Language-independent Query Representation for IR Model Parameter Estimation on Unlabeled Collections

Parantapa Goswami Massih-Reza Amini Eric Gaussier

Université Grenoble Alps,
CNRS-LIG/AMA
Grenoble, France

firstname.lastname@imag.fr

ABSTRACT

We study here the problem of estimating the parameters of standard IR models (as BM25 or language models) on new collections without any relevance judgments, by using collections with already available relevance judgements. We propose different query representations that allow mapping queries (with and without relevance judgments, from different collections, potentially in different languages) into a common space. We then introduce a kernel regression approach to learn the parameters of standard IR models individually for each query in the new, unlabeled collection. Our experiments, conducted on standard English and Indian IR collections, show that our approach can be used to efficiently tune, query by query, standard IR models to new collections, potentially written in different languages. In particular, the versions of the standard IR models we obtain not only outperform the versions with default parameters, but can also outperform the versions in which the parameter values have been optimized globally over a set of queries with target relevance judgements.

Categories and Subject Descriptors

H.3 [**Information Storage and Retrieval**]: Information Search and Retrieval - Search process; I.2 [**Artificial Intelligence**]: Learning - Parameter learning

General Terms

Algorithms, Experimentation, Theory

Keywords

IR Theory, Learning IR Parameters, Transfer learning

1. INTRODUCTION

In many situations, one has to deploy IR models on new collections (on new domains or languages) from scratch. In such cases, developing relevance judgments so as to adapt the retrieval models to the new collections considered is a costly operation. An alternative is of course to simply rely on default parameters of the IR models, hoping that the results obtained will be reasonable, *i.e.* not too far away from the ones obtained by "adapting" the models (as we will see in Section 4, default results are reasonable on several collections, however not on all of them). This "default" strategy is the one traditionally adopted in IR evaluation campaigns (as TREC or CLEF) when new collections and languages are introduced. In fact, each time a collection changes substantially, *e.g.* through the introduction of new documents, then the IR models that are employed, should be adapted so as to follow the potential evolution of the collection.

Of course, one would like to perform such an adaptation at a minimal cost, and, ideally, without resorting to new, specific relevance judgments, albeit using any relevance judgments available on known, past collections. This is precisely the problem we are investigating in this study, focusing on learning the underlying parameter(s) of standard IR models as BM25 [20], language models (LM) [18] and information-based models (LGD) [7]. Our focus on these particular IR models is motivated by the fact that these models are the most widely used in different IR tasks (as *ad hoc* IR, structured IR or social IR for example) and serve as components of learning to rank models deployed *e.g.* on web collections. Thus, the fundamental problem we address can be formulated as follows: *How to infer the parameter values of standard IR models (BM25, LM, LGD) on collections without any relevance judgments by using past labeled collections?*

We will solve this problem, which relates to different research fields, as model adaptation or transfer learning, by mapping queries from different collections into common vector spaces in which regression functions can be efficiently learned. Two key elements of our approach are: (a) The method proposed allows to obtain parameter values for a single query, so that IR models are optimized per query on the new collection and (b) the representations used are language-independent, and parameter values can be estimated for new collections, in new languages, for any standard IR models.

The remainder of the paper is organized as follows. We first discuss related work in Section 2. We then describe in Section 3 the approach developed for learning the parameter(s) of standard IR models on collections with no relevance judgments, using known collections with relevance judgments. We then illustrate several aspects of the method we propose in Section 4 on several collections, prior to conclude in Section 5.

2. RELATED WORK

IR models are generally defined with free parameters, as the parameters b and k_1, k_3 for BM25 [19], the Dirichlet smoothing coefficient μ for language models with Dirichlet prior [11] and the c parameter for LGD [7]. These parameters are query and collection dependent and hence their tuning is unavoidable to achieve good performance. The necessity of empirical tuning of the parameters of traditional models has been advocated in different studies, as in [26]. In the case where the relevance judgments exist, the optimal values of these free parameters are generally found by testing different parameter values from a predefined set of discrete values for each parameter on a set of queries with associated relevance judgments, and then selecting the parameter values that lead to the best performance with respect to the evaluation measure considered. This greedy search has been found to be competitive compared to simple learning strategies that optimize some differentiable IR measures [23]. This said, creating new test collections, or manually assigning relevance judgments for even a small set of queries, is a tedious task [3].

Other studies directly considered the problem of unsupervised parameter estimation [26, 22]. In [26], an automated *leave-one-out* likelihood method to estimate the Dirichlet prior smoothing parameter μ on unlabeled collection is proposed; the study in [22] does not aim at estimating standard IR model parameters but rather focuses on pseudo-relevance feedback and makes use of a mixture model with a regularized expectation maximization method to estimate the feedback parameter. If these methods have been shown to work well on several collections, they were however developed under the language model framework with a probabilistic view over the generation of words, and cannot be (at least directly) applied to IR models outside the language model family. Some studies also focus on unlabeled collections [21, 25], but with a different, evaluation-oriented goal, namely the one of ranking different IR models. The assumptions and methods used in these studies radically differ from ours.

In between supervised methods, making use of relevance judgements on the collection queried, and unsupervised methods, that do not rely on such relevance judgements, lie methods (often referred to as *transfer* or *cross-domain* learning) that aim at exploiting past relevance judgements, available on some *source* collections, to learn models on unlabeled, *target* collections. Many efforts have indeed already been made to conceive and develop different collections through existing competitions like TREC, and one can wonder whether such annotated collections can be used for unannotated ones.

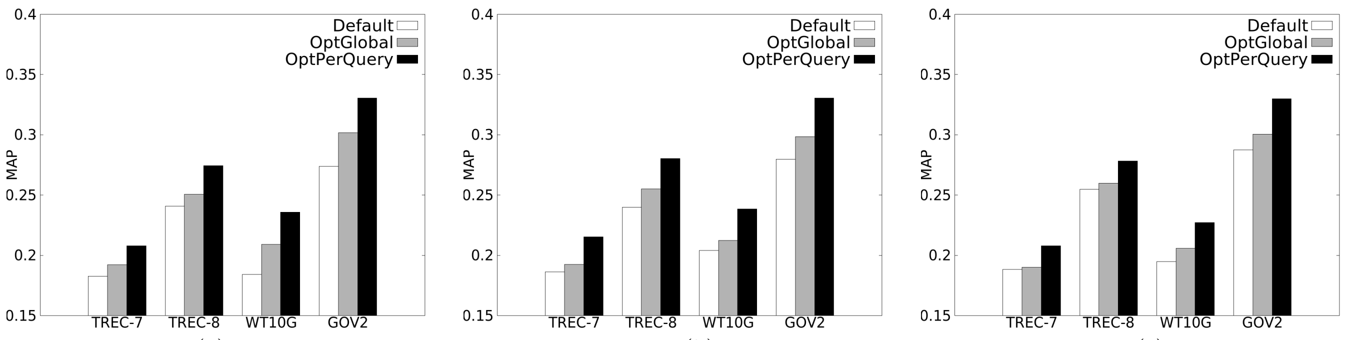
Several innovative studies have adapted cross-domain approaches to the IR learning to rank framework [5, 4, 8, 9]. While some of them made use of labeled queries in the source and a small set of labeled queries in the target to learn a ranking function [4], the others considered that only the source collection contains labeled information, while queries in the target collection have no associated relevance judgments. In [8, 2], the transfer learning is done by weighting documents in the source collection using unlabeled queries and documents from the target dataset. More precisely, a classifier is first learned aiming at discriminating source and target documents or queries on the basis of standard features similar to the ones used in learning to rank for IR. The score of the classifier on each source (query,document) pair is then used as an indicator of the proximity of this pair to the target collection. A ranking function is then learned on the source collection, with (query,document) pairs weighted according to their proximity to the target collection: the closer a pair is to the target collection, the more importance it will have in the learning process. This approach has been shown to work in the uncommon case where the source and target collections are close to each other. However, when the two collections are far away, then the model learned is not appropriate. To address this problem, [9] proposed a transfer technique approach that relies on relative relevance judgement pairs induced on the target collection from a set of source queries.

Our approach differs from these ones in that we are focusing on learning the parameter values of standard IR models, whereas default values of these models are used in the above studies. Standard IR models are used in many IR tasks and any procedure that can improve their performance on new, unlabeled collections will be beneficial to the IR community. Our approach also differs from the ones proposed in [26] and [22] (discussed above) in that we are proposing a solution that can be used for *any* standard IR model, and not only for the language model family. The representation we are using however bears some similarity with the one used in [26] as they both use query representations based on word frequency distributions over the collections considered (see Section 3). We are however using a richer summary of these distributions through the use of their three first moments. Similar to what is done in [22, 14], we are learning parameter values per query (different target queries may use different parameter values), and are relying, as [14], on a regression framework to do so. If this latter study is close in spirit to ours, it also has some significant differences. In particular, we consider here the standard *ad hoc* IR setting and IR models from different families, whereas [14] focuses on pseudo-relevance feedback for relevance models. Because of this difference, we rely on different features and query representations and, from them, on different regression functions (our query representation calls for kernel regression methods whereas standard regression functions are used in [14]).

3. LEARNING IR MODEL PARAMETERS ON UNLABELED COLLECTIONS

In the remainder, the unlabeled collection queried will be referred to as the *target* collection whereas a collection available with relevance judgements will be referred to as the *source* collection (possibly formed by merging several existing collections). We place ourselves here in a standard *ad hoc* IR setting, only assuming that one has access to standard tools on the target, unlabeled collection (stop-word list and stemming procedure). This corresponds to a standard, general situation, but the methodology we propose can directly be applied if one has access to additional resources and tools on both the source and target collections. Our goal is to learn the parameters of standard IR models on a target, unlabeled collection by transferring relevance information from a source collection, so to improve over the performance obtained with parameters set to their default values. Two questions that directly arise are:

1. What is the upper bound on the gain one can have



(a) `BM25` (b) `LM` (c) `LGD`

Figure 1: Performance of IR models with default parameter values (□), parameters optimized within the whole collection (▨) and per query basis (■). The results are in terms of `MAP` on `TREC-7,8`, `WT10G` and `GOV2` collections for (a) `BM25`, (b) `LM`, and (c) `LGD` IR models.

over the default strategy corresponding to setting the parameters to their default values? In other words, is it worth trying to improve over this strategy?

2. What are the upper bounds of methods estimating parameter values globally over all queries and of methods estimating those values query per query? In other words, is it worth trying to find parameter estimates individually for each query?

To answer these questions, we computed the mean average precision (`MAP`) of three standard IR models (`BM25`, the Dirichlet language model `LM` and the log-logistic model `LGD`) on four collections (`TREC-7`, `TREC-8`, `WT10G` and `GOV2`)[1] with three different settings:

(a) Using the default parameter values of each model;

(b) Selecting, in a given set of values (described in Table 2), the parameter value that provides the highest `MAP` on all target queries (this makes use of the relevance judgements on the target collection). The associated `MAP` corresponds to the upper bound of methods aiming at finding the best value globally for all queries;

(c) Selecting, in the same given set of values and for each query, the parameter value that provides the highest average precision for the query (as before, this makes use of the relevance judgements on the target collection). The mean of all the average precisions obtained in this way yields a `MAP` value that corresponds to the upper bound of methods aiming at finding the best value individually per query.

The results obtained are displayed in Figure 1. As one can note, the difference between default values and optimized ones varies according to the model and the collection considered. This difference is relatively small, in the range $[0, 2.5\%]$ for all collections and models, when the optimization is performed globally over all queries. It is more important when the optimization is performed query by query, even though the improvement varies from one collection to another: around 2 to 3% for all models on `TREC-7,8`, around 5 to 6% for all models on `GOV2` and `WT10G`, the difference being however less marked for `LGD`. These results show firstly that it may not be possible to obtain significant improvements over default values on all collections, as such values yield results close to the best possible ones, and secondly that higher gains can be expected by estimating the parameters per query.

[1]The collections are described in Section 4.

Estimating parameter values per query is not necessarily a difficult task. Indeed, from a source collection and a given representation of queries, one can learn, using the relevance judgements as in the setting (c) above, a regression function that can then be used to associate parameter values to new queries. However, in order to apply such a regression function on target queries, one needs to rely on a representation of queries common to both the source and target collections.

3.1 Query representation

We consider a source collection $\mathcal{S}$, composed of a set of documents $\mathcal{D}^s$, a set of n queries $\mathcal{Q}^s = \{q_1^s, \ldots, q_n^s\}$ and relevance judgments for each query in $\mathcal{Q}^s$. Furthermore, we consider a target collection $\mathcal{T}$, composed of a set of documents $\mathcal{D}^t$ and a set of u queries $\mathcal{Q}^t = \{q_1^t, \ldots, q_u^t\}$ without any relevance judgments. Each query q, in any collection, is constituted by a set of terms $q = \{w_1^q, \ldots, w_{|q|}^q\}$, corresponding to the standard representation of queries. It is however not possible to rely here on this simple query representation as different collections use different vocabularies, potentially from different languages.

Representation based on the tf distribution

In order to bypass the above problem, we extend the idea of [26], who assumed that the distribution of query words regarding relevant documents does not vary from source to target collections, by considering that two queries are similar if the words they contain have similar *idf* (inverse document frequency) and tf (term frequency) distributions over their collections. As the tf distributions of source and target words are often high-dimensional and may differ in length (as the collections contain different documents), we rely here on a summary of these distributions obtained through the estimates of their first (mean), second (standard deviation) and third (skewness) moments, respectively denoted by $\mu(q)$, $\sigma(w)$ and $sk(w)$ for a given word w. For a bounded distribution, like the term frequency distribution here, combination of the moments of all orders (from 0 to ∞) uniquely determines the distribution. Thus the first three moments can be considered to define a natural summary of the distribution of the term frequency scores of the word in the collection.

We finally obtain the following representation of a query in which each query word corresponds to a 4-dimensional vector[2]:

$$\begin{cases} q = \{\mathbf{w}_1^q, \ldots, \mathbf{w}_{|q|}^q\} \\ \mathbf{w}^q = (idf(w^q), \mu(w^q), \sigma(w^q), sk(w^q)) \end{cases} \quad (1)$$

where $idf(w)$ denote the inverse document frequency of the word w.

Learning extended representations with auto-encoders

We further investigate the possibility to enhance the proposed tf and idf based representation (Eq. 1) using an Auto-Encoder (AE). AEs [10] are a family of feed-forward neural networks that are trained to reconstruct the input data by performing two steps. In the first step, an input vector from $\mathbb{R}^d$ is projected to a space $\mathbb{R}^a$, called encoding, using non-linear bijective functions. In the second step, the encoded vector is projected into the original space of dimension d using again non-linear bijective functions. These models have been found effective to extract text or image representations as it has been shown that the neurons of the hidden layer are able to detect generic concepts [13, 12]. The AE model that we developed is trained using stochastic back-propagation algorithm [1] over all the 4-dimensional vector representation of terms ($d = 4$) that are in the source and the target collections[3]. After this step, any query q is then represented as a set of term vectors:

$$\begin{cases} q = \{\mathbf{w}_1^q, \ldots, \mathbf{w}_{|q|}^q\} \\ \mathbf{w}_j^q : \text{ vector found by the auto-encoder} \end{cases} \quad (2)$$

From these mappings we finally learn the association between query representations and parameter values, as described in the next section.

3.2 Learning the regression function

For each query in the source collection, one can obtain, as mentioned before, the optimal value of the parameter of each of the standard models `BM25`, `LM` and `LGD`. The vector collecting these optimal values for each query in $\mathcal{Q}^s$ will denoted by $\mathbf{c}_{opt}^s = (c_{q_1^s}, \ldots, c_{q_n^s})^\top$, where $^\top$ corresponds to the transpose operator. Using the association between queries in $\mathcal{Q}^s$ and $\mathbf{c}_{opt}^s$ and relying on a common vector space for queries, one can learn a regression function that maps any query in $\mathcal{Q}^t$ to a parameter value. Figure 2 illustrates this procedure.

In order to use regression methods, one first needs to define a common vector space for queries. We do so by introducing kernels that directly operate on the representations defined above.

Simple PDS kernels

Starting from the representations of queries defined above ($q = \{\mathbf{w}_1^q, \ldots, \mathbf{w}_{|q|}^q\}$, with $\mathbf{w}^q$ obtained either through the idf and tf distribution or from an auto-encoder), we first

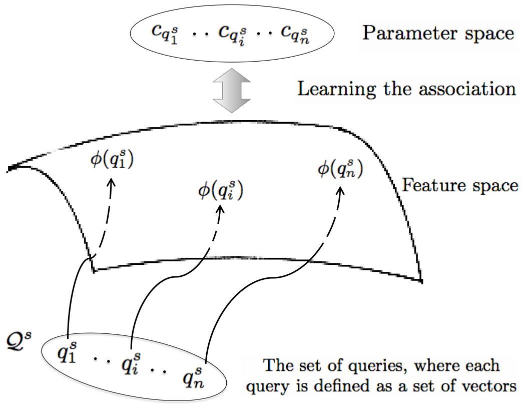

$c_{q_1^s} \cdot\cdot c_{q_i^s} \cdot\cdot c_{q_n^s}$
Parameter space
Learning the association
$\phi(q_1^s)$
$\phi(q_i^s)$
$\phi(q_n^s)$
Feature space
$\mathcal{Q}^s$
$q_1^s \cdot\cdot q_i^s \cdot\cdot q_n^s$
The set of queries, where each query is defined as a set of vectors

Figure 2: Each target query is mapped in the common vector space and an associated parameter of the IR model is predicted using the learned association model.

consider the following convolution kernel[4]:

$$\kappa_{\text{all}}(q, q') = \frac{1}{|q|}\frac{1}{|q'|}\sum_{i=1}^{|q|}\sum_{j=1}^{|q'|}\left\langle \mathbf{w}_i^q, \mathbf{w}_j^{q'} \right\rangle \quad (3)$$

Exploiting twice the bi-linearity of the dot product, one has:

$$\begin{aligned} \kappa_{\text{all}}(q, q') &= \frac{1}{|q|}\frac{1}{|q'|}\left(\left\langle \sum_{i=1}^{|q|}\mathbf{w}_i^q, \sum_{j=1}^{|q'|}\mathbf{w}_j^{q'} \right\rangle\right) \\ &= \left\langle \sum_{i=1}^{|q|}\frac{\mathbf{w}_i^q}{|q|}, \sum_{j=1}^{|q'|}\frac{\mathbf{w}_j^{q'}}{|q'|} \right\rangle \end{aligned}$$

The mapping $\boldsymbol{\phi}$ associated with the above kernel takes the form:

$$\boldsymbol{\phi}(q) = \sum_{i=1}^{|q|}\frac{\mathbf{w}_i^q}{|q|} \quad (4)$$

ϕ thus corresponds to the average of the vectors of the words present in q and κ_{all} amounts to the dot product between the average word vectors of each query:

$$\kappa_{\text{all}}(q, q') = \langle \boldsymbol{\phi}(q), \boldsymbol{\phi}(q') \rangle \quad (5)$$

From the form above, one can further define new kernels, by substituting the dot product in equation (5) by any valid PDS kernel. We consider here homogeneous polynomial kernels and Gaussian kernels, widely used in text processing, leading to:

$$\begin{aligned} \kappa_{\text{poly-}\delta}(q, q') &= (\langle \boldsymbol{\phi}(q), \boldsymbol{\phi}(q') \rangle)^\delta \\ \kappa_{\text{gau-}\sigma}(q, q') &= \exp(-\frac{||\boldsymbol{\phi}(q) - \boldsymbol{\phi}(q')||_2^2}{2\sigma^2}) \end{aligned}$$

where δ corresponds to the degree of the polynomial kernel and σ to the standard deviation of the Gaussian kernel.

These different kernels can be used with different kernel regression methods. In this study, we rely on kernel Support Vector Regression (kernel SVR) [24] as this method has been shown to perform well in practice.

[2]This corresponds to two additional dimensions wrt the idf and tf measures traditionally considered for query words.

[3]Note that the vectors of terms, as defined in equation 1, do not involve the knowledge of relevance judgments.

[4]The subscript "all" in κ_{all} denotes the fact that all pairs of words, from the two queries, are compared.

Kernel regression

Let κ denote any of the above-defined kernels and let $\mathbf{K}$ be the associated Gram (or kernel) matrix ($\mathbf{K}_{i,j} = \boldsymbol{\kappa}(q_i^s, q_j^s)$, $1 \leq i, j \leq n$). From the training set $\{(q_i^s, c_{q_i^s}),\ 1 \leq i \leq n\}$, the goal of kernel SVR is to learn a regression function in the kernel-induced feature space (potentially of infinite dimension) in order to associate to any new query q^t a parameter value c_{q^t}. The optimization problem for kernel SVR takes the following dual form (see for example [16]):

KSVR-opt:

$$\max_{\boldsymbol{\alpha},\boldsymbol{\alpha}'} -\epsilon(\boldsymbol{\alpha}' + \boldsymbol{\alpha})^\top \mathbf{1} + (\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{c}^s_{opt} - \frac{1}{2}(\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{K}(\boldsymbol{\alpha}' - \boldsymbol{\alpha})$$
$$\text{s. t. } 0 \leq \alpha_i,\ \alpha_i' \leq C (1 \leq i \leq n),\ (\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{1} = 0$$

where $\mathbf{1} \in \mathbb{R}^n$ is a vector containing only 1s and C and ϵ are hyper-parameters usually set through cross-validation (C serves as a regularization parameter and ϵ controls the accuracy with which errors are measured). $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathbb{R}^n$ are the parameters to be learned and are such that either α_i or α_i' is non null (this happens if q_i^s is a support vector), or they are both null (if q_i^s is not a support vector).

The above optimization problem is a convex quadratic programming problem that can be solved by any convex QP solver. The value predicted for a new query q^t in the target collection is obtained by:

$$c_{q^t} = \sum_{i=1}^{n} (\alpha_i' - \alpha_i)\boldsymbol{\kappa}(q_i^s, q^t) + b_i \qquad (6)$$

the offset b_l being defined for any support vector q_l^s by:

$$b_l = \sum_{i=1}^{n} (\alpha_i - \alpha_i')\boldsymbol{\kappa}(q_i^s, q_l^s) + c_{q_l^s} + \epsilon \qquad (7)$$

The overall process for predicting the parameter value of standard IR models on new, unlabeled collections can thus be summarized as follows. For each free parameter of the standard IR model under consideration:

1. Training step: Compute $\mathbf{c}^s_{opt}$ in $\mathcal{Q}^s$ and solve the KSVR-opt problem above with any convex QP solver;

2. Prediction: For each query of the target collection q^t, compute c_{q^t} through equations (6) and (7).

4. EXPERIMENTS

We present in this section experiments aimed at evaluating the validity of the approach described before.

4.1 Collections

We perform experiments on nine IR collections: one from CLEF[5], six from TREC[6] and two non-English collections from FIRE[7] containing Hindi and Bengali documents and queries. Basic statistics on these collections are provided in Table 1. We appended TREC-9 and TREC-10 Web tracks to experiment with WT10G, and TREC-2004 and TREC-2005 Terabyte tracks for experimenting with GOV2. Experiments are performed on the Terrier IR platform v3.5 (terrier.org) [17], and as for WEB oriented ad-hoc IR, we only considered the title of queries and dropped their descriptions. The preprocessing steps in creating an index include stemming using Porter stemmer and removing stop-words using the stop-word list provided by Terrier. On FIRE collections, we used the stop-word lists available at the corresponding website, but did not use any stemmer.

Collection	$\mathcal{N}$	l_{avg}	Index size	#queries
GOV2	25,177,217	646	19.6 GB	100
WT10G	1,692,096	398	1.3 GB	100
FIRE-BN	500,122	245	498.6 MB	50
TREC-3	741,856	261	427.7 MB	50
TREC-4	567,529	323	379.0 MB	50
TREC-5	524,929	339	378.0 MB	50
TREC-6,7,8	528,155	296	373.0 MB	50
FIRE-HN	331,599	178	225.5 MB	50
CLEF-3	169,477	301	126.2 MB	60

Table 1: Statistics of various collections used in our experiments, sorted by size.

In most of our experiments, we considered CLEF-3, TREC-3,4,5,6 as source collections, and used TREC-7,8, WT10G and GOV2 for testing. In order to see how the projection in the query space is dependent to the languages of the source and target collections, we also tested our strategy by learning the regression model on English collections WT10G, GOV2 and predicting model parameters on both FIRE collections, as well as the other way around by learning the regression model on FIRE collections and predicting model parameters on WT10G and GOV2. Results are evaluated using the mean average precision, MAP, and precision at 10 documents, P@10 (relevance judgments on the target collections are just used for evaluating the models). As MAP is the measure retained in our experiments to construct c^s_{opt} it is the one we privilege in our discussions (other measures, as P@10 or NDCG could of course be used to construct c^s_{opt} and estimate model parameters) whereas the P@10 is given as a illustrator of the behavior of the methods. A Wilcoxon statistical test, with a p-value of $p = 0.05$, is used to assess whether the difference, in terms of MAP, between two methods is significant or not.

4.2 Experimental setup

We used three different, widely used, IR ranking models, namely BM25, the language model with Dirichlet smoothing (LM), and the log-logistic information-based model LGD from the Divergence from Randomness family, and compared the performance of these models with their default values ($b = 0.75$ for BM25, $\mu = 2500.0$ for LM and $c = 1.0$ for LGD) against the values predicted by the proposed regression approach on target collections. In order to better evaluate the effect of parameter learning on each single free-parameter alone, we just considered the parameter b of BM25 and kept its two other parameters fixed to their default values ($k_1 = 1.2$ and $k_2 = 8.0$) in all of our experiments[8].

Finally, the number of neurons in the hidden layer of the auto-encoder was fixed by cross-validation on the source collections by considering numbers varying from 5 to 20. In all of our experiments, we used a server with an intel Xenon 1.8HGz processor and 16GB of RAM. The training time for learning the representations on this machine, with a maximum number of iterations fixed to 60 000 000, was less than

[5]www.clef-campaign.org
[6]trec.nist.gov
[7]www.isical.ac.in/~clia/

[8]The codes we used can be found on http://ama.liglab.fr/resourcestools/ir-parameter-learning/

			TREC-7		TREC-8		WT10G		GOV2	
			MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	def		$18.28^{\downarrow}$	*41.80*	$24.09^{\downarrow}$	*47.40*	$18.42^{\downarrow}$	*29.10*	$27.39^{\downarrow}$	*53.84*
	$\kappa_{\texttt{all}}$-SVR	$\star$	19.03	*43.01*	**24.95**	*47.20*	**20.52**	*30.90*	**30.31**	*58.79*
		$\dagger$	**19.05**	*43.20*	24.92	*46.60*	**20.52**	*31.10*	30.09	*58.89*
LM	def		$18.63^{\downarrow}$	*39.20*	$24.01^{\downarrow}$	*43.20*	$20.40^{\downarrow}$	*29.30*	$27.98^{\downarrow}$	*54.45*
	loo [26]		19.06	*43.40*	**25.48**	*45.40*	**21.24**	*30.90*	$28.46^{\downarrow}$	*53.64*
	$\kappa_{\texttt{all}}$-SVR	$\star$	19.14	*41.20*	$24.73^{\downarrow}$	*44.80*	21.02	*30.10*	29.44	*55.76*
		$\dagger$	**19.16**	*41.40*	$24.76^{\downarrow}$	*45.00*	21.06	*30.80*	**29.56**	*54.85*
LGD	def		18.82	*42.80*	25.47	*47.40*	$19.49^{\downarrow}$	*28.70*	$28.76^{\downarrow}$	*54.14*
	$\kappa_{\texttt{all}}$-SVR	$\star$	**19.00**	*44.20*	**25.64**	*46.60*	**20.02**	*29.10*	**29.62**	*56.46*
		$\dagger$	18.99	*43.80*	25.60	*46.40*	19.94	*29.11*	29.49	*56.26*

Table 3: Comparison of IR models with parameters set to their default values (def) against the predicted parameter values obtained with the leave-one-out strategy [26] (loo) for LM, and the proposed $\kappa_{\textbf{all}}$-SVR approach for all models in terms of MAP and P@10 (in %). For $\kappa_{\textbf{all}}$-SVR, results are presented when terms are coded as in equation 1 (in $\star$) or equation 2 (in $\dagger$). The regressor is trained to optimize the average precision of each query on the source collection. For MAP, best results are shown in bold and $\downarrow$ indicates that the result is worse than the best one according to a Wilcoxon rank sum test with $p < .05$.

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	κ_{all}	**19.03**	*43.01*	**24.95**	*47.20*	20.52	*30.90*	**30.31**	*58.79*
	$\kappa_{\text{poly-2}}$	18.98	*42.80*	24.89	*46.60*	20.50	*30.80*	29.93	*58.48*
	$\kappa_{\text{poly-3}}$	18.97	*42.80*	24.89	*46.80*	20.50	*30.80*	29.94	*58.38*
	$\kappa_{\text{poly-4}}$	18.97	*42.80*	24.90	*46.80*	20.49	*31.00*	29.94	*58.48*
	$\kappa_{\text{gau-10}}$	19.01	*43.00*	24.91	*47.00*	20.54	*31.10*	29.99	*58.18*
	$\kappa_{\text{gau-50}}$	19.00	*43.00*	24.91	*46.80*	20.58	*31.20*	29.99	*58.18*
	$\kappa_{\text{gau-100}}$	19.00	*43.00*	24.92	*46.80*	**20.59**	*31.20*	29.99	*58.18*

Table 4: Comparison of different kernels in SVR in terms of MAP and P@10 on TREC-7,8, WT10G and GOV2 for BM25. The different kernels used are (a) kernel $\kappa_{\textbf{all}}$, (b) polynomial kernels with $\delta = 2$, $\delta = 3$ and $\delta = 4$ ($\kappa_{\textbf{poly-}\delta}$) and (c) Gaussian kernels with $\sigma = 10.0$, $\sigma = 50.0$, $\sigma = 70.0$ and $\sigma = 100.0$ ($\kappa_{\textbf{gau-}\sigma}$).

b (BM25)	$0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.25, 1.5, 1.75, 2.0, 2.25, 2.5, 2.75, 3.0$
μ (LM)	$10, 25, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 10000$
c (LGD)	$0.1, 0.5, 1.0, 1.5, 2.0, 2.5, 3.0, 3.5, 4.0, 4.5, 5.0, 6.0, 7.0, 8.0, 9.0, 10.0, 20.0$

Table 2: Set of values considered to find the optimal parameter values on the source collection of the different IR models.

10 minutes.

The vectors $\mathbf{c}^s_{opt}$ of optimized parameter values on the source collection are constructed by selecting, for each source query and IR model, the value that provides the highest average precision. Table 2 gives the different values considered for each model parameter. As kernel SVR (κ_{all}-SVR), we used the LIBSVM[9] implementation of ϵ-SVR, by fixing ϵ to 0.1, the hyper-parameter C is found by cross-validation on the training set. Each query q here is represented by the vector $\phi(q)$ given in equation (4).

4.3 Experimental results

In this section, we first assess whether the method we propose improves over the one based on default parameter values. We further compare our approach to the two upper bounds introduced in Section 3, prior to illustrate its use on collections written in different languages. Lastly, we investigate the behavior of the method with respect to the number of queries used for training, and its execution time.

4.3.1 Learned values vs default ones

We first evaluate the proposed transfer approach for parameter tuning by comparing the performance of different IR models (a) with their default parameter values, denoted by def, and (b) with their parameter values estimated by the *leave-one-out* (loo) strategy [26] (for the LM model only), against SVR based on the simple kernel κ_{all} (see Section 3), denoted by κ_{all}-SVR, (for all IR models). For the latter, we considered both term representations based on the *tf* distributions (Eq. 1) and those learned with the auto-encoding model (Eq. 2). Table 3 summarizes these results.

We first note that the performance of IR models with their default values over different collections are in line with those reported in studies which also considered query titles [6, 15]. Further, the proposed method yields MAP results significantly higher according to a Wilcoxon rank sum test with $p < .05$ than the ones obtained with the default values on all models and collections except for LGD on TREC-7,8, for which there is no significant difference between the two. Further, considering the LM model, we see that both κ_{all}-SVR and loo [26] behave the same across different collections : the performance of LM with parameters found by loo are significantly

[9]`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`

			FIRE-HN		FIRE-BN	
			MAP	P@10	MAP	P@10
BM25	def		$29.46^{\downarrow}$	*45.40*	$14.09^{\downarrow}$	*24.80*
	κ_{all}-SVR	$\star$	30.29	*47.80*	15.32	*26.80*
		$\dagger$	**30.33**	*48.20*	**15.35**	*27.20*
LM	def		$26.82^{\downarrow}$	*46.40*	15.35	*25.20*
	κ_{all}-SVR	$\star$	**28.39**	*47.40*	**15.76**	*26.40*
		$\dagger$	28.03	*48.20*	15.75	*26.00*
LGD	def		**30.92**	*46.40*	**15.67**	*28.40*
	κ_{all}-SVR	$\star$	30.78	*48.80*	15.66	*28.20*
		$\dagger$	30.67	*49.00*	15.64	*27.80*

(a)

			WT10G		GOV2	
			MAP	P@10	MAP	P@10
BM25	def		$18.42^{\downarrow}$	*29.10*	$27.39^{\downarrow}$	*53.84*
	κ_{all}-SVR	$\star$	**20.75**	*32.50*	**30.17**	*58.08*
		$\dagger$	**20.75**	*32.50*	30.10	*58.48*
LM	def		$20.40^{\downarrow}$	*29.30*	$27.98^{\downarrow}$	*55.45*
	κ_{all}-SVR	$\star$	**21.05**	*30.60*	**29.33**	*55.15*
		$\dagger$	**21.05**	*30.50*	**29.33**	*54.85*
LGD	def		19.49	*28.70*	$28.76^{\downarrow}$	*54.14*
	κ_{all}-SVR	$\star$	**19.85**	*30.40*	**29.05**	*54.55*
		$\dagger$	19.81	*30.30*	28.88	*56.87*

(b)

Table 5: MAP and P@10 measures (in %) of IR models with their default parameters (def) and predicted ones using the κ_{all}-SVR strategy on *non-English* FIRE target collections when using WT10G and GOV2 as source collections (a), and on WT10G and GOV2 target collections when using *non-English* FIRE as source collections (b). For κ_{all}-SVR, results are presented when terms are coded as in equation 1 (in $\star$) or equation 2 (in $\dagger$). The regressor is trained to optimize the average precision of each query on the source collection. For MAP, best results are shown in bold and a result with $\downarrow$ is significantly worse than the best one according to a Wilcoxon rank sum test with $p < .05$.

better on TREC-8 while they are significantly better with parameters found by κ_{all}-SVR on GOV2. As both approaches model the distribution of terms, these results confirm that query terms are distributed likely the same over relevant documents on source and target collections. The difference is however that loo can only be applied to probabilistic IR models using maximum likelihood estimates while the distribution of terms with the κ_{all}-SVR strategy is coded in their vector representations allowing the approach to tune the parameters of more general IR models. Moreover, different term encodings we employed (Eq. 1 and Eq. 2) do not affect the performance of κ_{all}-SVR as the predicted parameters for different IR models with these two settings are slightly the same over different collections. These results suggest that the features space induced by κ_{all}-SVR captures the main information of word distributions by just using the initial word features.

As conjectured in Section 3, the difference between the default values and the approach proposed varies from one collection to the other, and from one model to the other. Indeed, as mentioned before, the difference is not significant for LGD on TREC-7 and TREC-8, which corresponds to two smallest differences between the default values and the upper bounds displayed in Figure 1. There is in this case little room for improvement over the default values.

We also compare the method proposed with the different kernels introduced in Section 3: κ_{all}, $\kappa_{\text{poly}-\delta}$ and $\kappa_{\text{gau-}\sigma}$ with different values for the parameters δ and σ. Table 4 shows those results in terms of MAP and P@10 for BM25[10]. As one can note, the different kernels yield very similar results, without any significant difference between them. This is not really surprising, but still needed experimental confirmation, as the space in which κ_{all} operates is based on (Eq. 1) in which linear kernels are likely to behave well; κ_{all} amounts to a dot product in this space, and the extra dimensions brought by the polynomial and Gaussian kernels are of no use here. We thus focus on κ_{all} in the remainder of this study.

[10]For sake of space, we did not report the same kind of results obtained with LM and LGD models.

4.3.2 Learning from a source dataset of different language than the unlabeled target collection

We investigate here whether κ_{all}-SVR behaves well when the source and target collections considered are written in different languages. To this end, we performed two kind of experiments:

1. We first trained κ_{all}-SVR on WT10G and GOV2 collections and tested it on non-English (Hindi and Bengali) FIRE collections;

2. We then performed the reciprocal experiment by training κ_{all}-SVR on both FIRE collections and testing on WT10G and GOV2 collections.

MAP and P@10 results are respectively reported in Table 5. Except for LGD on FIRE and WT10G collections, one can note that in all other cases, IR models with their predicted free-parameters perform significantly better than with their default values. Furthermore, one can note note that the MAP measures of all IR models for the collections WT10G and GOV2 are similar to the ones reported in Table 3. The representation of queries is based on *tf* moments of query words across a given collection, which allows one to use collections in different languages to learn the regressor, as illustrated in this experiment.

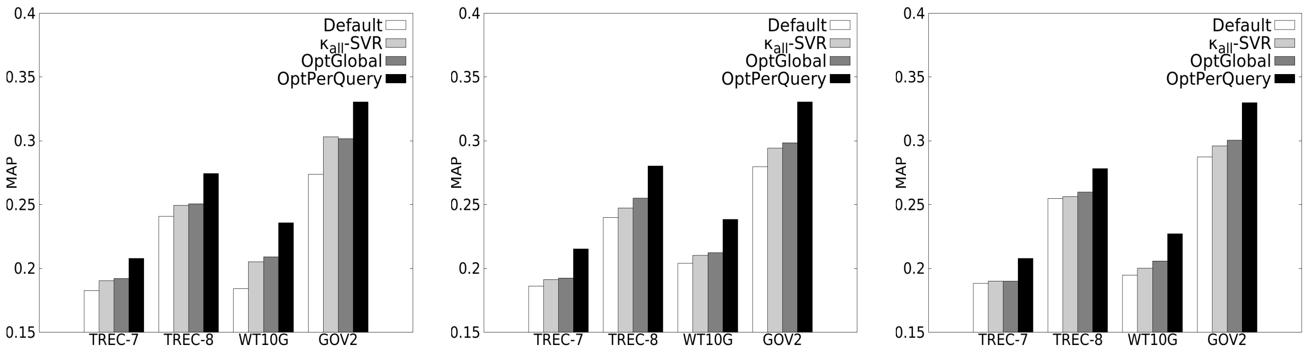
4.3.3 Learned values vs optimized ones

We now compare the results obtained by the method developed in this study and the ones corresponding to two "ideal" cases. The first ideal case corresponds to the scenario in which some target queries (50% in our experiment) are associated with relevance judgements and are used to estimate the parameters of each IR model while the remaining (50%) queries are used for testing. The second ideal case corresponds to the upper bounds already discussed in Section 3 where all the relevance judgements on the target collections are used to optimize the parameters either globally for all queries or query by query.

In the first case, we performed 10 random splits of the target queries, using 50% for training and 50% for testing, and report the average and variance obtained over the 10

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	10split$_{mean}$	18.20$^{\downarrow}$	*41.92*	24.53	*45.52*	19.61$^{\downarrow}$	*30.94*	**30.54**	*60.18*
	10split$_{var}$	0.029	*0.053*	0.121	*0.149*	0.046	*0.048*	0.022	*0.041*
	κ_{all}-SVR	**19.03**	*43.01*	**24.95**	*47.20*	**20.52**	*30.90*	30.31	*58.79*
LM	10split$_{mean}$	18.21$^{\downarrow}$	*42.24*	**25.45**	*45.36*	19.67$^{\downarrow}$	*30.76*	**30.19**	*56.79*
	10split$_{var}$	0.035	*0.115*	0.048	*0.059*	0.052	*0.062*	0.019	*0.096*
	κ_{all}-SVR	**19.14**	*41.20*	24.73	*44.80*	**21.02**	*30.10*	29.44	*55.76*
LGD	10split$_{mean}$	18.03$^{\downarrow}$	*43.04*	**26.08**	*45.84*	19.24$^{\downarrow}$	*29.08*	**30.26**	*59.01*
	10split$_{var}$	0.039	*0.167*	0.048	*0.079*	0.053	*0.037*	0.023	*0.088*
	κ_{all}-SVR	**19.00**	*44.20*	25.64	*46.60*	**20.02**	*29.10*	29.62	*56.46*

Table 6: Comparison of IR models with optimized parameter values using 10 random splits, against the predicted parameter values using κ_{all}-SVR in terms of MAP and P@10 (in %). The regressor is trained to optimize the average precision of each query on the source collection. For MAP, best results are shown in bold and a result with ↓ is significantly worse than the best one according to a Wilcoxon rank sum test with $p < .05$.



(a) BM25 (b) LM (b) LGD

Figure 3: Comparison of IR models with parameters optimized per query basis (■) and optimized within the whole collection (■) against the predicted parameter values using κ_{all}-SVR (■) and default parameter values (□). The results are in terms of MAP on TREC-7,8, WT10G and GOV2 collections for (a) BM25 (b) LM and (c) LGD.

splits. The training simply consists in selecting the parameter value that yields the best MAP value on the training set. We denote this method 10split. Table 6 displays the results obtained in this setting, together with the ones obtained by our approach. As one can note, the variance is always very small, for all models and collections and for both MAP and P@10, showing that the results remain stable even though the query set considered for training changes. Another interesting fact is that our approach, which does not make use of any relevance judgements on the collection queried, is either better or on a par with the 10split method which uses 25 (for TREC-7,8) or 50 (for WT10G and GOV2) queries with their relevance judgements on each collection.

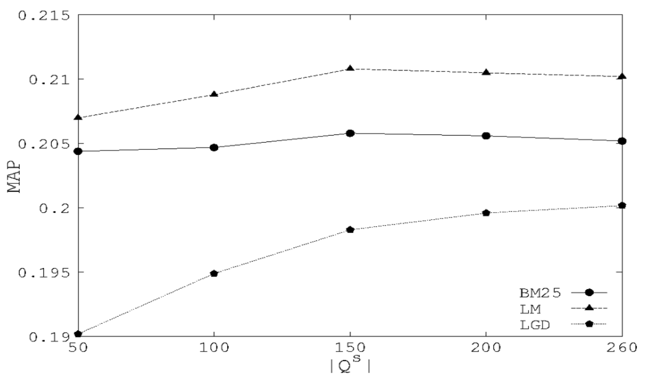
We finally compare our approach to the ideal situation when the relevance judgements of all target queries are used. Let us recall that this ideal situation provides both an upper bound for the methods selecting a global parameter value for the query set, and an upper bound for the methods selecting a parameter value for each query. Figure 3, which parallels Figure 1 of Section 3, shows the comparison of this ideal situation with respect to our approach and the one relying on default parameters. The first point that one can note is that the results obtained by κ_{all}-SVR are very close to the ones of the upper bound of the global optimization methods (second and third bars on each set of histograms). A Wilcoxon test revealed no significant difference between them, whatever the collection, whatever the model. This shows that our approach is at least as good as any method optimizing IR parameters globally over all queries, whether this method uses relevance judgements or not (the def and 10split strategies are such methods, outperformed, as we have seen before, by our approach).

The second point to notice is that our approach is still 2 to 4% below the upper bound for methods providing optimal parameters per query. This suggests that there is still room for improvement over the approach considered in this study.

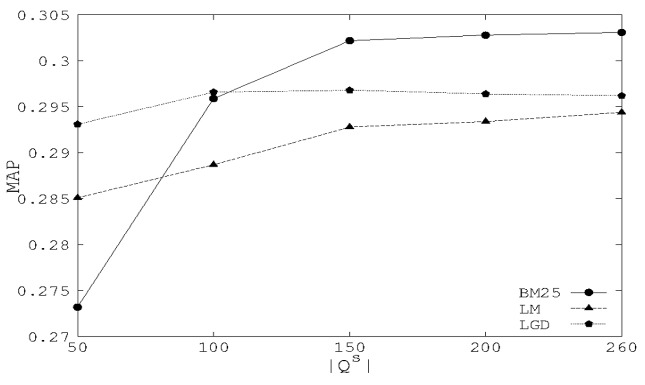
Finally, the predicted parameter values for all IR models obtained with κ_{all}-SVR are in the ranges of those used to learn the regressor (table 2) with the difference that these predicted real-value parameters are not exactly the same with the latter values.

4.3.4 The effect of the number of queries for training

We also analyze the behavior of the IR models for an increasing number of queries in $\mathcal{Q}^s$ for training the κ_{all}-SVR model. Figure 4, illustrates this by showing the evolution of MAP on WT10G and GOV2 with respect to the size of $\mathcal{Q}^s$. To create $\mathcal{Q}^s$, we randomly selected queries from CLEF-3, TREC-3,4,5,6 source collections. As expected all performance curves increase monotonically with respect to the additional training queries, though the increase reaches rapidly a plateau and 150 queries seem sufficient on both collections to learn the regressor at the basis of our method. The findings of these results are (a) that a simple linear model is



(a) WT10G



(b) GOV2

Figure 4: Evolution of MAP for BM25, LM and LGD on (a) WT10G and (b) GOV2 collections with respect to the size of $\mathcal{Q}^s$. The queries constituting $\mathcal{Q}^s$ are randomly sampled from the collections CLEF-3 and TREC-3,4,5,6.

sufficient to learn the association between the mapping of queries in the vector space (Equation 4) and their desired parameter values, and (b) that with a limited amount of source labeled queries, the κ_{all}-SVR model is able to predict accurate parameter values.

4.3.5 Time considerations

As our approach estimates a value for each query, it is important to measure the extra time needed, per query, for this estimation. The word vectors defined by Eq. 1 can be computed offline and stored in the traditional way IDF scores are stored. This means that the query representation defined by Eq. 4 can be computed with (almost) no extra burden. The step that requires additional time with respect to standard IR models is the one corresponding to the application of the regression function on the target query, given by Eq. 6.

As one can note, this step involves the computation of a limited number (at most n, the number of source queries) of dot products between 4-dimensional vectors. One can thus conjecture that the extra time for this step is limited. Here extra time taken for every query is first measured and then the average is calculated for all three models on TREC-7,8, WT10G and GOV2. Table 7 gives this average extra time. As one can note, this extra time is in the range $15 - 30$ milliseconds, and can be easily decreased by parallelizing the different dot products (by e.g. devoting one core to each source query).

	average extra time taken (milisec.)			
	TREC-7	TREC-8	WT10G	GOV2
BM25	31.22	31.03	16.81	16.99
LM	30.89	31.80	17.23	16.94
LGD	31.51	31.41	16.98	17.46

Table 7: Average extra time taken per query (in milliseconds) by κ_{all}-SVR over default parameter value settings.

5. CONCLUSION

We have presented in this paper a new method to predict, on a query by query basis, the values of the parameters of standard IR models on new collections for which no relevance judgments are available. To do so, we have first introduced a new representation of queries as a set of 4-dimensional vectors, that we have then extended through the use of auto-encoders. From these representations, we have obtained collection and language independent vector spaces, corresponding to feature spaces of PDS kernels between queries, in which we have learned regression functions.

Our experiments, conducted on standard collections, have revealed several points:

1. The method we propose significantly outperforms, in terms of MAP, the one using default parameter values (def) on the collections and models considered (the MAP is the measure optimized during training); it either significantly outperforms or is on a par with the method that uses half target queries with their relevance judgements to find the best parameter value over all queries (10split). These two methods (def and 10split) are instances of "global methods" which make use of the same parameter value for all queries of a given collection;

2. The number of source queries with relevance judgements required by our method is in the range 150 for all the collections and models we have considered. Furthermore, the source queries used need not be from a collection written in the same language as the one of the collection queried;

3. The extra time required by our method for each query lies in the range 15-30 milliseconds. This extra time corresponds to the application of the regression function on the query representation;

4. Lastly, the method can be easily applied to any IR standard model with few free parameters.

The proposed method, at its current stage, can predict the value of a single parameter. We wish to investigate the extension of the approach to the case where more than one parameter value needs be estimated, for example BM25 (which originally has three free parameters, b, k_1, and k_3). One obvious way is to train separate regressors for each of the parameters, but this works fine as long as there is no dependency between the parameters. Alternatively, this could be done, for example, by relying on a regression function predicting a vector of values so as to take into account the potential dependencies between the parameters considered.

Acknowledgment

This work was supported in part by the project AAP18 FUI Smart Support Center, and the LabEx PERSYVAL-Lab ANR-11-LABX-0025.

6. REFERENCES

[1] L. Bottou. Stochastic gradient tricks. In G. Montavon, G. B. Orr, and K.-R. Müller, editors, *Neural Networks, Tricks of the Trade, Reloaded*, Lecture Notes in Computer Science (LNCS 7700). Springer, 2012.

[2] P. Cai, W. Gao, A. Zhou, and K. Wong. Query weighting for ranking model adaptation. In *Proceedings of the* 49^{th} *Annual Meeting of the Association for Computational Linguistics (ACL)*, New York, USA, 2011. ACM.

[3] B. Carterette. Robust test collections for retrieval evaluation. In *Proceedings of the* 30^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM, 2007.

[4] D. Chen, Y. Xiong, J. Yan, G. Xue, G. Wang, and Z. Chen. Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3), June 2010.

[5] D. Chen, J. Yan, G. Wang, Y. Xiong, W. Fan, and Z. Chen. Transrank: A novel algorithm for transfer of rank learning. In *IEEE International Conference on Data Mining Workshops (ICDMW)*. IEEE Xplore, 2008.

[6] S. Clinchant and E. Gaussier. The BNB distribution for text modeling. In *Proceedings of the* 30^{th} *European Conference on Advances in Information Retrieval (ECIR)*. Springer, 2008.

[7] S. Clinchant and E. Gaussier. Information-based models for ad hoc ir. In *Proceedings of the* 33^{rd} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, USA, 2010. ACM.

[8] W. Gao, P. Cai, K. Wong, and A. Zhou. Learning to rank only using training data from related domain. In *Proceedings of the* 33^{rd} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, USA, 2010. ACM.

[9] P. Goswami, M. Amini, and E. Gaussier. Transferring knowledge with source selection to learn ir functions on unlabeled collections. In *Proceedings of the* 22^{nd} *ACM International Conference on Information & Knowledge Management (CIKM)*. ACM, 2013.

[10] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[11] F. Jelinek and R. Mercer. Interpolated estimation of markov source parameters from sparse data. In *Proceedings of the Workshop on Pattern Recognition in Practice*. 1980.

[12] Q. V. Le and T. Mikolov. Distributed representations of sentences and documents. In *Proceedings of the 31th International Conference on Machine Learning*, 2014.

[13] Q. V. Le, M. Ranzato, R. Monga, M. Devin, G. Corrado, K. Chen, J. Dean, and A. Y. Ng. Building high-level features using large scale unsupervised learning. In *Proceedings of the 29th International Conference on Machine Learning*, 2012.

[14] Y. Lv and C. Zhai. Adaptive relevance feedback in information retrieval. In *Proceedings of the* 18^{th} *ACM Conference on Information and Knowledge Management (CIKM)*, New York, USA, 2009. ACM.

[15] D. Metzler and O. Kurland. Experimental methods for information retrieval. In *Proceedings of the 35th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '12, New York, NY, USA, 2012. ACM.

[16] M. Mohri, A. Rostamizadeh, and A. Talwalkar. *Foundations of Machine Learning*. The MIT Press, 2012.

[17] I. Ounis, G. Amati, V. Plachouras, B. He, C. Macdonald, and C. Lioma. Terrier: A high performance and scalable information retrieval platform. In *Proceedings of ACM SIGIR'06 Workshop on Open Source Information Retrieval (OSIR 2006)*, New York, USA, 2006. ACM.

[18] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the* 21^{st} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, USA, 1998. ACM.

[19] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the* 17^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, USA, 1994. ACM.

[20] S. E. Robertson and H. Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4), 2009.

[21] I. Soboroff, C. Nicholas, and P. Cahan. Ranking retrieval systems without relevance judgments. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2001.

[22] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the* 29^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, New York, USA, 2006. ACM.

[23] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the* 15^{th} *ACM International Conference on Information and Knowledge Management*, New York, USA, 2006. ACM.

[24] V. Vapnik. *The Nature of Statistical Learning Theory*. Springer Verlag, 2000.

[25] S. Wu and F. Crestani. Methods for ranking information retrieval systems without relevance judgments. In *Proceedings of the 2003 ACM Symposium on Applied Computing*, 2003.

[26] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*,

22(2):179–214, 2004.