(x_d, x_d{}'))$$

where δ is the Kronecker delta function (equalling one if its two arguments are identical and zero otherwise).

In order to accommodate both continuous and categorical hyperparameters, we combine the Euclidean distance and the Hamming

distance into a mixture distance, the *Hamming Euclidean mixture distance* (HE distance). HE treats continuous and categorical hyperparameters differently; it is defined as

$$r_{HE} := \sqrt{\sum_{d \in \text{con}} (x_d - x_d')^2 + \sum_{d \in \text{cat}} (1 - \delta(x_d, x_d'))}$$

where *con* is the set of continuous dimensions and *cat* the set of categorical dimensions. It is easy to prove that HE is a distance (or metric). By replace the original distance function r_{SE} with r_{HE}, we get two new covariance functions,

$$K_{\text{HSE}}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-\frac{r_{HE}^2}{2})$$

$$K_{\text{HMatérn1}}(\boldsymbol{x}, \boldsymbol{x'}) = \exp(-r_{HE}).$$

Compared with SE and Matérn1, HSE and HMatérn1 are designed to handle heterogeneous space where the continuity property of different dimensions are not the same. To sum up, in this work, we test the performance of SE, HSE, Matérn1 and HMatérn1 as four representative covariance functions for continuous and categorical hyperparameters.

5 EXPERIMENT SETUP

5.1 Research questions

In the remainder of the paper we aim to answer the following three research questions:

- **RQ1** What is the most critical component of the Bayesian Optimization framework for identifying the best retrieval system configuration?
- **RQ2** How effective is Bayesian Optimization in searching the configuration space, and in finding configurations that generalize across queries and collections?
- **RQ3** How to explain the optimization behaviour of Bayesian Optimization in terms of exploration and exploitation?

5.2 Implementation of Retrieval System and Bayesian Optimization

We use Pyndri [32], a Python Interface to the *Indri* Search Engine [27], as the IR module in our pipeline, which is mainly decomposed to indexing, retrieval, and pseudo-relevance feedback. All the three modules are considered in our optimization experiments.

The objective function can be any retrieval effectiveness measure. In this work we optimize for the Mean Average Precision (MAP), Normalized Discounted Cumulative Gain (NDCG) and Mean Reciprocal Rank (MRR) and use *trec_eval* [1] for the computations.

We use *Pybo* [16] in our experiments, a Python package for Bayesian Optimization . *Pybo* supports the default version of the *SE covariance function* and the *Martén1 covariance function* which use Euclidean distance to measure the distance of two points; we implemented the *HSE covariance function* and the *HMartén1 covariance function* within *Pybo* ourselves.

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

5.3 Candidate Configurations

There are two major choices to make when indexing documents: the stopword list and the stemmer. Ferro and Silvello [11] has shown that the choice of different stopword lists, like that of indri, Lucene, Smart and Terrier, has limited effect to the performance of a IR system; however, having stopword list or not makes a big difference. On the other hand, indexing document collections takes a long time. Therefore for efficiency and convenience we only set two values for the categorical hyperparameter *stopper*: TRUE means using Indri stopword list and FALSE means not using any stopword list. The situation is similar for *stemmer*: TRUE means using Krovets stemmer and FALSE means not using any stemmer.

There are three retrieval models implemented in *Indri*: TF-IDF with BM25 term weighting, Okapi BM25, and Language Models. Language Models supports three different smoothing methods, Jelinek-Mercer (JM), Dirichlet (DIR), and two-stage smoothing (TS). Let *rm* denote a categorical hyperparameter that represents the retrieval model type and takes values in [TF-IDF, BM25, LM-JM, LM-DIR, LM-TS], that is the TF-IDF model with BM25 term weighting, the Okapi BM25 model, the Language Model with JM smoothing, the Language Model with Dirichlet smoothing, and the Language Model with two-stage smoothing respectively. There is a number of hyperparameters per model: $k1$ and b for TF-IDF, $k1$, $k3$, and b for Okapi BM25, λ_{col} and λ_{doc} for the JM smoothing, μ for the Dirichlet smoothing, and λ and μ for the two stage smoothing. The parameters of these models lay in a continuous space.

Indri also supports pseudo-relevance feedback models. We consider a binary hyperparameter *prf* that takes the value TRUE if pseudo-relevance feedback is used, and FALSE otherwise. There are four hyperparameters that need to be set for the pseudo-relevance feedback models, the number of feedback documents to be considered, *fbDocs*, the number of feedback terms, *fbTerms*, the Dirichlet smoothing parameter used for the feedback document language model,*fbMu*, and the weight of the original query, *fbOrigWeight*, in the mixture model between the original query and the feedback documents. In total, we have a conditional hyperparameter space of 18 dimensions (see Figure 3).

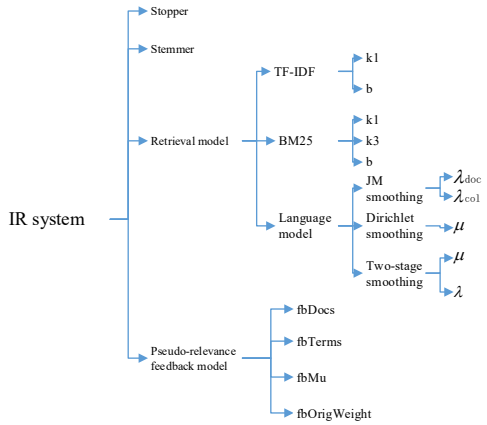
5.4 Test collections

We conduct our experiments on the ad-hoc test collections of TREC 5-8 and TREC Robust 2004, as well as the web test collections of TREC 2010-2012 in order to thoroughly evaluate the proposed method in diversified datasets. The details can be found in Table 1. As we can see, ad-hoc tracks and web tracks are quite different test collections. Web tracks have much more documents and the average length of documents is also longer, indicating they are more challenging compared with ad-hoc tracks.

5.5 Baselines

Manual Search. The first baseline is obtained by running *Indri* with its default hyperparameters. Then we manually select the best performing model among TF-IDF/BM25/Language Models, with/without pseudo-relevance feedback.

Grid Search. The second baseline is grid search. Grid search is the most widely used method for hyperparameter tuning. For each of the five retrieval model (TF-IDF, BM25, LM-JM, LM-DIR, LM-TS)



Hyperparamter	Type	Values
$Stopper$	Boolean	{True, False}
$Stemmer$	Boolean	{True, False}
rm	Integer	{TF-IDF, BM25, LM-JM, LM-DIR, LM-TS}
$k1$	Real value	[1,2]
b	Real value	[0,1]
$k1$	Real value	[1,10]
$k3$	Real value	[1,10]
b	Real value	[0,1]
λ_{doc}	Real value	[0,1]
λ_{col}	Real value	[0,1]
μ_{dir}	Real value	[0,3000]
μ_{ts}	Real value	[0,3000]
λ_{ts}	Real value	[0,1]
prf	Boolean	{True, False}
$fbDocs$	Integer	[1,50]
$fbTerms$	Integer	[1,50]
$fbMu$	Real value	[0,3000]
$fbOrigWeight$	Real value	[0,1]

Figure 3: Conditional hyperparameters and their search ranges in *Indri*

Table 1: Test collections

TREC	Document collection	No. of Doc.	Doc. Length Median	Doc. Length Mean	Topics
TREC 5	Volume 2 & 4	524,929	340	546	251-300
TREC 6	Volume 4 & 5	556,077	326	526	301-350
TREC 7	Volume 4 & 5 − CR*	528,155	328	480	351-400
TREC 8	Volume 4 & 5 − CR	528,155	328	480	401-450
TREC Robust 2004	Volume 4 & 5 − CR	528,155	328	480	301-450 601-700
TREC Web 2010	ClueWeb09	21,258,800	629	1096	51-100
TREC Web 2011	ClueWeb09	21,258,800	629	1096	101-150
TREC Web 2012	ClueWeb09	21,258,800	629	1096	151-200

* : Congressional Record documents.

in Section 5.3, we generate the grid points by evenly partitioning the search space of each parameter into 20 parts and making a complete combination of all the parameters of that retrieval model. Furthermore, based on the five sub-baselines, we easily get five more sub-baselines by setting *pfb* =TRUE and fixing *fbDocs* = 10, *fbTerms* = 10, *fbMu* = 2500, and *fbOrigWeight* = 0.5. For instance, the baseline LM-DIR + PFB contains 20 search points obtained by setting μ to the values partitioning $[0, 3000]$ into 20 parts and setting *pfb*=TRUE. For all the sub-baselines, *stopper* and *stemmer* are set to TRUE. This is a common practice in the use of grid search, and allows the reduction of the 6.6×10^{19} possible points of our hyperparameter space down to 1.8×10^4.

Random Search. The third baseline is random search [3]. For random search, we use 3 sampling designs, *Uniform*, *Latin*, *Sobol*, and we allow its number of iteration same with Bayesian Optimization .

6 RESULTS AND ANALYSIS

6.1 Decompose component effect of Bayesian Optimization

This experiment is designed to answer RQ1. In order to study which components of Bayesian Optimization has the strongest effect on the performance of Bayesian Optimization in finding optimal system configurations we run a full factorial experiment, using 54 joint strategies of Bayesian Optimization : the 3 initialization strategies × the 3 covariance functions × the 3 acquisition functions × the 2 selection strategies. The experiment was run on the Robust 2004 dataset. We measure performance on the basis of MAP and we run an analysis of variance (ANOVA) considering only the main effects of the Bayesian Optimization components (that is we ignore any interactions between them, since we do not have enough data points for a detailed analysis).

The results of the ANOVA can be seen in Table 2. The important observation lays in the last column of the table; this is the p-value that designates whether a factor has a significant effect when $p(>F)<0.05$, or there is not enough evidence to reach that conclusions otherwise. As we can observe, and in accordance to previous work [26] ANOVA suggests that the choice of the covariance function is the most important decision one needs to make when instantiating Bayesian Optimization , at least among the components we considered in this work. Therefore, we vary covariance functions and fix the rest strategies by selecting the respective best ones, that is *Sobol + EI + Incumbent* in the following sections.

Table 2: The effects of different components of Bayesian Optimization on objective function via ANOVA.

	SS	DF	F	$p(>F)$
Initialization strategy	0.000566	2.0	0.903685	0.412144
Covariance function	0.002976	2.0	4.748861	**0.013338**
Acquisition function	0.000647	2.0	1.032076	0.364368
Selection strategy	0.000543	1.0	1.734376	0.194376

6.2 Optimizing IR systems using Bayesian Optimization

This experiment is designed to answer RQ2. We first run the four methods on Robust 2004 or Web 2012 as the training set, then we tested the corresponding optimal configurations on TREC 5-8 or Web 2011-2012. In order to have a thorough study of Bayesian Optimization , we further experimented on the 2-dimensional space which is is λ and μ for the two stage smoothing, and the 18-dimensional space which includes the complete hyperparameters introduced in Section 5.3 respectively.

Table 3 shows the performance of the best configurations found by each method. As expected, there is no big difference among

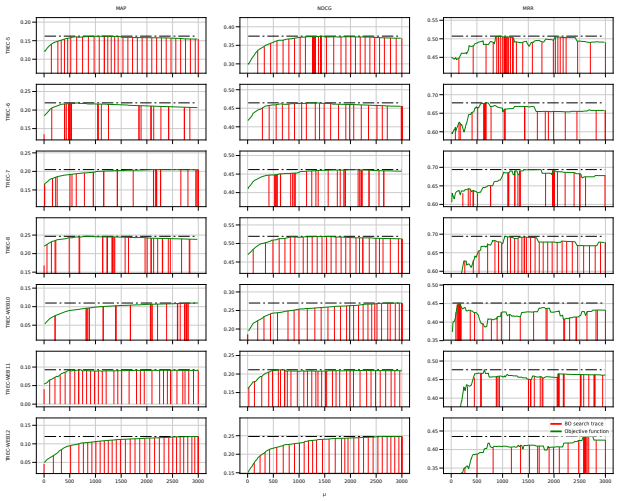
the four search methods in 2-dimensional space. It is because both grid search and random search are able to cover the space in low dimensional space. However, grid search needs more budget compared with Bayesian Optimization and random search. Therefore, Bayesian Optimization , grid search and random search can achieve comparable performance, but Bayesian Optimization and random search are more efficient in 2-dimensional spaces than grid search. The situation changes with the number of dimension increasing. We can see Bayesian Optimization performs slightly better than random search, grid search and manual method in 18-dimensional space. Random search is comparable with Bayesian Optimization in low (2) dimensional space, but it fails in high (18) dimensional space. However, there seems no significant difference among the four methods according to the results in Table 3 .

6.3 Optimization behaviour of Bayesian Optimization

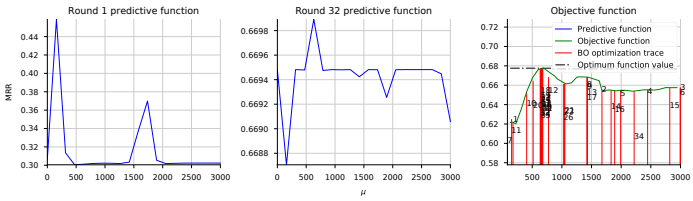
This experiment is designed to answer RQ3. We study how Bayesian Optimization searches points in the search space and how different covariance functions affect this behaviour. For visualisation convenience we take 1-dimensional and 2-dimensional hyperparameter space as examples.

The hyperparameter considered in 1-dimensional space is μ in language model with Dirichlet smoothing. We initialised the posterior of the surrogate model with 2 points, and iteratively searched 32 points. We also recorded the predictive function predicted by the surrogate model in round 1 and 32. The result is show in Figure 4. We can see that the intensive red lines tend to appear near the global optimum for most datasets and measures, indicating that Bayesian Optimization spends more efforts and exploits much near the global optimum. It is more obvious when the objective function is highly irregular like MRR. By "irregular" we mean there are many local optimums. On the right panel of Figure 4a, we find that Bayesian Optimization prefers exploitation near these local optimums and allows exploration in other areas, and finally it spends most efforts near the global optimum. Figure 4b shows the zoomed-in result on TREC 6 + MRR. The posterior of the surrogate model in round 1 does not know much about the objective function and predicts the same value for most points except the area near the two initial points. Whereas in round 32, the predictive function can model the rough trend quite well, where several local maximums and minimums are quite consistent with the real objective function. Overall speaking, Bayesian Optimization 's performance in 1-dimensional space is as good as we expect in terms of exploration and exploitation. Its preference of exploitation near global optimum makes it a reliable optimization approach.

The hyperparameters considered in 2-dimensional space are μ and λ in language model with two-stage smoothing. We initialised the posterior of the surrogate model with 4 points, and iteratively searched 32 points. Same as 1-dimensional case, we also recorded the predictive function in round 1 and 32. In Figure 5 we ploted the results of the SE covariance function and the Matérn1 covariance function respectively. The first two panels of Figure 5a shows the effect of Matérn1 on the predictive function. In round the surrogate model predicts that the lower left corner has the potential of getting high values. In round 32 the predictive function is quite rugged



(a) Overall optimization behaviour in a glance. Covariance function: Matérn1, objective functions: MAP, NDCG, MRR. The red vertical line associated with a number denotes Bayesian Optimization searches which point in which round. The green curve denotes the real objective function generated by 128 devisions of the interval [0, 3000]. The black dot line denotes the maximum objective function value.



(b) Zoomed-in result on TREC 6 + MRR. The first two panels are the predictive objective functions in round 1 and 32; the last panel is the real objective function. The red vertical line associated with a number denotes Bayesian Optimization searches which point and in which round.

Figure 4: Optimization behaviour of Bayesian Optimization in 1-dimensional space $X = [0, 3000]$.

because multiple local optimums are observed. It predicts the area around $(2800, 0.7)$ having a global optimum, which is quite consistent with the real objective function. The right panel compares the point traces of Bayesian Optimization , random search and manual search. At the beginning, Bayesian Optimization prefers exploration and tries to cover a large area of the search space. Later it prefers exploitation as we can see it spends more efforts near the global optimum. It is also interesting to compare the way how the two covariance functions model the real objective function. We know that the smoothness of the samples generated by the two covariance functions conforms to the order: SE >Matérn1. This is consistent with Figure 5. The real objective function in this case is quite smooth and seems to have one optimum, therefore SE is enough to model its irregularity. As an evidence we can see that more points are searched within green contour line on the most right side in Figure 5b compared with Figure 5a.

Table 3: The best performing configurations for all search methods.

Method	Budget	Train A			Test A			Train B			Test B		
		MAP	NDCG	MRR	MAP	NDCG	MRR	MAP	NDCG	MRR	MAP	NDCG	MRR
2-D													
Manual	-	2415	5144	6915	1986	4446	6091	1195	2508	4084	1002	2393	4305
Grid	400	2531	5239	7028	2058	4535	6255	1216	2523	4350	1002	2395	4458
Random	4+32	2529	5239	7020	2058	4534	6341	1215	2522	4254	1000	2395	4458
BO SE	4+32	2520	5239	7022	2010	4284	6258	1212	2518	4352	998	2388	4408
BO Matérn1	4+32	2527	5239	7020	2056	4514	6314	1204	2516	4353	1003	2394	4389
18-D													
Manual	-	2755	5507	6994	2221	4817	5744	1209	2509	4242	939	2282	4019
Grid	2480	2889	5616	7248	2256	4824	6184	1404	2620	4773	922	2311	4100
Random	36+64	2692	5454	7041	2084	4776	6309	1404	2620	4773	922	2311	4100
BO SE	36+64	2908	5499	7162	2301	4705	6077	1395	2576	4860	926	2281	3933
BO HSE	36+64	2924	5549	7032	2064	4737	6273	1395	2551	5181	935	2334	3931
BO Matérn1	36+64	2925	5571	7192	2283	4781	6168	1398	2593	5268	929	2372	4140
BO HMatérn1	36+64	2747	5661	7050	2161	4864	6301	1308	2618	5502	974	2315	4160

Note: Magnitude of numbers is 10^{-4}. Train A denotes TREC Robust 2004, Test A denotes TREC 5-8, Train B denotes TREC Web 2012, Test B denotes TREC Web 2010-2011.

To sum up, Bayesian Optimization balances exploitation and exploitation by spending more search budget near the global optimum or local optimums, which makes it a reliable optimization approach. This optimization behaviour is affected by the covariance function. If the objective function is very irregular like MRR, Matérn1 is recommended; while SE is recommended if relative objective functions like MAP and NDCG.

7 CONCLUSIONS AND FUTURE WORK

In this paper we study the problem of retrieval system optimization. We propose the use of Bayesian Optimization to jointly search and optimize over the hyperparameter space. Given the heterogeneous hyperparameters in retrieval systems we suggest the use of four covariance functions that can handle both continuous and categorical hyperparameters, the SE, HSE, Matérn1 and HMatérn1 covariance function. We analyze the effect of the different components of Bayesian Optimization and reach at the same conclusion as prior research on the topic [26] that it is the choice of the covariance functions that have the strongest effect on the performance of Bayesian Optimization . To demonstrate the effectiveness and efficiency of Bayesian Optimization to identify a good system configuration, we tested it on both the ad-hoc and the web test collections of TREC. In both collections we demonstrate that Bayesian Optimization outperforms manual tuning, grid search and random search, both in terms of the retrieval effectiveness of the best configuration found, and in terms of efficiency in finding this configuration. We further examined the optimization behaviour of Bayesian Optimization in terms of exploitation and exploitation. We found it spends more search budget near the global optimum or local optimums, and this optimization behaviour is affected by covariance functions of different smoothness.

One should note that for the Gaussian Process the bounds for all dimensions of the search space are axis-aligned, i.e. the search space is a *hyper-rectangle* [25]. This is contradictory with the conditional structure of IR hyperparameters, which means that our instantiation of Bayesian Optimization wastes time in searching in inactive dimensions. We leave the study of surrogate model that can solve conditional hyperparameters as a future work.

REFERENCES

[1] Timothy G. Armstrong, Alistair Moffat, William Webber, and Justin Zobel. 2009. Improvements That Don'T Add Up: Ad-hoc Retrieval Results Since 1998. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, New York, NY, USA, 601–610. https://doi.org/10.1145/1645953.1646031

[2] Richard E Bellman. 2015. *Adaptive control processes: a guided tour*. Princeton university press.

[3] James Bergstra and Yoshua Bengio. 2012. Random search for hyper-parameter optimization. *Journal of Machine Learning Research* 13, Feb (2012), 281–305.

[4] James Bergstra, Daniel Yamins, and David D Cox. 2013. Making a Science of Model Search: Hyperparameter Optimization in Hundreds of Dimensions for Vision Architectures. *ICML (1)* 28 (2013), 115–123.

[5] James S Bergstra, Rémi Bardenet, Yoshua Bengio, and Balázs Kégl. 2011. Algorithms for hyper-parameter optimization. In *Advances in Neural Information Processing Systems*. 2546–2554.

[6] 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* 21, 13 (2015), 1726–1745.

[7] Russel E Caflisch, William J Morokoff, and Art B Owen. 1997. *Valuation of mortgage backed securities using Brownian bridges to reduce effective dimension*. Department of Mathematics, University of California, Los Angeles.

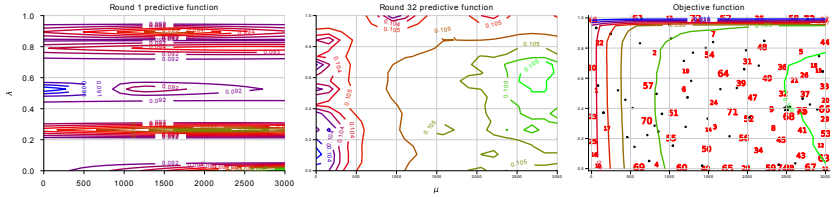
[8] Romain Deveaud, Josiane Mothe, and Jian-Yun Nia. 2016. Learning to Rank System Configurations. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management (CIKM '16)*. ACM, New York, NY, USA, 2001–2004. https://doi.org/10.1145/2983323.2983894

[9] Laurence Charles Ward Dixon and Giorgio Philip Szegö. 1978. *Towards global optimisation 2*. North-Holland Amsterdam.

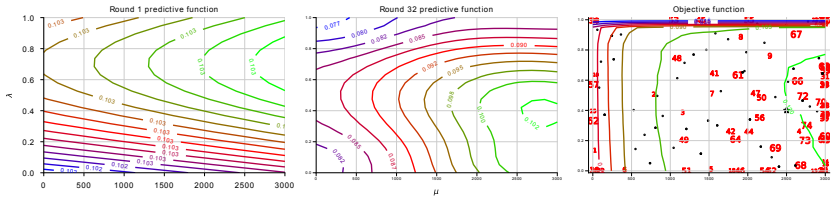
[10] Katharina Eggensperger, Matthias Feurer, Frank Hutter, James Bergstra, Jasper Snoek, Holger Hoos, and Kevin Leyton-Brown. 2013. Towards an empirical foundation for assessing bayesian optimization of hyperparameters. In *NIPS workshop on Bayesian Optimization in Theory and Practice*. 1–5.

[11] 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 and Development in Information Retrieval (SIGIR '16)*. ACM, New York, NY, USA, 25–34. https://doi.org/10.1145/2911451.2911530

[12] Parantapa Goswami and Eric Gaussier. 2013. Estimation of the Collection Parameter of Information Models for IR. In *Proceedings of the 35th European Conference on Advances in Information Retrieval (ECIR'13)*. Springer-Verlag, Berlin, Heidelberg, 459–470. https://doi.org/10.1007/978-3-642-36973-5_39



(a) Matérn1 + MAP.



(b) SE + MAP.

Figure 5: Optimization behaviour of Bayesian Optimization in 2-dimensional space $X = \{(\mu, \lambda)|\mu \in [0, 3000], \lambda \in [0, 1]\}$. Objective function: MAP, Data set: TREC-WEB12. The contour lines depict the predictive functions in round 1 and 32 on the first two panels, and the real objective function on last panel. The red numbers denote Bayesian Optimization searches which point and in which round; the black dots denote the point trace of random search; and the star denotes the point of manual search.

[13] Ben HE and Iadh Ounis. 2003. A Study of Parameter Tuning for Term Frequency Normalization. In *Proceedings of the Twelfth International Conference on Information and Knowledge Management (CIKM '03)*. ACM, New York, NY, USA, 10–16. https://doi.org/10.1145/956863.956867

[14] Ben He and Iadh Ounis. 2007. On Setting the Hyper-parameters of Term Frequency Normalization for Information Retrieval. *ACM Trans. Inf. Syst.* 25, 3, Article 13 (July 2007). https://doi.org/10.1145/1247715.1247719

[15] Ben He and Iadh Ounis. 2007. Parameter Sensitivity in the Probabilistic Model for Ad-hoc Retrieval. In *Proceedings of the Sixteenth ACM Conference on Conference on Information and Knowledge Management (CIKM '07)*. ACM, New York, NY, USA, 263–272. https://doi.org/10.1145/1321440.1321479

[16] Matthew W Hoffman and Bobak Shahriari. 2014. Modular mechanisms for Bayesian optimization. In *NIPS Workshop on Bayesian Optimization*. Citeseer.

[17] Frank Hutter, Holger H Hoos, and Kevin Leyton-Brown. 2011. Sequential model-based optimization for general algorithm configuration. In *International Conference on Learning and Intelligent Optimization*. Springer, 507–523.

[18] Frank Hutter and Michael A Osborne. 2013. A Kernel for Hierarchical Parameter Spaces. *arXiv preprint arXiv:1310.5738* (2013).

[19] Sadegh Kharazmi, Falk Scholer, David Vallet, and Mark Sanderson. 2016. Examining Additivity and Weak Baselines. *ACM Trans. Inf. Syst.* 34, 4, Article 23 (June 2016), 18 pages. https://doi.org/10.1145/2882782

[20] Yuanhua Lv and ChengXiang Zhai. 2009. A Comparative Study of Methods for Estimating Query Language Models with Pseudo Feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management (CIKM '09)*. ACM, New York, NY, USA, 1895–1898. https://doi.org/10.1145/1645953.1646259

[21] Jonas Mockus. 1994. Application of Bayesian approach to numerical methods of global and stochastic optimization. *Journal of Global Optimization* 4, 4 (1994), 347–365.

[22] Carl Edward Rasmussen and Christopher K. I. Williams. 2005. *Gaussian Processes for Machine Learning (Adaptive Computation and Machine Learning)*. The MIT Press.

[23] François Rousseau and Michalis Vazirgiannis. 2013. Composition of TF Normalizations: New Insights on Scoring Functions for Ad Hoc IR. In *Proceedings of the 36th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '13)*. ACM, New York, NY, USA, 917–920. https://doi.org/10.1145/2484028.2484121

[24] Jangwon Seo and W. Bruce Croft. 2010. Unsupervised Estimation of Dirichlet Smoothing Parameters. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '10)*. ACM, New York, NY, USA, 759–760. https://doi.org/10.1145/1835449.1835602

[25] Bobak Shahriari, Kevin Swersky, Ziyu Wang, Ryan P Adams, and Nando de Freitas. 2016. Taking the human out of the loop: A review of bayesian optimization. *Proc. IEEE* 104, 1 (2016), 148–175.

[26] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. 2012. Practical bayesian optimization of machine learning algorithms. In *Advances in neural information processing systems*. 2951–2959.

[27] Trevor Strohman, Donald Metzler, Howard Turtle, and W. Bruce Croft. 2005. *Indri: a language-model based search engine for complex queries*. Technical Report. in Proceedings of the International Conference on Intelligent Analysis.

[28] Kevin Swersky, David Duvenaud, Jasper Snoek, Frank Hutter, and Michael A Osborne. 2014. Raiders of the lost architecture: Kernels for Bayesian optimization in conditional parameter spaces. *arXiv preprint arXiv:1409.4011* (2014).

[29] Kevin Swersky, Jasper Snoek, and Ryan P Adams. 2013. Multi-task bayesian optimization. In *Advances in neural information processing systems*. 2004–2012.

[30] 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*. ACM, 585–593.

[31] Andrew Trotman, Antti Puurula, and Blake Burgess. 2014. Improvements to BM25 and Language Models Examined. In *Proceedings of the 2014 Australasian Document Computing Symposium (ADCS '14)*. ACM, New York, NY, USA, Article 58, 8 pages. https://doi.org/10.1145/2682862.2682863

[32] Christophe Van Gysel, Evangelos Kanoulas, and Maarten de Rijke. 2017. Pyndri: a Python Interface to the Indri Search Engine. In *Advances in Information Retrieval: 39th European Conference on IR Research, ECIR 2017*. Springer International Publishing.

[33] Ellen M. Voorhees and Donna Harman (Eds.). 2001. *Proceedings of The REtrieval Conference (TREC 1-9)*. Vol. NIST Special Publication. National Institute of Standards and Technology (NIST). http://trec.nist.gov/pubs.html

[34] Lidan Wang, Minwei Feng, Bowen Zhou, Bing Xiang, and Sridhar Mahadevan. 2015. Efficient hyper-parameter optimization for NLP applications. In *Proceedings of EMNLP*, Vol. 15. 2112–2117.

[35] Ziyu Wang and Nando de Freitas. 2014. Theoretical analysis of bayesian optimisation with unknown gaussian process hyper-parameters. *arXiv preprint arXiv:1406.7758* (2014).

[36] Dani Yogatama and Noah A Smith. 2015. Bayesian optimization of text representations. *arXiv preprint arXiv:1503.00693* (2015).

[37] Chengxiang Zhai and John Lafferty. 2001. A Study of Smoothing Methods for Language Models Applied to Ad Hoc Information Retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '01)*. ACM, New York, NY, USA, 334–342. https://doi.org/10.1145/383952.384019

[38] ChengXiang Zhai and John Lafferty. 2002. Two-stage Language Models for Information Retrieval. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '02)*. ACM, New York, NY, USA, 49–56. https://doi.org/10.1145/564376.564387