Learning Information Retrieval Functions and Parameters on Unlabeled Collections

Parantapa Goswami

UNIVERSITÉ DE GRENOBLE

THÈSE

Pour obtenir le grade de

DOCTEUR DE L'UNIVERSITÉ DE GRENOBLE

Spécialité : **Informatique**

Arrêté ministérial : 7 Aôut, 2006

Présentée par

Parantapa Goswami

Thèse dirigée par **Eric Gaussier**
et codirigée par **Massih-Reza Amini**

préparée au sein **Laboratoire d'Informatique de Grenoble**
et de **Ecole Doctorale Mathématiques, Sciences et Technologies de l'Information, Informatique**

Learning Information Retrieval Functions and Parameters on Unlabeled Collections

Thèse soutenue publiquement le **06 Octobre, 2014**,
devant le jury composé de :

Marie-Christine Rousset
Professeur, LIG, Présidente
Fabio Crestani
Professeur, Università della Svizzera Italiana, Rapporteur
Josianne Mothe
Professeur, IRIT, Rapporteur
ChengXiang Zhai
Professeur, University of Illinois at Urbana-Champaign, Examinateur
Patrick Gallinari
Professeur, LIP6, Examinateur
Eric Gaussier
Professeur, LIG, Directeur de thèse
Massih-Reza Amini
Professeur, LIG, Co-Directeur de thèse

ACKNOWLEDGMENTS

I express my deepest gratitude and indebtedness to my thesis supervisors Prof. Eric Gaussier and Prof. Massih-Reza Amini, who have supported me thoughout my thesis with utmost patience whilst allowing me the room to work in my own way. I attribute the level of this thesis to their motivation, enthusiasm, and immense knowledge. One simply could not wish for any better or friendlier supervisors.

Beside my advisors, I would like to thank the rest of the jury: Prof. Marie-Christine Rousset, Prof. Fabio Crestani, Prof. Josianne Mothe, Prof. ChengXiang Zhai and Prof. Patrick Gallinari, for their encouragement and insightful comments.

I also express my gratitude to Prof. Mandar Mitra of Indian Statistical Institute for his constant encouragement and valuable suggestions.

I am grateful to the faculty members of the AMA team for their friendly advice and invaluably constructive critisicm and for providing a rich and fertile environment to study and explore new ideas.

I am using this opportunity to express my gratitude to every fellow students of the team who are more than just colleagues. Our sometime constructive, sometime light discussions over lunch and coffee were always a fuel for all the studies and works. It is a pleasure to work with them.

I am deeply indebted to my father Prof. Partha Pratim Goswami, my mother Prof. Bhaswati Goswami, my grandfather Late Mr. Madhumoy Goswami and my grandmother Mrs. Dolly Goswami for their incredible patience and sacrifice in all my academic endeavors. And last, but not the least, to Kamalika, my wife, who shares my passions, thank you for your endless support.

ABSTRACT

The present study focuses on (a) predicting parameters of already existing standard IR models and (b) learning new IR functions.

We first explore various statistical methods to estimate the collection parameter of family of information based models (Chapter 2). This parameter determines the behavior of a term in the collection. In earlier studies, it was set to the average number of documents where the term appears, without full justification. We introduce here a fully formalized estimation method which leads to improved versions of these models over the original ones. But the method developed is applicable only to estimate the collection parameter under the information model framework.

To alleviate this we propose a transfer learning approach which can predict values for any parameter for any IR model (Chapter 3). This approach uses relevance judgments on a past collection to learn a regression function which can infer parameter values for each single query on a new unlabeled target collection. The proposed method not only outperforms the standard IR models with their default parameter values, but also yields either better or at par performance with popular parameter tuning methods which use relevance judgments on target collection.

We then investigate the application of transfer learning based techniques to directly transfer relevance information from a source collection to derive a "pseudo-relevance" judgment on an unlabeled target collection (Chapter 4). From this derived pseudo-relevance a ranking function is learned using any standard learning algorithm which can rank documents in the target collection. In various experiments the learned function outperformed standard IR models as well as other state-of-the-art transfer learning based algorithms.

Though a ranking function learned through a learning algorithm is effective still it has a predefined form based on the learning algorithm used. We thus introduce an exhaustive discovery approach to search ranking functions from a space of simple functions (Chapter 5). Through experimentation we found that some of the discovered functions are highly competitive with respect to standard IR models.

RÉSUMÉ

Dans cette thèse, nous nous intéressons (a) à l'estimation des paramètres de modèles standards de Recherche d'Information (RI), et (b) à l'apprentissage de nouvelles fonctions de RI.

Nous explorons d'abord plusieurs méthodes permettant, a priori, destimer le paramètre de collection des modèles dinformation (chapitre 2). Jusquà présent, ce paramètre était fixé au nombre moyen de documents dans lesquels un mot donné apparaissait. Nous présentons ici plusieurs méthodes destimation de ce paramètre et montrons quil est possible daméliorer les performances du système de recherche dinformation lorsque ce paramètre est estimé de faon adéquate.

Pour cela, nous proposons une approche basée sur l'apprentissage de transfert qui peut prédire les valeurs de paramètre de n'importe quel modèle de RI (chapitre 3). Cette approche utilise des jugements de pertinence d'une collection de source existante pour apprendre une fonction de régression permettant de prédire les paramètres optimaux d'un modèle de RI sur une nouvelle collection cible non-étiquetée. Avec ces paramètres prédits, les modèles de RI sont non-seulement plus performants que les młme modèles avec leurs paramètres par défaut mais aussi avec ceux optimisés en utilisant les jugements de pertinence de la collection cible.

Nous étudions ensuite une technique de transfert permettant d'induire des pseudo-jugements de pertinence des couples de documents par rapport à une requłte donnée d'une collection cible (chapitre 4). Ces jugements de pertinence sont obtenus grce à une grille d'information récapitulant les caractéristiques principale d'une collection. Ces pseudo-jugements de pertinence sont ensuite utilisés pour apprendre une fonction d'ordonnancement en utilisant n'importe quel algorithme d'ordonnancement existant. Dans les nombreuses expériences que nous avons menées, cette technique permet de construire une fonction d'ordonnancement plus performante que d'autres proposées dans l'état de l'art.

Dans le dernier chapitre de cette thèse (chapitre 5), nous proposons une technique exhaustive pour rechercher des fonctions de RI dans l'espace des fonctions existantes en utilisant un grammaire permettant de restreindre l'espace de recherche et en respectant les contraintes de la RI. Certaines fonctions obtenues sont plus performantes que les modèles de RI standards.

CONTENTS

1

INTRODUCTION

Archiving information and later finding them from the archives efficiently has been practiced since 3000BC [Singhal, 2001]. Back then, Sumerians designated special areas to store clay tablets with cuneiform inscriptions. They even designed efficient methodologies to find information from these archives. Nowadays with the invention of computers, it has become possible to store large amount of information. Hence, to access and retrieve information efficiently from such large collection, has become a necessity. In 1945, Vannevar Bush published an article titled "As We May Think" which gave birth to the concept of automated retrieval of information from large amount of stored data. The field of information retrieval has advanced very fast over last sixty years. Currently various information retrieval based applications have been developed and commercialized. Many of them, such as the Web search engines, have become an integrated part of day to day human life.

In this introductory chapter we present different basic notions of information retrieval which are relevant to this thesis. Starting with a very brief outline of information retrieval as a whole (Section 1.1), we will discuss basics of two standard IR models used in this thesis (Section 1.2). These models are associated with different free parameters. In Section 1.3 we mention different strategies commonly used by researchers to set these parameter values. We then move to explain the heuristic IR constraints proposed by the community which a good IR scoring function should satisfy (Section 1.4). Then we explore two effective applications of machine leaning in information retrieval (Section 1.5). Finally, we mention the research questions that we will investigate throughout this thesis (Section 1.6 and how this thesis is organized (Section 1.7).

1.1 INFORMATION RETRIEVAL

The term *information retrieval* (IR) is used very broadly. Just searching a phone number from the phone book of a mobile is a form of information retrieval, as well as searching a topic from the Web by some search engine. As an academic field of study, [Manning et al., 2008] defined information retrieval as:

> Information retrieval (IR) is finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers).

Here a *document* is a file containing significant text content with some minimal structures e.g. title, author, subject, etc. A set of similar documents is called a *collection*. Generally all activities of an IR system is performed on a collection. Though the term *information retrieval* is associated with textual retrieval, but the corresponding data can also be images or musics. In this thesis we focus only on textual data.

1.1.1 *Information Retrieval Procedure*

The retrieval process is interpreted in terms of two main components or sub-processes - *Indexing* and *Retrieval*. Figure 1 explains the overview of the procedure.

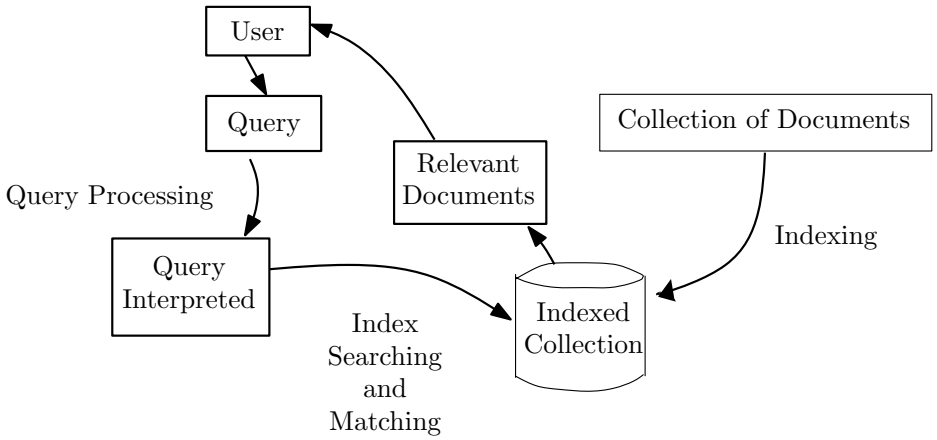


Figure 1: Outline of the IR Procedure.

Indexing This step involves processing each document in a collection and building a data structure so that they can be found efficiently. The documents are first parsed and preprocessed. Preprocessing commonly involves stopword removal and stemming. Finally each term in the document is inserted in the structure of indexed documents. Efficiency of the IR system depends on the data structure used to store indexed documents. Most commonly used data structure is an *Inverted Index*, where for a term w, inverted index stores a list of IDs of all documents containing w. Then the term set is organized in a suitable data structure, e.g. array or hash table or binary search tree. Structure of a typical inverted index is shown in Figure 2.

Retrieval An information need is specified via a query, which is first preprocessed with same preprocessing steps as indexing. This transformed query is a system representation for the original information need. Then the query terms are searched within the indexed terms and corresponding document lists are retrieved from the indexed data structure. The retrieved documents are ranked according to some ranking function (explained in Section 1.2) and the ranked list is returned to the user.

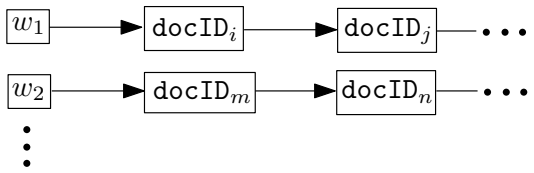


Figure 2: Structure of an Inverted Index.

1.1.2 *Evaluation of an IR System*

To evaluate the performance of an ad hoc IR system, it is generally tested on a collection typically consisting of a set of documents, a test suite of queries, a set of *relevance judgments*, which gives some form of assessment of each query-document pair, that tells the degree of relevance of the document to the query. Document collection and query suite must be of reasonable size. There exists a number of popular evaluation workshops, e.g. `TREC` (`trec.nist.gov`), `CLEF` (`www.clef-campaign.org`) and `FIRE` (`www.isical.ac.in/~clia/`), that facilitates research by providing the collections for comparing models and techniques.

The standard approach to IR system evaluation revolves around the notion of relevant and non-relevant documents. This relevance may be binary or graded. In the first case a document is either relevant or non-relevant to a query, where as for the second case relevancy of a document to a query is specified using a scale of numbers signifying its level of relevancy (e.g. 0 for 'not relevant', 1 for 'relevant', 2 for 'very relevant'). Throughout this thesis we will consider only binary relevance. So here we describe measures to indicate the performance of an IR system based on binary relevance.

1.1.2.1 *Evaluation of Unranked Retrieval Sets*

The two basic parameters for performance measurement of an IR system are *precision* and *recall*. These are initially defined for the simple case where the IR system returns only a set of documents, not a ranked list. These definitions can be extended for IR systems which returns a set of documents along with their ranks.

Precision is the fraction of the documents retrieved that are relevant to the user's information need.

$$Precision = \frac{\text{number of relevant documents retrieved}}{\text{total number of documents retrieved}}$$

Recall is the fraction of the successfully retrieved documents that are relevant to the query.

$$Recall = \frac{\text{number of relevant documents retrieved}}{\text{total number of relevant documents}}$$

1.1.2.2 *Evaluation of Ranked Retrieval Results*

In a ranked retrieval context, simple ratios like precision and recall defined above are not so effective. One has to consider the position of the document in the list. If a relevant document appears way below in the ranked list, then it should be penalized. This is taken into account in the evaluation measure named *Mean Average Precision* (MAP).

For a single query, *average precision* (AP) is defined as the average of the precision value obtained for a ranked set of documents after each relevant document is retrieved. Let the ranked list of documents returned by an IR system for a query q_j be $D = \{d_1, \ldots d_{|D|}\}$. Let rel_{jk} be 1, if $d_k \in D$ is relevant to query q_j, 0 otherwise. Average precision for query q_j is defined as:

$$\texttt{AP}_{q_j} = \sum_{i=1}^{|D|} \texttt{P@i} \times rel_{ji}$$

Here P@i is precision at i. It is defined as:

$$\texttt{P@i} = \frac{\text{number of relevant documents in the rank list up to } i^{th} \text{ position}}{i}$$

When no relevant document is retrieved for a query, the average precision value in the above equation is taken to be zero.

Let the complete query set be $\mathcal{Q}$. MAP of $\mathcal{Q}$ is the average of $\texttt{AP}_{q_j}$ for all $q_j \in \mathcal{Q}$. Thus:

$$\texttt{MAP} = \frac{1}{|\mathcal{Q}|} \sum_{j=1}^{|\mathcal{Q}|} \texttt{AP}_{q_j}$$

In Web search, when results are returned, most of the users do not go beyond the first page, and thus it is very important for the system to present maximum possible relevant documents within this page. If each page contains n documents, precision at n (by the expression P@n defined above) is an important evaluation measure. Typically a result page for a Web search engine contains ten results, so n = 10 is a natural choice. This gives another important evaluation measure, *precision at 10 documents* (P@10).

There exist other evaluation measures in the literature for graded relevancy, such as *normalized discounted cumulative gain* or NDCG. But as we consider only binary relevance in this thesis, MAP and P@10 are sufficient enough for our purposes.

1.2 STANDARD IR MODELS

To retrieve relevant documents from a pool of documents and later to rank them, one needs to associate some scores to the documents with respect to the query in hand. This is what an IR model does. Each IR model is associated with a *scoring function* (also called *ranking function*) which formulates a scoring strategy that uses various information about

the document in light of the query. Two main quantities are *term frequency* and *document frequency*. Term frequency of a term w in document d is defined as the number of times that w occurs in d. Whereas, document frequency of w is the number of documents in the collection that w occurs in. Scoring functions are sometimes referred to as *retrieval status value* or RSV, as it provides the value of the retrieval status of a document with respect to a query.

Many classical IR models are proposed in the literature. Most efficient and popular ones include vector space model, tf-idf model, probabilistic models, language models and more recently DFR models, information based models. Scores provided by these standard models are often used as features in learning to rank approach (Section 1.5.1).

Once a new model is developed, it is common practice to compare it with already existing models. In this thesis, when we will develop any approach, we will compare them with three standard models, namely Okapi `BM25`, language models and information based models. Scores calculated using these three models will also be used as features to learn a ranking function in Chapter 4. First two models are discussed briefly in next two sections whereas the information model is discussed in Chapter 2.

1.2.1 *Okapi BM25*

The *Okapi BM25* ranking model [Robertson and Walker, 1994, Zaragoza et al., 2004] is based on probabilistic retrieval framework and has become one of the most popular benchmark models in IR research. The main idea of this model is based on the simple observation that a term appearing in a small distinct number of documents with high frequency is far more important than the terms appearing in a large number of documents with very low frequency. Thus a document can be called important with respect to a term, if the term occurs frequently in that document, but the term itself occurs rarely throughout the collection. These terms are conventionally called the *elite* terms and clearly, such a term has a high term frequency in the document in question but low document frequency.

For a query q and a document d, $s(q,d)$ gives a score to d to indicate how important it is to the query q. Both q and d are actually a string of individual terms. `BM25` assumes a bag-of-word approach which ignores any relation between the terms in a document or a query, and hence each query term is processed independently as a single term query. Thus assuming $s(w,d)$ assigns a score to d with respect to w, one has:

$$s(q,d) = \sum_{w \in q \cap d} s(w,d)$$

To score a document with respect to a term, [Robertson and Walker, 1994] considered three components:

1. From the observation above, it can be said that the importance of term w in document d is proportional to the term frequency of w in d (denoted by t_w^d). But different documents in the collection have different lengths and thus raw occurrences are normalized and following is the term frequency factor considered:

$$\frac{(k_1+1)\times t_w^d}{k_1\left((1-b)+b\frac{l_d}{l_{avg}}\right)+t_w^d}$$

   where l_d is the length of document d, l_{avg} is the average document length in the collection and k_1 and b are two free parameters.

2. Again from the observation above, it can be said that the importance of term w is inversely proportional to the document frequency of w (denoted by $\mathcal{N}_w$). Taking into account this fact and assuming a 2-Poisson mixture model for elite and non-elite terms, [Robertson and Walker, 1994] derived the *inverse* document frequency factor:

$$ln\left(\frac{\mathcal{N}-\mathcal{N}_w+0.5}{\mathcal{N}_w+0.5}\right)$$

   where $\mathcal{N}$ is the number of documents in the collection.

3. Finally, the score is weighted by the number of occurrences of the term w inside the query q itself (denoted by t_w^q). But here also the raw occurrence is normalized as following:

$$\frac{(k_3+1)\times t_w^q}{k_3+t_w^q}$$

   where k_3 is a free parameter.

Combining all the components the score of document d for query q calculated by `BM25` thus becomes:

$$s(q,d)=\sum_{w\in q\cap d}\frac{(k_3+1)\times t_w^q}{k_3+t_w^q}\,\frac{(k_1+1)\times t_w^d}{k_1\left((1-b)+b\frac{l_d}{l_{avg}}\right)+t_w^d}\,ln\left(\frac{\mathcal{N}-\mathcal{N}_w+0.5}{\mathcal{N}_w+0.5}\right) \tag{1.2.1}$$

Now, to successfully deploy `BM25` one has to properly fix its free parameter values. We will discuss the details of that in Section 1.3.

1.2.2 *Language Models*

Language modeling approach proposed by Ponte and Croft [Ponte and Croft, 1998] assumes that a query q is close to a document d if and only if q can be *generated* using the language model associated to d. Formally, assuming a query $q=\{w_1^q,\ldots,w_K^q\}$ and a

document $d = \{w_1^d, \ldots, w_n^d\}$, the goal is to estimate the probability that language model of d can generate the observed query q, $P(d|q)$. Applying Bayes' rule one has:

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)}$$

Here $P(q|d)$ is the conditional probability of the query q given the document d, $P(d)$ is the *a priori* probability that d is relevant and $P(q)$ is the *a priori* probability that q can be generated. For any query, the quantity $P(q)$ is equal for all the documents in the collection, thus does not affect the retrieval process. Moreover, one should not give any *a priori* privilege to a particular document in terms of relevance, hence the distribution $P(d)$ is assumed to be uniform and no longer has any effect on the retrieval process. Thus the expression for language model reduced to $P(q|d)$. Commonly, a *unigram* language model is assumed which essentially means that given a document, the terms in the query are independent from each other, thus giving:

$$P(q|d) = \prod_{i=1}^{K} P(w_i^q|d)$$

Lastly, assigning zero probabilities to documents where the term under consideration does not occur poses a risk of vanishing the entire probability. Instead, for those documents, the probability $P(w|d)$ is assigned with the probability of presence of the term in the collection, that is:

$$\forall w \in q, w \notin d, P(w|d) = c_d P(w|\mathcal{C})$$

The normalizing factor $c_d = \frac{1-\sum_{w\in d} P(w|d)}{1-\sum_{w\in d} P(w|\mathcal{C})}$ ensures that $\sum_{w\in\mathcal{C}} P(w|d) = 1$. With all necessary components, now the query can be decomposed into the terms it contains:

$$ln\ P(q|d) = \sum_{i=1}^{K} ln\ P(w_i^q|d) = \sum_{i:w_i\in d} ln\ P(w_i^q|d) + \sum_{i:w_i\notin d} ln\left(c_d P(w_i^q|\mathcal{C})\right)$$

The expression $\sum_{i=1}^{K} ln\ P(w_i^q|\mathcal{C})$ can also be decomposed as following:

$$\sum_{i=1}^{K} ln\ P(w_i^q|\mathcal{C}) = \sum_{i:w_i\in d} ln\ P(w_i^q|\mathcal{C}) + \sum_{i:w_i\notin d} ln\ P(w_i^q|\mathcal{C})$$

After substituting and rearranging one has:

$$ln\ P(q|d) = \sum_{i:w_i\in d} ln\left(\frac{P(w^q|d)}{c_d P(w^q|\mathcal{C})}\right) + Klnc_d + \sum_{i=1}^{K} ln\ P(w_i^q|\mathcal{C})$$

The last term of the above equation is independent of the documents and can be ignored from the retrieval process. So the final form becomes:

$$ln\ P(q|d) = \sum_{i:w_i\in d} ln\left(\frac{P(w^q|d)}{c_d P(w^q|\mathcal{C})}\right) + Klnc_d \tag{1.2.2}$$

Now for a given query all the documents in the collection are ranked based on the score $ln\ P(q|d)$. This score can be calculated by estimating the probability $P(w^q|d)$. Commonly, $P(w^q|\mathcal{C})$ estimated by the ratio of total number of occurrences of w in the collection $\mathcal{C}$ and the size of the dictionary. There exist various methods to estimate $P(w^q|d)$, but here we discuss the method using *Dirichlet prior*.

Maximum *a posteriori* method using Dirichlet prior assumes that a language model is a multinomial distribution. In this method, the estimated probability is deduced to be [Zhai and Lafferty, 2001]:

$$P_{dir}(w|d) = \frac{t_w^d + \mu P(w^q|\mathcal{C})}{l_d + \mu} \tag{1.2.3}$$

Here t_w^d is the term frequency of w in d, l_d is the length of d and μ is a free parameter called as *smoothing parameter*.

Another popular smoothing method is *Jelinek-Mercer*. But in this thesis we will use the Dirichlet prior version of language model as it yields better results than the one based on Jelinek-Mercer smoothing [Ponte and Croft, 1998, Zhai and Lafferty, 2004].

1.3 MODEL PARAMETERS AND HOW TO DEAL WITH THEM

Like many mathematical models, IR models also have some free parameters. Proper adjustment of the parameters allows the models to get adapted with the collections they are applied on. The standard models we discussed in the previous sections also have free parameters: b, k_1 and k_3 for `BM25`, smoothing parameter μ for language model with Dirichlet prior. The values of the free parameters used in the models may be fixed empirically or by educated guesses from previous experiences or even randomly. Here we will discuss some strategies commonly used to set the parameter values.

1.3.1 *Default Values*

The most common setting is the one where the parameters are assigned to some "default" values. Mostly these values are educated guesses and observed to perform well in various experiments and thus it is assumed that they will perform reasonably well on new collections as well.

As for example, in `BM25` the parameter k_1 (Eq. 1.2.1) controls the document term frequency scaling, $k_1 = 0$ signifies a binary model without any term frequency whereas a large value means no normalization and raw term frequency. Keeping in mind the purpose of variables (as well as by experiments), researches have determined that `BM25` should work well if $k_1 \in [1.2, 2]$ [Manning et al., 2008]. Similarly the suggested range for query term frequency scaling parameter k_3 is also $[1.2, 2]$[Manning et al., 2008]. Lastly the parameter b controls the scaling by document length and $0 \leq b \leq 1$. $b = 0$ means no length

normalization and $b = 1$ means complete term weight normalization by the document length. Again, b is set to 0.75 by researchers which keeps partial balance (yet not full) towards normalization by the document length [Manning et al., 2008].

The popular IR experimental simulation platform Terrier [Ounis et al., 2006] assigns: $b = 0.75$, $k_1 = 1.2$, $k_3 = 8$ for `BM25` and $\mu = 2500$ for language models with Dirichlet prior. Other IR platforms also assign similar default values to these parameters.

1.3.2 *Parameter Tuning*

Default parameter values generally perform reasonably well on any collection, but still they cannot yield the best performance from the associated models. Setting these parameters to a proper value for a particular collection has attracted attention from many researchers. Many techniques have been developed to automatically set the parameter values. This procedure is commonly referred to as *parameter tuning*. Parameters can be tuned in a supervised or an unsupervised manner.

1.3.2.1 *Supervised Tuning*

Supervised tuning involves some relevance information on the collection. The values for the parameters are searched to maximize the performance of the model (in terms of any evaluation measure, e.g. `MAP`). This search can be either manual or automated e.g. linear or grid search. Once the best parameter value is obtained, that value is used for the model. But it will be a wrongful evaluation if the performance of the model is reported on a test collection which is also used to tune parameters. One possible solution is to have one or more validation collections on which parameters are tuned, and then the models are tested on a separate test collection. This method is able to provide an unbiased estimate of performance.

Let us assume that a collection $\mathcal{C}$ has a set of queries $\mathcal{Q}$, and a model `M` is to be evaluated on $\mathcal{C}$. Suppose `M` has a free parameter $\mathfrak{p}$ which needs to be tuned. Following are two commonly used supervised parameter tuning techniques, among many, which use a single collection both for tuning and performance evaluation, but also avoid the issue of wrongful evaluation. Note that $\mathfrak{p}$ can also be a set of parameters (e.g. for `BM25`, $\mathfrak{p} = \{b, k_1, k_3\}$), and in those cases following methods can be applied separately on each of the parameters of the set. But for simplicity reason, we here assume a single parameter.

k Fold Crossvalidation: The query set is sequentially partitioned into k disjoint subsets $\mathcal{Q}_1, \ldots, \mathcal{Q}_k$ each containing $\frac{1}{k}$ portion of all the queries. Sequential partition means that $\mathcal{Q}_1$ contains the first $\frac{1}{k}$ portion of the queries from the original set $\mathcal{Q}$, $\mathcal{Q}_2$ contains the second $\frac{1}{k}$ portion and so on. Of these k subsets, a single subset is retained for testing the model, and the remaining $(k-1)$ subsets are used for tuning the parameter $\mathfrak{p}$. For example, in the first step $\mathfrak{p}$ is tuned using the queries of

$\mathcal{Q}_2 \cup \mathcal{Q}_3 \cup \ldots \cup \mathcal{Q}_k$ and the model M is evaluated on $\mathcal{Q}_1$ using the tuned value of $\mathfrak{p}$. The tuning means searching a value for $\mathfrak{p}$ from a range of possible values which will maximize the performance of M on the tuning query set in terms of some evaluation measure. Similarly, in the second iteration $\mathcal{Q}_2$ is retained for testing and rest is used for tuning $\mathfrak{p}$ and so on. This cross-validation process is repeated k times, with each of the k subsets used exactly once for testing. Each time performance of M is noted. After k folds, the average of the performance of M on all the subsets is reported.

Random Split: First the query set $\mathcal{Q}$ is split into two disjoint subsets $\mathcal{Q}_1$ and $\mathcal{Q}_2$, each containing 50% of the queries. This split is done randomly, meaning those 50% queries are selected randomly. $\mathcal{Q}_1$ is associated with the relevance judgments. The parameter $\mathfrak{p}$ is tuned on $\mathcal{Q}_1$ following a similar method as k fold crossvalidation. For that $\mathfrak{p}$ is assigned a value from a range of possible values and performance of M is evaluated using $\mathcal{Q}_1$ on $\mathcal{C}$. The range of values are searched for which performance of M is maximized on $\mathcal{Q}_1$. Once the "best" value for $\mathfrak{p}$ is identified, M is tested with that value of $\mathfrak{p}$ on the other set $\mathcal{Q}_2$. Note that relevance judgment on $\mathcal{Q}_2$ is not used for tuning and only used for final evaluation. Often this procedure is repeated multiple times and the average of the performances are reported.

Chapter 3 will discuss more about supervised approaches of parameter tuning. For different experiments that we will conduct throughout this thesis, in Chapters 2 and 5 we will use k fold crossvalidation assuming $k = 5$, in Chapter 3 we will use random split.

1.3.2.2 *Unsupervised Tuning*

Unlike supervised approaches, unsupervised parameter tuning methods try to fix parameter values using only statistical information gathered from the collection and does not use any relevance information. Unsupervised techniques mostly deploy statistical methods like estimation. Being unsupervised, these techniques do not need any prior training and commonly do not involve any learning at all. But these techniques are mostly developed under a particular framework and tailored to estimate parameters in that framework only.

Notable work in this direction is "leave-one-out" likelihood estimation of smoothing parameter μ of language model with Dirichlet prior [Zhai and Lafferty, 2004], expectation maximization estimation of relevance feedback parameter [Tao and Zhai, 2006]. In Chapter 2 we will develop an unsupervised parameter estimation method for the collection parameter of information based models (described in Section 2.2).

1.4 HEURISTIC RETRIEVAL CONSTRAINTS

IR models we presented here along with other probabilistic models are generally developed under different frameworks. But interestingly they share some common hypothetical

properties observed by the researchers. This enables these models to be expressed in a single framework. These empirical hypotheses were first formalized by Fang et. al. [Fang et al., 2004]. The conditions derived from the hypotheses lays a guideline of how a *good* IR function should behave. When IR functions are analyzed and described through these conditions, it is called *axiomatic approach to IR*.

[Clinchant and Gaussier, 2011] studied these conditions and expressed them in analytical forms in terms of first and second order derivatives of the IR scoring function. These analytical forms enable one to easily determine if a IR scoring function is satisfying the hypothetical conditions. [Amini and Gaussier, 2013] summarizes these constraints.

An IR scoring function, also referred to as retrieval status value (RSV), scores a document d with respect to a query q. Mostly each term in the query is assumed to be independent of each other and the score of d for q is the sum of the scores d gets for all the terms $w \in q$. As we have seen for standard IR models, a scoring function is a function of several variables namely term frequency (t_w^d), a collection based statistics of w (generally document frequency $\mathcal{N}_w$), length of the document (l_d) and a set of parameters (θ). Thus a standard RSV can be written as:

$$\texttt{RSV}(q,d) = \sum_{w \in q} a(t_w^q) h(t_w^d, \mathcal{N}_w, l_d, \theta)$$

Here a is a function of occurrences of w in the query and is usually set to the identity function. h scores document d with respect to an individual query term w based on t_w^d, $\mathcal{N}_w$, l_d, and θ. The form of h depends on the IR model considered.

The conditions are categorized into two categories. The first group of constraints defines the general form of the function h, hence are called *form conditions*. Whereas constraints from the second group aim to adjust the function h satisfying the form conditions by regulating the interaction between term frequency and document length. Hence this group is called *adjustment conditions*.

Four form conditions are as follows:

TF Effect aims to capture the fact that the documents with higher occurrences of the query term is more important and should get a higher score. This gives that the score h of a document d should increase with the number of occurrences of term w. Analytically:

$$\forall(\mathcal{N}_w, l_d, \theta), \quad \frac{\partial h}{\partial t_w^d} > 0 \qquad \text{(TF Effect)}$$

Concavity Effect suggests that the increase of score with t_w^d must be restricted for high t_w^d values. As for example, the increase from 100 to 101 is much less significant than the increase from 1 to 2, as the increase in first case is only 1% where as in the second

case it is 100%. Mathematically this can be ensured by imposing a concave form on the function h. Analytically:

$$\forall(\mathcal{N}_w, l_d, \theta), \quad \frac{\partial^2 h}{\partial (t_w^d)^2} < 0 \qquad \text{(Concavity Effect)}$$

IDF Effect ensures that scores for very frequent terms in the collection are weighed down. These terms occur in many documents and are most commonly used terms, thus have little importance. Indeed these are the *non-elite* terms as described in BM25 framework (Section 1.2.1). Here the term *IDF* stands for *inverse document frequency*. Hence the score h of document d must decrease when the associated term w has a high document frequency. Analytically:

$$\forall(t_w^d, l_d, \theta), \quad \frac{\partial h}{\partial \mathcal{N}_w} < 0 \qquad \text{(IDF Effect)}$$

One can note that, if document frequency is replaced by any other collection based statistics of the term w (e.g. collection frequency), the same condition holds.

Document Length Effect takes into account the difference of length of the documents in the collection. Suppose two documents of different length have the same number of occurrences of the term w. Then the scoring function should penalize the longer document over the shorter one as the longer one tends to cover additional irrelevant topics. So the score h of d must decrease as length of d increases. Analytically:

$$\forall(t_w^d, \mathcal{N}_w, \theta), \quad \frac{\partial h}{\partial l_d} < 0 \qquad \text{(Document Length Effect)}$$

In this thesis we will use above four form constraints and when developing scoring functions (Chapter 5) we will ensure that these constraints are always satisfied. We will also try to verify the effectiveness of these conditions for defining an IR scoring function by empirically validating them (Section 5.6.1).

Though we will not explicitly use the adjustment constraints in this thesis, we briefly mention here two of them defined by [Fang et al., 2004]:

1. Document Length Effect penalizes longer documents, whereas the first adjustment constraint avoids over-penalizing long documents. It is defined as:

   > Let q be a query. $\forall k > 1$, if d_1 and d_2 are two documents such that $l_{d_1} = k \times l_{d_2}$ and for all words w, $t_w^{d_1} > t_w^{d_2}$, then $\texttt{RSV}(q, d_1) \geq \texttt{RSV}(q, d_2)$.

   This constraint ensures that redundancy within a document is not penalized. Suppose a new document d_2 is generated by concatenating document d_2 k times with itself, then the score of the new document d_2 should not be lower than the original d_1.

2. Suppose w is a query word, d_1 and d_2 are two documents. Let d_1 be longer than d_2 and the excess length of d_1 is caused by the query word w. Under this scenario, document length effect will punish d_1 for its higher length. The goal is to promote the high occurrences of the query word w, and thus in this case one has to ensure that d_1 is not penalized just because it is longer. So a longer document must not be penalized over a shorter document if the excess length is due to the occurrences of the query term. This condition is defined as:

   Let $q = w$ be a single word query, if $t_w^{d_1} > t_w^{d_2}$ and $l_{d_1} = l_{d_2} + (t_w^{d_1} - t_w^{d_2})$, then $\texttt{RSV}(q, d_1) > \texttt{RSV}(q, d_2)$.

1.5 MACHINE LEARNING AND IR

Machine learning pioneers the concept of automated *learning* from data with either small or none manual intervention. Technically a learning algorithm is supplied with proper representation of data (called *feature vector*) and sometimes with some training examples to learn from or sometimes without any such examples. The first case where training data, often in form of labels associated with the original data, are available is referred to as *supervised learning*, whereas the later case is called *unsupervised learning* as no annotation or label is available with the data for learning and the algorithm must learn on the run.

In IR the main goal is to distinguish, for a given query, the relevant documents from non-relevant documents. Thus key IR processes can be seen as a classification task. Instead of developing term and document frequency based scoring function by hand, there is a possibility of defining retrieval as a classification task and deploying machine learning tools to learn a scoring function. To do so, scores of different standard IR models can be used as features in the learning problem and the classifier can be supplied with samples of relevant and non-relevant documents for different queries on which the classifier can be trained. Then this classifier can predict scores for different document for any unseen query. Clearly this is a supervised approach and we briefly describe this approach in the next section. Another popular approach is to apply supervised heuristic search techniques like genetic algorithm and genetic programming to find an effective scoring function. We will discuss this approach in Section 5.2.

Apart from the ranking itself, machine learning has been applied to other IR components, such as collection fusion, user modeling, query formulation etc.

1.5.1 *Learning to Rank*

A traditional IR scoring function does not need any *a priori* training like learning functions do. Once they are deployed (as for example `BM25` in Eq. 1.2.1 or language model in Eq. 1.2.3), they can start scoring the documents in the collection without any prior labeled

data (relevance judgments). Though sometimes some labeled data is required to tune the free parameters of these models (Section 1.3).

In contrast to these conventional approaches, *learning to rank* makes use of machine learning tools to automatically learn a scoring function from a family of functions[1]. Of course, to build such a scoring function some labeled information is needed first to train the system. This approach is motivated by the increasing use of web-based search engines where it is possible to easily accumulate users' click through data which serve as relevance information from which the system can be trained. Learning to rank approaches can also be adopted for the ad hoc IR task by relying on various existing test collections (e.g. TREC).

Like all other supervised machine learning algorithms, learning to rank also has two distinct phases, namely training and testing.

1.5.1.1 *Training*

Training data associates query, documents and some labels in the form of relevancy of the documents to the queries. Relevancy (hence the labels) can be binary or graded. Note that graded relevance can always be transformed into binary, but the opposite is not true. Throughout this thesis we will only consider binary relevance. So the set of labels is $\mathcal{Y} = \{0, 1\}$, where 1 means relevant and 0 means non-relevant. Suppose $\mathcal{Q}$ is a query set and $\mathcal{D}$ is a document set inside a collection $\mathcal{C}$. For every pair (q_i, d_j), where $q_i \in \mathcal{Q},\ d_j \in \mathcal{D}$, there is a associated relevancy label $y_{ij} \in \mathcal{Y}$. Thus the training set consists of:

$$\mathcal{T} = \{\big((q_i, d_j),\ y_{ij}\big)\ ;\ q_i \in \mathcal{Q},\ d_j \in \mathcal{D},\ y_{ij} \in \mathcal{Y}\}$$

A feature vector is defined over the pair (q_i, d_j), denoted by $\mathbf{f}(q_i, d_j)$. In literature various quantities are used as features. Three categories of features are considered when building $\mathbf{f}(q_i, d_j)$: query independent features (e.g. document frequency in collection), query related features (e.g. length of the query) and features related to both query and document (e.g. term frequency or scores of other standard IR models).

Once a (q_i, d_j) pair gets represented by a proper feature vector, the training set takes the form based on the type of learning to rank algorithm to be used. [Liu, 2009] divided different learning to rank approaches into three broad categories:

Pointwise approaches assume that every query-document pair has a ordinal score and thus the ranking problem is formulated as regression problem which can provide an absolute score to the query-document pair. Thus the training data for a query document pair is of the form:

$$\mathcal{T} = \{\big(\mathbf{f}(q_i, d_j),\ y_{ij}\big)\ ;\ q_i \in \mathcal{Q},\ d_j \in \mathcal{D},\ y_{ij} \in \mathbb{R}\}$$

[1] The learned function is normally a classifier capable of predicting classification probabilities which in turn is used for ranking, thus is the name "learning to rank".

As ranking is now a regression problem, the relevance is a real score. Normally higher the score, more the document d_j is relevant to the query q_i.

Pairwise approaches take document pairs and a pairwise classifier is trained by minimizing the misordering of these document pairs and the trained classifier is used to assign positive and negative labels of new document pairs to determine their relative ranking. Goal of the trained classifier is to predict positive labels for those document pairs whose first document is more relevant than the second one, and to predict negative labels for other pairs. Suppose there is a pair of documents $d, d' \in \mathcal{D}$ and with respect to query $q_i \in \mathcal{Q}$, the document d is more relevant than the document d'. Commonly the assciation between the pair (d, d') and the query q_i is represented as the difference between the feature vectors $\mathbf{f}(q_i, d)$ and $\mathbf{f}(q_i, d')$. The feature vector $\mathbf{f}(q_i, d) - \mathbf{f}(q_i, d')$ is thus assigned a positive label and the feature vector $\mathbf{f}(q_i, d') - \mathbf{f}(q_i, d)$ is assigned a negative label. Training data for a pairwise algorithm takes the form:

$$\mathcal{T} = \left\{ \begin{matrix} ((\mathbf{f}(q_i, d) - \mathbf{f}(q_i, d'))\,, +1)\,; \\ ((\mathbf{f}(q_i, d') - \mathbf{f}(q_i, d))\,, -1)\,; \end{matrix} \quad q_i \in \mathcal{Q},\ (d, d') \in \mathcal{D} \right\}$$

[Joachims, 2002] used support vector machine (SVM) to classify the order of pairs of documents and utilize the classifier in the ranking task. This version of SVM using pairwise approach is popularly known as *RankingSVM*.

Listwise approaches consider the entire ranking list of each query both for training and prediction. One sub-category of these algorithms considers rank list of all documents associated with a query along with their degrees of relevance or scores and tries to directly optimize widely used IR evaluation measures. An alternative approach considers different permutations of the documents associated with a single query. Here the loss function measures the discrepency between a given permutation with the ground truth permutation. These algorithms can distinguish between documents in the ranking list of different queries and their positions in the list.

With the training set in hand, now the goal is to learn a scoring function $h(q, d)$ which can assign a score to a feature vector $\mathbf{f}(q, d)$ which is also considered as the score given to the pair (q, d) itself. For this step any standard regressor or classifier can be used.

1.5.1.2 *Testing*

Let q_{new} is an unseen query, and $q_{new} \notin \mathcal{Q}$. For q_{new} the documents of set $\mathcal{D}$ have to be ranked. A test set is prepared from the feature vectors in accordance with the algorithm used for learning. In case of both pointwise and pairwise algorithms, for all pairs of the form (q_{new}, d_j), where $d_j \in \mathcal{D}$ the test set is of the form:

$$\mathcal{S} = \{\mathbf{f}(q_{new}, d_j);\ d_j \in \mathcal{D}\}$$

Every pair in the test set is given a score using the learned scoring function h. For pointwise algorithms h is a regressor function whereas for pairwise algorithms it is a classifier. The pairs are sorted based on the scores and a ranking list is prepared from the associated documents obtained from the sorted pairs.

Detailed literature surveys about these three approaches are discussed in Section 4.2. In Chapter 4 we will develope a pairwise learning to rank algorithm which also uses the concept of *transfer learning*, described in the next section.

1.5.2 *Transfer Learning*

Most machine learning algorithms are trained and used on data with the same underlying distribution and representation, whereas real world applications often involve data for which the distribution and representation change over time. Technically those data is referred to as a different *domain*. As for example, an object recognition algorithm trained on some commercial photos of objects, suppose from Amazon, seldom works well on images of the same objects but taken with low quality cameras, suppose by customers [Saenko et al., 2010]. *Transfer learning* is deployed to alleviate this type of situations. For a given learning task and two source and target domains, algorithms under this paradigm aims to improve the performance of the predictive function on the target by using labeled instances from source domain. The predictive function can either be learned directly on the source and then adapted to be used in the target domain, or the source labeled information can be represented in the target domain and the predictive function is learned in the target domain. Transfer learning is defined by [Pan and Yang, 2010] as:

> Given a source domain $\mathcal{D}^s$ and learning task $\mathfrak{T}^s$, a target domain $\mathcal{D}^t$ and a learning task $\mathfrak{T}^t$, transfer learning aims to help improve the learning of the target predictive function $h^t(.)$ in $\mathcal{D}^t$ using the knowledge in $\mathcal{D}^s$ and $\mathfrak{T}^s$, where $\mathcal{D}^s \neq \mathcal{D}^t$, or $\mathfrak{T}^s \neq \mathfrak{T}^t$.

Based on this definition, [Pan and Yang, 2010] categorized transfer learning approaches into three categories:

Inductive Transfer Learning. Under this setting, source and target tasks are different ($\mathfrak{T}^s \neq \mathfrak{T}^t$), irrespective of the status of the source and target domains. Here some labeled data in the target domain is necessary to *induce* the predictive function $h(.)$ to be applied in the target domain.

Transductive Transfer Learning. The opposite of the previous case is considered in this setting where the source and target tasks are identical but the domains are different ($\mathcal{D}^s \neq \mathcal{D}^t$) [Arnold et al., 2007]. Here no labeled data in the target domain is available. Source and target domains can differ in two ways: the feature space between them are different and the knowledge transfer is instance based

[Sugiyama et al., 2008], or the feature space is same but the data distributions are different and the knowledge transfer is feature based [Blitzer et al., 2008]. The later case is popularly known as *domain adaptation* in literature.

Unsupervised Transfer Learning. Lastly, this setting focuses on unsupervised learning tasks in the target domain, as for example clustering. Here source and target tasks are different but related. Like all unsupervised learning problems, in this case also neither source nor target domain contains any labeled data.

1.5.2.1 *Transfer Learning applied to IR*

In the field of IR, a domain is essentially a collection and the knowledge that needs to be transferred is the relevance judgments. The learning task for both source and target domains is ranking, and one has $\mathfrak{T}^s = \mathfrak{T}^t$. In the normal learning to rank setup described above in Section 1.5.1, one has a collection $\mathcal{C}$, some training queries $\mathcal{Q}$ with relevance judgments and a document set $\mathcal{D}$ on which a scoring function $h(.)$ is learned. Then $h(.)$ is applied to a new query $q_{new} \notin \mathcal{Q}$ to rank documents from the same document set $\mathcal{D}$ in the same collection $\mathcal{C}$. Thus one must have some annotated data (in the form of relevance judgments) on $\mathcal{C}$ to deploy a learning to rank algorithm. If a new collection is obtained or even if the old collection is modified, for example by introduction of new documents, the learned function needs to be retrained from scratch and for that new annotations have to be created to build new training data.

Now, instead of one single collection, we have two collections, namely source collection $\mathcal{C}^s$ and target collection $\mathcal{C}^t$. $\mathcal{C}^s$ has a query set $\mathcal{Q}^s$ and a document set $\mathcal{D}^s$, and for every query document pair in the source collection (q_i^s, d_j^s) there is an associated relevance label $y_{ij}^s \in \mathcal{Y}$. Here $\mathcal{Y} = \{0, 1\}$ is the set of binary relevance labels. By representing the (q_i^s, d_j^s) pairs using some feature vector $\mathbf{f}$, one can define a training set just like a normal learning to rank setting (Section 1.5.1):

$$\mathcal{T} = \{\Big(\mathbf{f}(q_i^s, d_j^s),\ y_{ij}^s\Big)\ ;\ q_i^s \in \mathcal{Q}^s,\ d_j^s \in \mathcal{D}^s,\ y_{ij}^s \in \mathcal{Y}\}$$

The target collection $\mathcal{C}^t$ has a document set $\mathcal{D}^t$. For a new query q_{new}^t intended on $\mathcal{C}^t$, the aim is to rank documents in $\mathcal{D}^t$. But in contrast with source, the target collection do not have any labeled data (or relevance judgments). In general transfer learning framework, the feature space of source domain may be different than the target domain. We here assume that the same feature vector $\mathbf{f}$ can be calculated on the source as well as on the target collection, as three catogories of features mentioned in Section 1.5.1 can be computed from basic collection statistics of any collection irrespective of different dictionaries and different word distributions of those collections. Thus the training set becomes:

$$\mathcal{S} = \{\mathbf{f}(q_{new}^t, d_j^t);\ d_j^t \in \mathcal{D}^t\}$$

Going by the definition of transfer learning and the fact that here the learning task for both source and target domains is same, which is ranking, we can apply transfer learning when $\mathcal{C}s \neq \mathcal{C}t$. Moreover, as the feature space of both source and target domains is same, the problem of ranking falls under the category of *domain adaptation* and *feature based transductive transfer learning*.

The transfer learning paradigm is mainly developed for classification and regression tasks. In Chapter 3 we will use transfer learning for regression whereas in Chapter 4 we will apply transfer learning for classification or more precisely for the ranking task. The set up and framework of each of these tasks are detailed in those chapters.

1.6 RESEARCH QUESTIONS

We here concentrate on two major directions for IR research, firstly to improve already existing IR models through proper tuning of the associated free parameters, and to learn new IR models.

Researchers have explored both avenues of parameter tuning, supervised as for example in [Taylor et al., 2006, Lv and Zhai, 2009] and unsupervised as for example in [Zhai and Lafferty, 2004, Tao and Zhai, 2006]. As most unsupervised approaches are designed for a particular framework, for a newly developed model these models are not applicable. Thus one has to develop one such parameter estimation technique from scratch using basic statistical theories.

One such newly developed IR framework is the family of information based models [Clinchant and Gaussier, 2010] which relies on a parameter named collection parameter (details in Section 2.2) that directly impacts the performance of the model. In [Clinchant and Gaussier, 2010], very little was explored regarding this parameter, which gives rise to an obvious question, is it possible to tune this parameter on collections without relevance judgments? This is realistic as one may want to deploy the model on a new collection where no relevance judgment is available.

Unsupervised techniques for parameter tuning mostly rely on statistical methods developed under a particular IR framework. Technique developed for one framework is not applicable to another. As for example the model which was developed to predict smoothing parameter of language model using "leave one out estimation" in [Zhai and Lafferty, 2004] cannot be applied to predict collection parameter for information based models. This again turns the focus towards supervised approaches, which use relevance judgments to tune parameters and work for any parameter of any IR model. If one wants to tune parameters of an IR model on a new collection where no relevance judgment is available, these supervised tuning methods do not work anymore. Creating relevance judgment for a new collection is not an easy solution either. It involves manual effort and thus

very expensive and tedious task [Carterette, 2007]. Thus is it possible to tune IR model parameters without any relevance judgment?

To answer this question, a machine learning paradigm namely transfer learning (Section 1.5.2) is explored which studies the "transfer" of labeled information from *source* domain to a *target* domain. Application of this paradigm enables one to utilize already available relevance information on past collections (as TREC collections) to tune or learn parameters which can be applied to new collections. But different collections have different vocabularies and different word distributions. Tuning on one collection and applying on another may not help directly. Then how can one use the parameter values tuned in one collection (with the relevance judgments) on another? One solution is to represent the source and target queries in the space where they no longer depend on word distribution or dictionary of the individual collections. This helps in two aspects, firstly it enables the transfer of tuned parameter values from source to target collection, and it sets the parameter values for each separate query (unlike commonly used tuning methods like k fold or "random split", where a single parameter value is set for an entire set of queries) thus giving the models the possibility to adapt at a finer granularity.

Beyond the transfer of tuned parameter values through a common representation space of queries, one can ask whether it is possible to transfer the relevance information for the learning to rank framework? This can be solved in two slightly different strategies. The first one is to learn a ranking model in the source collection using the relevance judgments and then to transfer the model so that it can adapt to the target collection. [Gao et al., 2010, Cai et al., 2011b] showed that this is indeed possible. But the drawback is that the learned model depends too much on the source collection. If not transferred from a suitable source, the model may not work well on a target collection. This leaves another possibility, which is to transfer the relevance information from source to target collection, that is to build some "pseudo-relevance" on the target based on the source collection, then train a model on target collection using this pseudo-relevance information. This should not be sensitive to the selection of the source collection and, as nothing is modified within the learning algorithm, it is possible to apply any learning to rank algorithm out of the box.

In practice learning to rank is very effective [Liu, 2009] and if deployed properly it can learn good ranking functions. But those learned functions will always remain in a predefined format depending on the learning algorithm used. As for example, for ranking SVM the form the function will be related to the kernel used (linear, polynomial, Gaussian etc.). The learning algorithm only learns optimized weights to different variables laid out in the predefined format. Thus is it possible to learn a function where there is no restriction on its form? In other words can new IR functions be generated using some basic variables and operations? This form of learning can be called "discovery", because it is equivalent to searching a function from the function space. Due to the huge (potentially infinite) dimension of this space, the intuitive solution is some heuristic search technique. Evolutionary approaches as genetic algorithm [Gordon, 1988] and genetic programming

[Fan et al., 2000] can be used here, but they are bound not to consider some valid and may be interesting functions. So the question that arises here is, is it possible to explore the function space exhaustively? [Maes et al., 2011] proposed an intelligent exhaustive discovery approach to find good solution for famous multi-arm bandit problem which can be applied to find IR scoring functions as well. This exhaustive approach helps to explore simpler yet effective functions. As these functions are generated without any restriction on their form, they are not so easy to interpret theoretically. Thus this method can find functions which cannot be developed from theoretical intuitions.

In a nut-shell, the research questions that we attempt to answer in this thesis can be summarized as:

1. How to set the value of the collection parameter of information based models?
2. Is it possible to transfer parameter values of standard IR models tuned on past labeled collections to a new collection without any relevance judgments?
3. Is it possible to learn new IR functions on collections without any relevance judgments by transferring the relevance information from past labeled collections?
4. Is it possible to exhaustively search the function space in order to discover new effective IR functions?

1.7 THESIS OUTLINE AND CONTRIBUTIONS

This thesis distinctly explores two different problem areas and is thus divided into two parts. Part I (Chapters 2 and 3) deals with the problem of identifying proper parameter values for different IR models either by unsupervised or by supervised methods. Part II (Chapters 4 and 5) concentrates on learning new IR functions either by applying learning algorithms or by simply intelligently generating new functions from basic variables and operations.

In order to venture different possibilities of unsupervised parameter tuning, we concentrated on the collection parameter of the information based model. In Chapter 2 we explore various methods to estimate the collection parameter of the information based models for ad hoc information retrieval (described in Section 2.2). In previous studies, this parameter was set to the average number of documents where the word under consideration appears. We introduce here a fully formalized estimation method for both the log-logistic and the smoothed power law models that leads to improved versions of these models in IR. Furthermore, we show that the previous setting of the collection parameter of the log-logistic model is a special case of the estimated value proposed here.

Though the methods of Chapter 2 are able to estimate the value of the collection parameter successfully, they are not applicable to estimate other parameters, for example, free

parameters of different models. In Chapter 3 we present a new method based on the transfer learning paradigm to predict the values of the parameters of standard IR models on new collections (target collection) without any relevance judgments, by using already available relevance judgments on some other collections (source collections). This learning is achieved by first mapping queries (with and without relevance judgments, from different collections, potentially in different languages) into a common space corresponding to the feature space of a positive definite symmetric kernel between queries. Standard kernel regression functions, as kernel support regression, can then be used to learn a mapping between queries lying in the kernel induced feature space and IR parameter values. We furthermore introduce a simple positive definite kernel from which one can develop more complex kernels. Our experiments, conducted on standard English and Indian IR collections, show (a) that there is no significant difference between the different kernels considered, (b) that 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 judgments (even though no target relevance judgments are used in the method we propose), and (c) that our approach is collection and language independent and can be used to efficiently tune, query by query, standard IR models to new collections, potentially written in different languages.

Application of transfer learning in a regression task to learn parameter values inspired us to explore the possibilities of directly transferring relevance information between collections, which brings us to Part II of the thesis. In Chapter 4 we propose a general approach to learn a ranking function on a target collection without relevance judgments by transferring knowledge from a source collection having such information. The relevance information in the source collection is summarized in a grid that provides, for each term frequency and document frequency values of a word in a document, an empirical estimate of the relevance of that document. We then, propagate this information from a source to a target collection using the grid and obtain a first pool of pairwise preferences over the pairs of documents in the latter. The algorithm then iteratively learns a ranking function on the target collection and assigns pairwise preferences to its documents using the scores of the learned function. Our approach can be coupled with easy in hand transfer strategies, and we further propose a simple source selection procedure in order to choose the best associated source collection for each query in a target dataset. We show the effectiveness of our approach through a series of extensive experiments which proved that the proposed approach yields results consistently and significantly above state-of-the-art IR functions as well as a state-of-the-art transfer ranking approaches.

The functions learned in Chapter 4 assumes a predefined form based on the learning algorithm used. At this point we ask the question if it is possible to learn new IR scoring functions which does not assume any predefined form. In Chapter 5 we develop an approach to search and discover functions for IR ranking from a space of simple functions. In general all IR ranking models are based on two basic variables, namely, normalized

term frequency and document frequency. Here a grammar for generating all possible functions is defined which consists of the two above mentioned variables and basic mathematical operations and functions: addition, subtraction, multiplication, division, logarithm, exponential and square root. The large set of functions generated by this grammar is filtered by checking mathematical feasibility and satisfiability to heuristic constraints on IR scoring functions proposed by the community. Obtained candidate functions are tested on various standard IR collections and several simple but promising scoring functions are identified. We show that these newly discovered functions are outperforming other state-of-the-art IR scoring models through extensive experimentation on several IR collections. We also compare the performance of functions satisfying IR constraints and those which do not, where the earlier set of functions clearly outperforms the later set. In doing so we also empirically validate these constraints.

Finally Chapter 6 summarizes the contributions of this thesis and discusses possible perspectives obtained from the thesis.

Part I

Predicting IR Model Parameters on Unlabeled Collections

2

ESTIMATION OF THE COLLECTION PARAMETER OF INFORMATION MODELS

In this chapter we explore various methods to estimate the collection parameter of the information based models for *ad hoc* information retrieval. In previous studies, this parameter was set to the average number of documents where the word under consideration appears. We introduce here a fully formalized estimation method for both the log-logistic and the smoothed power law models that leads to improved versions of these models in IR. Furthermore, we show that the previous setting of the collection parameter of the log-logistic model is a special case of the estimated value proposed here.

2.1 INTRODUCTION

Clinchant and Gaussier [Clinchant and Gaussier, 2010] introduced the family of information-based models for *ad hoc* information retrieval. One of the main ideas behind this family of models is based on the *information* content of a term, which measures how much a term deviates in a document from its average behavior in the collection. The more a term deviates in a document from its average behavior in the collection, the more likely it is *significant* or *informative* for this particular document. The retrieval status value of a document is then computed as the weighted average of information content of the query terms present in the document. To describe the *average behavior* of a term w, a "bursty" probability distribution with a single parameter λ_w is introduced, describing the distribution of term w in the collection. We call λ_w *collection parameter* of w as it regulates the way term w behaves in the collection, given the probability distribution. In [Clinchant and Gaussier, 2010], λ_w was simply set to the average number of documents where the word w appears, without much justification, except from the fact that such a setting allows the obtained model to satisfy the IDF effect [Fang et al., 2004].

We explore in this chapter different theoretical frameworks to estimate the collection parameter of information-based models, focusing on the two probability distributions proposed in previous work for this framework, namely the log-logistic distribution and the smoothed power law distribution. The estimation we finally propose is theoretically well motivated, provides an explanation to the setting used before and yields state-of-the-art performance in *ad hoc* information retrieval.

Using statistical methods to estimate model parameters is not novel in the field of IR. [Zhai and Lafferty, 2001] considered "leave-one-out" likelihood method to estimate Dirichlet prior smoothing parameter μ, and [Tao and Zhai, 2006] used regularized expectation maximization method to estimate the balance parameter of relevance feedback on language models. Similarly, here we aim to develop estimation techniques under the information model framework that will be able to predict the collection parameter λ_w defined under this framework.

We first present in section 2.2 a brief overview of information models and the role played by the collection parameter. In section 2.3, various estimation techniques are explored for estimating the collection parameter λ_w. Section 2.4 provides the experimental details and 2.5 experimentally validates the models with their collection parameter estimated properly. These experiments show that when the collection parameter is properly estimated, a gain in performance is obtained.

2.2 INFORMATION MODELS AND COLLECTION PARAMETER

In this section we will give a short overview of the Information-based models proposed by [Clinchant and Gaussier, 2010] and the probability distributions associated to it. We will also derive the formal constraints for these probability distributions based on the heuristic IR constraints (described in previous chapter Section 1.4), that needs to be satisfied by the estimation functions. The notations used in this chapter are summarized in table 1. Here w represents a term.

Notation	Description
t_w^d	term frequency of term w in document d
t_w^q	number of occurrences of term w in query q
x_w^d	normalized version of term frequency of w in d
l_d	length of document d in number of terms
l_q	length of document q in number of terms
l_{avg}	average document length in the collection
$\mathcal{N}$	number of documents in the collection
$\mathcal{N}_w$	number of documents in the collection that contains w, $N_w = \sum_d I(t_w^d > 0)$

Table 1: Notations

2.2.1 *Information Model*

The term *information* in information models, means *Shannon information*. If the behavior of a term w in a document d is as expected by its behavior in the whole collection, the

probability of occurrence of w in the document, p is high. So the Shannon information of w in the document, $-\log(p)$ is low. But, the word w is more informative for a document if p is low with respect to the collection distribution. Based on this theory, the retrieval status value, which scores a document d with respect to a query q, of information based models is formulated as:

$$RSV(q,d) = \frac{1}{l_q} \sum_{w \in q \cap d} -t_w^q \log P(X_w \geq x_w^d | \lambda_w)$$

This ranking function calculates the mean information the document d brings to the query q. Equivalently it calculates the mean of the document information brought by each query term. Since the term $\frac{1}{l_q}$ is same for all the documents, it does not affect the ranking and hence can be dropped. Thus the above RSV takes the following form:

$$RSV(q,d) = \sum_{w \in q \cap d} -t_w^q \log P(X_w \geq x_w^d | \lambda_w) \tag{2.2.1}$$

Different components of this model are as follows:

1. x_w^d is a normalization function depending on the term frequency t_w^d, of w in d, and on the length, l_d, of d. This is because in IR, documents which are compared and ranked are of different lengths. So, in almost all IR models, instead of taking raw term frequency values the normalized versions are used. Following [Clinchant and Gaussier, 2010], in this work it is defined as:

$$x_w^d = t_w^d \log\left(1 + c\frac{l_{avg}}{l_d}\right) \tag{2.2.2}$$

   where c is the smoothing parameter.

2. P is a probability distribution defined over a random variable, X_w, associated to each word w. This probability distribution must be:

   - Continuous, the random variable under consideration, x_w^d, being continuous;
   - Compatible with the domain of x_w^d, i.e. if $x_{\min}$ is the minimum value of x_w^d, then $P(X_w \geq x_{\min} | \lambda_w) = 1$;
   - Bursty, i.e. representing the property that when a word occurs in a document, it is much likely to appear again. Formally it should be such that: $\forall \epsilon > 0,\ g_\epsilon(x) = P(X \geq x + \epsilon | X \geq x)$ is strictly increasing in x;

3. λ_w is a collection-dependent parameter of P, set in [Clinchant and Gaussier, 2010] to $\lambda_w = \frac{N_w}{N}$. We want in this study to provide a sound estimation for this parameter.

2.2.2 *Information Model and Heuristic IR Constraints*

Heuristic retrieval constrains proposed by [Fang et al., 2004] are conditions that a good IR functions should satisfy. These constraints are descried in Section 1.4.

From the definition of x_w^d (Eq. 2.2.2) it can be clearly deduced that:

$$\frac{\partial x_w^d}{\partial t_w^d} > 0; \;\; \frac{\partial x_w^d}{\partial l_d} < 0; \frac{\partial^2 x_w^d}{\partial (t_w^d)^2} \geq 0$$

For the information model family $h = \log P(X_w \geq x_w^d | \lambda_w)$.

- Indeed $P(X_w \geq x_w^d | \lambda_w)$ is a decreasing function with respect to x_w^d. Thus:

$$\frac{\partial P}{\partial x_w^d} < 0 \Rightarrow \frac{\partial h}{\partial x_w^d} > 0$$

  Since we have $\frac{\partial x_w^d}{\partial t_w^d} > 0$, we can deduce $\frac{\partial h}{\partial x_w^d} > 0$ by applying chain rule for derivatives, giving that h satisfies TF effect.

- As mentioned above, P is a bursty distribution. [Clinchant and Gaussier, 2010] presented that a bursty distribution like P always satisfies $\frac{\partial^2 P}{\partial (t_w^d)^2} > 0$, thus giving $\frac{\partial^2 h}{\partial (t_w^d)^2} < 0$. This proves that h satisfies the concavity effect.

- Again applying chain rule for derivatives:

$$\frac{\partial h}{\partial l_d} = \frac{\partial h}{\partial x_w^d} . \frac{\partial x_w^d}{\partial l_d}$$

  We already have $\frac{\partial h}{\partial x_w^d} > 0$ and $\frac{\partial x_w^d}{\partial l_d} < 0$, hence it can be said that $\frac{\partial h}{\partial l_d} < 0$. So, h satisfies document-length effect as well.

The fourth important effect, the IDF effect, has to be enforced by the setting of λ_w. In the next section we discuss this constraint in light of λ_w.

2.2.3 *IDF Effect and λ_w*

IDF effect stipulates that while assigning a retrieval score to a document, there should be a constraint to weigh down the effect of those terms on the scoring which have a high document or collection frequency, as these terms have lower discrimination power. The IDF effect is satisfied only with a proper choice of λ_w. So here we formalize a constraint based on IDF effect which must be followed by λ_w .

For information model family $h = -logP(X_w \geq x_w^d | \lambda_w)$. The IDF effect ensures that $\frac{\partial h}{\partial N_w} < 0$, which again implies $\frac{\partial P}{\partial N_w} > 0$. Now P is a function of λ_w and in turn λ_w is a

function of $\mathcal{N}_w$ (Section 2.2.1), as it gets estimated by a suitable technique. Henceforth the notation λ_w is treated as a function which actually estimates the collection parameter. So, we can write:

$$\frac{\partial P}{\partial \mathcal{N}_w} = \frac{\partial P}{\partial \lambda_w} . \frac{\partial \lambda_w}{\partial \mathcal{N}_w}$$

But, we have $\frac{\partial P}{\partial \mathcal{N}_w} > 0$. Hence, the IDF effect becomes:

$$\frac{\partial P}{\partial \lambda_w} . \frac{\partial \lambda_w}{\partial \mathcal{N}_w} > 0 \tag{2.2.3}$$

Above is the form of IDF effect which the estimation function λ_w must follow. If P decreases with the parameter λ_w, then the estimation function must decrease with $\mathcal{N}_w$. On the other hand if P increases with parameter λ_w, then the estimation function must also increase with $\mathcal{N}_w$.

2.2.4 *Probability Distributions in Information Models*

As one can note, Equation 2.2.1 computes the information brought by the document on each query word ($-\log P(X_w \geq x_w^d | \lambda_w)$) weighted by the importance of the word in the query ($\frac{t_w^q}{l_q}$). In order to define a proper IR model, one needs to choose a particular bursty distribution. Two such distributions have been proposed and studied by [Clinchant and Gaussier, 2010], and we will rely on them here. These are the log-logistic and smoothed power law distributions. In this section we derive the forms of IDF effect for these distributions which must be followed by any estimation method.

2.2.4.1 *Log-logistic Distribution*

Log-logistic distribution, denoted by `LGD`, is a power law distribution which is also bursty. For $X \geq 0$, `LGD` is defined by:

$$P_{LGD}(X < x | r, \beta) = \frac{x^\beta}{x^\beta + r^\beta}$$

where β is the shape parameter and r is the scale parameter. [Clinchant and Gaussier, 2010] considered a restricted form, where $\beta = 1$. The distribution takes the following form, where the collection parameter $\lambda_w = r$.

$$P_{LGD}(X \geq x_w^d | \lambda_w) = \frac{\lambda_w}{x_w^d + \lambda_w}$$

To check the nature of P_{LGD} with respect to λ_w, we take the following derivative:

$$\begin{aligned} \frac{\partial P_{LGD}}{\partial \lambda_w} &= \frac{(x_w^d + \lambda_w) - \lambda_w}{(x_w^d + \lambda_w)^2} \\ &= \frac{x_w^d}{(x_w^d + \lambda_w)^2} > 0 \quad [\because x_w^d > 0, \lambda_w > 0\ ] \end{aligned}$$

So, for log-logistic distribution the IDF effect becomes:

$$\frac{\partial \lambda_w}{\partial \mathcal{N}_w} > 0 \tag{2.2.4}$$

i.e. the estimation function should be an increasing function of $\mathcal{N}_w$.

2.2.4.2 *Smoothed Power-Law Distribution*

Smoothed power-law distribution, denoted by SPL is also a power law distribution which is bursty. For $X > 0$, SPL is defined as:

$$P_{SPL}(X > x_w^d | \lambda_w) = \frac{(\lambda_w)^{\frac{x_w^d}{x_w^d+1}} - \lambda_w}{1 - \lambda_w}, (0 < \lambda_w < 1)$$

To check the nature of P_{SPL} with respect to λ_w, we take the following derivative:

$$\frac{\partial P_{SPL}}{\partial \lambda_w} = \frac{\left(\frac{x_w^d}{x_w^d+1}(\lambda_w)^{-\frac{1}{x_w^d+1}} - 1\right)(1-\lambda_w) + \left((\lambda_w)^{\frac{x_w^d}{x_w^d+1}} - \lambda_w\right)}{(1-\lambda_w)^2}$$

It is given that $0 < \lambda_w < 1$. So the term $(1 - \lambda_w)$ is always positive, as is $(1 - \lambda_w)^2$. Now let,

$$\begin{aligned}
f(\lambda_w) &= \left(\frac{x_w^d}{x_w^d+1}(\lambda_w)^{-\frac{1}{x_w^d+1}} - 1\right)(1-\lambda_w) + \left((\lambda_w)^{\frac{x_w^d}{x_w^d+1}} - \lambda_w\right) \\
&= \frac{x_w^d}{x_w^d+1}(\lambda_w)^{-\frac{1}{x_w^d+1}}(1-\lambda_w) + (\lambda_w)\frac{x_w^d}{x_w^d+1} - 1 \\
\frac{\partial f}{\partial \lambda_w} = f'(\lambda_w) &= -\frac{x_w^d}{(x_w^d+1)^2}(\lambda_w)^{-\frac{x_w^d+2}{x_w^d+1}}(1-\lambda_w) - \frac{x_w^d}{x_w^d+1}(\lambda_w)^{-\frac{1}{x_w^d+1}} + \frac{x_w^d}{x_w^d+1}(\lambda_w)^{-\frac{1}{x_w^d+1}} \\
&= -\frac{x_w^d}{(x_w^d+1)^2}(\lambda_w)^{-\frac{x_w^d+2}{x_w^d+1}}(1-\lambda_w)
\end{aligned}$$

Since, $x_w^d > 0$, $\frac{x_w^d}{(x_w^d+1)^2} > 0$. It is also given that $0 < \lambda_w < 1$. So, the terms $(\lambda_w)_w^{-\frac{x_w^d+2}{x_w^d+1}}$ and $1 - \lambda_w > 0$ are also positive. Hence, $f'(\lambda_w) < 0$ which implies that $f(\lambda_w)$ is strictly decreasing with respect to λ_w. Now, $f(0) = -1$ and $f(\lambda_w)$ is strictly decreasing. This concludes that, for $\lambda_w \in (0,1)$, $f(\lambda_w)$ is always negative.

We have, $\frac{\partial P_{SPL}}{\partial \lambda_w} = \frac{f(\lambda_w)}{(1-\lambda_w)^2}$. For $\lambda_w \in (0,1)$, the denominator is always positive and numerator is always negative. Hence, $\frac{\partial P_{SPL}}{\partial \lambda_w} < 0$.

For smoothed power law distribution the IDF effect becomes:

$$\frac{\partial \lambda_w}{\partial \mathcal{N}_w} < 0 \tag{2.2.5}$$

i.e. the estimation function should be a decreasing function of $\mathcal{N}_w$.

2.3 ESTIMATION OF THE COLLECTION PARAMETER

We review here three important methods for estimating the collection parameter λ_w on the basis of idf effect derived in the previous section. Firstly, most widely used and standard method, *maximum likelihood estimation*, which in this case fails to yield any valid estimate for both distributions. Secondly, *Kaplan-Meier estimation*, used mainly in survival analysis, also fails to yield any valid estimate for smoothed power law distribution. Lastly, *generalized method of moments*, which is able to estimate the parameters for both distributions.

2.3.1 *Maximum Likelihood Estimation*

Maximum likelihood is a standard choice for estimating the parameters of probability distributions. However, this method does not always yield an estimate as the likelihood may be maximum at the bounds of the definition domain of the parameter. If the bounds are not valid values for the parameter, then the estimate is undefined. This is exactly what happens here for both the log-logistic and smoothed power law distributions.

2.3.1.1 *Log-logistic Distribution*

For the log-logistic distribution, the probability density function is given by:

$$\begin{aligned} PDF(\lambda_w, x_w^d) &= \frac{\frac{1}{\lambda_w}}{(1+\frac{x_w^d}{\lambda_w})^2} \\ &= \frac{\lambda_w}{(x_w^d+\lambda_w)^2} \end{aligned}$$

Thus likelihood function of log-logistic distribution is:

$$\begin{aligned} L(\lambda_w, X_w) &= \log \prod_d \frac{\lambda_w}{(x_w^d+\lambda_w)^2} \\ &= \sum_d \log \frac{\lambda_w}{(x_w^d+\lambda_w)^2} \\ &= \mathcal{N} \log \lambda_w - \sum_d 2.\log(x_w^d+\lambda_w) \end{aligned}$$

The word w is present in $\mathcal{N}_w$ documents. So for $\mathcal{N}-\mathcal{N}_w$ documents $x_w^d = 0$. So the above expression can be decomposed into two parts:

$$\sum_d 2.\log(x_w^d+\lambda_w) = \sum_{d|x_w^d>0} 2.\log(x_w^d+\lambda_w) + \sum_{d|x_w^d=0} 2.\log(x_w^d+\lambda_w)$$

Clearly, the expression $\sum_{d|x_w^d>0} 2.\log(x_w^d + \lambda_w)$ contains $\mathcal{N}_w$ terms and the expression $\sum_{d|x_w^d=0} 2.\log(x_w^d + \lambda_w)$ contains $\mathcal{N} - \mathcal{N}_w$ terms. For any λ_w, we have:

$$\sum_{d|x_w^d=0} 2.\log(x_w^d + \lambda_w) = 2.(\mathcal{N} - \mathcal{N}_w)\log\lambda_w$$

Replacing the values in the log-likelihood of LGD:

$$\begin{aligned} L(\lambda_w, X_w) &= \mathcal{N}\log\lambda_w - 2.(\mathcal{N} - \mathcal{N}_w)\log\lambda_w - \sum_{d|x_w^d>0} 2.\log(x_w^d + \lambda_w) \\ &= (2\mathcal{N}_w - \mathcal{N})\log\lambda_w - 2\sum_{d|x_w^d>0} \log(x_w^d + \lambda_w) \end{aligned}$$

If the documents frequency of w is below half the number of documents in the collection, that is $\mathcal{N}_w < \frac{\mathcal{N}}{2}$, then $L(\lambda_w, X_w)$ is maximized when $\lambda_w \to 0$. In practice, $\mathcal{N}_w < \frac{N}{2}$ for most of the terms, and the maximum likelihood estimation method does not yield any valid estimate.

2.3.1.2 *Smoothed Power Law Distribution*

Log likelihood of smoothed power law is:

$$\begin{aligned} L(\lambda_w, X_w) &= \log\prod_d \Big[\frac{-\log\lambda_w}{1-\lambda_w}\frac{(\lambda_w)^{\frac{x_w^d}{x_w^d+1}}}{(x_w^d+1)^2}\Big] \\ &= -N\log\frac{\log\lambda_w}{1-\lambda_w} + \sum_d \log\frac{(\lambda_w)^{\frac{x_w^d}{x_w^d+1}}}{(x_w^d+1)^2} \\ &= -N\log\log\lambda_w + N\log(1-\lambda_w) + \sum_d \Big[\frac{x_w^d}{x_w^d+1}\log\lambda_w - 2\log(x_w^d+1)\Big] \end{aligned}$$

Taking derivative of $L(\lambda_w, X_w)$ with respect to λ_w:

$$\begin{aligned} \frac{\partial L}{\partial\lambda_w} = L'(\lambda_w, X_w) &= -N\frac{1}{\log\lambda_w}\frac{1}{\lambda_w} - N\frac{1}{1-\lambda_w} + \sum_d \frac{x_w^d}{x_w^d+1}\frac{1}{\lambda_w} \\ &= \frac{1}{\lambda_w}\Big[-N\frac{1}{\log\lambda_w} - \frac{N\lambda_w}{1-\lambda_w} + \sum_d \frac{x_w^d}{x_w^d+1}\Big] \end{aligned}$$

Now equating $L'(\lambda_w, X) = 0$ to find the value of λ_w where the likelihood is maximum:

$$\begin{aligned} & \frac{1}{\lambda_w}\Big[-N\frac{1}{\log\lambda_w} - \frac{N\lambda_w}{1-\lambda_w} + \sum_d \frac{x_w^d}{x_w^d+1}\Big] = 0 \\ \implies & -N\frac{1}{\log\lambda_w} - \frac{N\lambda_w}{1-\lambda_w} + \sum_d \frac{x_w^d}{x_w^d+1} = 0 \qquad [\because \lambda_w > 0] \\ \implies & \frac{\lambda_w}{1-\lambda_w} = \frac{1}{N}\sum_d \frac{x_w^d}{x_w^d+1} - \frac{1}{\log\lambda_w} \end{aligned}$$

Let, $g(\lambda_w) = \frac{\lambda_w}{1-\lambda_w}$ and $h(\lambda_w) = -\frac{1}{\log \lambda_w} = -(\log \lambda_w)^{-1}$. For a particular term x_w^d is constant. So, $c = \frac{1}{N}\sum_d \frac{x_w^d}{x_w^d+1}$ is also a constant. The above equation can be written as:

$$g(\lambda_w) = c + h(\lambda_w)$$

Given that the range of λ_w is $(0, 1)$. $g(\lambda_w)$ and $h(\lambda_w)$ both increases from 0 to ∞ as λ_w goes from 0 to 1. So, $c + h(\lambda_w)$ goes from c to ∞ as λ_w goes from 0 to 1. Figure 3 shows how $g(\lambda_w)$ and $c + h(\lambda_w)$ behaves as λ_w goes from 0 to 1.

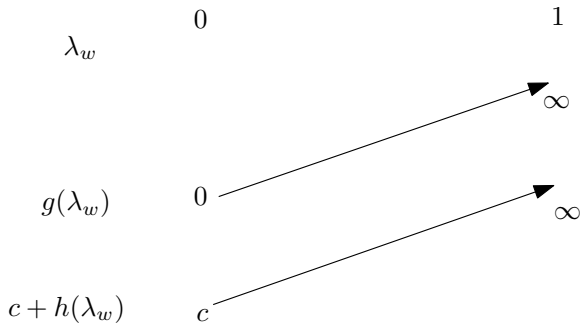


Figure 3: Nature of $g(\lambda_w)$ and $= c + h(\lambda_w)$.

If $g(\lambda_w)$ and $c + h(\lambda_w)$ crosses each other, the equation will have a solution for λ_w. For that we need to show that for at least one value of λ_w, $g(\lambda_w) > h(\lambda_w)$.

We start with the following standard logarithmic inequality:

$$\log(1 + p) > \frac{p}{p+1}$$

By replacing p by $p - 1$:

$$\log p > \frac{p-1}{p} \tag{2.3.1}$$

If possible, let us assume $g(\lambda_w) > h(\lambda_w)$. Then,

$$\begin{aligned}
& \frac{\lambda_w}{1-\lambda_w} > -\frac{1}{\log \lambda_w} \\
\implies \quad & \frac{\lambda_w}{\lambda_w - 1} < \frac{1}{\log \lambda_w} \qquad [\text{ multiplying both side by } -1\ ] \\
\implies \quad & \frac{\lambda_w \log \lambda_w}{\lambda_w - 1} > 1 \qquad [\because 0 < \lambda_w < 1, \therefore \log \lambda_w < 0\ ] \\
\implies \quad & \lambda_w \log \lambda_w < \lambda_w - 1 \qquad [\because 0 < \lambda_w < 1, \therefore \lambda_w - 1 < 0\ ] \\
\implies \quad & \log \lambda_w < \frac{\lambda_w - 1}{\lambda_w}
\end{aligned}$$

But from the logarithmic inequality 2.3.1, this is not true. Hence $g(\lambda_w) < h(\lambda_w)$ and $c > 0$.

$$g(\lambda_w) < c + h(\lambda_w)$$

Thus, here again, the maximum likelihood method does not yield an estimate for λ_w.

2.3.2 *Kaplan Meier Estimation*

The Kaplan-Meier estimation method [Kaplan and Meier, 1958] as designed for survival analysis, where the primary interest is the survival function of the form $P(T > t)$ which is the probability that a member from a given population will have a lifetime exceeding time t. For a sample set of size N, sorting the samples according to their observed death time $t_1 < \ldots < t_i < \ldots < t_N$. Here t_i is the death time of the i^{th} sample. Then the survival probability of a sample after time t_i, $P(T > t_i)$ is estimated as:

$$P(T > t_i) = \prod_{r=1}^{i} \frac{N - r}{N - r + 1}$$

The form of survival function $P(T > t)$ is of course the one used in the information models: $P(X_w > x_w^d | \lambda_w)$. The Kaplan-Meier method proposes an estimate for all the quantities $P(X_w > x_w^d | \lambda_w)$ corresponding to the different observed values of x_w^d in a given collection. Let the sorted (non-decreasing) normalized term frequencies for a term w in the documents in the collection be $x_w^{d_1} \leq x_w^{d_2} \leq \ldots \leq x_w^{d_{\mathcal{N}}}$ (one can always obtain this form by renumbering the documents in the collection). Let the 'time' be the domain of normalized term frequency values and 'observed death times' are normalized term frequency values. Hence, for $i \leq \mathcal{N}$, the probability $P(X \geq x_w^{d_i})$ can be estimated through the Kaplan Meier estimator:

$$P(X \geq t_w^{d_i}) = \prod_{r=1}^{i} \frac{\mathcal{N} - r}{\mathcal{N} - r + 1}$$

The bias of this estimator increases when $x_w^{d_i}$ increases [Balakrishnan and Rao, 2004] and, to rely on an estimate with lower bias, i is chosen such that $x_w^{d_i}$ is the first non zero normalized term frequency value; hence $i = \mathcal{N} - \mathcal{N}_w$. According to the Kaplan-Meier estimation formula, one thus has:

$$P_{est}(X \geq x_w^{d_i}) = \frac{\mathcal{N} - i}{\mathcal{N}}$$

We are now going to see how this applies to the log-logistic and smoothed power law distributions.

2.3.2.1 *Log-logistic Distribution*

Equating the probability estimated by Kaplan-Meier to its theoretical value leads, for the log-logistic distribution, to:

$$\begin{aligned} & \frac{\lambda_w}{x_w^{d_i} + \lambda_w} = \frac{\mathcal{N} - i}{\mathcal{N}} = \frac{\mathcal{N}_w}{\mathcal{N}} \\ \Longrightarrow \quad & \lambda_w = \frac{\mathcal{N}_w}{\mathcal{N} - \mathcal{N}_w} x_w^{d_i} \end{aligned} \tag{2.3.2}$$

To verify whether this estimation technique is satisfying the idf effect for LGD (Eq. 2.2.4):

$$\begin{aligned}\frac{\partial \lambda_w}{\partial \mathcal{N}_w} &= \frac{(\mathcal{N}-\mathcal{N}_w)-\mathcal{N}_w.(-1)}{(\mathcal{N}-\mathcal{N}_w)^2}x_w^d \\ &= \frac{(\mathcal{N}-\mathcal{N}_w)+\mathcal{N}_w}{(\mathcal{N}-\mathcal{N}_w)^2}x_w^d = \frac{\mathcal{N}}{(\mathcal{N}-\mathcal{N}_w)^2}x_w^d\end{aligned}$$

Since, $\mathcal{N}$, $\mathcal{N}_w$ and x_w^d all are non-negative, $\frac{\mathcal{N}}{(\mathcal{N}-\mathcal{N}_w)^2}x_w^d > 0$, thus giving $\frac{\partial \lambda_w}{\partial \mathcal{N}_w} > 0$. The IDF criterion is thus satisfied, which shows that this estimate can be used for the log-logistic information model. The model obtained with this estimate will be referred to as LGD_{KM}.

2.3.2.2 *Smoothed Power Law Distribution*

Equating the estimated probability by Kaplan Meier estimation to the probability obtained by SPL function:

$$\begin{aligned}&\frac{\lambda_w^{\left(\frac{x_w^{d_i}}{x_w^{d_i}+1}\right)}-\lambda_w}{1-\lambda_w} = \frac{\mathcal{N}-i}{\mathcal{N}} = \frac{\mathcal{N}_w}{\mathcal{N}} \\ \implies\ &\mathcal{N}\lambda_w^{\left(\frac{x_w^{d_i}}{x_w^{d_i}+1}\right)} - \mathcal{N}\lambda_w = \mathcal{N}_w - \mathcal{N}_w\lambda_w \\ \implies\ &\mathcal{N}\lambda_w^{\left(\frac{x_w^{d_i}}{x_w^{d_i}+1}\right)} - \mathcal{N}_w = \mathcal{N}\lambda_w - \mathcal{N}_w\lambda_w \\ \implies\ &\mathcal{N}\lambda_w^{\left(\frac{x_w^{d_i}}{x_w^{d_i}+1}\right)} - \mathcal{N}_w = \lambda_w(\mathcal{N}-\mathcal{N}_w)\end{aligned}$$

Recall that for SPL, the domain of λ_w is $(0,1)$. The above equation however does not always have a solution in $(0,1)$. Indeed, let us consider a plausible setting where for a word normalized term frequency $x_w^{d_i} = 1$ and $\mathcal{N}_w = \frac{\mathcal{N}}{2}$. Now substituting $x_w^{d_i}$ and $\mathcal{N}_w$ in the above equation one has:

$$\begin{aligned}&\mathcal{N}\,(\lambda_w)^{\frac{1}{2}} - \frac{\mathcal{N}}{2} = \lambda_w\mathcal{N} - \frac{\lambda_w\mathcal{N}}{2} \\ \Rightarrow\ &2\mathcal{N}\sqrt{\lambda_w} - \mathcal{N} = \lambda_w\mathcal{N} \\ \Rightarrow\ &\lambda_w - 2\sqrt{\lambda_w} + 1 = 0 \\ \Rightarrow\ &\left(\sqrt{\lambda_w}-1\right)^2 = 0\end{aligned}$$

This equation has only one solution, namely $\lambda_w = 1$, which is outside the domain of λ_w. Thus we have shown a counter example where the equation do not have any solution for λ_w in $(0,1)$ under a reasonable and probable settings. Hence, Kaplan-Meier estimate does not always yield a valid solution for the smoothed power law.

Even if the Kaplan-Meier estimate can be used for the log-logistic distribution, it still suffers from the fact that it is based on a quantity, $x_w^{d_i}$, which is not robust as it consists

of a single observation (i.e. a single document) in the collection for any given word. We are going to present another estimation method which does not suffer from the same drawback and which yields valid estimates for both the log-logistic and the smoothed power law distributions.

2.3.3 *Generalized Method of Moments*

The probability that a term w is present in a document d is given, under the log-logistic and smoothed power law distributions, by $P(x_w^d \geq 1|\lambda_w)$. But according to the definition of x_w^d (Eq. 2.2.2):

$$\begin{aligned} & x_w^d = t_w^d \log\left(1 + c\frac{l_{avg}}{l_d}\right) \\ \Longrightarrow \qquad & t_w^d = \frac{x_w^d}{\log\left(1+c\frac{l_{avg}}{l_d}\right)} \end{aligned}$$

which enables to rewrite the probability equation as:

$$P(t_w^d \geq 1|\lambda_w) = P(x_w^d \geq \log\left(1 + c\frac{l_{avg}}{l_d}\right)|\lambda_w)$$

The expectation of observing a term w in documents of the collection is thus:

$$\sum_d P(x_w^d \geq \log\left(1 + c\frac{l_{avg}}{l_d}\right)|\lambda_w)$$

The generalized method of moments simply amounts here in equating this expectation with the actual number of documents in which the term occurs, $\mathcal{N}_w$, leading to the following constraint:

$$\mathcal{N}_w = \sum_d P(x_w^d \geq \log\left(1 + c\frac{l_{avg}}{l_d}\right)|\lambda_w) = \sum_d P(x_w^d \geq \alpha_d|\lambda_w) \tag{2.3.3}$$

where $\alpha_d = \log\left(1 + c\frac{l_{avg}}{l_d}\right)$.

The application of generalized method of moments is not novel in IR. Johnson and Kotz [Johnson et al., 1993], for example, used a variant of the standard method of moments in which the empirical variance is replaced by the inverse document frequency. Church and Gale [Church and Gale, 1995] used the term "generalized method of moments" to denote a method in which the parameter of a probability distribution is set according to a constraint making use of a quantity observed in the collection (as the IDF, which is related to $\mathcal{N}_w$).

The main advantage of this method is that it relies on a robust quantity, namely $\mathcal{N}_w$, to estimate the value of λ_w. Contrary to the Kaplan-Meier estimate which relies on a quantity observed in only one document and subject to variations (related to the fact, for example,

that an author may have used a different term, or less occurrences of the term, to express the same idea), here one relies on the number of documents in which the term occurs, which is more robust to these types of variations. We are now going to see how the above constraint is expressed in the log-logistic and smoothed power law distributions.

2.3.3.1 *Log-logistic Distribution*

For the log-logistic distribution, Equation 2.3.3 is expressed as:

$$\begin{aligned} & \mathcal{N}_w = \textstyle\sum_d \frac{\lambda_w}{\alpha_d + \lambda_w}, \quad \lambda_w > 0 \\ \Longrightarrow \quad & \underbrace{\frac{\mathcal{N}_w}{\lambda_w}}_{f(\lambda_w)} = \underbrace{\sum_d \frac{1}{\alpha_d + \lambda_w}}_{g(\lambda_w)}, \quad \lambda_w > 0 \end{aligned} \tag{2.3.4}$$

Clearly $f'(\lambda_w) < 0$, $f''(\lambda_w) > 0$, $g'(\lambda_w) < 0$, $g''(\lambda_w) > 0$, hence both f and g are concave functions, and are such that $g(0) = \sum_d (\alpha_d)^{(-1)} < f(0) = +\infty$. Figure 4 illustrates how the functions f and g behave. Let a be a strictly positive number; we have $f(a) = \frac{\mathcal{N}_w}{a}$ and

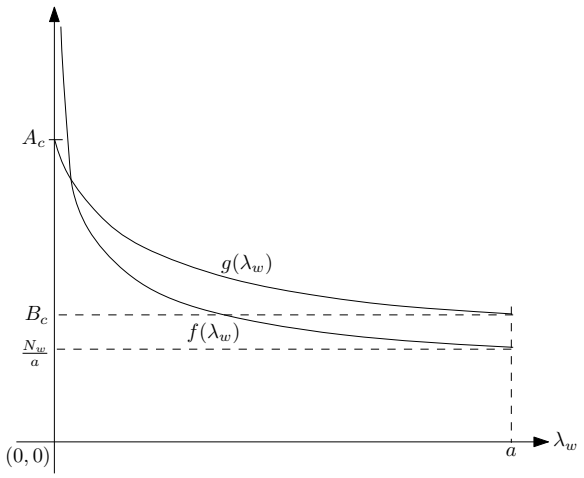


Figure 4: Condition to have a solution for λ_w in LGD using generalized method of moments.

$g(a) = \sum_d (\alpha_d + a)^{(-1)}$. A solution to Equation 2.3.4, for all terms w, exists in $(0, a)$ if and only if $g(a) > \frac{\mathcal{N}_w}{a}$ for all terms w, i.e. if and only if:

$$a \sum_d \frac{1}{\log\left(1 + c\frac{l_{avg}}{l_d}\right) + a} > \mathcal{N}_w^{max} \tag{2.3.5}$$

As $\lim_{a \to +\infty} a \sum_d \frac{1}{\log\left(1 + c\frac{l_{avg}}{l_d}\right) + a} = \mathcal{N} > \mathcal{N}_w^{max}$, a solution will necessarily exist for a sufficiently large. The free parameter (as the smoothing coefficient of language models, k_1 in BM25 and c in the information models) is usually optimized on a subset of queries for which one has associated relevance judgments. We adopt here the same strategy, with an additional step checking whether the above condition is satisfied:

1. Choose a value range for c;
2. Choose a large, as $a = 100$ (one thus has $\lambda_w \in (0, a)\ \forall w$);
3. For each value of c, if it satisfies the condition 2.3.5, estimate λ_w for each w by solving Equation 2.3.4;
4. If the current value of c does not satisfy condition 2.3.5, increase a and go back to step 3.

Two remarks need be made here:

(a) Equation 2.3.4 can be solved by standard methods, as Newton-Raphson method or using a simple (yet less efficient) dichotomy process;

(b) λ_w is estimated for w in the collection, so that the treatment of subsequent queries is efficient and straightforward.

The IDF criterion for LGD stipulates that λ_w should increase with $\mathcal{N}_w$ (Eq. 2.2.4). We now show that this criterion is satisfied with the above estimate. We know that λ_w is a solution of the equation:

$$\mathcal{N}_w = \sum_d \frac{\lambda_w}{\alpha_d + \lambda_w} = h(\lambda_w)$$

Taking derivative it is trivial to show that $h'(\lambda_w) > 0$, so h is an increasing function of λ_w. But from the equality above, if $\mathcal{N}_w$ increases, so does $h(\lambda_w)$. This directly gives that $\mathcal{N}_w$ increases with λ_w, thus satisfying IDF criterion for the estimate obtained with the above method. We will refer to the model obtained, for the log-logistic distribution, through the generalized method of moments as LGD_{GM}.

2.3.3.2 *Smoothed Power Law Distribution.*

For smoothed power law distribution, Equation 2.3.3 is expressed as:

$$\begin{aligned}
& \mathcal{N}_w = \sum_d \frac{\lambda_w^{\frac{\alpha_d}{\alpha_d+1}} - \lambda_w}{1-\lambda_w}, \quad \lambda_w \in (0,1) \\
\implies \quad & (1-\lambda_w)(\mathcal{N} - \mathcal{N}_w) + \mathcal{N}_w = \sum_d \lambda_w^{\frac{\alpha_d}{\alpha_d+1}}, \quad \lambda_w \in (0,1) \\
\implies \quad & \underbrace{\lambda_w(\mathcal{N} - \mathcal{N}_w) + \mathcal{N}_w}_{f(\lambda_w)} = \underbrace{\sum_d \lambda_w^{\frac{\alpha_d}{\alpha_d+1}}}_{g(\lambda_w)}, \ \lambda_w \in (0,1) \qquad (2.3.6)
\end{aligned}$$

Clearly f is a linear increasing function. As $g'(\lambda_w) < 0$ and $g''(\lambda_w) > 0$, g is a concave function. We also have $g(0) < f(0)$ and $g(1) = f(1) = \mathcal{N}$. To have a solution of

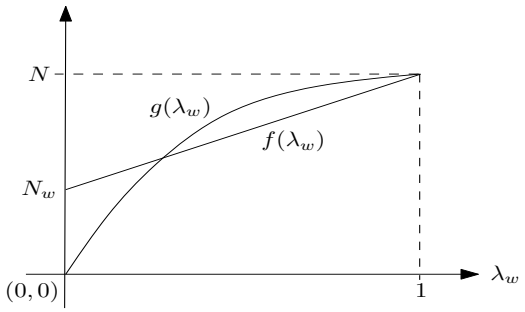


Figure 5: Condition of to have a solution for λ_w in SPL using generalized method of moments.

Equation 2.3.6 for CP, $f(\lambda_w)$ and $g(\lambda_w)$ must cross at a point in $(0,1)$, because $\lambda_w \in (0,1)$. The situation is explained in Figure 5. As $f > g$ at $\lambda_w = 0$, after crossing it must be $f < g$. Since $f(1) = g(1)$, g must be above f in the neighborhood of 1. For a very small positive constant ϵ the condition for having a solution of Equation 2.3.6 for λ_w becomes $f(1-\epsilon) - g(1-\epsilon) < 0$. One has:

$$\begin{aligned} f(1-\epsilon) &= (1-\epsilon)\mathcal{N} + \epsilon\mathcal{N}_w \text{ and} \\ g(1-\epsilon) &= \sum_d (1-\epsilon)^{\frac{\alpha_d}{\alpha_d+1}} \approx \sum_d 1 - \epsilon^{\frac{\alpha_d}{\alpha_d+1}} \\ &= \mathcal{N} - \sum_d \epsilon \frac{\alpha_d}{\alpha_d+1} \end{aligned}$$

Thus $f(1-\epsilon) - g(1-\epsilon) = \epsilon(\mathcal{N}_w + \sum_d \frac{\alpha_d}{\alpha_d+1} - \mathcal{N})$. Since $\epsilon > 0$, g is above f in the neighborhood of 1, if and only if:

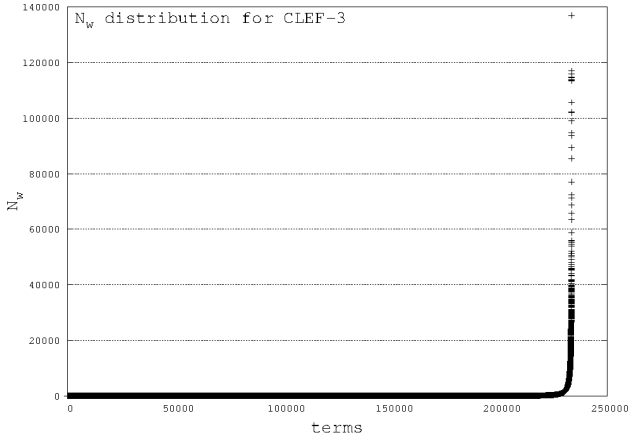
$$\begin{aligned} & \left(\mathcal{N}_w + \sum_d \frac{\alpha_d}{\alpha_d+1} - \mathcal{N}\right) > 0 \\ \Rightarrow \quad & \sum_d \frac{\log(1 + c\frac{l_{avg}}{l_d})}{1 + \log(1 + c\frac{l_{avg}}{l_d})} < N - \mathcal{N}_w \end{aligned}$$

For all terms w, the above condition can be generalized to:

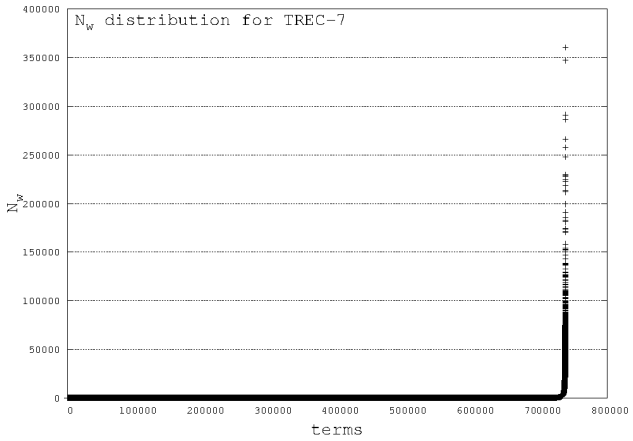
$$\sum_d \frac{\log(1 + c\frac{l_{avg}}{l_d})}{1 + \log(1 + c\frac{l_{avg}}{l_d})} < N - \mathcal{N}_w^{max} \tag{2.3.7}$$

which provides a constraint on the admissible values of c. For all the values of c compatible with this constraint, one can then estimate λ_w by solving Equation 2.3.6, which can be done using standard methods.

However, the constraint expressed by Equation 2.3.7 is strong as soon as there are terms in the collections which occur in many documents. The presence of such terms limits the range in which accurate estimates for λ_w can be found for the other terms. It turns out in



(a) CLEF3



(b) TREC7

Figure 6: Distribution of $\mathcal{N}_w$ on 2 collections with horizontal line set at $p = 0.05\%$ of total number of terms.

practice that very few terms appear in many documents, as illustrated in Figure 6 which displays the values of $\mathcal{N}_w$ for all the terms in two different collections that will be detailed in Section 2.4.1. The shapes for the other collections are identical to the ones displayed here. In order not to limit the range of possible values for c, we recommend in practice to set λ_w to $\frac{\mathcal{N}_w}{\mathcal{N}}$ (i.e. the setting recommended in [Clinchant and Gaussier, 2010]) for the p most frequent terms, as measured by $\mathcal{N}_w$, of the collection, and to use the generalized method of moments for the other terms, with $\mathcal{N}_w^{max}$ taken from this latter set. In our experiments, we set p to 0.05% of total number of terms in the collection but none of the query terms encountered belong to this set of highly frequent terms.

Lastly, the estimate obtained through this method also satisfies the IDF criterion. We have:

$$\mathcal{N}_w = \sum_d \frac{\lambda_w^{\frac{\alpha_d}{\alpha_d+1}} - \lambda_w}{1 - \lambda_w} = h(\lambda_w)$$

As the function $h(\lambda_w)$ has the form of smoothed power law itself which is a concave function and increases with λ_w. Using a similar logic used to prove the idf effect for LGD_{GM}, it can be said that when $\mathcal{N}_w$ increases, so does λ_w, thus satisfying the idf effect given in Equation 2.2.5. We will refer to the model obtained, for the smoothed power law distribution, through the generalized method of moments as SPL_{GM}.

2.4 EXPERIMENTAL SETUP

In this section we describe all the necessary details regarding the experimental validation of the estimation methods.

2.4.1 *Collections*

For the experimentation, we used five standard IR collections from two major evaluation campaigns, namely TREC (`trec.nist.gov`) and CLEF (`www.clef-campaign.org`). Table 2 shows basic statistics of different collections used in the experimentations, namely `TREC-3,6,7,8` and `CLEF-3` AdHoc Task in English. Collections are indexed using Terrier IR Platform v3.5 [Ounis et al., 2006](`http://www.terrier.org`). All the collections are preprocessed before creating an index. This prepossessing includes stemming using standard Porter Stemmer, already implemented in Terrier, and removing stop-words using the stopword list provided by Terrier.

Collection	$\mathcal{N}$	l_{avg}	Index size	# queries
TREC-3	741856	261	427.7 MB	50
TREC-6	528155	296	373.0 MB	50
TREC-7	528155	296	373.0 MB	50
TREC-8	528155	296	373.0 MB	50
CLEF-3	169477	301	126.2 MB	60

Table 2: Statistics of the different collections used in our experiments, sorted by their Index size.

2.4.2 *IR Models and Corresponding Parameters*

Models with estimates parameters LGD_{KM}, LGD_{GM}, SPL_{GM} (Section 2.3) are tested and evaluated against:

- Original log-logistic model (denoted by LGD) and smoothed power law model (denoted by SPL) proposed in [Clinchant and Gaussier, 2010] with $\lambda_w = \frac{\mathcal{N}_w}{\mathcal{N}}$.

- Standard Okapi BM25 [Robertson and Zaragoza, 2009]
- Language model with Dirichlet prior (denoted by LM) and we restrict ourselves here to this version of the language model, which yields better results than the one based on Jelinek-Mercer smoothing [Ponte and Croft, 1998][Zhai and Lafferty, 2001].

Mean average precision (MAP) and *precision at 10 documents* (P@10) measures are used for evaluation.

For every model on each collection we performed 5 fold cross validation meaning that the query set is sequentially partitioned into 5 subsets of equal size. Of the 5 subsets, a single subset is retained for testing the model, and the remaining 4 subsets are used as training set to chose the best free parameter values of each model which optimize MAP and P@10 respectively. The free parameters are:

- c for all versions of information based models (original LGD, SPL along with LGD_{KM}, LGD_{GM}, SPL_{GM}),
- b and k_1 for BM25,
- smoothing parameter μ for LM.

The best parameter are chosen during training phase from a set of values. These set of values for different parameters are shown in Table 3. Then with those trained values, performance is measured on the set kept aside for testing. The cross-validation process is then repeated 5 times, with each of the 5 subsets used exactly once for testing. Each time a query-wise average precision is calculated for each set. After 5 folds, average precision of all the queries are obtained and MAP is calculated. For each fold we also calculate the P@10 value. We report the average of those 5 values as the final P@10 value. During comparison, to find if the performances two models are significantly different we used a paired two-sided t-test at the 0.1 level. We used query-wise average precision values for all the queries obtained after 5 folds of the cross-validation method for this statistical significance testing.

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
k_1 (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
c (all LGD, all SPL)	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
μ (LM)	10, 25, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 10000

Table 3: Sets of values of different parameters used for tuning standard IR models through 5 fold cross validation.

Note that BM25 has another free parameter namely k_3. This parameter is used in normalization of t_w^q, the number of occurences of the query term in the query itself (Eq. 1.2.1). If

$t^q_w = 1$, which is almost always the case in the all the queries provided by TREC and CLEF collections, then the whole normalized t^d_w weighting term becomes a constant and thus irrerelevant in the ranking. So instead of tuning, this parameter is assigned to its default value implemented in Terrier, which is 8. Lv and Zhai[Lv and Zhai, 2012] proposed a method, based on the log-logistic distribution to estimate k_1. But we rely here on an optimization that likely to provide a better estimate.

2.4.3 *Implementation of Estimation Methods*

All the experiments are performed on Terrier IR Platform v3.5 [Ounis et al., 2006](http://www.terrier.org) as all standard models are integrated there. As mentioned before we used indexing component of Terrier to index the collections. We also used the evaluation component to determine the performance of different models.

We implemented our estimation models inside Terrier framework. We used standard Newton's method to solve fixed point Equations 2.3.4 for LGD_{GM} and 2.3.6 for SPL_{GM} for λ_w. Note that λ_w has a clear analytical form for LGD_{KM} (Eq. 2.3.2) and can be determined directly. The 5 fold cross validation method of free parameter tuning is implemented using shell script and inside the script Terrier is used as main retrieval and evaluation module.

2.5 RESULTS

We conducted experiments in two phases. Firstly information-based models with estimated λ_w are compared against their original versions with $\lambda_w = \frac{\mathcal{N}_w}{\mathcal{N}}$ (Section 2.5.1). That is original versions of both log-logistic (LGD) and smoothed power law (SPL) distributions are compared against the same with the λ_w estimated with Kaplan-Meier estimation (LGD_{KM}) and generalized method of moments (LGD_{GM}, SPL_{GM}). Then we compare the information-based models with estimated λ_w (LGD_{KM}, LGD_{GM}, SPL_{GM}) against other two standard IR models considered here, namely BM25 and language model (LM) mentioned in Section 2.5.2.

2.5.1 *Comparison with Original Information-based Models*

Table 4 presents the experimental results comparing LGD with LGD_{KM}, LGD_{GM} and SPL with SPL_{GM}. Here the best results of each category of information model for every collection are given in bold font. The MAP values marked with a $^{\uparrow}$ are significantly better than the respective original version of information models.

	TREC-3		TREC-6		TREC-7	
	MAP	P@10	MAP	P@10	MAP	P@10
LGD	0.2456	0.4840	0.2449	0.4040	0.1895	0.4420
LGD_{KM}	0.2483	0.5000	**0.2466**	0.4040	**0.1896**	**0.4440**
LGD_{GM}	**0.2560**$^{\uparrow}$	**0.5180**$^{\uparrow}$	0.2457	**0.4060**	0.1892	0.4340
SPL	0.2517	0.5240	0.2503	**0.4040**	0.1844	**0.4500**
SPL_{GM}	**0.2677**$^{\uparrow}$	**0.5460**$^{\uparrow}$	**0.2519**	**0.4040**	**0.1909**	0.4480

	TREC-8		CLEF-3	
	MAP	P@10	MAP	P@10
LGD	0.2582	**0.4560**	0.3855	0.3136
LGD_{KM}	0.2577	0.4500	0.3847	**0.3192**
LGD_{GM}	**0.2587**	**0.4560**	**0.3951**	0.3153
SPL	0.2556	0.4620	0.3890	0.3282
SPL_{GM}	**0.2628**$^{\uparrow}$	**0.4700**	**0.4042**$^{\uparrow}$	**0.3394**$^{\uparrow}$

Table 4: LGD_{KM}, LGD_{GM} versus LGD and SPL_{GM} versus SPL. Best results are given in bold, and a result with $^{\uparrow}$ are significantly better than the respective original version of information models according to a paired two-sided t-test at the 0.1 level.

As one can see, the new models developed here outperform their original version in most cases, the difference being significant for the smoothed power law model in three collections out of five. The fact that LGD_{GM} and LGD_{KM} are not really significantly better than LGD can be explained by the fact that LGD represents an approximation of both the models. Indeed, if $l_d \approx l_{avg}$ then α_d becomes a constant ($\alpha_d = \alpha$) and the solution to Equation 2.3.4 is $\lambda_w = \frac{\alpha \mathcal{N}_w}{\mathcal{N} - \mathcal{N}_w}$, which is also the form used in LGD_{KM}. As, for most terms (and for all the query terms in all the collections considered here) $\mathcal{N}_w \ll \mathcal{N}$, we have: $\lambda_w = \frac{\alpha \mathcal{N}_w}{\mathcal{N}}$. This is the setting of λ_w proposed in [Clinchant and Gaussier, 2010] with a constant factor which however does not change the ranking of documents. Note that the same does not hold for SPL and SPL_{GM}.

2.5.2 *Comparison with Standard IR Models*

The comparison of LGD_{KM}, LGD_{GM} and SPL_{GM} with Okapi BM25 and the Dirichlet language model (LM) is presented in Table 5. Here also the best results for each collection are given in bold, and a result with $^{\downarrow}$ is significantly worse than the best model.

As one can note, the SPL_{GM} model provides the best results on four collections out of five for the MAP, and three collections out of five for P@10. Moreover, when SPL_{GM} is the best model, its performance is significantly better than the ones of other models in most cases. When it is not the best model, the difference with the best model is not significant. This shows that the new versions of the information models introduced in this paper do not

	TREC-3		TREC-6		TREC-7	
	MAP	P@10	MAP	P@10	MAP	P@10
BM25	**0.2732**	**0.5620**	0.2376$^{\downarrow}$	0.3940$^{\downarrow}$	0.1908	0.4300
LM	0.2685	0.5580	0.2427$^{\downarrow}$	0.3900$^{\downarrow}$	0.1888	0.4240$^{\downarrow}$
LGD_{KM}	0.2483$^{\downarrow}$	0.5000$^{\downarrow}$	0.2466$^{\downarrow}$	0.4040	0.1896	0.4440
LGD_{GM}	0.2560$^{\downarrow}$	0.5180$^{\downarrow}$	0.2457$^{\downarrow}$	**0.4060**	0.1892	0.4340$^{\downarrow}$
SPL_{GM}	0.2677	0.5460	**0.2519**	0.4040	**0.1909**	**0.4480**

	TREC-8		CLEF-3	
	MAP	P@10	MAP	P@10
BM25	0.2589$^{\downarrow}$	0.4640	0.3988	0.3338
LM	0.2543$^{\downarrow}$	0.4520$^{\downarrow}$	0.3938$^{\downarrow}$	0.3158$^{\downarrow}$
LGD_{KM}	0.2577	0.4500$^{\downarrow}$	0.3847$^{\downarrow}$	0.3192$^{\downarrow}$
LGD_{GM}	0.2587$^{\downarrow}$	0.4560$^{\downarrow}$	0.3951$^{\downarrow}$	0.3153$^{\downarrow}$
SPL_{GM}	**0.2628**	**0.4700**	**0.4042**	**0.3394**

Table 5: LGD_{KM}, LGD_{GM} and SPL_{GM} versus standard IR models, BM25 and LM. Best results are given in bold, and a result with $^{\downarrow}$ is significantly worse than the best model according to a paired two-sided t-test at the 0.1 level.

only outperform their original counterpart, but also that they provide state-of-the-art results, outperforming other IR models on most of the collections retained here.

2.6 CONCLUSION

The collection parameter of the family of information models determines the nature of a term in the collection. In earlier studies, it was assigned to the average number of documents where the term appears, even though this setting was not fully justified. In this chapter we have explored various techniques to properly estimate this collection parameter and have shown that the generalized method of moments provide valid estimate for both the log-logistic and smoothed power law distributions, compatible with the IDF criterion. These estimates also yield state-of-the-art results in our IR experiments, significantly improving over the original setting in case of the smoothed power law. Furthermore, the new version of the SPL model developed here regularly outperforms all the other models in most cases (which was not the case for its original version), leading to a new, state-of-the-art IR model.

In this chapter we have shown that using generalized method of moments it is possible to successfully estimate the collection parameter λ_w of information based models. One can note that this estimation technique uses only statistical data from the collection and does not need any relevance judgment to estimate λ_w. But at the same time the whole estimation method was developed solely for the collection parameter and it is

not applicable to estimate any other parameters. As for example, the free parameter c of the models used here. In this work we used a cross validation method to tune this parameter c along with other free parameters (b and k_1 of BM25 and μ of LM). This tuning approach uses relevance judgment for selecting the best parameter on the training set before applying that parameter to the test set. But this situation is unrealistic in the sense that under practical situations relevance judgment is seldom available for a real life collection. Moreover, even for the test collections, preparing relevance judgment needs manual effort and hence is very costly.

In the next chapter we will address precisely these drawbacks and we will try to device a technique that:

(a) is applicable to predict any free parameter of any IR model, and

(b) does not use the relevance judgment on the querying collection, instead it uses relevance judgment already available on some other collection potentially in different languages.

3

QUERY-BASED TRANSFER LEARNING OF STANDARD IR MODEL PARAMETERS

This chapter presents a new method to predict the values of the parameters of standard IR models on new collections (target collection) without any relevance judgments, by using already available relevance judgments on some other collections (source collections). This learning is achieved by first mapping queries (with and without relevance judgments, from different collections, potentially in different languages) into a common space corresponding to the feature space of a positive definite symmetric kernel between queries. Standard kernel regression functions, as kernel support regression, can then be used to learn a mapping between queries lying in the kernel induced feature space and IR parameter values. We furthermore introduce a simple positive definite kernel from which one can develop more complex kernels. Our experiments, conducted on standard English and Indian IR collections, show (a) that there is no significant difference between the different kernels considered, (b) that 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 judgments (even though no target relevance judgments are used in the method we propose), and (c) that our approach is collection and language independent and can be used to efficiently tune, query by query, standard IR models to new collections, potentially written in different languages.

3.1 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 3.7, default results are reasonable on several collections, however not on all of them). As discussed in Section 1.3.1 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. Though this does not necessarily mean that the IR model will change, but that it may change and thus needs to be given the opportunity to get adapted.

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 [Robertson and Zaragoza, 2009], language models (LM) [Ponte and Croft, 1998] and log-logistic information-based models (LGD) [Clinchant and Gaussier, 2010]. 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 in this chapter is the first research question asked in Section 1.6 and 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?*

One solution, as explained in the previous chapter, is to use various statistical estimation techniques e.g. maximum likelihood estimation or expectation maximization. But these estimation techniques generally take different forms for different parameters and frameworks and they are tailored and thus suitable to estimate a particular parameter under a particular framework. As for example, technique developed to estimate the smoothing parameter μ in language models with Dirichlet prior in [Zhai and Lafferty, 2001] cannot be applied to estimate parameters of BM25. Same applies to SPL_{GM} and other information models developed in the previous chapter.

However, we presents here a novel method that can predict parameters of any IR function for any given query even if no relevance judgments are available for such queries. The only necessary knowledge about unseen queries comes from unlabeled text corpora and a set of available queries for which the relevance judgment information exists on some source collections. Our method also relates to different research fields as *model adaptation* and *transfer learning*, where a model learned on a source collection is biased towards a target collection or where relevance information from a source collection is propagated to a target collection. *Model adaptation* and *transfer learning* are, in many studies, considered to be the same; we adopt this viewpoint here even though the term transfer learning is more appropriate in our case. Our parameter prediction approaches map queries from different collections into common vector spaces. In those spaces regression functions can be efficiently learned on training data obtained using relevance information from source. Then the same regression function can predict parameters for the target queries which are also mapped in the same common vector space. Three key elements of our approach are:

- Unlike pointwise learning to rank methods for IR that directly learn a ranking model, our approach aims at estimating the value of the parameters of a standard IR model;
- The method proposed allows to obtain parameter values for a single query, so that IR models are optimized per query on the new collection;
- The representations used are language-independent, so that parameter values can be estimated for new collections, in new languages.

We performed rigorous experimentation using nine different English collections from TREC and CLEF campaigns and two non-english Indian language collections from FIRE campaign. We found that our method not only outperforms the models with default parameter values, in some cases it provides better results than the ones one can obtain by tuning the parameters of the model with relevance judgments. This latter result may be due to the fact that the prediction is made in a continuous space whereas standard tuning procedures use a fixed set of discrete parameter values. Our learning model does not require any manually defined hyperparameters and unseen queries are mapped using a positive-definite kernel over the space induced by queries for which relevance judgments exist on some source collections. Using this set, we then use a regression model to learn the association between the kernel-based feature representation of queries and their optimized associated parameter using the relevance judgments. On the top of this model, we also investigate how different IR scoring functions with their predicted parameter values can be combined using two existing transfer learning strategies.

The remainder of the chapter is organized as follows. We first discuss related work in Section 3.2. Then we explore why parameter prediction is required (Section 3.3. The framework and the approach developed for learning the parameter(s) of standard IR models on collections with no relevance judgments, using known collections with relevance judgments are described in Sections 3.4 and 3.5 respectively. We then illustrate several aspects of the method we propose and its effectiveness in Sections 3.6 and 3.7 on several collections, prior to conclude (Section 3.8).

3.2 RELATED WORK

IR models are generally defined with some free parameters, for example the b and k for BM25 [Robertson and Walker, 1994], the Dirichlet smoothing coefficient μ for language models [Jelinek and Marcer, 1980] and parameter c for LGD [Clinchant and Gaussier, 2010]. These parameters are query and collection dependent and hence their tuning is unavoidable to achieve good performance within the collection. [Zhai and Lafferty, 2001] explained the necessity of empirical tuning of the parameters for traditional models like vector space model or BM. Due to the absence of a direct combined modeling or representation of queries and documents, it becomes extremely hard for the model to incorporate special characteristics through the parameters which address them. To this end language

models have the advantage of directly representing both queries and documents through statistical language models. Harnessing this representation [Zhai and Lafferty, 2001] proposed an automated "leave-one-out" likelihood method to estimate Dirichlet prior smoothing parameter μ. Effectiveness of pseudo-relevance feedback over language model framework is examined by [Tao and Zhai, 2006] where a mixture model is proposed and regularized expectation maximization method is used to estimate the parameter which determines the balance between the original query against its feedback based counterpart. Both the methods are applicable across the collection meaning a single parameter value for all the queries in the collection. [Lv and Zhai, 2009] used an logistic regression model to predict the balance parameter of pseudo-relevance feedback for each query separately and observed significant improvement. This proves the effectiveness of querywise parameter setting.

However, 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 w.r.t the evaluation measure considered. This greedy search has been found to be competitive compared to simple learning strategies that optimize some differentiable IR measures [Taylor et al., 2006]. This said, the complexity of the greedy search increases exponentially with respect to the number of free-parameters making the search unfeasible in some extreme cases. Any method of tuning that utilize relevance judgment information is very much unrealistic in many real retrieval scenarios [Bennett et al., 2008]. Furthermore, the use of relevance judgments has motivated a large amount of research in the learning to rank framework [Liu, 2009].

But generating relevance judgments on a new test collection mostly involves manual effort and assigning relevance judgments manually, for even a small set of queries, is a very costly task [Carterette, 2007]. Some works directly considered the problem of unsupervised parameter estimation [Zhai and Lafferty, 2001, Tao and Zhai, 2006] which do not need any relevance information and relies only on statistical data available from the collection. But these methods were developed under the language model framework with a probabilistic view over the generation of words. These techniques are thus not applicable to other non-probabilistic IR models.

Some other works utilized a paradigm called *transductive transfer learning* which deals with the problem of using knowledge from some existing source domain, mostly in the form of labeled instances, to either learn or improve the learning of a prediction function for a target domain with no labeled data. This paradigm is developed both for classification and regression tasks. The two main approaches to transductive transfer learning are either instance-based [Sugiyama et al., 2008] or feature-based (also referred to as domain adaptation in the literature) [Blitzer et al., 2008]. In instance-based approaches, the optimization of the objective function for the target domain is generally carried out by adding different penalty values to each labeled instance in the source domain. For

example, [Huang et al., 2009] proposed a kernel-mean matching algorithm to learn these penalty values by matching the means between the source and the target domain data in a reproducing-kernel Hilbert space. [Sugiyama et al., 2008] proposed an algorithm which directly estimates these penalty values by minimizing the Kullback-Leibler divergence between the probability distributions of instances in the source and the target domains. On the other hand, the overall aim in feature-based approaches is to propagate, in some manner, the label information from the source to the target domain. For example, [Dai et al., 2007] proposed a co-clustering algorithm to diffuse the label information across different domains, and [Xing et al., 2008] studied the cross-domain transfer learning problem under a spectral classification framework in which the objective function takes into account the consistency between the supervision provided in the source domain and the structure of the target domain. A review of the work, prior to 2010, done in transfer learning for classification and regression may be found in [Pan and Yang, 2010].

Several studies have been done to adapt cross domain approaches to learning to rank framework [Chen et al., 2008a, Chen et al., 2010, Gao et al., 2010]. But in this chapter we will concentrate on using transfer learning to predict parameter values for different standard IR models and to the best of our knowledge this study is the first to do so.

Lastly, our approach bears some similarities with the study described in [Lv and Zhai, 2009] aiming at learning, on a query by query basis, the parameter of feedback models. The main difference between this study and ours lies in the fact that we are interested in the parameters of IR models in a standard *ad hoc* retrieval scenario, whereas [Lv and Zhai, 2009] focuses on the feedback coefficient in a typical relevance feedback scenario. This difference entails that the representation spaces used in each case radically differ: we use feature spaces induced by positive definite kernels defined on a new query representation whereas [Lv and Zhai, 2009] uses an explicit feature space capturing the properties of the query, the documents in the feedback set and the relation between the two.

3.3 BEYOND DEFAULT PARAMETERS

Our goal here is to predict the parameters of standard IR models on the target collection by transferring the relevance information known on source collection. Two questions that directly arise are:

1. Is it worth doing so, or, in other words, is it possible to improve upon the default values usually used in IR models?
2. Should one learn global values that will be used for all target queries or should one learn different values for each query?

To answer these questions, we computed the mean average precision (`MAP`) of three standard IR models namely `BM25`, the Dirichlet language model (denoted by `LM`) and

the log-logistic distribution of information-based model family (denoted by LGD) on four collections namely TREC-7, TREC-8, WT10G and GOV2 (details of these collection are given in Section 3.6.1) with three different settings. Like all standard available collections, we here consider that a set of test queries are provided with a collection. Three different settings are:

Default: This setting uses fixed default parameter values of each model on all the test collections and is most commonly used setting. These default values, as explained in Section 1.3.1, are observed to perform well in various experiments and thus assumed to be reasonably well also on a new collection.

OptGlobal: This setting selects a single parameter value for all the target queries in the target collection that provides the highest MAP on that collection. This strategy searches for the best parameter value from a range or a set of values. Clearly this makes use of the relevance judgments on the target collections. The associated MAP corresponds to the upper bound of methods aiming at finding the best parameter value globally for all queries.

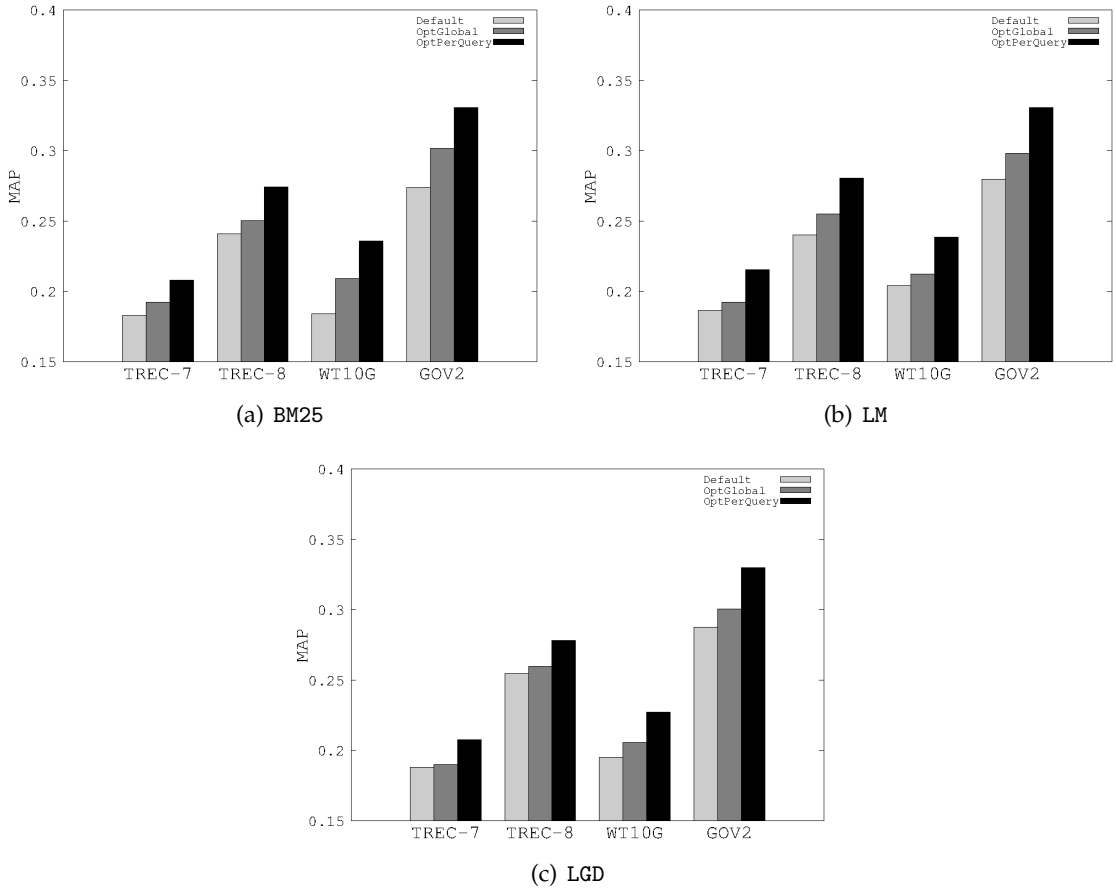
OptPerQuery: This setting selects a parameter value for each query in the query set of the target collection that provides the highest average precision for the query under consideration. Here also the best parameter value is searched from a range or a set of values. Again as before, this makes use of the relevance judgments on the target collection. The mean of all the average precision obtained for every query in this way yields a MAP value that corresponds to the upper bound of methods aiming at finding the best parameter value individually per query.

The results obtained are displayed in Figure 7. As one can note, the difference between default values and optimized ones varies according to the models and collections 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 (OptGlobal). It is more important when the optimization is performed query by query (OptPerQuery), even though the value varies from one collection to another: around $2 - 3\%$ for all models on TREC-7,8, around $5 - 6\%$ for all models on GOV2 and WT10G, the difference being however less marked for LGD. These results show that:

1. 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,

2. higher gains can be expected by optimizing the parameters per query.

The sets of values used to search and select the best parameter are described in 3.6, along with other experimental details.



(a) BM25 (b) LM

(c) LGD

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

3.4 FRAMEWORK

The situation we are interested in consists of a source collection $\mathcal{C}^s$, 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 $\mathcal{C}^t$, composed of a set of documents $\mathcal{D}^t$ and a set of m queries $\mathcal{Q}^t = \{q_1^t, \ldots, q_m^t\}$. Each query q, in any collection, is constituted by a set of terms $q = \{w_1^q, \ldots, w_{|q|}^q\}$.

The methodology we follow thus relies on the following steps:

1. For all IR models and using the selection method OptPerQuery (Section 3.3) on the source collection, we first create the vectors of optimized parameter values of each query in $\mathcal{Q}^s$, denoted by $\mathbf{c}_{opt}^s = (c_{q_1^s}, \ldots, c_{q_n^s})^\top$, where $^\top$ denotes the transpose

operator. Thus each value $c_{q_i^s} \in \mathbf{c}^s_{opt}$ corresponds to the parameter value that produces best MAP for query $q_i^s \in \mathcal{Q}^s$.

2. A regression model is learned using the association between queries in $\mathcal{Q}^s$ and $\mathbf{c}^s_{opt}$.

3. Finally the learned regression model is used to predict a parameter value for each query in $\mathcal{Q}^t$.

Figure 8 roughly illustrates this procedure.

The last steps of the above procedure however require that the queries in $\mathcal{Q}^s$ and $\mathcal{Q}^t$ lie in a common space. For this purpose, it is not possible to directly rely on the standard representation of queries as the vocabulary used in the different collections can be radically different. Indeed, as mentioned above, one may be interested in predicting the parameter values of standard IR models on collections written in different languages, with potentially no (or little) vocabulary overlap with the collections encountered so far. Moreover, different queries may contain different number of terms, thus disabling their representation directly in a fixed dimension vector space. We explore the details of this problem in next section.

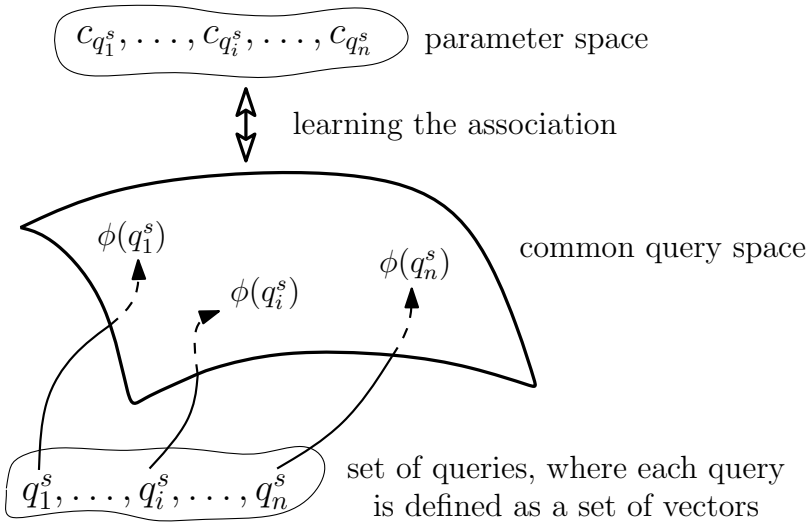


Figure 8: Each target query is mapped in the common vector space and an associated parameter of the IR model is predicted using the learned association model.

3.5 LEARNING IR MODEL PARAMETERS

As seen in the previous section it is not possible here to directly represent queries by the terms they contain as different collections rely on different vocabularies, potentially from different languages and may content different number of terms. In order to bypass this problem, in this section we first present a query representation using the statistical data of

different terms in the query (Section 3.5.1). This enables the query to be represented as a set of vectors, where each vector represents a single term contained in the query. Then we describe the mapping of these queries (either source or target) in a common vector space. We will consider two such common vector spaces based on kernel and similarity measures between queries. We will also discuss some of the regression models that can be used in these spaces. These elements are respectively described in Sections 3.5.2 and 3.5.3.

3.5.1 *Query Representation*

Each query term $w^q \in Q$ is represented as a 4-dimensional vector:

$$\mathbf{w^q} = (idf(w^q), \mu(w^q), \sigma(w^q), sk(w^q)) \in \mathbb{R}^4 \tag{3.5.1}$$

where $idf(w^q)$ denote the inverse document frequency of w^q, and $\mu(w^q)$, $\sigma(w^q)$ and $sk(w^q)$ are the empirical estimates of the first (mean), second (standard deviation) and third (skewness) moments of term frequency distribution of w^q on the collection associated with q. The $idf(w^q)$ can be any standard function to calculate the inverse document frequency of w^q. In the present study we have assumed $idf(w^q) = \frac{\mathcal{N}}{\mathcal{N}_w}$, where $\mathcal{N}_w$ is the document frequency of w^q in the collection and $\mathcal{N}$ is the number of documents in the collection.

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. It is common practice in IR to represent a term with its term frequency and inverse document frequency. Following the same path, here also the term w^q is represented with its $idf(w^q)$ and three moments which essentially summarizes its term frequency distribution. Thus the vector of Equation 3.5.1 is an acceptable way of representing the word w^q.

Now each term is statistically summarized within a vector and from that, each query q is represented as a set of such term vectors:

$$q = \{\mathbf{w^q}_1, \ldots, \mathbf{w^q}_{|q|}\} \tag{3.5.2}$$

For example, starting from the query (extracted from the `WT10G` collection described in Section 3.6.1) q:*lava lamps*, one gets as representative vectors for *lava* and *lamps*:

$$\begin{aligned} \text{lava} &: (6.7762, 0.0012, 0.0050, 137.0581)^\top \\ \text{lamps} &: (5.6177, 0.0059, 0.0355, 81.8577)^\top \end{aligned}$$

where $^\top$ denotes the transpose and the first three moments are calculated over normalized TF scores. For readability reasons, all elements are truncated after the fourth decimal. The final representation for the query is then:

$$q = \{(6.7762, 0.0012, 0.0050, 137.0581)^\top, (5.6177, 0.0059, 0.0355, 81.8577)^\top\}$$

3.5.2 *Mapping Queries into a Common Feature Space Through PDS Kernels*

Though each query q now has an enriched representation (Eq. 3.5.2), it is still not possible to learn the association between queries and the optimized parameter values. Because queries are defined as a set of term vectors and they do not lie in a fixed vector space. To overcome this problem, we map both source and target queries into common vector spaces. We introduce here a collection-independent vector representation of queries through the use of kernels, which opens the door to kernel regression methods as kernel Support Vector Regression (kernel SVR) [Vapnik, 2000] or Kernel Ridge Regression (KRR) [Saunders et al., 1998]. Among these methods, we focus in this paper on kernel SVR, as this technology has been shown to perform well in practice.

We denote by κ a positive definite symmetric (PDS) kernel taking as input a pair of queries represented, as described above, by a set of 4-dimensional word vectors (Eq. 3.5.2) and taking values in $\mathbb{R}_+$. Any such PDS kernel is associated with a mapping $\boldsymbol{\phi}$ that maps queries into a vector space $\mathcal{F} \subseteq \mathbb{R}^p$ (with p potentially infinite) such that: $\kappa(q, q') = \langle \boldsymbol{\phi}(q), \boldsymbol{\phi}(q') \rangle$, where $\langle ., . \rangle$ denotes the dot product. As long as κ is a valid PDS kernel, $\mathcal{F}$ defines a feature (and a vector) space common to all queries and in which a regression function can be learned, through kernel regression methods as kernel SVR. Note that the mapping $\boldsymbol{\phi}$ and the feature space $\mathcal{F}$ may not be explicitly defined; all is required is the knowledge of κ, and this knowledge is sufficient to learn a regression function in $\mathcal{F}$.

To formalize let H denote the family of linear functions in the induced feature space $\mathcal{F}$:

$$H = \{h_{\boldsymbol{\omega}} : q \mapsto \boldsymbol{\omega} \cdot \boldsymbol{\phi}(q) + b,\ \boldsymbol{\omega} \in \mathbb{R}^p,\ b \in \mathbb{R}\}$$

where $\boldsymbol{\phi}$ denotes the mapping, from the input space to the feature space, associated with a positive definite symmetric (PDS) kernel κ. $\boldsymbol{\phi}(q)$ thus represents the vector representation of q in $\mathcal{F}$ and the kernel κ defines a similarity between queries in the input space that implicitly represents a dot product between the vector representation of queries. The optimization problem for kernel SVR is usually defined as:

$$\min_{\boldsymbol{\omega}, b} \frac{1}{2}||\boldsymbol{\omega}||^2 + C \sum_{i=1}^{n} |c_{q_i^s} - (\langle \boldsymbol{\omega}, \boldsymbol{\phi}(q_i^s) \rangle + b)|_\epsilon$$

where $\boldsymbol{\omega} \in \mathcal{F}$ and $b \in \mathbb{R}$ are the weights to be learned and $|\ |_\epsilon$ represents the ϵ-insensitive loss which for $\epsilon > 0$ is defined as: $|y - y'|_\epsilon = \max(0, |y - y'| - \epsilon)$. This loss function has been shown to lead to sparse solutions, with few support vectors. As one can not always specify the mapping $\boldsymbol{\phi}$, a dual formulation, involving the kernel κ is usually adopted (see for example [Mohri et al., 2012]). Thus the optimization problem takes the following form (see for example [Mohri et al., 2012]):

KSVR-opt:

$$\max_{\boldsymbol{\alpha}, \boldsymbol{\alpha}'} -\epsilon(\boldsymbol{\alpha}' + \boldsymbol{\alpha})^\top \mathbf{1} + (\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{c}^s - \frac{1}{2}(\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{K}(\boldsymbol{\alpha}' - \boldsymbol{\alpha})$$

$$\text{s. t. } 0 \leq \alpha_i,\ \alpha_i' \leq C (1 \leq i \leq n),\ (\boldsymbol{\alpha}' - \boldsymbol{\alpha})^\top \mathbf{1} = 0$$

where $\mathbf{c}^s_{opt} \in \mathbb{R}^n$ is the vector of optimized parameter values for the source queries, $\mathbf{1} \in \mathbb{R}^n$ is a vector containing only 1s, $\mathbf{K}$ is the kernel (or Gram) matrix ($\mathbf{K}_{i,j} = \boldsymbol{\kappa}(q^s_i, q^s_j)$, $1 \leq i, j \leq n$) and $\boldsymbol{\alpha}, \boldsymbol{\alpha}' \in \mathbb{R}^n$ are the parameters to be learned and are such that either α_i or α'_i is non null (this happens if q^s_i is a support vector), or they are both null (if q^s_i 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:

$$c_{q^t} = \sum_{i=1}^{n} (\alpha'_i - \alpha_i) \boldsymbol{\kappa}(q^s_i, q^t) + b \tag{3.5.3}$$

the offset b being defined for any support vector q^s_l by:

$$b = \sum_{i=1}^{n} (\alpha_i - \alpha'_i) \boldsymbol{\kappa}(q^s_i, q^s_l) + c_{q^s_l} + \epsilon \tag{3.5.4}$$

As one can note, in the above formulation, source and target queries are mapped into a common vector space through the mapping $\boldsymbol{\phi}$ associated to the PDS kernel used. In Section 3.5.4, we will describe a simple yet effective PDS kernel that we retained for our experiments and for which it is possible to explicitly define the mapping $\phi(.)$. But at the same time it is important to note that any PDS kernel over the representation for queries we have defined above can be used.

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:
   (a) Compute the optimal value c_{q^s} of the free parameter for each query q^s in $\mathcal{Q}^s$ (*e.g.* using a line search);
   (b) Solve the KSVR-opt problem above with any convex QP solver.
2. Prediction:
   (a) For each query q^t, compute c_{q^t} through Equation 3.5.3 with b defined by Equation 3.5.4.

3.5.3 *Mapping Queries into a Common Similarity-based Query Space*

In the previous parameter learning approach, the mapping is embedded in the kernel based optimization problem. A more conventional way to tackle the parameter learning problem is to explicitly represent each query in a vector space, and then to learn the

association between the vectors of source queries and the parameter values through a classical regression model, such as random forests.

To do so, we propose a simple mapping technique that relies on a collection $\mathcal{C}^r$, composed of a set of documents $\mathcal{D}^r$ and a set of p queries $\mathcal{Q}^r = \{q_1^r, \ldots, q_p^r\}$. We now define a similarity measure, denoted as $sim(.,.)$ in the following, between two queries each defined as a set of word vectors (Section 3.5.1). In this case, the mapping of any query q would be:

$$\forall q, \boldsymbol{\phi}_r(q) = (sim(q, q_1^r), \ldots, sim(q, q_p^r))^\top \in \mathbb{R}^p \tag{3.5.5}$$

The process for learning and predicting the parameter value can then be summarized as follows:

For each free parameter of the standard IR model under consideration:

1. Training step:
   (a) For each query q^s in $\mathcal{Q}^s$, compute the optimal value c_{q^s} of the free parameter for the IR model of interest;
   (b) Represent the query using a query set $\mathcal{Q}^r$ and a similarity function $sim(.,.)$ as in Equation 3.5.5;
   (c) Learn the association between the new representation of queries in $\mathcal{Q}^s$, given by $\{\boldsymbol{\phi}_r(q_1^s), \ldots, \boldsymbol{\phi}_r(q_n^s)\}$ and their corresponding optimal parameter values, $\{c_{q_1^s}, \ldots, c_{q_n^s}\}$, using a regression model $\mathcal{M} : \mathbb{R}^m \to \mathbb{R}_+$.
2. Prediction:
   (a) For each query q^t in the target collection $\mathcal{Q}^t$, use the same query set $\mathcal{Q}^r$ and the similarity function $sim(.,.)$ as in the previous step to represent q^t; $\boldsymbol{\phi}_r(q^t)$ Equation 3.5.5;
   (c) Apply the learned model $\mathcal{M}$ on $\boldsymbol{\phi}_r(q^t)$ to predict the free parameter of the IR model for q^t.

In the next section we will exhibit a similarity function, which turns out to be a PDS kernel as well.

3.5.4 *Simple PDS Kernels*

We introduce in this section several kernels that can be used within the framework defined before. Let us recall that the representation of queries we are starting from is a set of 4-dimensional vectors: $q = \{\mathbf{w^q}_1, \ldots, \mathbf{w^q}_{|q|}\}$, with $\mathbf{w^q} = (idf(w^q), \mu(w^q), \sigma(w^q), sk(w^q)) \in \mathbb{R}^4$. On this representation, we first define the following similarity measure that computes

the normalized sum of all pairwise dot products between the vectors of two queries q and $q^{'}$:

$$\kappa_{\text{all}}(q, q^{'}) = \frac{1}{|q|}\frac{1}{|q^{'}|}\sum_{i=1}^{|q|}\sum_{j=1}^{|q^{'}|}\left\langle \mathbf{w^q}_i, \mathbf{w}_j^{q^{'}} \right\rangle \tag{3.5.6}$$

κ_{all} defines a similarity measure between queries in the sense that κ_{all} will likely be higher for queries containing words with similar distributions in the collections considered than for queries containing words with very different distributions. The subscript "all" in κ_{all} denotes the fact that all pairs of words, from the two queries, are compared. κ_{all} can thus be used to map queries into the common similarity-based query space introduced previously. It is important to note that we do not need here a precise definition of what a similarity measure is, as the development presented in Section 3.5.3 is valid for any real-valued function defined over a pair of queries.

Note that in the definition of κ_{all} only the word vectors of the queries are required and there is no restrictions that they have to be from the same collection. This enables us to compare two queries from two different collections may be even in different languages.

The following property shows that κ_{all} also defines a valid PDS kernel, so that it can also be used to map queries into a common feature space as discussed in Section 3.5.2.

Property 1. *κ_{all} is a PDS kernel.*

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

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

Therefore, κ_{all} corresponds to a dot product in the 4-dimensional vector space introduced earlier; it thus defines a PDS kernel. □

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

$$\boldsymbol{\phi}(q) = \sum_{i=1}^{|q|}\frac{\mathbf{w}_i^q}{|q|} \tag{3.5.7}$$

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

$$\kappa_{\text{all}}(q, q') = \langle \boldsymbol{\phi}(q), \boldsymbol{\phi}(q') \rangle \tag{3.5.8}$$

We will use this simple PDS kernel in our experiments, both for the mapping into a common feature space (Eq. 3.5.8)and for the mapping into a common similarity-based query space (Eq. 3.5.7).

From the form given in Equation 3.5.8, one can further define new kernels, by substituting the dot product by any valid PDS kernel. We consider here homogeneous polynomial kernels and Gaussian kernels, widely used in text processing, leading to:

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

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

The different kernels (κ_{all}, $\kappa_{\text{poly-}\delta}$, and $\kappa_{\text{gau-}\sigma}$) will be used for solving the KSVR-opt problem and for predicting new parameter values as defined by the regression function in Equation 3.5.3.

3.6 EXPERIMENTAL SETUP

We present in this section experimental details including description of collections used (Section 3.6.1), IR models used and associated different free parameters (Section 3.6.2), different modes of selecting a proper parameter value considered in this work (Section 3.6.3 and finally the implementation details of the regression method proposed (Section 3.6.4.

3.6.1 *Collections*

Experiments are performed on eleven standard IR collections from three different languages, namely English, and two Indian languages Hindi and Bengali. It is interesting to note that these three languages not only have different vocabulary, but their scripts are also different. Basic statistics on these collections are provided in Table 6.

We used nine English collections here, one is from CLEF[1] and remaining eight are from TREC[2]. Among these collection, TREC-6, TREC-7 and TREC-8 use the same document sets (TREC disks 4 and 5) but different query sets, whereas WT10G, TREC-3,4,5, CLEF-3 and GOV2 use unique document sets and unique query sets. We appended TREC-9 Web and TREC-10 Web tracks and used the combined track to experiment with WT10G. Similarly TREC-2004 Terabyte and TREC-2005 Terabyte tracks are combined and used for experimenting with GOV2.

Two non-English collections used here are from latest corpus available in FIRE[3] campaign. Among various other Indian language collections we used Bengali (denoted by FIRE-BN

[1] www.clef-campaign.org

[2] trec.nist.gov

[3] www.isical.ac.in/~clia/

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 TREC-7 TREC-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 6: Statistics of various collections used in our experiments, sorted by size.

and Hindi (denoted by FIRE-HN collections. These are used to show the language independency of our approach proposed here. Hence, any collection in any other language can be used in this purpose.

Standard preprocessing steps to create an index are performed using Terrier IR platform v3.5 (terrier.org) which include stemming using Porter stemmer and removing stopwords 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. As we will do a comparative study, not using any stemmer on FIRE collections will not affect our purposes.

In most of our experiments, we considered CLEF-3, TREC-3,4,5,6 as source collections, and used TREC-7,8 as well as 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 source 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 collections. For the similarity-based regression model we used both the FIRE collections as representation set $\mathcal{Q}^r$ and all source and target queries are represented in a 100-dimensional vector space (because combined query set of two FIRE collections contains 100 queries in total).

Results are evaluated using the mean average precision MAP and precision at 10 documents P@10 measures (relevance judgments on the target collections, are just used for evaluating the models). We furthermore use the Wilcoxon statistical test with a p-value of $p = 0.05$ to assess whether the difference between two methods is significant or not.

3.6.2 *Standard IR Models and Free Parameters*

We used three different, widely used, IR ranking models, namely Okapi BM25 (denoted by BM25) [Robertson and Walker, 1994], the language model with Dirichlet smoothing (denoted by LM) [Jelinek and Marcer, 1980], and the log-logistic information-based model (denoted by LGD) [Clinchant and Gaussier, 2010] from the Divergence from Randomness family. These models contain free parameters and goal here is to identify most appropriate value for these parameters.

BM25 has three free parameters, b, k_1 and k_3, whereas the method proposed here can predict only one parameter at its current stage. In the experiments we just considered the parameter b and kept the two other parameters fixed to their default values ($k_1 = 1.2$ and $k_3 = 8.0$). The parameter k_3 is used in normalization of t^q_w, the number of occurrences of the query term in the query itself (Eq. 1.2.1). If $t^q_w = 1$, which is almost always the case in the short queries provided by TREC and CLEF collections, then the whole normalized t^d_w weighting term becomes a constant and thus irrelevant in the ranking. Among k_1 and b, we are particularly interested in b, because it controls the effect of document length in the term frequency normalization, the role similar to the parameter c in LGD.

3.6.3 *Parameter Values*

Our goal is to compare the performance of standard IR models described in the previous section with their default parameter values against the values predicted by the proposed regression approach on target collections. We also wish to position our method with respect to other popular parameter tuning methods which uses relevance information on the target collection. Thus following parameter value settings are used here:

- The default value setting offers a fixed value for the parameters and performs fairly well on all occasions. In our experimentation, we use values provided as default in Terrier IR framework: $b = 0.75$ for BM25, $\mu = 2500.0$ for LM and $c = 1.0$ for LGD.

- For tuning the parameters using relevance judgment the query set $\mathcal{Q}^t$ is split into two disjoint subsets each containing 50% of the queries selected randomly. Among these two subsets one is associated with the relevance judgments and are used to identify the best performing parameter value for all IR models from a predefined set of values. This sets of predefined parameter values for three IR models are given in Table 7. Note that the sets of values are the same as the ones used in Chapter 2 (Table 3). This step simply consists in selecting the parameter value that yields the best MAP value on the first subset of queries. Then the second subset is used for testing the models with identified parameter values. We performed 10 such random splits of the target queries and hence denote this method as 10split.

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 7: Set of values considered for the free parameter of the different IR models. These values are used to find the optimal parameter values on the source collection to construct the training set, and on the target collection (a) to construct the global and per query upper bounds (Section 3.3) and (b) to find the optimal parameter value from a subset of the target queries (method `10split`).

- The parameters are predicted using the regression model proposed here. Details of its implementation are given in the next section.

- Parameter values are optimized either globally for all the queries in the query set (`OptGlobal`) or per query basis (`OptPerQuery`) as described in Section 3.3. These optimization methods use relevance judgments and provide upper bounds of performance achievable by changing the free parameter values. Both `OptGlobal` and `OptPerQuery` select best parameter values from the sets shown in Table 7.

3.6.4 *Implementation of Kernel Regression Model*

Each query q here is represented by the 4-dimensional vector $\phi(q)$ which expression is given in Equation 3.5.7. First step is to compute the vectors $\mathbf{w^q}$ for all $w^q \in q$. This can be done during index construction of the collection. While indexing a collection $\mathcal{C}$, for each word in the dictionary and all the documents where the word occurs, corresponding term frequency scores along with the document frequency of the word are computed and stored in the postings lists. At the same time the components of the vectors (idf and first three moments of tf scores) can be calculated and stored in the same postings list. Hence for a query word the corresponding vector can be accessed from its index and needs not to be calculated on the spot every time a query is issued and thus incurs only a constant memory access time.

3.6.4.1 *Training Data*

The regression models require training data to learn an association between the queries and corresponding optimal parameter values. Hence we need to compute the vectors $\mathbf{c}^s_{opt}$ of optimized parameter values on the source collections. In a source collection $\mathcal{C}^s$, suppose q^s_i is a query in the set $\mathcal{Q}^s$. To determine the best parameter value $c^s_{q^s_i}$ for q^s_i of one of the models (mentioned in Section 3.6.2), retrieval process is run for q^s_i on $\mathcal{C}^s$ using each of the parameter values in the sets provided in Table 7. In each retrieval run the result is evaluated in terms of average precision. The parameter value for which we obtain

the highest average precision is selected as the optimized parameter value $c^s_{q^s_i}$ for q^s_i. The relevance judgment on $\mathcal{C}^s$ is used for these evaluations. For retrieval step we use Terrier as all models as well as evoluation modules are already implemented. We used shell scripts to automatically supply all the parameter values to Terrier and to determine the best average precision.

Once we have optimized parameters for all the queries in $\mathcal{Q}^s$, that is the vector $\mathbf{c}^s_{opt}$, the training set for kernel regression model (Section 3.5.2) can be built just by associating a query $q_i \in \mathcal{Q}^s$ with the corresponding best parameter value $c^s_{q^s_i}$ of the vector $\mathbf{c}^s_{opt}$. But for similarity-based regression model (Section 3.5.3) first the similarity measure $\kappa_{\text{all}}(.,.)$ (Eq. 3.5.6) is employed to represent queries with respect to a given representation set $\mathcal{Q}^r$ (Eq. 3.5.5). In this representation each query is now a $|q^r|$-domensional vector. Now training data for similarity-based regressor is built by associating a query $q_i \in \mathcal{Q}^s$ represented in the similarity-based space with the corresponding best parameter value $c^s_{q^s_i}$ of the vector $\mathbf{c}^s_{opt}$.

In general a collection contains very few queries (50 to 100) in its query set (Table 6) which is very few for learning a regressor. To alleviate this problem, we consider multiple collections as sources (mentioned in Section 3.6.1). For each source collection we independently calculate the vector $\mathbf{c}^s_{opt}$ and the corresponding training data for all the queries in the corresponding query set using the method described above. Then we merge all these individial training data to obtain a larger one. This does not affect the overall learning process as each learning example, that is the query and the associated parameter values are obtained independently from each other and on the corresponding collection only.

3.6.4.2 *Learning a Regressor*

As mentioned in Section 3.5.2, we mainly focus on kernel support vector regression as our kernel regression method. We denote it as `k-SVR`. LIBSVM[4] implementation of ϵ-`SVR` is used. The ϵ is fixed to 0.1. The hyperparameter C is found by cross-validation on the training set and the LIBSVM implementation includes this cross-validation module.

For our second approach presented in Section 3.5.3, we used random forests as the regression model (denoted by `RF` in the following). Random forest regressor is used as this is a similarity measure based approach and the used similarity measure may not necessarily be a kernel. Here we used Python Scikit-learn[5] implementation of random forest regressor. Different hyperparameters of random forest regressor are set by cross-validation. These hyperparameters are, number of trees in the forest (`n_estimators`), number of features to consider when looking for the best split (`max_features`), maximum depth of the tree till which the nodes are to be expanded (`max_depth`) and minimum number of samples required to split an internal node (`min_samples_split`). We used a python script to implement this cross-validation step.

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

[5] `http://scikit-learn.org`

For experimentation we consider three IR models each having a single free parameter (Section 3.6.2) totalling three parameters. For each of these parameters, a separate training set is prepared and eventually three separate regression models are learned, thus providing a dedicated regressor to predict the values of each of the parameter.

3.6.4.3 *Prediction*

Once three separate regressors are learned (either `k-SVR` or `RF`) they are used for predicting the parameter values for the target queries. For a target query q^t the corresponding word vectors can be accessed from the index of the corresponding target collection. Then the query representation $q^t = \{w^{q^1}, \ldots, w^{q^{|q|}}\}$ is passed to the kernel regressor `k-SVR` which predicts the parameter value. For the similarity-based regressor the query q^t is first represented in the similarity-based common space using similarity measure defined in Equation 3.5.6 with respect to the representation set $\mathcal{Q}^r$ (Eq. 3.5.5). This representation is then passed to `RF` to get a prediction of the parameter value.

This predicted value is used for the retrieval run using Terrier for q^t on the target collection $\mathcal{C}^t$. This procedure is followed for all the queries in the query set $\mathcal{Q}^t$. The results are merged and evaluated using Terrier's evaluation module. `MAP` and `P@10` are reported to evaluate the performance of the predicted parameters.

Note that relevance judgments on the target is only used for evaluating the final performance of the predicted parameters. For parameter prediction step itself no relevance information is used.

3.7 RESULTS

Here we first assess the effect of different kernels used in the regression model in Section 3.7.1. In Section 3.7.2 we do a comparative study among the two methods proposed here namely `k-SVR` and `RF`. Then we inspect whether the method we propose improves over the one based on default parameter values (Section 3.7.3) and how it performs against popular method of parameter tuning and two upper bounds introduced in Section 3.3 (Section 3.7.4). We also consider how our method performs in terms of extra time burden it incurs (Section 3.7.5).Then we illustrate its use on collections written in different languages (Section 3.7.6). Lastly, in Section 3.7.7, we investigate the behavior of the method with respect to the number of queries used for training.

3.7.1 *Different Kernels*

We first compare the method proposed with the different kernels introduced in Section 3.5: κ_{all}, $\kappa_{\text{poly}-\delta}$ and $\kappa_{\text{gau-}\sigma}$. We tried a varied range of values for the parameters δ and σ. For $\kappa_{\text{poly}-\delta}$ we tried with $\delta = 2, 3, 4$, that is second, third and fourth degree polynomial

kernel. For $\kappa_{\text{gau-}\sigma}$ we tried with $\sigma = 10, 50, 70, 100$ which is a considerably large range of values. The results are reported in Table 8 in terms of MAP and P@10.

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	κ_{all}	**0.1903**	**0.4300**	**0.2495**	**0.4720**	0.2052	0.3090	**0.3031**	**0.5879**
	$\kappa_{\text{poly-2}}$	0.1898	0.4280	0.2489	0.4660	0.2050	0.3080	0.2993	0.5848
	$\kappa_{\text{poly-3}}$	0.1897	0.4280	0.2489	0.4680	0.2050	0.3080	0.2994	0.5838
	$\kappa_{\text{poly-4}}$	0.1897	0.4280	0.2490	0.4680	0.2049	0.3100	0.2994	0.5848
	$\kappa_{\text{gau-10}}$	0.1901	**0.4300**	0.2491	0.4700	0.2054	0.3110	0.2999	0.5818
	$\kappa_{\text{gau-50}}$	0.1900	**0.4300**	0.2491	0.4680	0.2058	**0.3120**	0.2999	0.5818
	$\kappa_{\text{gau-70}}$	0.1900	0.4300	0.2492	0.4680	0.2058	0.3120	0.2999	0.5818
	$\kappa_{\text{gau-100}}$	0.1900	**0.4300**	0.2492	0.4680	**0.2059**	**0.3120**	0.2999	0.5818
LM	κ_{all}	**0.1914**	**0.4120**	0.2473	**0.4480**	0.2102	0.3010	**0.2944**	0.5476
	$\kappa_{\text{poly-2}}$	**0.1914**	**0.4120**	0.2478	**0.4480**	0.2109	0.3040	0.2933	**0.5485**
	$\kappa_{\text{poly-3}}$	**0.1914**	**0.4120**	0.2478	**0.4480**	0.2109	0.3040	0.2933	**0.5485**
	$\kappa_{\text{poly-4}}$	**0.1914**	**0.4120**	0.2478	**0.4480**	0.2109	0.3040	0.2933	**0.5485**
	$\kappa_{\text{gau-10}}$	**0.1914**	**0.4120**	**0.2480**	**0.4480**	**0.2110**	**0.3050**	0.2934	0.5465
	$\kappa_{\text{gau-50}}$	**0.1914**	**0.4120**	0.279	**0.4480**	0.2109	0.3040	0.2933	**0.5485**
	$\kappa_{\text{gau-70}}$	0.1914	0.4120	0.2478	0.4480	0.2109	0.3040	0.2933	0.5485
	$\kappa_{\text{gau-100}}$	**0.1914**	**0.4120**	0.2478	**0.4480**	0.2109	0.3040	0.2933	**0.5485**
LGD	κ_{all}	**0.1900**	**0.4420**	0.2564	**0.4660**	0.2002	**0.2910**	0.2962	0.5646
	$\kappa_{\text{poly-2}}$	0.1897	**0.4420**	**0.2571**	0.4640	**0.2024**	0.2890	0.2975	0.5718
	$\kappa_{\text{poly-3}}$	0.1897	**0.4420**	0.2563	0.4600	0.2022	**0.2910**	0.2982	0.5724
	$\kappa_{\text{poly-4}}$	0.1897	**0.4420**	0.2568	0.4600	**0.2024**	**0.2910**	**0.2984**	**0.5728**
	$\kappa_{\text{gau-10}}$	0.1899	0.4380	0.2558	0.4620	0.2001	0.2900	0.2947	0.5606
	$\kappa_{\text{gau-50}}$	0.1899	0.4380	0.2560	0.4640	0.1998	**0.2910**	0.2949	0.5626
	$\kappa_{\text{gau-70}}$	0.1899	0.4380	0.2560	0.4640	0.1998	0.2890	0.2949	0.5626
	$\kappa_{\text{gau-100}}$	0.1899	0.4380	0.2560	0.4640	0.1998	0.2890	0.2949	0.5626

Table 8: Comparison of different kernels in SVR in terms of MAP and P@10 on TREC-7,8, WT10G and GOV2. The different kernels used are (a) kernel κ_{all}, (b) polynomial kernels with $\delta = 2$, $\delta = 3$ and $\delta = 4$ ($\kappa_{\text{poly-}\delta}$) and (c) Gaussian kernels with $\sigma = 10.0$, $\sigma = 50.0$, $\sigma = 70.0$ and $\sigma = 100.0$ ($\kappa_{\text{gau-}\sigma}$). Best results are shown in bold and a bold result with † is significantly better than the others at p-value threshold of 0.05.

As one can note, the different kernels yield very similar results, without any significant difference between them. The reason is that the space in which κ_{all} operates is only a 4-dimensional space 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. This result is not really surprising, but still needed experimental confirmation. We thus focus on κ_{all} in the remainder of this study.

3.7.2 *Kernel-based vs Similarity-based Regression Models*

We now compare between both parameter learning strategies, operating in the embedded query space using k-SVR (Section 3.5.2), and in the explicit similarity-based vector space using RF (Section 3.5.3). As mentioned in Section 3.6.1 the representation set $\mathcal{Q}^r$ is constituted of all queries in the two FIRE collections and we used all query sets from CLEF-3 and TREC-3,4,5,6 datasets as source collections. Table 9 summarizes results on TREC-7,8, WT10G and GOV2 collections obtained with the three IR models with their free-parameters predicted using these two approaches.

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	k-SVR	**0.1903**$^{\uparrow}$	**0.4300**	**0.2495**$^{\uparrow}$	**0.4720**	**0.2052**	**0.3090**	**0.3031**$^{\Uparrow}$	**0.5879**$^{\Uparrow}$
	RF	0.1895	0.4280	0.2473	0.4700	0.2017	0.3030	0.2921	0.5616
LM	k-SVR	**0.1914**$^{\uparrow}$	**0.4120**$^{\Uparrow}$	**0.2473**$^{\uparrow}$	**0.4480**$^{\Uparrow}$	**0.2102**	**0.3010**$^{\uparrow}$	**0.2944**$^{\Uparrow}$	**0.5576**$^{\Uparrow}$
	RF	0.1884	0.3940	0.2447	0.4380	0.2079	0.2960	0.2868	0.5494
LGD	k-SVR	**0.1900**	0.4420	**0.2564**	**0.4660**$^{\Uparrow}$	0.2002	0.2910	0.2962	0.5646
	RF	0.1887	**0.4460**	0.2551	0.4540	**0.2036**	**0.2950**	**0.2990**	**0.5707**

Table 9: MAP and P@10 measures of different standard IR models on TREC-7,8, WT10G and GOV2 collections, when the free-parameters of these models are predicted with k-SVR and RF regression functions. The source collections used to train the regression functions are CLEF-3 and TREC-3,4,5,6 datasets. For building the similarity based query vector space queries of both FIRE collections are used as representation set $\mathcal{Q}^r$. Best results are shown in bold. A result with $^{\uparrow}$ is significantly better than the other at 0.05 level and a result with $^{\Uparrow}$ is significantly better at 0.025 level according to Wilcoxon rank sum test.

From these results, it comes out that in all cases except five, learning the kernel based regression model in the embedded query space performs better than the regression model operating in the similarity based vector space. These five exceptions are all for LGD: for P@10 on TREC-7 and for both MAP and P@10 on WT10G and GOV2. But for any of these five cases the difference is not significant. Whereas, for most other cases k-SVR is significantly better than RF according to Wilcoxon rank sum test.

There are different factors which may affect the results of the RF model, among which there is the size of the query set, $\mathcal{Q}^r$, as well as the linguistic difference between queries in this set and those in the source and target collections. In order to verify these points, we changed $\mathcal{Q}^r$ by selecting queries from TREC-7 and TREC-8 datasets at random. Table 10 shows the MAP measures of LGD and LM models in the case where the free-parameters of

these models are predicted with RF, on WT10G and GOV2 collections for different size of the representation set, $\mathcal{Q}^r$.

$\lvert\mathcal{Q}^r\rvert$	WT10G		GOV2	
	LM	LGD	LM	LGD
50	0.2061	0.2028	0.2854	0.2989
75	0.2071	0.2037	0.2860	0.2994
100	0.2076	0.2046	0.2866	0.3002

Table 10: MAP measure of LGD and LM models, when the free-parameters are predicted with RF, on WT10G and GOV2 datasets for an increasing size of the representation set, $\mathcal{Q}^r$, constituted by queries of TREC-7 and TREC-8 collections.

We can see that in this case, the MAP measures of both models improves very slightly (only $1-2\%$) when the size of $\mathcal{Q}^r$ passes from 50 to 100. Moreover, performance of the models are almost same when $\lvert\mathcal{Q}^r\rvert = 100$ irrespective of the representation set be FIRE collection or TREC-7,8 (compare last line of Table 10 and performance of RF in Table 9). Thus the main conclusion from these results, is that for creating the query vector space, the performance of RF are not much affected by the language and the number of queries in $\mathcal{Q}^r$.

Summarizing, it is clearly evident from the experiments that k-SVR performs significantly better than RF on almost all cases that we experimented on. Thus from now on we will mainly focus on k-SVR to further continue our study.

3.7.3 *Learned Values vs Default Ones*

We then evaluate the validity of the approach proposed by comparing the different IR models:

(a) with their parameter set to their default values (Section 3.6.3), denoted by def, and

(b) with their parameter values estimated by SVR based on the simple kernel κ_{all} (Section 3.5.4), denoted by k-SVR.

The results obtained are reported in Table 11.

As one can note, the method proposed yields MAP results significantly higher than the ones obtained with the default values on all models and collections except for LGD on TREC-8, for which there is no significant difference between the two. The same holds for the P@10 measure, with two exceptions this time: LGD and BM25 on TREC-8. Note however that in our setting the regressor is trained to optimize the average precision of each query on the source collection and is thus in line with MAP and not P@10. It is of course straight

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	def	0.1828	0.4180	0.2409	**0.4740**	0.1842	0.2910	0.2739	0.5384
	k-SVR	**0.1903**$^{\uparrow}$	**0.4300**$^{\uparrow}$	**0.2495**$^{\uparrow}$	0.4720	**0.2052**$^{\Uparrow}$	**0.3090**$^{\uparrow}$	**0.3031**$^{\Uparrow}$	**0.5879**$^{\Uparrow}$
LM	def	0.1863	0.3920	0.2401	0.4320	0.2040	0.2930	0.2798	0.5545
	k-SVR	**0.1914**$^{\uparrow}$	**0.4120**$^{\Uparrow}$	**0.2473**$^{\uparrow}$	**0.4480**$^{\uparrow}$	**0.2102**$^{\uparrow}$	**0.3010**	**0.2944**$^{\Uparrow}$	**0.5576**$^{\uparrow}$
LGD	def	0.1882	0.4280	0.2547	**0.4740**	0.1949	0.2870	0.2876	0.5414
	k-SVR	**0.1900**$^{\uparrow}$	**0.4420**$^{\uparrow}$	**0.2564**	0.4660	**0.2002**	**0.2910**	**0.2962**$^{\uparrow}$	**0.5646**$^{\Uparrow}$

Table 11: Comparison of IR models with parameters set to their default values (def) against the predicted parameter values using k-SVR in terms of MAP and P@10. The regressor is trained to optimize the average precision of each query on the source collection. Best results are shown in bold. A result with $^{\uparrow}$ is significantly better than the other at 0.05 level and a result with $^{\Uparrow}$ is significantly better at 0.025 level according to Wilcoxon rank sum test.

forward to directly optimize the P@10, but we want to assess here whether the method tuned towards MAP also behaves well on P@10.

Lastly, as conjectured in Section 3.3, 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-8, which corresponds to the smallest difference between the default values and the upper bounds displayed in Figure 7. Thus in this case there is very little room for improvement over the default values.

3.7.4 *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 method 10split (Section 3.6.3) and is a standard procedure to tune the parameters. The second ideal case corresponds to the upper bounds already discussed in Section 3.3 where all the relevance judgments on the target collections are used to optimize the parameters either globally for all queries (OptGlobal) or query by query (OptPerQuery).

Table 12 displays the results obtained by 10split, together with the ones obtained by kernel regression approach k-SVR. As discussed in Section 3.6.3, for 10split method 10 random splits are performed on target query set and here we report mean (denoted by $\texttt{10split}_m$) as well as the variance (denoted by $\texttt{10split}_v$) of MAP and P@10 yielded by each of these splits. One can note that 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

		TREC-7		TREC-8		WT10G		GOV2	
		MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	10split$_m$	0.1820	0.4192	0.2453	0.4552	0.1961	**0.3094**	**0.3054**	**0.6018**$^{\uparrow}$
	10split$_v$	0.00029	0.00053	0.00121	0.00149	0.00046	0.00048	0.00022	0.00041
	k-SVR	**0.1903**$^{\uparrow}$	**0.4300**$^{\uparrow}$	**0.2495**	**0.4720**$^{\uparrow}$	**0.2052**$^{\uparrow}$	0.3090	0.3031	0.5879
LM	10split$_m$	0.1821	**0.4224**	**0.2545**	**0.4536**	0.1967	**0.3076**	**0.3019**	**0.5679**
	10split$_v$	0.00035	0.00115	0.00048	0.00059	0.00052	0.00062	0.00019	0.00096
	k-SVR	**0.1914**$^{\uparrow}$	0.4120	0.2473	0.4480	**0.2102**$^{\Uparrow}$	0.3010	0.2944	0.5576
LGD	10split$_m$	0.1803	0.4304	**0.2608**	0.4584	0.1924	0.2908	**0.3026**	**0.5901**$^{\Uparrow}$
	10split$_v$	0.00039	0.00167	0.00048	0.00079	0.00053	0.00037	0.00023	0.00088
	k-SVR	**0.1900**$^{\uparrow}$	**0.4420**$^{\uparrow}$	0.2564	**0.4660**	**0.2002**	**0.2910**	0.2962	0.5646

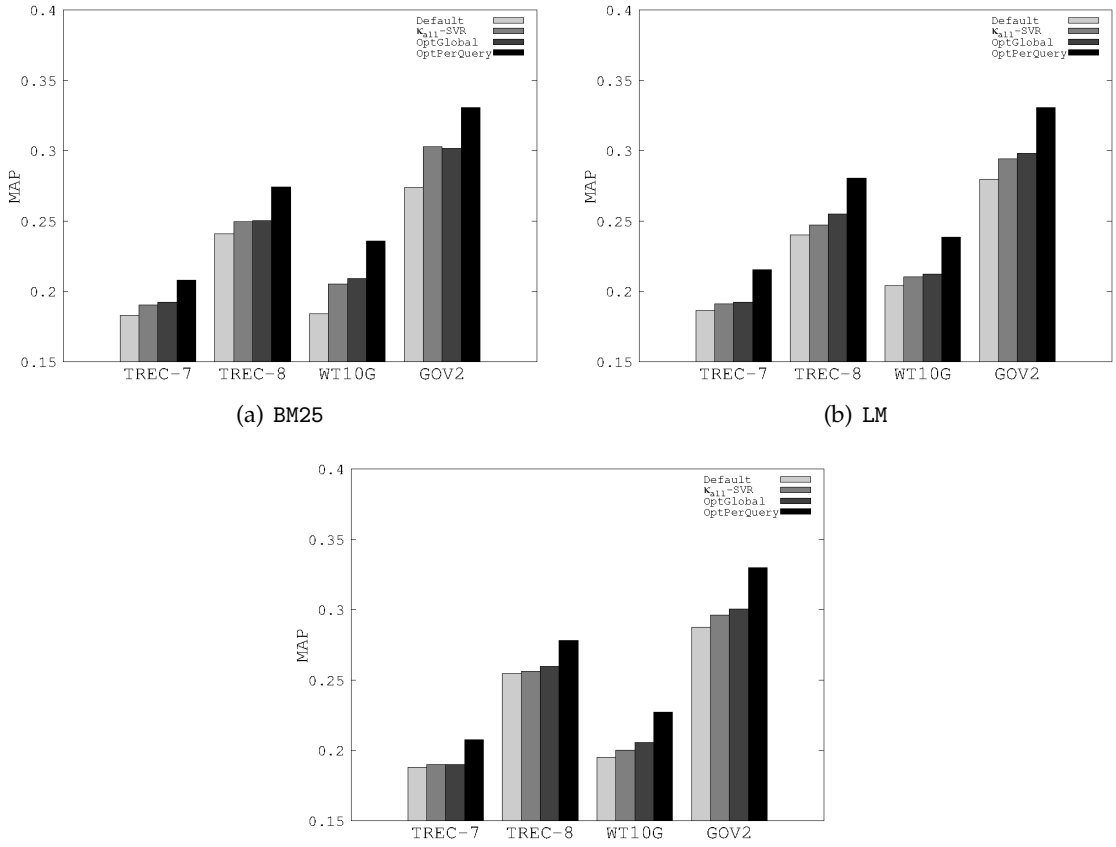
Table 12: Comparison of IR models with optimized parameter values using 10 random splits, against the predicted parameter values using k-SVR in terms of MAP and P@10. The regressor is trained to optimize the average precision of each query on the source collection Best results are shown in bold. A result with $^{\uparrow}$ is significantly better than the other at 0.05 level and a result with $^{\Uparrow}$ is significantly better at 0.025 level according to Wilcoxon rank sum test.

our approach, which does not make use of any relevance judgments on the collection queried (target), is either better or at par with the 10split method which uses 50% of the queries (25 for TREC-7,8 and 50 for WT10G and GOV2) with their relevance judgments on each collection.

We finally compare our approach to the ideal situation when the relevance judgments of all queries in the target collection 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 9, which parallels Figure 7 of Section 3.3, 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 k-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 judgments 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.



(a) BM25 (b) LM

(c) LGD

Figure 9: Comparison of IR models with parameters optimized per query basis (■) and optimized within the whole collection (■) against the predicted parameter values using k-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.

3.7.5 *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. 3.5.1 can be computed offline and stored in the traditional way IDF scores are stored during indexing (as explained in Section 3.6.4). This means that the query representation defined by Eq. 3.5.7 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. 3.5.3. 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. Table 13 gives this extra time, per query, for all collections and models considered.

	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 13: Average extra time taken per query (in milliseconds) by k-SVR over default parameter value settings.

Clearly, 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). These results show that k-SVR method can be deployed with almost none extra time burden compared to the default parameter setting.

3.7.6 *Breaking the Language Barrier*

We investigate here the language independency of the regression method. We perform experiments to validate whether k-SVR behaves well when the source and target collections considered are written in different languages. To this end, we performed two kinds of experiments:

1. We first trained k-SVR on WT10G and GOV2 collections and tested it on non-English FIRE collections;
2. We then performed the reciprocal experiment by training k-SVR on both FIRE collections and testing on WT10G and GOV2 collections.

Note that here we are using three languages in total, English, Hindi and Bengali. MAP and P@10 results are respectively reported in Table 14 and Table 15. 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 that the MAP measures of all IR models for the collections WT10G and GOV2 are similar to the ones reported in Table 11. 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 these experiments.

It can be said that the parameter prediction method k-SVR can be applied on target collections written in different languages. Thus availability of training data in any language (and eventually multiple languages as the case we studied here) enables k-SVR to predict better parameter values for any unseen query in any language.

		FIRE-HN		FIRE-BN	
		MAP	P@10	MAP	P@10
BM25	def	0.2946	0.4540	0.1409	0.2480
	k-SVR	**0.3029**$^{\uparrow}$	**0.4780**$^{\uparrow}$	**0.1532**$^{\uparrow}$	**0.2680**$^{\uparrow}$
LM	def	0.2682	0.4640	0.1535	0.2520
	k-SVR	**0.2839**$^{\uparrow}$	**0.4740**$^{\uparrow}$	**0.1576**$^{\uparrow}$	**0.2640**$^{\uparrow}$
LGD	def	**0.3092**	0.4640	**0.1567**	**0.2840**
	k-SVR	0.3078	**0.4880**$^{\uparrow}$	0.1566	0.2820

Table 14: Comparison of default parameter values (def) and predicted ones (k-SVR) in terms of MAP and P@10 when *non-English* FIRE collections are used as target and WT10G and GOV2 as source. Best results are shown in bold and a bold result with $^{\uparrow}$ is significantly better than the others at p-value threshold of 0.05 according to Wilcoxon rank sum test.

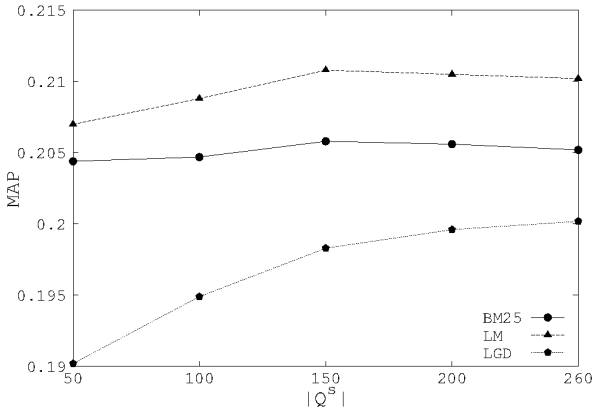
		WT10G		GOV2	
		MAP	P@10	MAP	P@10
BM25	def	0.1842	0.2910	0.2739	0.5384
	k-SVR	**0.2075**$^{\uparrow}$	**0.3250**$^{\uparrow}$	**0.3017**$^{\uparrow}$	**0.5808**$^{\uparrow}$
LM	def	0.2040	0.2930	0.2798	**0.5545**
	k-SVR	**0.2105**$^{\uparrow}$	**0.3060**$^{\uparrow}$	**0.2933**$^{\uparrow}$	0.5515
LGD	def	0.1949	0.2870	0.2876	0.5414
	k-SVR	**0.1985**	**0.3040**$^{\uparrow}$	**0.2905**$^{\uparrow}$	**0.5455**$^{\uparrow}$

Table 15: Comparison of default parameter values (def) and predicted ones (k-SVR) in terms of MAP and P@10 when WT10G and GOV2 are used as target and *non-English* FIRE collections as source. Best results are shown in bold and a result with $^{\uparrow}$ is significantly better than the others at p-value threshold of 0.05 according to Wilcoxon rank sum test.

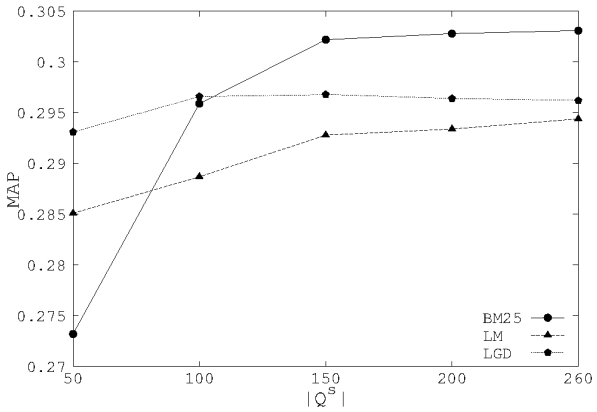
3.7.7 *The Effect of the Number of Queries for Training*

We also analyze the behavior of the IR models for an increasing number of queries in $\mathcal{Q}^s$ for training the k-SVR model. This will also help to reveal the number of queries necessary to train the k-SVR properly. To create $\mathcal{Q}^s$, we randomly selected queries from CLEF-3, TREC-3,4,5,6 source collections to include in the training set. After $\mathcal{Q}^s$ is finalized the training set is prepared by the method described in Section 3.6.4.

Figure 10, illustrates this by showing the evolution of MAP on WT10G and GOV2 with respect to the size of $\mathcal{Q}^s$. 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) WT10G



(b) GOV2

Figure 10: 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.

(a) a simple linear model is sufficient to learn the association between the mapping of queries in the vector space (Equation 3.5.7) and their desired parameter values, and

(b) with a limited amount of source labeled queries, the k-SVR model is able to predict accurate parameter values and concluding from the results of Section 3.7.6, these limited amount of source queries can be gathered from collections written in any language.

3.8 CONCLUSION

We have presented two learning strategies 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, each vector summarizing the behavior of a query word in the collection associated to the query. From this representation, we have shown how to build two collection and language independent vector spaces: the first one corresponds to the feature space of a PDS kernel between queries, and the second one to the similarity scores between the query under consideration and a set of *representational* queries. We have then shown how to learn standard state-of-the-art regression functions in these spaces: kernel support vector regression for the feature space, and random forests for the query similarity space. Other kernel regression methods can of course be used in the feature space, as other regression methods can be used in the query similarity space. Finally, we have introduced a simple PDS kernel that we used in the construction of both spaces. We also considered several traditional and popular PDS kernels (polynomial and Gaussian) to be used in the feature space.

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

1. Among two approaches we have developed, the one based on the feature space (as opposed to the query similarity space) behaves slightly better and does not depend on a set of "external" queries that can bring a bias in the representation chosen.

2. The feature space method 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 judgments 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.

3. In the collections considered, we have also found that our method is at least as good as any global method, potentially using all the relevance judgments on the target collection.

4. The feature space method is still below the upper bound provided by optimizing parameter values per query basis using all the relevance judgments on the target collection, and there is 2% to 4% room for improvement.

5. The extra time required by our method for each query lies in the range 15-30 milliseconds. This extra time, which can be decreased through parallel computation, corresponds to the application of the regression function on the query representation.

6. The number of source queries with relevance judgments 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.

7. Our approach is collection and language independent and can be used to predict parameter values of standard IR models to new collections even written in other languages, as illustrated on the FIRE collections.

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

Two methods proposed here make use of the paradigm known as *transfer learning* in literature. They use relevance judgments on source collection first to determine the optimal parameter values for different IR models and transfer the knowledge of these parameter values through a regression model. This transfer became possible because of the common representation of the queries on some vector space defined here. This approach of transferring knowledge opens the door to transfer relevance information directly between collections, instead of transferring it through the parameter values. But in order to do that a proper representation of complete collections is required. We focus on this problem in the next chapter where we try to design a data structure that will enable us to summarize and represent the relevance information of a complete collection so that it can be transferred to a target collection. Once we are able to infer some relevance information on the target any standard ranking algorithm can be deployed to learn a ranking function.

Part II

Discovering IR Functions Through Transfer Learning and Exhaustive Exploration

4

KNOWLEDGE TRANSFER IN IR: LEARNING TO RANK ON UNLABELLED COLLECTIONS

We propose a general approach to learn a ranking function on a target collection without relevance judgments by transferring knowledge from a source collection having such information. The relevance information in the source collection is summarized in a grid that provides, for each term frequency and document frequency values of a word in a document, an empirical estimate of the relevance of that document. We hence, propagate this information from a source to a target collection using the grid and obtain a first pool of pairwise preferences over the pairs of documents in the latter. The algorithm then iteratively learns a ranking function on the target collection and assigns pairwise preferences to its documents using the scores of the learned function. Our approach can be coupled with easy in hand transfer strategies, and we further propose a simple source selection procedure in order to choose the best associated source collection for each query in a target dataset. We show the effectiveness of our approach through a series of extensive experiments on `CLEF-3` and several collections from `TREC` and show that the proposed approach yields results consistently and significantly above state-of-the-art IR functions as well as a state-of-the-art transfer ranking approach.

4.1 INTRODUCTION

Many applications of information retrieval (IR), as *ad hoc* retrieval, routing or collaborative filtering, rely on ranking functions, and IR has a long tradition in designing such functions. More recently, IR researchers have explored the possibility to learn ranking functions from relevance judgments on query-document pairs, an approach known as learning to rank. Such an approach has been extremely productive and has led to new ranking functions that are built on a set of comprehensive features, including standard IR functions originally developed for unlabeled collections. In learning to rank for IR, the training data consists of a set of queries, a set of retrieved documents associated to each query, as well as relevance judgments for all query-document pairs. In many cases however, acquiring precise relevance judgments is a time consuming operation that requires an important manual effort. To alleviate this problem, researchers have explored ways to transfer known annotations, in the form of relevance information, from one source

collection to a target collection, a field of research often referred to as cross-domain adaptation. Two situations have been considered: (a) a few relevance judgments are available on the target collection that can be combined with the information of the source collection to learn an appropriate ranking function [Chen et al., 2008a, Chen et al., 2008b, Chen et al., 2010], and (b) no relevance judgments are available on the target collection and the only information one can exploit to learn a new ranking function is the one contained in the source collection [Gao et al., 2010, Cai et al., 2011b, Cai et al., 2011a].

Our work fits within this latter line of research and aims at learning an IR function on a collection with no relevance judgments. This is a typical situation in IR, in competitions like the TREC IR tracks for example, but also in retrieving information on new web collections related to new domains. In both cases, one is dealing with the problem of ranking documents with respect to different information needs for which no relevance judgments are available. In both cases however, some effort to gather (automatically or manually) relevance judgments for past collections and domains (and thus past queries) have already been made. The studies presented in [Gao et al., 2010, Cai et al., 2011b, Cai et al., 2011a] have shown that it was indeed possible to learn a good ranking IR function in this scenario. However, they have also shown that the function learned by their method on the target domain heavily depends on the source domain used.

We introduce in this study a new framework for transferring information from a source collection so as to learn a ranking function on a target collection. Our framework differs from the preceding ones and aims at learning the IR ranking function directly on the target domain. We refer to this research field as *transfer learning to rank* and will study several aspects of the problem, as the one related to the adequacy of the source collection for a given target collection and we will also explore the possibilities to select the appropriate source collection from which to transfer relevance information. Note that we used a similar transfer learning technique in the previous chapter to predict free parameters for IR models. However, here we attempt to learn the ranking function itself by directly transferring the relevance information from source to target instead of doing so through the parameter values.

The transfer learning to rank problem is addressed here by propagating relevance information from a source collection to a target collection, so as to obtain pairwise preferences over the pairs of documents in the latter. This propagation is achieved by first constructing a grid associating normalized Document Frequency (DF) and Term Frequency (TF) values to a relevance score in the source collection. The individual scores of each term in a (query, document) pair of the target collection are then looked-up in the grid and combined to form a global score for the query-document pair. This score is finally used to build pairwise preference judgments on the target collection. The ranking algorithm then trains a ranking function as the combination of baseline ranking scores, using pairwise preferences obtained previously. These two steps of pseudo-labeling, using the current ranking function, and pairwise learning are then iterated until a convergence criterion is

reached. The set of features as well as the model and algorithm that form the basis of our approach are standard components of learning to rank approaches.

Using CLEF-3, TREC-3, TREC-4, TREC-5, TREC-6, TREC-7, TREC-8, WT10G and GOV2 collections, we show that our approach consistently and significantly improves over state-of-the-art IR functions as well as other state-of-the-art transfer learning approaches. We also propose a simple source selection strategy which using the grid information chooses for each query in the target collection the most similar source collection before learning. This is to study the situations in which the source selection would be most profitable, in terms of a distance between the source and target collections. Our analyses show that the efficiency of transfer learning depends, to a certain extent, on the adequacy of the source collection with respect to the target one.

The remainder of the paper is organized as follows: Section 4.2 positions our work with respect to the state-of-the-art. Section 4.3 describes the problem of knowledge transfer from a source domain to a target domain for learning to rank. In Section 4.4 we present the details of the grid which forms the basis of the knowledge transfer we consider here, as well as the self-learning algorithm we consider. The experimental setup and the validation of the approach proposed here are presented and discussed in Section 4.5 and Section 4.6 respectively. Finally, we summarize the main findings of this study in Section 4.7.

4.2 RELATED WORK

Learning to rank approaches have been mainly developed within a supervised learning paradigm, and they are traditionally grouped into three main categories: pointwise, listwise and pairwise [Liu, 2009].

Pointwise approaches [Crammer and Singer, 2001, Harrington, 2003, Li et al., 2008] assume that each query-document pair has an ordinal score. Ranking is then formulated as a regression problem, in which the rank value of each document is estimated as an absolute quantity. In the case where relevance judgments are given as orders or pairwise preferences (rather than relevance degrees), it is usually not straightforward to apply these algorithms for learning. Moreover, the two main problems reported in the literature are that pointwise based techniques do not consider the interdependency among documents, so that the position of documents in the final ranked list is missing in the regression-like loss functions used for parameter tuning [Chapelle et al., 2011]. Furthermore, these algorithms ignore the association of some training documents with the same query, and in some extreme cases, the ranking loss function may be dominated by queries having a large number of associated retrieved documents [Chen et al., 2009]. On the other hand, listwise approaches [Valizadegan et al., 2009, Xu and Li, 2007, Xu et al., 2008] take the entire ranked list of documents for each query as a training instance. As a direct consequence, these approaches are able to differentiate documents from different queries, and consider their position in the output ranked list at the training stage. However, listwise techniques

aim to directly optimize a ranking measure, so they generally face a major problem of dealing with non-convex, non-differentiable and discontinuous functions which in some cases may be difficult to calibrate with respect to Average Precision and the Expected Reciprocal Rank, which are widely used in IR evaluations [Calauzènes et al., 2012]. Finally, in pairwise approaches [Cohen et al., 1999, Freund et al., 2003, Joachims, 2002], the ranked list is decomposed into a set of document pairs. Ranking is therefore considered as the classification of pairs of documents, such that a classifier is trained by minimizing the number of misorderings in ranking. In the test phase, the classifier assigns a positive or negative class label to a document pair that indicates which of the documents in the pair should be better ranked than the other one. SVM is undoubtedly one of the most popular classifiers used to perform binary classification on the pairs of documents for ranking [Cao et al., 2006, Herbrich et al., 1999, Joachims, 2002]. Other adaptation of popular classifiers to pairwise ranking, like RankBoost which minimizes the exponential loss over document pairs [Freund et al., 2003], has also attracted attention in the recent past. More recently, some work considered a smooth approximation to the gradient of the ranking loss instead of searching for a smooth and convex approximation to the ranking loss itself [Burges, 2010], while others considered a direct optimization of the ranking loss function [McAllester et al., 2010].

Creating new test collections, or manually assigning relevance judgments for even a small set of queries, is a tedious task [Carterette, 2007]; at the same time, many efforts have 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. This is precisely the point investigated in *transductive transfer learning* mainly developed for classification and regression tasks. For a given learning task and two source and target domains, approaches proposed under this paradigm also aim to improve the learning of a target predictive function (or simply to learn this function) by using the knowledge in the source domain (mostly in the form of labeled instances), as well as unlabeled examples from the target domain. Different approaches to transductive transfer learning and corresponding works done in those directions are discussed in Sections 1.5.2 and 3.2. A survey on the studies made in transfer learning till 2010 is provided in [Pan and Yang, 2010].

Recently, some innovative studies have extended the cross-domain learning to ranking [Chen et al., 2008a, Chen et al., 2010, Gao et al., 2010]. While some of them made use of both labeled queries in the source and a small set of labeled queries in the target to learn a ranking function [Chen et al., 2010], others considered that only the source collection contains labeled information, while queries in the target collection have no associated relevance judgments [Gao et al., 2010, Cai et al., 2011b]. This is also the setting we consider in this study. In these latter approaches, 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 retained in this study. 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 be in the learning process.

This approach has been shown to work in the case where the source and target collections are close to each other. In the case where they contain different ⋆ **subjects (?)** ⋆, the ranking function learned with the weighted (query,document) pairs will not lead to efficient rankings as illustrated in table 22. This remark is valid for all transfer learning approaches we are aware of (including ours), and justifies the need for a source selection procedure. We know of no (direct) ways to select source collections with the approach outlined above. Indeed, as the source (query, document) weights vary from one source collection to another, one has to select the source collection, and hence a ranking function, per target query, meaning that no single ranking functions can be used on target queries. In contrast, our approach directly learns the ranking function on the target collection from a set of queries, after having selected the source collection for transfer for each of these queries. In our early study, we noticed that this transfer strategy might be efficient in some interesting cases where the target set and the source collections might be typically different [Goswami et al., 2013]. In this work, we present a general algorithm that encompasses our early model and which can be easily adapted to more complex settings.

4.3 FRAMEWORK

We consider here a source collections $\mathcal{C}^s$ composed of a set of documents $\mathcal{D}^s$, a set of queries Q^s and relevance judgments for each query in Q^s. These relevance judgments are assumed to be binary, which is the most common situation. We also consider a target collection $\mathcal{C}^t$, composed of a set of documents $\mathcal{D}^t$ and a set of m queries $Q^t = \{q_1^t, q_2^t, \ldots, q_m^t\}$ without any relevance judgments.

Our goal here is to transfer relevance information from $\mathcal{C}^s$ to $\mathcal{C}^t$ and then learn a ranking function on $\mathcal{C}^t$. To do so, we propose:

(a) to induce relative relevance judgments between documents in $\mathcal{C}^t$ by transferring relevance information from $\mathcal{C}^s$,

(b) to learn a ranking function in $\mathcal{C}^t$ from the relative relevance judgments obtained previously.

As in traditional learning to rank approaches for IR, each query-document pair in the target collection $(q^t, d) \in Q^t \times \mathcal{D}^t$ is represented as a K-dimensional feature vector $\mathbf{f}$:

$$\mathbf{f}(q^t, d) \stackrel{def}{=} (f_1(q^t, d), \ldots, f_K(q^t, d)) \tag{4.3.1}$$

The vector attributes, $f_k(.,.); k \in \{1, \ldots, K\}$, considered in this work include standard features used in document retrieval as well as three state-of-the-art IR scoring functions. These features are presented in more details in Section 4.5.2.

For each query q^t in the target collection, the transfer of relevance information from the source collection to the target one results in a set of relative relevance judgment pairs for documents in $\mathcal{D}^t$. If d, d' are two documents in $\mathcal{D}^t$, then the relative relevance judgment is of the form $d \succ_{q^t} d'$, where $\succ_{q^t}$ denotes a preference relationship and means *more relevant to query q^t than*. That is $d \succ_{q^t} d'$ means d is more relevant with respect to query q^t than d. From these sets, one can then construct a ranking function $h : \mathbb{R}^K \to \mathbb{R}$ that assigns a score to documents in $\mathcal{D}^t$ for each query $q^t \in Q^t$. Similarly to previous learning to rank studies, we focus here on linear ranking functions:

$$h_{\omega}^{(\mathbf{f}(q^t,d))} = \langle \omega, \mathbf{f}(q^t, d) \rangle \tag{4.3.2}$$

where $\langle .,. \rangle$ stands for an inner product and the weight vector ω represents the model parameters. Table 16 gives the notations used in the chapter.

Notation	Description
t_w^d	term frequency of term w in a document d
t_w^q	number of occurrences of term w in a query q
$t_w^{\mathcal{C}}$	number of occurrences of term w in a collection $\mathcal{C}$
x_w^d	normalized version of term frequency
$N_w(\mathcal{C})$	document frequency of term w in collection $\mathcal{C}$
$y_w(\mathcal{C})$	normalized version of document frequency
$z_w(\mathcal{C})$	inverse document frequency of term w in $\mathcal{C}$
$N(\mathcal{C})$	number of documents in collection $\mathcal{C}$
$N_R^q(\mathcal{C})$	number of relevant documents in $\mathcal{C}$ for query q
$\mathcal{R}(q)$	Set of documents relevant to query q
l_d	length of document d in # of terms
l_q	length of query q in # of terms
$l_{\mathcal{C}}$	length of collection $\mathcal{C}$ in # of terms
$l_{avg}(\mathcal{C})$	average length of documents in collection $\mathcal{C}$

Table 16: Notations

4.4 TRANSFER LEARNING FOR RANKING

In the following sections, we first show how to obtain relative relevance judgments from a source collection $\mathcal{C}^s$ for a target collection $\mathcal{C}^t$ and then propose a learning algorithm to transfer the obtained relevance information (Section 4.4.1, 4.4.2, 4.4.3). We then extend our

algorithm so that a single source collection can be selected for a given target query from a set of available source collections (Section 4.4.4).

4.4.1 *Transferring relevance information*

As mentioned above, our goal is to transfer relevance information from a source collection to a target one, so as to be able to develop a pairwise ranking algorithm on the target collection. For each query, we thus want to obtain relative relevance judgments between document pairs. One possibility would be to tune an IR model, say BM25, on the source collection $\mathcal{C}^s$, and then use its scores on the target collection $\mathcal{C}^t$ to form document pairs, that is if $\texttt{BM25}(q^t, d_{q^t}) > \texttt{BM25}(q^t, d'_{q^t})$ then $d_{q^t} \succ_{q^t} d'_{q^t}$. The problem with this approach is that classical IR scoring functions constitute an important part of the vector features in the representation of query-document pairs, i.e. in $\mathbf{f}(.,.)$ (see e.g. [Cao et al., 2006]). Constructing a training set with BM25 and then using it as one of the attributes to learn a ranking function h will result in h behaving as BM25. To avoid this, one can think of:

(a) using a first IR scoring function, tuned on the source collection, to build the training set,

(b) relying on different IR scoring functions to form attributes of the examples in the target collection.

This procedure would avoid the problem mentioned above if the two scoring functions yield different rankings.

	BM25 vs LM	BM25 vs LGD	LM vs LGD
TREC-6	0.9998	0.9998	0.9998
TREC-7	0.9997	0.9997	0.9997
TREC-8	0.9997	0.9997	0.9998

Table 17: Kendall τ rank correlation coefficient between 3 state-of-the-art IR scoring functions on three collections.

Table 17, displays the Kendall τ rank correlation coefficients for three state-of-the-art IR scoring functions, on three different collections. The three scoring functions considered are: the language model with Dirichlet smoothing [Zhai and Lafferty, 2001], denoted as LM, BM25 [Robertson and Zaragoza, 2009], and the log-logistic model of the information-based family [Clinchant and Gaussier, 2010], denoted LGD. The three collections are TREC-6, TREC-7 and TREC-8, further described in Section 4.5.1. For each query in each collection, the Kendall τ rank correlation coefficient is computed between any two scoring functions, and then average value over all queries within a collection is reported in this table. Maximum possible value of Kendall τ rank correlation coefficient is 1, which indicates

that the two scoring functions under observation yield exactly the same ranking. As one can note, the values taken by the correlation coefficient here are very high, meaning that all functions provide very similar rankings.

Therefore, using any of these functions to construct a training set and other ones to form attributes for learning will again result in a function h equivalent to the IR scoring functions. We thus have to rely on a different strategy that avoids resorting to IR scoring functions, and makes explicit use of the relevance information provided by a source collection.

A word w can be characterized by two quantities which constitute the basis of all IR scoring functions. Firstly the normalized document frequency which corresponds to:

$$y_w(\mathcal{C}^s) = \frac{N_w(\mathcal{C}^s)}{N(\mathcal{C}^s)}$$

Secondly normalized number of occurrences of w in any document d of the collection considered, which is set, following [Amati and van Rijsbergen, 2002a] and [Clinchant and Gaussier, 2010], to:

$$x_w^d = t_w^d \log(1 + \frac{l_{avg}(\mathcal{C}^s)}{l_d})$$

In the remainder, we will use the term NDF for $y_w(\mathcal{C})$ and NTF for x_w^d interchangeably.

Suppose w is a term in a query q and a d is a document where w occurs. We wish to estimate the contribution of w to the relevance of d to q. Now if a query q^s is identified in the source collection $\mathcal{C}^s$ with same (NDF, NTF) values as w, the relevance can be estimated through the proportion of relevant documents with respect to q^s. One can recall, this is possible because relevance judgements for the sources are availble. The proportion of relevant documents q^s in source collection $\mathcal{C}^s$ that have the same (NDF, NTF) values as $y_w(\mathcal{C})$ and x_w^d is given by:

$$\frac{|\{d' \in \mathcal{D}^s, \exists(q' \in \mathcal{Q}^s, w' \in q'), d' \in \mathcal{R}(q') \wedge \text{eq}((w', d'), (w, d))\}|}{|\{d' \in \mathcal{D}^s, \exists(q' \in \mathcal{Q}^s, w' \in q' \cap d'), \text{eq}((w', d'), (w, d))\}|} \tag{4.4.1}$$

where $\text{eq}((w', d'), (w, d))$ means the situation where NDF and NTF of w and w' are same, that is:

$$y_{w'}(\mathcal{C}^s) = y_w(\mathcal{C}) \wedge x_{w'}^{d'} = x_w^d$$

$|\{\}|$ denotes the cardinality of a set. Let d' be a document in source document set $\mathcal{D}^s$, $q^{'}$ be a query in source query set Q^s and w' be a term in query $q^{'}$. The numerator of the above ratio represents the number of relevant documents with respect to query $q^{'}$ under the situation where any term w' in $q^{'}$ has same NDF and NTF as term w in the original query q under question. It is an estimate of the number of relevant documents to q with respect to source collection $\mathcal{C}$ and is determined by finding a similar query $q^{'}$ in Q^s in terms of (NDF, NTF) for which relevance judgement is known. Similarly denominator of the ratio represents the number of all the documents retrieved for source query $q^{'}$ which is similar to original query q in terms of same (NDF, NTF) of the words contained by

the queries. Hence the complete ratio represents the proportion of relevant documents of target query q with respect to a similar source query $q^{'}$. In other words the quantity estimates the probability that d is relevant to q knowing w and the relevance information in $\mathcal{C}^s$.

However, as typical IR collections only contain few queries, very few words will have exactly same (NDF, NTF) values and the above estimate will likely not be robust. One way to avoid this problem is to consider regions in the (NDF, NTF) space into which the different values are considered equivalent. This amounts to discretize the NDF and NTF values, e.g. by defining intervals on each value range and then by assigning a representative value (commonly the median) to each of these regions. If the real normalized document frequency value of w in a collection $\mathcal{C}$ is $y_w(\mathcal{C})$, then $[y_w(\mathcal{C})]_{dis}$ is the representative value of that NDF region where $y_w(\mathcal{C})$ belongs. Similarly for term frequency value x_w^d, the representative value of the NTF value region where x_w^d is denoted by $[x_w^d]_{dis}$.

The probability that document d is relevant to query q knowing term w and a source collection $\mathcal{C}^s$ can now be rewritten as:

$$\hat{P}(d \in \mathcal{R}(q)|w;\mathcal{C}^s) = \frac{|\{d' \in \mathcal{D}^s, \exists(q' \in \mathcal{Q}^s, w' \in q'), d' \in \mathcal{R}(q') \wedge \text{eq}_{dis}((w',d'),(w,d))\}|}{|\{d' \in \mathcal{D}^s, \exists(q' \in \mathcal{Q}^s, w' \in q' \cap d'), \text{eq}_{dis}((w',d'),(w,d))\}|} \tag{4.4.2}$$

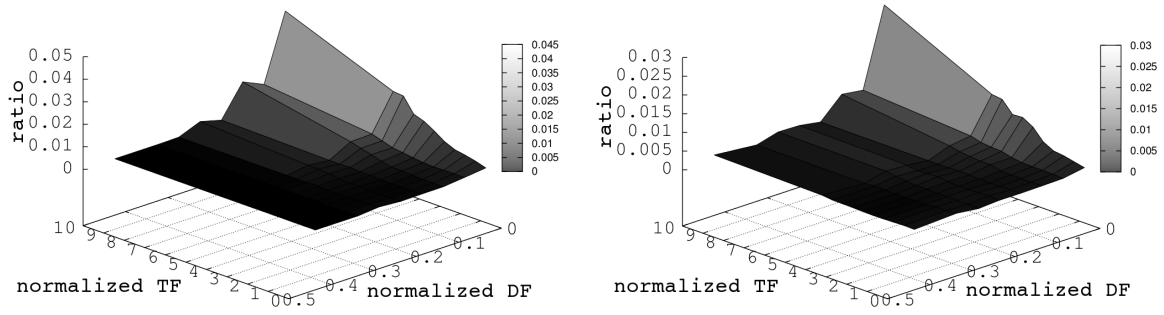
where $\text{eq}_{dis}((w',d'),(w,d))$ means the situation where both NDF and NTF of w and w' falls in the same discrete region, that is:

$$[y_{w'}(\mathcal{C}^s)]_{dis} = [y_w(\mathcal{C})]_{dis} \wedge [x_{w'}^{d'}]_{dis} = [x_w^d]_{dis}$$

Above ratio is a similiar one with the previous one (Eq. 4.4.1). The only difference is that now the (NDF, NTF) equality condition is relaxed to a discrete region instead of exact equality. As the previous ratio (Eq 4.4.1), it also estimates the probability that d is relevant to q knowing w and the relevance information in $\mathcal{C}^s$. Due to the discretization, this ratio no longer suffers from the robustness of estimation issue like the previous one.

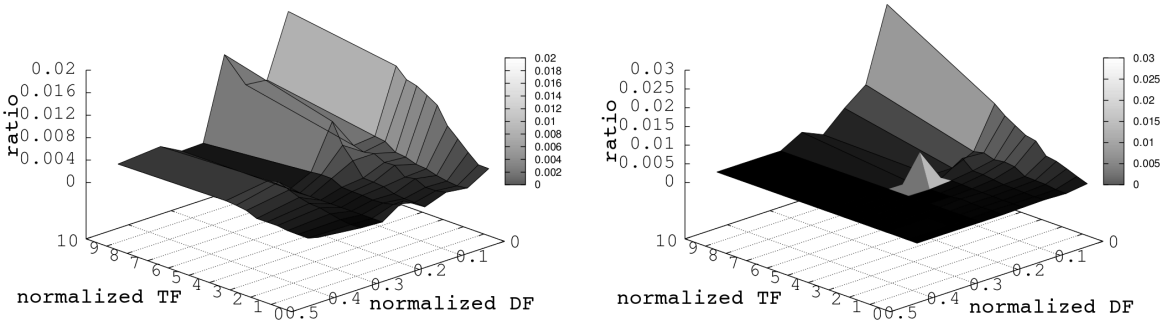
Figure 11 displays the proportion of relevant documents in 88 different regions of the normalized (NDF, NTF) space for the `TREC-3`, `TREC-4`, `TREC-6` and `CLEF-3` collections (described in Section 4.5.1). Here, the DF dimension has been discretized into 8 discrete values in steps of 0.05; the TF one into 11 values in steps of 0.5. This difference in discretization is due to the difference in stretch for the two scores. We finally obtain a 11×8 structure for any source collection, based on 8 discrete values for NDF and 11 discrete values for NTF along with the proportion of relevant documents for each region (details are in Section 4.5.2). We call such curves *grids* in the following.

The goal is to infer relative relevance rankings or a preference relationship between document pairs (Section 4.3) of a target collection. In order to do that one needs to derive a global score for each document on the complete query. But from the definition of grid it is clear that the information contained in the grid only concerns the contribution of a



(a) TREC-3 Grid

(b) TREC-4 Grid



(c) TREC-6 Grid

(d) CLEF-3 Grid

Figure 11: Proportion of relevant documents in each region of the normalized (DF, TF) space, for the TREC-3, TREC-4, TREC-6 and CLEF-3 collections. A gray scale is used to differentiate the proportions (the whiter, the higher). The small peak in the CLEF-3 grid is however rendered whiter for readability reasons.

single query term to the relevance of a document. A global score or retrieval status value, $RSV_{\mathcal{C}^s}(q_t, d)$, for a document d in the retrieved set of q^t can be computed on the basis of the (log) probability that d is relevant to q^t, that is:

$$RSV_{\mathcal{C}^s}(q_t, d) = \log P(d \in \mathcal{R}(q^t)|q^t)$$

Using Bayes formula and assuming query terms are independent of one another, one has:

$$\begin{aligned} P(d \in \mathcal{R}(q^t)|q^t) &= \frac{P(q^t|d \in \mathcal{R}(q^t)P(d \in \mathcal{R}(q^t))}{P(q^t)} \\ &= \frac{P(d \in \mathcal{R}(q^t))}{P(q^t)} \prod_{w \in q^t} P(w|d \in \mathcal{R}(q^t))^{x_w^{q^t}} \end{aligned}$$

Reapplying Bayes formula on the second term:

$$P(d \in \mathcal{R}(q^t)|q^t) = \frac{P(d \in \mathcal{R}(q^t))}{P(q^t)} \prod_{w \in q^t} \left(\frac{P(d \in \mathcal{R}(q^t)|w)P(w)}{P(d \in \mathcal{R}(q^t))} \right)^{x_w^{q^t}}$$

The term $P(d \in \mathcal{R}(q^t))$ is independent of query term w, hence:

$$\begin{aligned} P(d \in \mathcal{R}(q^t)|q^t) &= \frac{\big(P(d \in \mathcal{R}(q^t))\big)^{1-|q^t|}}{P(q^t)} \prod_{w \in q^t} (P(w))^{x_w^{q^t}} \prod_{w \in q^t} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}} \\ &= \frac{\big(P(d \in \mathcal{R}(q^t))\big)^{1-|q^t|}}{P(q^t)} P(q^t) \prod_{w \in q^t} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}} \\ &= \big(P(d \in \mathcal{R}(q^t))\big)^{1-|q^t|} \prod_{w \in q^t} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}} \end{aligned}$$

The last two steps are for the fact that $\prod_{w \in q^t} P(w) = P(q^t)$. In the absence of any information on the query q^t, all the documents in the collection have the same probability of being relevant. Thus the term $P(d \in \mathcal{R}(q^t))$ is independent of query q^t, so it does not affect the retrieval process and hence can be ignored. Now a document d will either contain the term w or it will not. Hence the events $w \in q^t \cap d$ and $w \in q^t \backslash d$ are mutually exclusive. This leads to:

$$P(d \in \mathcal{R}(q^t)|q^t) = \prod_{w \in q^t \cap d} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}} \times \prod_{w \in q^t \backslash d} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}}$$

The quantity $P(d \in \mathcal{R}(q^t)|w)$, for $w \in q^t \cap d$, can be estimated by $\hat{P}(d \in \mathcal{R}(q^t)|w; \mathcal{C}^s)$ using the grid of $\mathcal{C}^s$. For $w \in q^t \backslash d$, we have: $P(d \in \mathcal{R}(q^t)|w) = P(d \in \mathcal{R}(q^t))$. This is simply the probability of the relevancy of document d to the query q^t and can be estimated as:

$$P(d \in \mathcal{R}(q^t)) = \frac{|\mathcal{R}(q^t)|}{N(\mathcal{C}^t)}$$

But relevance judgments are not available in the target collection $\mathcal{C}^t$. One does not have a direct access to this quantity, but it can be estimated through the average of the proportion of relevant documents for queries in the source collection:

$$P(d \in \mathcal{R}(q^t)) \approx \frac{1}{Q^s} \sum_{q^s \in Q^s} \frac{|\mathcal{R}(q^s)| \text{ in } \mathcal{C}^s}{N(\mathcal{C}^s)}$$

One can note that now the term $P(d \in \mathcal{R}(q^t)$ depends on query term w as well as the the documents, hence it cannot be ignored as it affects the retrieval process directly. Replacing this estimated quantites in the above equation of $P(d \in \mathcal{R}(q^t)|q^t)$:

$$P(d \in \mathcal{R}(q^t)|q^t) = \prod_{w \in q^t \cap d} \big(P(d \in \mathcal{R}(q^t)|w)\big)^{x_w^{q^t}} \times \prod_{w \in q^t \backslash d} \left(\frac{1}{Q^s} \sum_{q^s \in Q^s} \frac{|\mathcal{R}(q^s)| \text{ in } \mathcal{C}^s}{N(\mathcal{C}^s)} \right)^{x_w^{q^t}}$$

From the definition of retrieval status value we thus finally obtain:

$$
\begin{aligned}
RSV_{\mathcal{C}^s}(q_t, d) &= \log P(d \in \mathcal{R}(q^t)|q^t) \\
&= \log \left(\prod_{w \in q^t \cap d} \left(P(d \in \mathcal{R}(q^t)|w)\right)^{x_w^{q^t}} \times \prod_{w \in q^t \setminus d} \left(\frac{1}{Q^s} \sum_{q^s \in Q^s} \frac{|\mathcal{R}(q^s)| \text{ in } \mathcal{C}^s}{N(\mathcal{C}^s)} \right)^{x_w^{q^t}} \right) \\
RSV_{\mathcal{C}^s}(q_t, d) &= \sum_{w \in q^t \cap d} x_w^{q^t} \log(\hat{P}(d \in \mathcal{R}(q^t)|w; \mathcal{C}^s)) \\
&\quad + \sum_{w \in q^t \setminus d} x_w^{q^t} \log \left(\frac{1}{Q^s} \sum_{q^s \in Q^s} \frac{|\mathcal{R}(q^s)| \text{ in } \mathcal{C}^s}{N(\mathcal{C}^s)} \right) \qquad (4.4.3)
\end{aligned}
$$

It is important to note here that, for a target query q^t, $RSV_{\mathcal{C}^s}(q^t, d)$ is based on the information brought by source queries *similar* to q^t. Indeed, the first term in $RSV_{\mathcal{C}^s}(q^t, d)$ makes use of those query-document pairs in $\mathcal{C}^s$ that contain words with similar (NDF, NTF) values as the ones in (d, q^t).

The procedure we use for transferring relevance information from a source collection $\mathcal{C}^s$ to a target query q^t can then be summarized as follows :

1. Construct the grid for $\mathcal{C}^s$ that provides the proportion of relevant documents in all the regions (as defined above) of the (NDF, NTF) space;
2. For all documents in the target collection, compute $RSV_{\mathcal{C}^s}(q^t, d)$ according to Eq. 4.4.3;
3. If two documents d and d' in the retrieved set of q^t are such that $RSV_{\mathcal{C}^s}(q^t, d)$ is sufficiently larger than $RSV_{\mathcal{C}^s}(q^t, d')$, then assume that $d \succ_{q^t} d'$.

Of course, the grid for any source collection can be constructed off-line, and does not need to be re-constructed for each new query. The details of step 3 will be discussed in Section 4.4.2.

	BM25 vs $\mathcal{G}_{\mathcal{C}^s}$	LM vs $\mathcal{G}_{\mathcal{C}^s}$	LGD vs $\mathcal{G}_{\mathcal{C}^s}$
TREC-6	0.8843	0.8842	0.8843
TREC-7	0.8711	0.8710	0.8710
TREC-8	0.8112	0.8111	0.8111

Table 18: Kendall τ rank correlation coefficient between 3 IR scoring functions and the grid score.

Interestingly, the correlation between the rankings provided by a grid and IR scoring functions is significantly lower than the one between IR scoring functions themselves. It can be seen in Table 18 that displays the Kendall τ rank correlation coefficient between

the scoring function given in Eq. 4.4.3 using grid $\mathcal{G}_{\mathcal{C}^s}$ built from the TREC-3 collection and the three standard IR scoring functions mentioned above (these rank coefficients are again computed on TREC-6, TREC-7 and TREC-8). Table 17 displays the same between three state-of-the-art IR scoring functions themselves. Lower correlation between grid based scoring function and IR scoring functions infers that the ranking generated using grid score is significantly different than the ranking by standard IR models.

We are thus in a position where learning with the relative relevance judgments obtained by the grid would lead to a different (and hopefully better) ranking function than state-of-the-art IR scoring functions.

4.4.2 *Learning a pairwise classifier*

We now transform the transferred information to create training data to learn the ranking function h (Eq. 4.3.2). The relevance information transferred from the source collection to the target one takes the form of pairwise relative judgments on which one can learn a pairwise classifier. To do so, for any query q^t in Q^t and any instance pair $(d, d') \in \mathcal{D}^t$, one first constructs a pseudo-label $\tilde{z}(d, d')$ as:

$$\tilde{z}_{d,d'} = \begin{cases} +1 \text{ if } RSV_{\mathcal{C}^s}(q^t, d) \geq RSV_{\mathcal{C}^s}(q^t, d') + \delta \\ -1 \text{ if } RSV_{\mathcal{C}^s}(q^t, d) \leq RSV_{\mathcal{C}^s}(q^t, d') - \delta \end{cases} \tag{4.4.4}$$

where $\mathcal{C}^s$ is the source collection. The pseudo-label $\tilde{z}(d, d')$ is $+1$ if score of d with respect to the source collection $\mathcal{C}^s$ is significantly higher than that of d'. This actually means that the document d is much more relevant with respect to the source $\mathcal{C}^s$ in comparison with d'. In Eq. 4.4.4 δ represents a margin over the grid scores which defines *significance difference* in the scores, that is $\tilde{z}(d, d')$ is $+1$ or -1 if the absolute difference of the scores of d and d' is atleast δ. This also avoids possible noise in transfer by filtering out documents with very similar score. Because document pairs with very little score difference are basically very similar and thus unfit to define a proper and effective preferential relationship.

However, as illustrated in [Cao et al., 2006], queries with more retrieved documents will be associated with more examples and the learned function will have a tendency to be biased undesiredly towards those queries. To avoid that, we randomly sample the set of examples associated to each query so as to have exactly N_p examples for each query. Here N_p is an abbreviation of *number of pairs* of examples. The final training set, $\mathcal{T}$, for learning the pairwise classifier is then collected from all instance pairs in $\mathcal{D}^t$ for all queries $q^t \in Q^t$ and their associated pseudo-labels:

$$\mathcal{T} = \{(\mathbf{f}(q^t, d) - \mathbf{f}(q^t, d'), \tilde{z}_{d,d'}); q^t \in Q^t, (d, d') \in \mathcal{D}^t\}$$

The function h is then learned from this set, which contains $N_p \times |Q^t|$ pseudo-labeled vectors, using a standard ranking SVM algorithm [Cao et al., 2006, Herbrich et al., 1999, Joachims, 2002]. It has to be noted however that, because of the fixed number of examples

per query imposed here, the query bias of ranking SVM highlighted in [Cao et al., 2006] is not present.

4.4.3 *A self-learning improvement*

We now introduce the iterative approach we have followed to learn the ranking function h (Eq. 4.3.2) using training data $\mathcal{T}$.

A description of the different steps of the iterative approach is given in Algorithm 4.1. For each query q^t, in the target domain, pseudo-relevance judgments are assigned to N_p random pairs chosen from the retrieved set of q^t (Eq. 4.4.4) based on the grid information obtained from source collection $\mathcal{C}^s$. The pairwise learning/pseudo-labeling steps are then iterated by alternately training a new pairwise classifier on the training set built from all the sets of document pairs and their associated pseudo-labels and assigning pseudo labels to randomly chosen document pairs for queries in the target collection, Q^t. These pseudo-label assignments also follow from Equation 4.4.4, but using this time the pairwise preferences given by the current ranking model rather than the grid. This algorithm is an instance of the discriminant CEM algorithm [McLachlan, 1992, p. 39] and it is easy to show that it converges to a discriminant version of the log-classification-likelihood over document pairs. Other variants of the discriminant CEM algorithm have been used in a more traditional semi-supervised learning setting, and applied to various IR problems [Amini and Gallinari, 2002].

In a typical transfer scenario for IR, e.g. [Cai et al., 2011b], rely on a single source to transfer knowledge. This brings stagnancy in terms of information obtained from source. [Gao et al., 2010] observed that when source is not similar to the target domain the transfer can be highly inefficient. This is experimentally proved in Section 4.6.1. To be more specific, a typical transfer learning algorithm uses same source collection for all the target queries. But this approach prevents one from taking into account the fact that queries are usually different from one another and that different collections generally display different query types. So to harness the complete effectiveness of knowledge transfer we propose a simple extension of Algorithm 4.1 in the following section which takes into account multiple sources to transfer knowledge.

4.4.4 *Transferring relevance information by source selection*

As one can note from Figure 11, the grids obtained from the three collections may differ significantly. As for example grid of `TREC-6` (Figure 4.11(c)) is very much different in shape than the other grids. Moreover, grids can differ on several regions, as for example the small peak exhibited on the `CLEF-3` grid around the (DF, TF) values of 0.1 and 6.5 (rendered whiter in Figure 4.11(d)). Relevance preferences extracted from the grid from `TREC-3`

ALGORITHM 4.1: Transductive Learning to Rank (TLR)

Given :

– A target collection $\mathcal{C}^t$ with a set of queries $Q^t = \{q_1^t, q_2^t, \ldots, q_m^t\}$ with a set of retrieved documents $\mathbf{d}_{q_i^t} = \{d^1_{q_i^t}, d^2_{q_i^t}, \ldots, d^n_{q_i^t}\}$ for each query $q_i^t \in Q^t$;

– $\mathcal{G}_{\mathcal{C}^s}$, grid information obtained from a source collection $\mathcal{C}^s$ (equation 4.4.3);

– Number of pairs of documents to be selected N_p;

– A margin $\delta > 0$;

– The expected precision $\epsilon > 0$;

Initialization:

– $j \leftarrow 0$;

– $\forall q^t \in Q^t$, select N_p pairs at random from $\mathcal{D}_{q^t} = \{(d_{q^t}, d'_{q^t}) \in \mathbf{d}_{q^t} \times \mathbf{d}_{q^t}\}$;

– Set $\mathcal{T}(0) = \{(\mathbf{f}(q^t, d_{q^t}) - \mathbf{f}(q^t, d'_{q^t}), \tilde{z}^{(0)}_{d_{q^t}, d'_{q^t}}); q^t \in Q^t, (d_{q^t}, d'_{q^t}) \in \mathcal{D}_{q^t}\}$ where $\forall (d_{q^t}, d'_{q^t}) \in \mathcal{D}_{q^t}$:

$$\tilde{z}^{(0)}_{d_{q^t}, d'_{q^t}} = \begin{cases} +1 \text{ if } \mathcal{G}_{\mathcal{C}^s}(d_{q^t}) \geq \mathcal{G}_{\mathcal{C}^s}(d'_{q^t}) + \delta \\ -1 \text{ if } \mathcal{G}_{\mathcal{C}^s}(d_{q^t}) \leq \mathcal{G}_{\mathcal{C}^s}(d'_{q^t}) - \delta \end{cases}$$

repeat

– $j \leftarrow j + 1$;

– Train $h^{(j)}_{\omega}$ on $\mathcal{T}(j-1)$;

– $\forall q^t \in Q^t$, select N_p pairs at random from $\mathcal{D}_{q^t} = \{(d_{q^t}, d'_{q^t}) \in \mathbf{d}_{q^t} \times \mathbf{d}_{q^t}\}$;

– Set $\mathcal{T}(j) = \{(\mathbf{f}(q^t, d_{q^t}) - \mathbf{f}(q^t, d'_{q^t}), \tilde{z}^{(j)}_{d_{q^t}, d'_{q^t}}); q^t \in Q^t, (d_{q^t}, d'_{q^t}) \in \mathcal{D}_{q^t}\}$ where $\forall (d_{q^t}, d'_{q^t}) \in \mathcal{D}_{q^t}$:

$$\tilde{z}^{(j)}_{d_{q^t}, d'_{q^t}} = \begin{cases} +1 \text{ if } h^{(j)}_{\omega}(\mathbf{f}(q^t, d_{q^t}) - \mathbf{f}(q^t, d'_{q^t})) \geq \delta \\ -1 \text{ if } h^{(j)}_{\omega}(\mathbf{f}(q^t, d_{q^t}) - \mathbf{f}(q^t, d'_{q^t})) \leq -\delta \end{cases}$$

until $||\omega^{(j)} - \omega^{(j-1)}|| < \epsilon$;

Output : The final ranking function $h^{(j)}_{\omega}$

will miss the behavior of CLEF-3 query-document pairs in this particular region. This is related to the fact, mentioned above, that different collections tend to use different types of queries, which motivates our will to select, for each target query, a source collection that is appropriate for transferring information.

Instead of a single source we now consider here a set of source collections. Suppose $\{\mathcal{C}^{s_1}, \cdots, \mathcal{C}^{s_L}\}$ is a set of L source collections, where i^{th} source $\mathcal{C}^{s_i}$ is composed of a set

of documents $\mathcal{D}^{s_i}$, a set of queries Q^{s_i} and binary relevance judgments for each query in $Q^{s_i}, 1 \leq i \leq L$. Same as before $\mathcal{C}^t$ is a target collection, with a set of documents $\mathcal{D}^t$ and a set of m queries $Q^t = \{q_1^t, q_2^t, \ldots, q_m^t\}$ without any relevance judgments.

Let q^t be a target query. We will first select a source for q^t from the set $\{\mathcal{C}^{s_1}, \cdots, \mathcal{C}^{s_L}\}$ and then transfer information by learning a ranking function as described in Section 4.4.1.

Let q^t be a target query, w a word in q^t and $NTF_1, \cdots, NTF_n$ are n different discrete NTF values (or representative values of each NTF region). As before, $[x_w^d]_{dis}$ denotes the discrete NTF value associated to x_w^d and $[y_w(\mathcal{C})]_{dis}$ the discrete NDF value associated to $y_w(\mathcal{C})$.

For any collection $\mathcal{C} \in \mathcal{C}^t \cup \{\mathcal{C}^{s_1}, \cdots, \mathcal{C}^{s_L}\}$, let us consider the n-dimensional vector $\mathbf{s}(w, \mathcal{C})$. The i^{th} component of this vector corresponds to the proportion of number of documents for which x_w^d falls within the i^{th} discrete NTF value range of $\mathcal{C}$:

$$\mathbf{s}(w, \mathcal{C})_i = \frac{|\{d \in \mathcal{C},\ s.t.\ [x_w^d]_{dis} = NTF_i\}|}{y_w(\mathcal{C})},\ 1 \leq i \leq n$$

One can note that the calculation of the vector $\mathbf{s}(w, \mathcal{C})$ for any collection $\mathcal{C}$ does not involve any relevance information and hence can be calculated even for target collection. $\mathbf{s}(w, \mathcal{C})$ thus indicates how w is distributed in the documents of $\mathcal{C}$[1], an information that can be summarized through the skewness value. The skewness of a vector $\mathbf{s}(w, \mathcal{C})$ is defined by:

$$sk(\mathbf{s}) = \frac{\frac{1}{n}\sum_{i=1}^{n}(s_i - \bar{s})^3}{\left(\frac{1}{n}\sum_{i=1}^{n}(s_i - \bar{s})^2\right)^{\frac{3}{2}}} \tag{4.4.5}$$

where n corresponds to the dimension of the vector, s_i to its i^{th} coordinate and $\bar{s}$ to the mean of its coordinates. Indeed, the skewness measures the asymmetry of an empirical distribution and aims at assessing whether the mass of a distribution is concentrated on the right or the left tail. The more similar two distributions are, similar their shapes will be and also closer their skewness values will be. So the score of a collection $\mathcal{C}$ with respect to the word w is defined as the skewness of the vector $\mathbf{s}(w, \mathcal{C})$. And the score of $\mathcal{C}$ with respect to the whole query q^t is the average or sum of skewness values of each word in q.

If the absolute value of the difference between skewness of two distributions is low, then the distributions have the same shape. Going by this logic, we measure the difference of skewness values of the vector for target collection $\mathbf{s}(w, \mathcal{C}^t)$ and same of all the source collections $\mathbf{s}(w, \mathcal{C}^{s_i})$, $1 \leq i \leq L$. The source having minimum difference displays the closest (NDF, NTF) distribution to the words contained in q^t and is selected as the source

[1] In other words, this vector indicates the spread of w in the documents of $\mathcal{C}$. The $\mathbf{s}$ of notation $\mathbf{s}(w, \mathcal{C})$ is an abbreviation of the word *spread*

for q^t. The source collection used to build pairwise relevance judgments for q^t is thus defined as:

$$\mathcal{C}^s_{q^t} = \underset{\mathcal{C} \in \{\mathcal{C}^{s_1}, \cdots, \mathcal{C}^{s_L}\}}{\operatorname{argmin}} \sum_{w \in q^t} |sk(\mathbf{s}(w, \mathcal{C}^t)) - sk(\mathbf{s}(w, \mathcal{C}))| \tag{4.4.6}$$

With the aid of source selection approach, now the initial training set $\mathcal{T}(0)$ is to be built on the pseudo-labels collected from the selected source for a particular target query. Algorithm 4.1.1 summarizes the steps for querywise source selection which are to be added at the beginning of Algorithm 4.1:

ALGORITHM 4.1.1: Querywise Source Selection

forall the $\forall q^t \in Q^t$ **do**

- Select the source $\mathcal{C}^s_{q^t}$ (Eq. 4.4.6) ;
- Select N_p pairs (d_{q^t}, d'_{q^t}) at random from the retrieved set of q^t ;
- Set $\mathcal{T}(0) = \{(\mathbf{f}(q^t, d_{q^t}) - \mathbf{f}(q^t, d'_{q^t}), \tilde{z}^{(0)}_{d_{q^t}, d'_{q^t}}); q^t \in Q^t\}$ where:

$$\tilde{z}^{(0)}_{d_{q^t}, d'_{q^t}} = \begin{cases} +1 \text{ if } RSV_{\mathcal{C}^s_{q^t}}(q^t, d_{q^t}) \geq RSV_{\mathcal{C}^s_{q^t}}(q^t, d'_{q^t}) + \delta^{(0)} \\ -1 \text{ if } RSV_{\mathcal{C}^s_{q^t}}(q^t, d_{q^t}) \leq RSV_{\mathcal{C}^s_{q^t}}(q^t, d'_{q^t}) - \delta^{(0)} \end{cases}$$

end

Here for each query q^t, in the target collection, its most similar source is first selected (Eq. 4.4.6), then pseudo-relevance judgments are assigned to N_p random pairs chosen from the retrieved set of q^t (Eq. 4.4.4) and the initial training set $\mathcal{T}(0)$ is build. After that the steps are same as described in Algorithm 4.1.

4.5 EXPERIMENTAL SETUP

Various experiments are performed aimed at evaluating to which extend the knowledge transfer presented above can help to learn an efficient ranking function on the target domain. To this end, we considered different collections with different associated query sets, sharing or not the same document sets.

4.5.1 *Collections*

Experiments are done using nine standard IR datasets from the TREC and CLEF[2] evaluation campaigns. Simple statistics of the data sets used are shown in table 19. Among these data sets, TREC-6, TREC-7 and TREC-8 use the same document sets (TREC disks 4 and

[2] http://www.clef-campaign.org

5) but different query sets, whereas WT10G, TREC-3,4,5 CLEF-3 and GOV2 use unique document sets and unique query sets. We appended TREC-9 Web and TREC-10 Web tracks and used the combined track to experiment with WT10G. Similarly TREC-2004 Terabyte and TREC-2005 Terabyte tracks are combined and used for experimenting with GOV2. Collections are indexed using Terrier IR Platform v3.5[Ounis et al., 2006] (http://www.terrier.org). Collections are preprocessed while indexing and the preprocessing steps include stemming using Porter stemmer and removing stop-words using the stopword list provided by Terrier 3.5.

Collection, $\mathcal{C}$	$N(\mathcal{C})$	$l_{avg}(\mathcal{C})$	Index size	$\|Q\|$
GOV2	25,177,217	646	19.6 GB	100
WT10G	1,692,096	398	1.3 GB	100
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	528,155	296	373.0 MB	50
TREC-7				50
TREC-8				50
CLEF-3	169,477	301	126.2 MB	60

Table 19: Statistics of various collections used in our experiments sorted by size.

4.5.2 *Implementing Transfer learning using Grids*

To constitute the grid $\mathcal{G}_{\mathcal{C}^s}$ for a source collection $\mathcal{C}$, we used *labeled* set of queries, along with their associated retrieved documents, from $\mathcal{C}^s$. The list of associated retrieved documents contains all the documents which contain at least one of the query words. We used Terrier to obtain the list. First both NDF and NTF dimensions are discretized where we considered a step of 0.05 along the NDF dimension, and of 0.5 along the NTF dimension (the NDF scores are lower than the NTF scores, hence different scales are used in two dimensions). For every query term $w \in q^s$ such that $q^s \in Q^s$ and for every document $d \in \mathcal{D}^s$ a term-document pair (w, d) can be obtained. A (NDF, NTF) value pair is associated to every (w, d) pair. These (NDF, NTF) value pairs are points in a $NDF - NTF$ space. Every single (NDF, NTF) point is assigned to a discrete region where it belongs to. Furthermore, as in all the test collections we considered, very few terms have NDF values above 0.35 and NTF values above 5, all the data points above these two values are grouped in the same interval. It is checked if the document d involed in the (NDF, NTF) pair is relevant or not to the query q^s where the involved term w belongs. The relevance judgement on the source collection is used in this step. By this processing of the (NDF, NTF) points the number of relevant points as well as total number of points

per discrete region are counted. Once these counts are obtained the $\hat{P}(d \in \mathcal{R}(q)|w;\mathcal{C}^s)$ is calculated using Equation 4.4.2 for each *NDF-NTF* region and the grid is constructed. In the experimental results below, we designate the transfer from a source collection $\mathcal{C}^s$ to a target collection $\mathcal{C}^t$ using the grid built over the source collection, $\mathcal{G}_{\mathcal{C}^s}$, by: $\mathcal{C}^s \overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$.

The proposed transfer learning to rank approach (Algorithm 4.1) is denoted as $\texttt{TLR}_{\texttt{ppr}}$. Here ppr is an abbreviation of *pairwise pseudo-relevance*, as the proposed transfer learning to rank approach here uses source grids to assign a pseudo-relevance between a pair of documents in the target.

For this pseudo-relevance based label assignments (Eq. 4.4.4), we fixed the initial margin $\delta^{(0)}$ to 10% of the diameter of the scoring intervals returned by the aggregation function $RSV_{q^t}^{\mathcal{C}^s}$ (Eq. 4.4.3). This choice was motivated by our observations that the proposed algorithm is not too sensitive to this threshold value, as for all not so high threshold values allowing the constitution of pseudo-labeled pairs we obtain merely the same performance result. The increasing step μ is also fixed to 10% of the scoring intervals returned either by the aggregation or the ranking function learned at each step. The precision ϵ is set to 10^{-3}. We also fixed N_p, the number of document pairs for each query to be added in the training set, to 150. For high value of this margin ($\delta \geq 60\%$ of the diameter of the scoring intervals) it becomes extremely hard and eventually impossible to find N_p document pairs (d, d') per target query with score difference more than δ. But for all target collections we experimented with the margin never reached that high and almost always the algorithm terminated after three or four iterations.

For learning the ranking function, we employed SVM on the pairwise representation of documents using the *SVMLight* [Joachims, 1999] implementation. In order to avoid overfitting the pseudo-label assignments obtained from the grid (Eq.4.4.4) of each of the ranking functions found iteratively in algorithm 4.1, we fixed the hyperparameter C of the SVM to 10^{-4}.

Furthermore, in order to define the attributes, we used the three IR models (BM25, LM, LGD) in the feature vector created for each document-query pair (Eq. 4.3.1). The inherent idea behind this choice is to evaluate in which cases the combination of *standard* scoring functions would be beneficial on a new target collection using the grid. The other vector attributes we considered are standard features used in document retrieval [Nallapati, 2004]. All these features are depicted in Table 20. Note that this list contains two among three types of features mentioned in Section 1.5.1. Features numbered 2, 3 are query independent quantites and rest are quantities related to both query and documents.

In our *transductive* transfer learning setting, we use all the *unlabeled* set of queries, i.e. queries without relevant judgments, and their associated retrieved document lists in the target collection for training. Note that true relevance judgments provided with the target collections are only used for final evaluation. In order to compare the performance of the algorithms we computed the mean average precision (MAP) and the average precision at the 10^{th} document (P@10) across queries.

Features			
1.	$\sum_{w \in q \cap d} \log(1 + t_w^d)$	2.	$\sum_{w \in q \cap d} \log(1 + \frac{l_{\mathcal{C}}}{t_w^{\mathcal{C}}})$
3.	$\sum_{w \in q \cap d} \log(z_w(\mathcal{C}))$	4.	$\sum_{w \in q \cap d} \log(1 + \frac{t_w^d}{l_d})$
5.	$\sum_{w \in q \cap d} \log(1 + \frac{t_w^d}{l_d}.z_w(\mathcal{C}))$	6.	$\sum_{w \in q \cap d} \log(1 + \frac{t_w^d}{l_d}.\frac{l_{\mathcal{C}}}{t_w^{\mathcal{C}}})$
7.	$\texttt{BM25}(q, d)$	8.	$\texttt{LM}(q, d)$
9.	$\texttt{LGD}(q, d)$		

Table 20: Features in the vector representation of (q, d), see table 16 for notations.

Finally, experiments are performed on Terrier IR platform v3.5 as all standard modules are integrated. We implemented our models inside this framework and used other necessary standard modules by Terrier, mainly the indexing and the evaluation components. Some auxiliary shell scripts are used to coordinate between SVMLight and Terrier as for example preparing training data for SVMLight and supplying results produced by SVMLight to Terrier etc.

4.5.3 *Standard IR Models*

To validate the transfer learning to rank approach $\texttt{TLR}_{\texttt{ppr}}$, we considered the following models for comparison:

- The learning to rank approach using training data from the related domain proposed in [Cai et al., 2011b] and denoted as $\texttt{TLR}_{\texttt{qw}}$. Here qw stands for *query weighting*, as in this model the source queries are assigned a weight with respect to the target domain which are eventually used to learn an IR scoring function. This approach is based on a classification criterion, and in the results presented in [Cai et al., 2011b] it came out that the transfer learning for ranking is highly dependent to the source collection. Hence, in our experiments we analyze the impact of a pure ranking approach for transfer learning and also the effect of source selection for this task.
- Furthermore, we considered as baseline models the three standard IR models: language model with Dirichlet smoothing [Jelinek and Marcer, 1980] denoted as LM, BM25 [Robertson and Walker, 1994], and the log-logistic model of the information-based family [Clinchant and Gaussier, 2010], denoted as LGD. For the language model, we restricted ourselves to the Dirichlet smoothing variant as it yielded better results than the other language models on all of the experiments we performed. Furthermore, because relevance information is not available in the target collection, we fixed the hyper-parameters of the these models to their default values provided

within the Terrier IR platform, that is: for BM25, $b = 0.75$, $k_1 = 1.2$ and $k_3 = 8.0$; for LM, the smoothing parameter μ is set to 2500; for LGD, the parameter c is fixed to 1.0.

- The aggregation function found over the grid of a source collection (Eq. 4.4.3) denoted as $\texttt{RSV}_{\mathcal{C}^s}$.

4.6 RESULTS

We start our evaluation by comparing the transfer knowledge based ranking algorithms $\texttt{TLR}_{\texttt{ppr}}$ (Section 4.4) and state-of-the-art standard IR models in Section 4.6.1. This first evaluation will provide a first insight into the effectiveness of cross-domain knowledge transfer for learning to rank. We proceed further by comparing $\texttt{TLR}_{\texttt{ppr}}$ with a state-of-the-art transfer learning algorithm $\texttt{TLR}_{\texttt{qw}}$ proposed by [Cai et al., 2011b] in Section 4.6.2. Then we analyze the effect of different choice of sources (Section 4.6.2.1) and the effect of size of the sources (Section 4.6.3) on knowledge transfer. Finally in Section 4.6.4 we venture the effect of source selection per target query on $\texttt{TLR}_{\texttt{ppr}}$.

4.6.1 *Knowledge transfer for ranking*

We start our evaluation by analyzing the gains provided by our transductive ranking algorithm $\texttt{TLR}_{\texttt{ppr}}$ over standard IR models when we use the grid information from CLEF-3, TREC-3, TREC-4 and TREC-6 taken as source collections. Each time a single source is used for knowledge transfer. We used TREC-7, TREC-8, WT10G and GOV2 as target collections.

We have reported the MAP and P@10 of all the ranking models as well as the ranking provided by the grid itself (using Eq. 4.4.3). Table 21 summarizes these results. We use bold face to indicate the best performance on a target collection. The symbol $^{\downarrow}$ indicates that performance is significantly worse than the best result, according to a Wilcoxon rank sum test at a p-value threshold of 0.05[Rice, 2006].

From these results in Table 21 it becomes clear that:

1. The transfer ranking algorithm $\texttt{TLR}_{\texttt{ppr}}$ consistently improves over other standard IR models on MAP and P@10. Moreover, these results are *robust over the choice of the source collection*.

2. For WT10G and GOV2 the performance of $\texttt{TLR}_{\texttt{ppr}}$ are notably better than standard IR models. For all the source collections, $\texttt{TLR}_{\texttt{ppr}}$ is always significantly better than the the standard IR models on GOV2. However the impact of the choice of source is more on GOV2 in comparison to other target collections. For WT10G, when TREC-4 and TREC-6 is used as source $\texttt{TLR}_{\texttt{ppr}}$ is significantly better. These observations lead to two remarks, firstly $\texttt{TLR}_{\texttt{ppr}}$ seems to perform better on larger target collections

$\texttt{CLEF-3} \overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$

	$\mathcal{C}^t \equiv$ TREC-7		$\mathcal{C}^t \equiv$ TREC-8		$\mathcal{C}^t \equiv$ WT10G		$\mathcal{C}^t \equiv$ GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
$\text{RSV}_{\mathcal{C}^s}$	$0.149^{\downarrow}$	$0.354^{\downarrow}$	$0.214^{\downarrow}$	0.470	$0.147^{\downarrow}$	$0.218^{\downarrow}$	$0.219^{\downarrow}$	$0.443^{\downarrow}$
BM25	$0.183^{\downarrow}$	$0.418^{\downarrow}$	0.241	0.472	$0.184^{\downarrow}$	0.291	$0.272^{\downarrow}$	$0.533^{\downarrow}$
LM	0.186	$0.392^{\downarrow}$	0.240	$0.432^{\downarrow}$	0.204	0.293	$0.278^{\downarrow}$	$0.549^{\downarrow}$
LGD	0.188	0.428	0.255	0.474	0.195	0.287	$0.285^{\downarrow}$	$0.536^{\downarrow}$
TLR_{ppr}	**0.194**	**0.442**	**0.260**	**0.480**	**0.207**	**0.308**	**0.299**	**0.571**

$\texttt{TREC-3} \overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$

	$\mathcal{C}^t \equiv$ TREC-7		$\mathcal{C}^t \equiv$ TREC-8		$\mathcal{C}^t \equiv$ WT10G		$\mathcal{C}^t \equiv$ GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
$\text{RSV}_{\mathcal{C}^s}$	$0.154^{\downarrow}$	$0.384^{\downarrow}$	$0.222^{\downarrow}$	$0.456^{\downarrow}$	$0.156^{\downarrow}$	$0.227^{\downarrow}$	$0.234^{\downarrow}$	$0.453^{\downarrow}$
BM25	$0.183^{\downarrow}$	$0.418^{\downarrow}$	0.241	0.472	$0.184^{\downarrow}$	0.291	$0.272^{\downarrow}$	$0.533^{\downarrow}$
LM	0.186	$0.392^{\downarrow}$	0.240	$0.432^{\downarrow}$	0.204	0.293	$0.278^{\downarrow}$	$0.549^{\downarrow}$
LGD	$0.188^{\downarrow}$	0.428	0.255	0.474	0.195	$0.287^{\downarrow}$	$0.285^{\downarrow}$	$0.536^{\downarrow}$
TLR_{ppr}	**0.196**	**0.442**	**0.262**	**0.476**	**0.208**	**0.311**	**0.306**	**0.576**

$\texttt{TREC-4} \overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$

	$\mathcal{C}^t \equiv$ TREC-7		$\mathcal{C}^t \equiv$ TREC-8		$\mathcal{C}^t \equiv$ WT10G		$\mathcal{C}^t \equiv$ GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
$\text{RSV}_{\mathcal{C}^s}$	$0.156^{\downarrow}$	$0.378^{\downarrow}$	$0.227^{\downarrow}$	$0.450^{\downarrow}$	$0.159^{\downarrow}$	$0.228^{\downarrow}$	$0.239^{\downarrow}$	$0.447^{\downarrow}$
BM25	$0.183^{\downarrow}$	$0.418^{\downarrow}$	0.241	0.472	$0.184^{\downarrow}$	$0.291^{\downarrow}$	$0.272^{\downarrow}$	$0.533^{\downarrow}$
LM	0.186	$0.392^{\downarrow}$	0.240	$0.432^{\downarrow}$	0.204	0.293	$0.278^{\downarrow}$	$0.549^{\downarrow}$
LGD	0.188	0.428	0.255	0.474	$0.195^{\downarrow}$	$0.287^{\downarrow}$	$0.285^{\downarrow}$	$0.536^{\downarrow}$
TLR_{ppr}	**0.193**	**0.456**	**0.261**	**0.474**	**0.207**	**0.315**	**0.309**	**0.579**

$\texttt{TREC-6} \overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$

	$\mathcal{C}^t \equiv$ TREC-7		$\mathcal{C}^t \equiv$ TREC-8		$\mathcal{C}^t \equiv$ WT10G		$\mathcal{C}^t \equiv$ GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
$\text{RSV}_{\mathcal{C}^s}$	$0.155^{\downarrow}$	$0.390^{\downarrow}$	$0.224^{\downarrow}$	$0.448^{\downarrow}$	$0.145^{\downarrow}$	$0.211^{\downarrow}$	$0.223^{\downarrow}$	$0.434^{\downarrow}$
BM25	$0.183^{\downarrow}$	$0.418^{\downarrow}$	0.241	0.472	$0.184^{\downarrow}$	$0.291^{\downarrow}$	$0.272^{\downarrow}$	$0.533^{\downarrow}$
LM	0.186	$0.392^{\downarrow}$	$0.240^{\downarrow}$	$0.432^{\downarrow}$	0.204	0.293	$0.278^{\downarrow}$	$0.549^{\downarrow}$
LGD	$0.188^{\downarrow}$	0.428	0.255	0.474	$0.195^{\downarrow}$	0.287	$0.285^{\downarrow}$	$0.536^{\downarrow}$
TLR_{ppr}	**0.196**	**0.446**	**0.262**	**0.476**	**0.208**	**0.308**	**0.301**	**0.562**

Table 21: MAP and P@10 measures on different target collections when using CLEF-3, TREC-3, TREC-4 and TREC-6 as source data sets respectively. The best result is in bold, and a $\downarrow$ indicates a result that is statistically significantly worse than the best, according to a Wilcoxon rank sum test at a p-value threshold of 0.05.

and secondly when given a choice to select a source per query (Algorithm 4.1.1) probably the most affected target is GOV2.

3. Finally, the MAP and P@10 measures of rankings obtained from the scores derived from the grid, $\text{RSV}_{\mathcal{C}^s}$ (Eq. 4.4.3) are in the same range even though below the performance (in terms of MAP and P@10) of standard IR models in most cases. Moreover, for almost all source target combinations presented here TLR_{ppr} performs significantly better than $\text{RSV}_{\mathcal{C}^s}$. This fact suggests that the self-learning improvement over the grid scores for pairwise learning helps to boost the performance significantly.

The consistent improvements obtained with TLR_{ppr} show that it is possible to learn an efficient combination of state-of-the-art IR scoring functions from the relevance judgments provided by a source collection. The information provided by the grid and the state-of-the-art IR scoring functions is thus complementary, which is in line with the preliminary analysis we did on the ranking correlations between the grid scores and the baseline models (Section 4.4.1, Table 18). However, our results also show that the transfer learning based approach proposed here is somewhat robust on the choice of the source collection used. But it also revealed that venturing the effect of choice of source on larger collection may be interesting, as well as the effect of the size and the diversity in the source collection.

4.6.2 *The effect of a ranking criterion to build pseudo-pairs on the target collection*

Now we compare TLR_{ppr} with TLR_{qw}, proposed in [Cai et al., 2011b]. TLR_{qw} is based on classification criterion to constitute the pseudo relevance-pairs to learn the ranking model on the target collection. Table 22, shows MAP results on TREC-7, TREC-8, WT10G and GOV2 taken as target collections using CLEF-3, TREC-3, TREC-4 and TREC-6 respectively as source collection for knowledge transfer.

For all the cases except one (on GOV2 using TREC-6 as source) TLR_{ppr} performs significantly better than TLR_{qw}, especially when TREC-4 is used as source. From these results, it can be seen that the performance of TLR_{qw}, on a given target collection, varies significantly depending on the source collection in use. These results also suggest that the efficiency of transfer learning may highly depend on the adequacy of the source collection with respect to the target one, and failure to identify a proper source for transfer may result in *negative transfer* [Pan and Yang, 2010].

TLR_{qw} first classifies between source and target domain query-document pairs by learning a hyperplane and then predicts the weights of source queries with respect to target domain based on the distances of the source query-document pairs from this hyperplane. Then TLR_{qw} incorporates the weights into source ranking model to rank documents in target domain. So the knowledge is transferred through the model using a classification criterion, making TLR_{qw} extremely sensitive to the similarity of source and target domain. This is evident in the performance of TLR_{qw} when TREC-4 is source. On the other hand

		Targets ($\mathcal{C}^t$)			
		TREC-7	TREC-8	WT10G	GOV2
CLEF-3 $\overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$	TLR_{qw}	0.186	0.253	$0.177^{\downarrow}$	$0.271^{\downarrow}$
	TLR_{ppr}	**0.194**	**0.260**	**0.208**	**0.299**
TREC-3 $\overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$	TLR_{qw}	$0.163^{\downarrow}$	$0.218^{\downarrow}$	$0.189^{\downarrow}$	$0.276^{\downarrow}$
	TLR_{ppr}	**0.196**	**0.262**	**0.208**	**0.306**
TREC-4 $\overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$	TLR_{qw}	$0.077^{\downarrow}$	$0.099^{\downarrow}$	$0.091^{\downarrow}$	$0.163^{\downarrow}$
	TLR_{ppr}	**0.193**	**0.261**	**0.207**	**0.309**
TREC-6 $\overset{\mathcal{G}_{\mathcal{C}^s}}{\curvearrowright} \mathcal{C}^t$	TLR_{qw}	0.189	0.259	0.205	**0.302**
	TLR_{ppr}	**0.196**	**0.262**	**0.208**	0.301

Table 22: Comparison between the knowledge transfer based ranking algorithms TLR_{ppr} (Section 4.4) and TLR_{qw}. MAP measures are shown on four different collections taken as target collections and when only one fixed source is used. Best results are shown in bold, and a $\downarrow$ indicates a result that is statistically significantly worse than the best, according to a Wilcoxon rank sum test at a p-value threshold of 0.05.

TLR_{ppr} uses the source grid for building a ranking model in the target domain itself, thus incorporating a pure ranking approach, making it less susceptible to the similarity of source and target domains.

4.6.2.1 *The effect of the distance between collections*

As seen in table 21, TLR_{ppr} is robust to the choice of sources. But TLR_{ppr} deploys a self-learning iterative algorithm on the information obtained based on the source grid and it is evident from the same table that the performance of the original grid based $\text{RSV}_{\mathcal{C}^s}$ depends highly on the selected source. In general $\text{RSV}_{\mathcal{C}^s}$ performs worse when CLEF-3 is used as source and performs better when TREC-4 is used as source. When TREC-6 is source, $\text{RSV}_{\mathcal{C}^s}$ performs better on smaller targets but is significantly worse on larger targets. A natural explanation for this is that TREC-6 collection, and more precisely the grid one can construct from it, is better suited for TREC-7, TREC-8 than for WT10G, GOV2.

To understand the effect of a source on knowledge transfer, first a similarity measure between collections is required. As described previously, a grid is a summary of a collection. So the comparison between the shapes of two grids should represent the comparison between the underlying collections. Going by this intuition, we now describe similarity measures between collections which are computed from the corresponding grids.

Following observations can be made while transferring knowledge based on grids:

- A grid constitutes 88 discrete regions over normalized (NDF, NTF) space; 8 discrete values are considered along the normalized DF axis and 11 along the normalized TF

axis. Each region has a corresponding proportion value of relevant documents in that particular region (Eq. 4.4.2). These proportions carry the relevance information that is getting transferred from the source to the target. When the shape of a grid is referred, actually the shape provided by these values are considered.

- Each discrete *NDF* region constitutes a number of query terms (i.e. the terms having normalized DF that falls in that region) that contributed to that portion of the grid. One can note that, $\texttt{RSV}_{\mathcal{C}^s}$ (hence $\texttt{TLR}_{\texttt{ppr}}$ as well) fetches information from a source based on target query terms (Algorithm 4.1). Only those portion of a source grid contribute to the transfer where some target query terms exist. Other portions of the source grid is redundant to the target under consideration. Hence, when measuring similarity between a source and a target grid, different regions should be weighted based on the number of target query terms present in that region.
- Each grid region actually corresponds to a certain number of documents (both relevant and non relevant) that actually contains the terms having the corresponding normalized DF and TF values for that region. The proportion values of relevant documents are actually an estimate of the probability that document d is relevant to target query q^t. Hence, more the number of documents in a region, better would be the estimate. Similarly, the ratio of number of documents in a region of the source grid against the target grid determines the relative effectiveness of the estimate. Larger this ratio, better would be the estimation. Hence each region should be weighted by the number of document ratio between source and target to measure the similarity between the grids.

Taking into account the above observations, we now define a similarity measure between a source and a target grid, namely $\mathcal{G}_\text{s}$ and $\mathcal{G}_\text{t}$. Suppose $\mathcal{G}_\text{s}(i,j)$ (or $\mathcal{G}_\text{t}(i,j)$) is the proportion of relevant documents in the region corresponding to i^{th} discrete normalized DF region and j^{th} discrete normalized TF region of grid $\mathcal{G}_\text{s}$ (or $\mathcal{G}_\text{t}$), where $1 \leq i \leq 8$ and $1 \leq j \leq 11$. Normalized number of terms present in the i^{th} discrete normalized DF region of $\mathcal{G}_\text{t}$ is denoted by $\#_n\texttt{terms-}\mathcal{G}_\text{t}(i)$. Number of documents present in the region corresponding to i^{th} discrete normalized DF region and j^{th} discrete normalized TF region of grid $\mathcal{G}_\text{s}$ (or $\mathcal{G}_\text{t}$) is denoted by $\texttt{\#docs-}\mathcal{G}_\text{s}(i,j)$ (or $\texttt{\#docs-}\mathcal{G}_\text{t}(i,j)$). So we define the distance between $\mathcal{G}_\text{s}$ and $\mathcal{G}_\text{t}$ as:

$$\text{dist}(\mathcal{G}_\text{t}, \mathcal{G}_\text{s}) = \sqrt{\sum_{i=1}^{8}\sum_{j=1}^{11}\left(\mathcal{G}_\text{t}(i,j) - \mathcal{G}_\text{s}(i,j)\right)^2 \times \#_n\texttt{terms-}\mathcal{G}_\text{t}(i) \times \frac{\texttt{\#docs-}\mathcal{G}_\text{t}(i,j)}{\texttt{\#docs-}\mathcal{G}_\text{s}(i,j)}}$$

This distance is calculated by an euclidean distance of grid values between the same regions of two grids weighted by number of target query terms present in the region and ratio of number of documents in that region. One can note that the number of document ratio between the source grid and the target grid is proportional to the similarity between two grids, thus inversely proportional to the distance. Hence in the distance equation above the reverse of the ratio is considered.

We now define similarity between $\mathcal{G}_t$ and $\mathcal{G}_s$ as the inverse of their distances :

$$\mathsf{sim}(\mathcal{G}_t, \mathcal{G}_s) = \frac{1}{\mathsf{dist}(\mathcal{G}_t, \mathcal{G}_s)} \tag{4.6.1}$$

The weighted similarity measure given by the above equation between different source (CLEF-3, TREC-3, TREC-4, TREC-6) and target grids (TREC-7, TREC-8, WT10G, GOV2) used here is presented in Table 23. Note that here relevant judgemnt on different target collections are used to measure the similarity. But at the same time, it is only for analysis purpose by which we are trying to understand the behaviour of transfer learning algorithms.

	TREC-7	TREC-8	WT10G	GOV2
CLEF-3	0.9	1.3	0.9	0.2
TREC-3	2.4	3.6	1.7	0.3
TREC-4	2.5	3.8	2.4	0.4
TREC-6	2.5	2.1	0.4	0.1

Table 23: Weighted similarity measures between different 3D grids calculated by Equation 4.6.1.

These results are consistent with the performance of the grid scores $\mathtt{RSV}_{\mathcal{C}^s}$ on target collections that we considered in our experiments. Indeed, as shown in table 21, the more a source collection is similar to a target collection (Table 23), the higher is the MAP performance of $\mathtt{RSV}_{\mathcal{C}^s}$ for those source and target collections. One can note that though $\mathtt{RSV}_{\mathcal{C}^s}$ is susceptible to the similarity of source and target collections, but final ranking function learned by the self-learning iterative algorithm is able to shed off this dependency.

4.6.3 *The effect of size and diversity in the source collection*

Here we analyze the behavior of $\mathtt{TLR}_{\mathtt{ppr}}$ for growing amounts of queries and their associated relevance judgments in the source collection.

Let $\mathcal{C}^1$ and $\mathcal{C}^2$ be two collections with document sets $\mathcal{D}^1$ and $\mathcal{D}^2$ and query sets Q^1 and Q^2 respectively. *Augmentation* of two collections $\mathcal{C}^1$ and $\mathcal{C}^2$ (denoted by $\mathcal{C}^1 \oplus \mathcal{C}^2$) means merging their document sets ($\mathcal{D}^{12} = \mathcal{D}^1 \cup \mathcal{D}^2$) and appending the associated query sets ($Q^{12} = \{Q^1, Q^2\}$). This eventually leads to the merging of the retrieved document sets by the queries in Q^{12} as well as merging of corresponding relevance judgements. Clearly $\mathcal{C}^1 \oplus \mathcal{C}^2$ itself is a collection with a document set $\mathcal{D}^{12}$ and query set Q^{12} along with relevance judgements. Thus it is possible to build a grid using $\mathcal{C}^1 \oplus \mathcal{C}^2$.

We first start experimenting with the CLEF-3 collection as the source, then we advance with CLEF-3⊕TREC-3 as source and finally CLEF-3⊕TREC-3⊕TREC-6 as source. At each

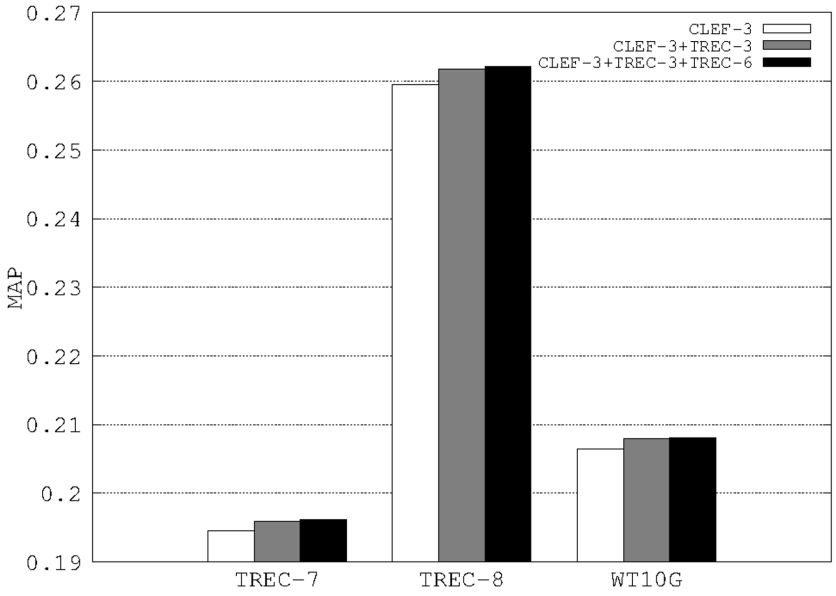


Figure 12: Evolution of MAP on TREC-7, TREC-8 and WT10G taken as target collections; when the grid is built up gradually over CLEF-3 (□), CLEF-3 ⊕ TREC-3 (■), and, CLEF-3 ⊕ TREC-3 ⊕ TREC-6 (■). ⊕ represents the merging operation.

step a new grid is build from the corresponding augmented source collection to transfer knowledge. Figure 12 illustrates the evolution of MAP for these successive source augmentations on TREC-7, TREC-8 and WT10G taken as target collections.

On all these three target collections, one can notice that the improvements of MAP for the TLR_{ppr} model when augmenting CLEF-3 with TREC-3 are more substantial than when augmenting TREC-6 to the set of previously augmented source data sets, even though TREC-6 is 5 times larger in size than CLEF-3 (Table 19). To analyze the reason behind it we caluculated the weighted similarities (Eq. 4.6.1) between the grids from TREC-7, TREC-8 and WT10G and those obtained after augmenting incrementally the three source collections, which are CLEF-3, CLEF-3⊕TREC-3 and CLEF-3⊕TREC-3⊕ TREC-6. Table 24 shows the calculated weighted similarities.

	TREC-7	TREC-8	WT10G
CLEF-3	0.9	1.3	0.9
CLEF-3 ⊕ TREC-3	2.6	3.8	2.0
CLEF-3 ⊕ TREC-3 ⊕ TREC-6	3.7	4.8	1.9

Table 24: Weighted similarity measures calculated by Equation 4.6.1 between different 3D grids obtained after augmeting incrementally CLEF-3, TREC-3 and TREC-6 source collections. ⊕ represents the merging operation.

As it can be seen, the grids corresponding to different target collections and the incrementally augmented source collections become more and more closer as more source collections are added. There is a jump in the similarity after the first augmentation of (CLEF-3 and TREC-3) over CLEF-3 alone which is inline with the first jump of MAP values in Figure 12. The highest improvement here is thus obtained through the merging of two dissimilar collections (CLEF-3 and TREC-3) and not through the addition of many more queries from TREC-6. This suggests that the diversity of the source collections has more impact in the transfer of pairwise preferences on the target collections than the size of the source data sets.

4.6.4 *The effect of source selection*

In Section 4.6.1 we discussed how the choice of source may or may not affect the performance of TLR_{ppr} on different target collections. In this section we further explore this idea using an extension of TLR_{ppr} in the form of source selection (Algorithm 4.1.1). This extension is denoted by $\text{TLR}^{\text{ss}}_{\text{ppr}}$, where the ss is the abbreviation of *source selection*.

We now use TREC-3, TREC-4, TREC-5, TREC-6 and CLEF-3 as the pool of source collections from which each target query can select the most appropriate source for knowledge transfer using Algorithm 4.1.1. We measured the MAP and P@10 of all the models on TREC-7, TREC-8, WT10G and GOV2. Note that for TLR_{qw} the source selection step is not trivial to carry out as the transfer is done as a whole from a source to a target instead of individual target queries. For comparison, we included best MAP performance for TLR_{ppr} on each target collection over different sources (Table 21). Table 25 summarizes these results. We use bold face to indicate the best performances. The symbol $^{\downarrow}$ (respectively $^{\Downarrow}$) indicates that performance is significantly worse than our approach, according to a Wilcoxon rank sum test used at a p-value threshold of 0.05 (respectively 0.01) [Rice, 2006].

	$\mathcal{C}^t \equiv$ TREC-7		$\mathcal{C}^t \equiv$ TREC-8		$\mathcal{C}^t \equiv$ WT10G		$\mathcal{C}^t \equiv$ GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
BM25	$0.183^{\downarrow}$	$0.418^{\Downarrow}$	$0.243^{\Downarrow}$	0.472	$0.184^{\Downarrow}$	$0.291^{\downarrow}$	$0.272^{\Downarrow}$	$0.533^{\Downarrow}$
LM	0.186	$0.392^{\Downarrow}$	$0.240^{\downarrow}$	$0.432^{\Downarrow}$	0.204	$0.293^{\downarrow}$	$0.278^{\Downarrow}$	$0.549^{\Downarrow}$
LGD	$0.188^{\downarrow}$	0.428	$0.255^{\Downarrow}$	0.474	$0.195^{\downarrow}$	$0.287^{\downarrow}$	$0.285^{\Downarrow}$	$0.536^{\Downarrow}$
TLR_{ppr}	0.196	**0.446**	0.262	**0.476**	0.208	0.311	0.309	0.579
$\text{TLR}^{\text{ss}}_{\text{ppr}}$	**0.197**	**0.446**	**0.263**	0.468	**0.210**	**0.318**	**0.312**	**0.582**

Table 25: MAP and P@10 measures on different target collections where query wise source data sets are selected from CLEF-3, TREC-3,4,5,6. The best results are in bold, and a $^{\downarrow}$ (respectively $^{\Downarrow}$) indicates a result that is statistically significantly worse than the best result, according to a Wilcoxon rank sum test used at a p-value threshold of 0.05 (respectively 0.01).

Following points can be observed from these results:

1. Like the original $\texttt{TLR}_{\texttt{ppr}}$ algorithm, the source selection extension $\texttt{TLR}^{\texttt{ss}}_{\texttt{ppr}}$ also improves consistently and significantly over other standard IR models on MAP and P@10 in most cases.

2. $\texttt{TLR}^{\texttt{ss}}_{\texttt{ppr}}$ is almost always slightly better than the original $\texttt{TLR}_{\texttt{ppr}}$ but not significantly better. This again proves that $\texttt{TLR}_{\texttt{ppr}}$ is robust against the choice of source selection, and that the main improvement comes from the constitution of pseudo-labeled pairs which is performed using the ranking strategy of (Eq. 4.4.4).

3. Among different targets, $\texttt{TLR}^{\texttt{ss}}_{\texttt{ppr}}$ gets maximum improvement over $\texttt{TLR}_{\texttt{ppr}}$ on GOV2, thus implying source selection has maximum impact on this target. This is inline with the observation in Section 4.6.1 (Table 21) where the impact of the choice of source is more notable on GOV2 in comparison to other target collections.

4.7 CONCLUSION

We have studied in this chapter the problem of learning a ranking function from collections without relevance information. To do so, we have used a transfer learning approach by which we try to derive relative pseudo-relevance judgments in a target collection from absolute relevance judgments available in the source collection. This derivation relies on a grid that associates to each (NDF, NTF) value of a term in a query-document pair and a relevance score, which is then combined over all query terms. A ranking SVM system is then deployed on the obtained relative pseudo-relevance judgments, and further improved through a self-learning mechanism. The experiments we have conducted show that the ranking function obtained in this way consistently and significantly outperforms state-of-the-art IR ranking functions in the majority of cases on a whole range of TREC collections with respect to the MAP and the P@10 measures.

Our approach directly learns a ranking function on the target collection as opposed to previous approaches developed in the same setting and which learned the ranking function on re-weighted version of the source collection. Thus this approach allows the method to adapt to more complex setting, as well as it can be coupled with easy in hand transfer strategies for ranking. We employed a simple source selection procedure in order to choose the best associated source collection for each query in a target collection. Compared to a state-of-the-art transfer learning approach for ranking, our results suggest that the central significant step to make the transfer work is to use a ranking strategy to create a first pool of pseudo-labeled pairs for each query in the target collection.

The conventionally used term "learning a function" actually means learning the weights associated with the variables of a function whose form is predefined. As for example, in learning to rank scenario (as well as the case we studied here) ranking SVM

[Herbrich et al., 1999] assumes a linear form as given in Equation 4.3.2, which again comes from defining a minimization problem over the hinge loss function. Though use of different kernels (e.g. polynomial, gaussian etc.) enables to assume a non-linear form, but these kernels themselves have predefined forms which is again imposed on the learned ranking function. Similar is true for any standard ranking algorithm. To this end the question is, whether it is possible to learn a ranking function which has no restriction on its form. This question leads us to the next chapter where we investigate this problem.

5

EXPLORING THE SPACE OF IR FUNCTIONS

In this chapter we propose an approach to search and discover formulas for IR ranking from a space of simple formulas. In general all IR ranking models are based on two basic variables, namely, term frequency and document frequency. Here a grammar for generating all possible formulas are defined which consists of two above said variables and basic mathematical operations - addition, subtraction, multiplication, division, logarithm, exponential and square root. The large set of formulas generated by this grammar are filtered by checking mathematical feasibility and satisfiability to heuristic constraints on IR scoring functions proposed by the community. Obtained candidate formulas are tested on various standard IR collections and several simple but promising scoring formulas are identified. We show that these newly discovered formulas are outperforming other state-of-the-art IR scoring models through extensive experimentation on several IR collections. We also compare the performance of formulas satisfying IR constraints and those which do not, where the earlier set of formulas clearly outperforms the later set. In doing so we also empirically validate these constraints.

5.1 INTRODUCTION

Developing new term-document scoring functions that outperform already existing traditional scoring schemes is one of the most interesting and popular research area in theoretical information retrieval (IR). Many state-of-the-art IR scoring schemes have been developed since the dawn of IR research, such as the vector space model [Salton and McGill, 1983], the language model [Ponte and Croft, 1998], BM25 [Robertson and Zaragoza, 2009], and, more recently, the HMM [Metzler and Croft, 2005], DFR [Amati and van Rijsbergen, 2002a], information-based models [Clinchant and Gaussier, 2010] and learning to rank [Liu, 2009]. All these scoring schemes were developed along one what could be called a "theoretical line", in which theoretical principles guide the development of the scoring function, and then assessed on different standard IR test collections. There is however a chance, or more precisely a risk, that some high performing scoring schemes will not come into light through such an approach, as they are not so intuitive and/or are not so easily explainable theoretically. Quoting [Fan et al., 2004]:

> *T*here is no guarantee that existing ranking functions are the best/optimal ones available. It seems likely that more powerful functions are yet to be discovered.

The motivation of finding the *best* or *optimal* IR scoring function has led the researchers to explore the space of IR functions in a more systematic way. But, such attempts have always been limited by the complexity of search space with regard to the current computational power. One can note that the search space is of infinite dimension containing potentially all real functions. The first attempts to this exploration were based on genetic programming and genetic algorithms, which were seen as a way to automatically learn IR functions by exploring parts of the solution space stochastically [Gordon, 1988, Pathak et al., 2000, Cummins and O'Riordan, 2006b]. Even though all of these methods enlarge the space of scoring functions, they are still limited in two aspects: first, they usually assume that the IR scoring function takes a particular form (e.g. linear or polynomial), and they require some training set in order to learn the parameters of the function given a particular collection. Two questions, directly addressed in the current study, thus remain open:

a) Is it possible to explore the space of IR scoring functions in a more systematic (i.e. exhaustive) way?

b) Is it possible to find a function that behaves well on all (or most) collections, and thus dispenses from re-training the function each time a new collection is considered?

To answer those questions, we introduce an automatic discovery approach based on the systematic exploration of a search space of simple closed-form mathematical functions. This approach is inspired from the work of [Maes et al., 2011] on multi-armed bandit problems and is here coupled with the use of heuristic IR constraints [Fang et al., 2004] to prune the search space and so, limiting the computational requirements. Such a possibility was mentioned in [Cummins and O'Riordan, 2006b] but has not been tried to the best of our knowledge.

We performed extensive experiments on `CLEF-3`, `TREC-3`, `TREC-5`, `TREC-6`, `TREC-7`, `TREC-8`, `WT10G` and `GOV2` to evaluate the performance of the scoring functions discovered by our search strategy. We show that these functions are simple yet effective on most of the collections and performs significantly better than other state-of-the-art IR scoring functions. While going through the search space we also stored the functions that does not comply with the heuristic IR constraints and compared them with the functions that satisfies those constraints. This experiments show that later set of functions significantly outperforms the first set, thus empirically validating the heuristic IR constraints.

Rest of the chapter is organized as follows: Section 5.2 discusses the previous work done to explore the space of IR functions and places our work with respect to the previous approaches. Section 5.3 introduces the function generation process using the search strategy we have followed and Section 5.4 describes the method of selecting best performing functions from the pool of generated functions. Sections 5.5 and 5.6 present the experiments and results obtained. Finally, Section 5.7 concludes the chapter.

5.2 RELATED WORK

Use of heuristic and intelligent search algorithms in the field of IR is not naive. The first attempts to this exploration were based on genetic algorithm [Goldberg, 1989] and genetic programming [Koza, 1992]. Genetic algorithm is a heuristic optimization strategy inspired by the principles of biological evaluation. It considers each solution to be an individual and iteratively tries to find the optimal solution by means of genetic operations (crossover and mutation) on the existing population to create more diverse and efficient generation of population. Genetic programming is an extension of genetic algorithm where each individual is essentilly a program (or algorithm) to solve the problem. It applies same techniques to evolve a program which can best perform a given task based on some fitness criteria.

First direct application of genetic algorithm in IR comes in the form of finding suitable document descriptions and in general, modifying the document indexing [Gordon, 1988, Blair, 1990]. This approach considers multiple complete descriptions of a single document so that it can be found on the basis of the description that best describes the document with respect to a query. Genetic algorithm is used to create a new generation of descriptions from the existing ones on the basis of feedback generated from past queries which enables to have better retrieval results for a group of users searching for similar topics. [Vrajitoru, 1998] has analyzed the effect of crossover operation of genetic algorithm on the model described by [Gordon, 1988] and has shown that the proper choice of crossover operator can significantly improve the basic model. This similar approach was used by [Gordon, 1991], where the genetically generated document subject descriptions were used to form clusters of co-relevant documents for a set of queries. This document clustering problem was also addressed by [Raghavan and Agarwal, 1987]. [Yang et al., 1992] presents another direction where genetic algorithm is applied to the query space to find the optimal query term weights inside a relevance feedback framework. Genetic algorithm was applied by [Petry et al., 1994] to build and modify weighted Boolean queries to improve the performance in terms of precision and recall. Moreover, they also studied the effect of fitness function on the overall performance. [Chen, 1995] assumed documents to be individuals (chromosomes) and binary presence of keywords (0 or 1) as genes and used genetic algorithm to optimize these keywords so that the system can suggest most relevant document to the user.

Above mentioned genetic algorithm based techniques concentrates mainly on document and query representations and their modifications. [Pathak et al., 2000] attempted to use genetic algorithm directly to the retrieval procedure by means of learning the *best* weights of a linear weighted combination of different scoring (or matching) functions which yielded promising results. A similar problem was studied by [Billhardt et al., 2002], where genetic algorithm was used in two fold. First to select a set of candidate matching functions (or *retrieval experts*) from a pool of different matching functions, and then to find the weight of these candidate functions in a linear combination. This brings us to the point

of finding the ranking function itself which can be directly applied as a retrieval model. But genetic algorithm by definition can only operate on individuals with a fixed-length string description which disables it to explore the function space where functions of different lengths can be involved.

One major advantage of genetic programming over genetic algorithm is the flexibility of data representation. Individuals considered by genetic programming are not constrained by the fixed length representaion. [Fan et al., 2000] was the first to apply genetic programming to explore the IR function space and the problem was revisited by a later work in [Fan et al., 2004]. Each function is represented by a tree (more specifically parse-tree) structure which enables the easy parsing and implementation. Functions are composed of literals, i.e. some basic IR variables (e.g. *tf*, *df* etc.) and operations ($+$, $\times$, / and log). A set of random functions are generated to serve as initial population. Some queries and corresponding labeled documents are used for training and cross validaion and performance of each generated scoring function is measured in terms of *average precision*. This measured performance is used as the fitness value. Iterative genetic programming method is deployed on the initial population which involves two genetic operations. Firstly, *reproduction*, which copies top 10% funcions of the current generation with respect to the fitness value to the next generation without any alteration. Secondly, *crossover*, where some randomly selected functions from the new generation exchange subtrees. This ensures that the new genration contains functions different from the parent generation. [Fan et al., 2004] simulated this method for thirty generation and selected the best performing function based on the performance over the cross validation set. This method is summrazied in Figure 13

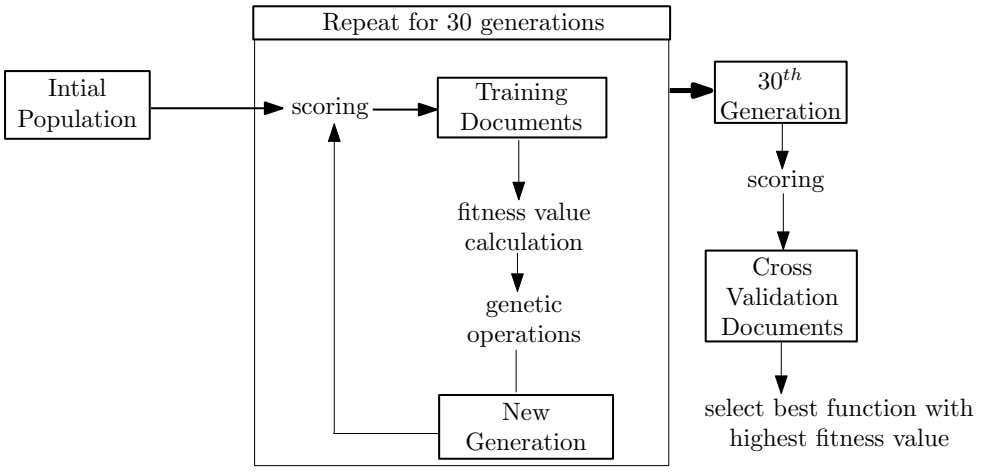


Figure 13: Overview of genetic programming based approach to ranking function discovery problem

The same problem was studied by [Oren, 2002] where mutation operation was also used along with reproduction and crossover. [Cummins and O'Riordan, 2006b] attempted to explore the search space more systematically by evolving scoring functions in lo-

cal (within-document) and global (collection-wide) domains and later combining them. [Cummins and O'Riordan, 2006a] uses a similar strategy by considering a term weighting scheme to have three parts, a global part, a term-frequency part and a normalization part. These techniques allow to search each part of the term-weighting scheme separately, thus reducing the complexity of the process. These approaches show the power of automated function discovery using machine intelligence tools. But, being a non-deterministic method, the solutions generated by these genetic programming based approaches are often difficult to analyze. [Cummins and O'Riordan, 2005] tries to analyze the weighting schemes generated by this approach and [Cummins and O'Riordan, 2006a] provides metrices to measure distances between rank lists generated by different generated solutions and thus explaining their position in the solution space.

More recently, researchers have focused on particular function forms as linear combinations or well-defined kernel functions, the parameters of which are learned from some training data. This approach has been highly successful in IR, through the various "learning to rank" methods proposed so far: pointwise, e.g. [Crammer and Singer, 2001], pairwise, e.g. [Cohen et al., 1999, Freund et al., 2003, Joachims, 2002], or list wise approaches, e.g. [Valizadegan et al., 2009]. [Yeh et al., 2007] applied genetic programming to learning to rank scenario where an efficient ranking function is searched given the pairwise training data. But, as pointed out in Section 4.7 of previous chapter, this approach of learning a ranking function involves finding out weights of variables of functions with a certain form.

Here another search strategy is proposed which explores the search space exhaustively but systematically. This search technique to find a suitable solution for a particular problem is not novel in computer science. This approach was first introduced to find solution to the multi-arm-bandit problem [Maes et al., 2011, Maes et al., 2012]. [Castronovo et al., 2012] used this technique to find solutions for Markov decision problems.

The difference between the automated strategy explained in this chapter and the genetic programming based approach are mainly two folds. As mentioned above, solutions yielded by genetic programming based methods are often very complex because of its non-deterministic nature. Moreover, there is another issue associated with genetic programming, namely "code bloat" – the fact that almost all genetic programming algorithms have a tendency to produce larger and larger functions along the iterations. As for example, the consensus scoring function reported in [Fan et al., 2004] for a TREC ad-hoc task is as follows:

$$\frac{\log\left(t^{d}_{\mathbf{w}} \times \left(t^{avg}_{\mathbf{w}} + \frac{t^{d}_{\mathbf{w}}}{\log((t^{d}_{\mathbf{w}})^{2} \times t^{avg}_{\mathbf{w}})} + \frac{t^{d}_{\mathbf{w}} \times N}{\mathcal{N}_{\mathbf{w}}} \times \frac{t^{avg}_{\mathbf{w}} \times (t^{max}_{\mathbf{w}} + uq)}{\mathcal{N}_{\mathbf{w}}}\right)\right)}{uq + 2 \times t^{max}_{\mathbf{w}} + 0.373}$$

where, $t^{d}_{\mathbf{w}}$ is term frequency, $\mathcal{N}_{\mathbf{w}}$ is document frequency, $t^{avg}_{\mathbf{w}}$ is the average term frequency within the document, $t^{max}_{\mathbf{w}}$ is the maximum term frequency in the entire collection, N is the number of documents in the collection and uq is the number of unique term in the document. Thus there is a high risk of missing simple functions of high quality.

Whereas our method is deterministic and explores the entire search space till a given length (explained in 5.3.4), thus it does not miss simple yet effective solutions. Genetic programming enforces a selection step based on the fitness function (in most literature MAP or similar evaluation measure) at every generation which in some sense is optimization based on some already labeled data (relevance judgment). Our approach initially generates the set of all suitable candidate functions and later test them directly, hence involving no learning step. Moreover, our approach uses the heuristic IR constraints [Fang et al., 2004] to restrict the solution space at the beginning to get only *good* candidate functions and which in turn limits the computational requirements. This also enables us to study the effect of these constraints on the retrieval system. [Zheng and Fang, 2009] and [Fang and Zhai, 2011] experimentally evaluated the performances of IR models based on the heuristic IR constraints. We also provide an empirical proof of the effectiveness of these constraints in the light of the framework presented here.

5.3 FUNCTION GENERATION

Retrieval status value (RSV) scores a document $\mathbf{d}$ with respect to a query $\mathbf{q}$. It in turn does a weighted scoring of the document in question with respect to every term $\mathbf{w}$ in the query $\mathbf{q}$.

$$RSV(\mathbf{d},\mathbf{q}) = \sum_{\mathbf{w}\in\mathbf{q}} t_{\mathbf{w}}^{\mathbf{q}} \quad g(\mathbf{d},\mathbf{w})$$

The function g assigns a positive score to the document with respect to every query term. We call $g(\mathbf{d},\mathbf{w})$ a *scoring function*. In this section we present the proposed discovery strategy for function generation and the validity verification steps that we deploy to find a set of candidate scoring functions from the space of IR functions. All the notations are summarized in Table 26.

Notation	Description
$t_{\mathbf{w}}^{\mathbf{d}}$	term frequency of term $\mathbf{w}$ in document $\mathbf{d}$
$t_{\mathbf{w}}^{\mathbf{q}}$	number of occurrences of term $\mathbf{w}$ in query $\mathbf{q}$
$x_{\mathbf{w}}^{\mathbf{d}}$	normalized version of term frequency
$\mathcal{N}_{\mathbf{w}}$	document frequency of term $\mathbf{w}$
$y_{\mathbf{w}}$	normalized version of document frequency
$\mathcal{N}$	number of documents in a given collection
$l_{\mathbf{d}}$	length of document $\mathbf{d}$ in number of terms
l_{avg}	average length of documents in a given collection

Table 26: Notations

5.3.1 *Variables*

All classical IR scoring functions to score a document **d** with respect to a query term **w** consists of two basic variables:

- The *term frequency* $t^{\mathbf{d}}_{\mathbf{w}}$ of term **w** in document **d** is defined as the number of times that **w** occurs in **d**.
- The *document frequency* $\mathcal{N}_{\mathbf{w}}$ of a term **w** is the number of documents in the collection that **w** occurs in.

However it is well known that normalized versions of these variables yield better results. There exist numerous normalization schemes available in literature, for example language models use relative term counts [Ponte and Croft, 1998] and the BM25 model uses the Okapi normalization [Robertson and Zaragoza, 2009]. For this work, we selected a common scheme, which is the one used in DFR and in information based models [Clinchant and Gaussier, 2010].

Thus, we consider the following variables:

1. Normalized term frequency given by:

$$x^{\mathbf{d}}_{\mathbf{w}} = t^{\mathbf{d}}_{\mathbf{w}} \log\left(1 + c\frac{l_{avg}}{l_{\mathbf{d}}}\right) \tag{5.3.1}$$

   Here $c \in \mathbb{R}$ is a multiplying factor. Note that this variable incorporates both $t^{\mathbf{d}}_{\mathbf{w}}$ and $l_{\mathbf{d}}$. For simplicity, unless otherwise stated it is written as x from now on;

2. Normalized document frequency given by:

$$y_{\mathbf{w}} = \frac{\mathcal{N}_{\mathbf{w}}}{\mathcal{N}}$$

   For simplicity, unless otherwise stated it is written as y from now on;

3. A constant real valued parameter $k \in \mathbb{R}$.

5.3.2 *Operations*

A scoring function can be seen as a combination of the basic quantities x, y and k, and some unary and binary mathematical operations. The operations considered here are most commonly used in all standard existing IR scoring functions. Again, this list is modifiable and the proposed approach is independent of the operations used.

Here in this work unary operations can be logarithm, exponentiation with respect to e ($exp()$), square root or unary negation. Binary operations can be any four basic mathematical operations (addition, subtraction, multiplication, division) or exponentiation with

respect to any real number. Variables and operations considered here are summarized in Table 27.

Variables	normalized tf (x), normalized df (y), constant (k)	
Operations	binary:	$+, -, \times, \div, pow$
	unary:	$log, exp, sqrt$, unary negation

Table 27: Variables and operations considered.

5.3.3 *The Grammar*

A *grammar* in a formal language is a finite set of recursive rewriting rules (called *productions*) to generate strings in the formal language under consideration. The set of rules are used to generate strings using the *alphabet* of the language so that the generated strings follow the *syntax* of the language. The *alphabet* (conventionaly denoted by Σ) of a language is a non-empty set of symbols from which the strings are generated. The symbols in the alphabet set Σ are often called *terminal symbols* as they appear in the strings. On the other hand another set (conventionaly denoted N) of auxiliary symbols is also considered, which are called *nonterminal symbols*. These are used as placeholders for patterns of terminal symbols and gets replaced by the group of terminal symbols it represents according to a production rule. Formally a grammar $\mathcal{G}$ is defined by a four tuple (N, Σ, P, S), where N and Σ are set of nonterminal and the alphabet respectively as described above. P is the finite set of production rules and S is a special nonterminal called *start symbol* which acts as the initial symbol to start with to generate a string by the grammar.

A grammar is called a *context free grammar* if all the production rules of P are of the form $X \to \omega$ where X is a single nonterminal ($X \in N$) and ω is a string of terminals and nonterminals ($\omega \subseteq \Sigma \cup N$). Context free grammars are particularly very useful to formalize mathematical formulas.

Here we use a context free grammar to generate syntactically correct functions. We define a grammar $\mathcal{G}$ as following:

- $N = \{\mathbb{B}(\cdot,\cdot), \mathbb{U}(\cdot), \mathbb{S}, g\}$
- $\Sigma = \{x, y, k, +, -, \times, \div, pow, log, exp, sqrt, -(\cdot)$
- $S = g$
- P is described by Figure 14.

The $-(\cdot)$ signifies the unary negation operation (e.g. $-y$). Thus, by this grammar $\mathcal{G}$ a function g may be a binary expression $\mathbb{B}(g, g)$, or a unary expression $\mathbb{U}(g)$, or a symbol $\mathbb{S}$.

$$
\begin{aligned}
g &::= \mathbb{B}(g, g) \mid \mathbb{U}(g) \mid \mathbb{S} \\
\mathbb{B} &::= + \mid - \mid \times \mid \div \mid pow \\
\mathbb{U} &::= log \mid exp \mid sqrt \mid -(.) \\
\mathbb{S} &::= x \mid y \mid k
\end{aligned}
$$

Figure 14: Grammar $\mathcal{G}$ to generate scoring functions

5.3.4 *Validity Verification*

Grammar $\mathcal{G}$ generates functions which are syntactically sound. After this first combination of the variables and operations we look at the validity of the generated functions. This validity verification is a four-step process.

1. *Presence of both x and y*: We consider it to be mandatory that an IR scoring function must contain both basic variables, x and y. This imposition is supported by the fact that the scoring functions either with only term frequency or with only document frequency perform poorly with respect to functions containing both of them. All functions with only one of these variables are rejected. One can note that the presence of the constant k is kept optional.

2. *Domain of definition*: This step ensures that a function generated by the grammar is mathematically well defined, that is to verify that all the operations used in a function are well defined. Bad operations include logarithm or square root of a negative number and division by zero.

3. *Positiveness*: Here we check that a generated function is non-negative valued. We restrict ourselves only to non-negative functions because it prunes the search space and concentrates on scoring functions which assign non-negative scores to documents. We can always get negative valued functions from a non-negative valued function either by multiplying it with -1 or subtracting a negative constant from it.

4. *IR constraints*: Heuristic IR constraints [Fang et al., 2004] proposed by the community guides how a "good" IR scoring function should behave analytically. These constraints were discussed in Section 1.4. In this step it is checked if a generated function satisfies the heuristic IR constraints or not. For ease of implementation and interpretebality we use the analytical form of the constraints proposed in

[Clinchant and Gaussier, 2011, Clinchant and Gaussier, 2010]. That is for the generated function $g(x, y)$, we must have[1]:

$$\frac{\partial g}{\partial t^d_{\mathbf{w}}} > 0 \qquad \text{tf effect}$$
$$\frac{\partial^2 g}{(\partial t^d_{\mathbf{w}})^2} < 0 \qquad \text{concavity effect}$$
$$\frac{\partial g}{\partial \mathcal{N}_{\mathbf{w}}} < 0 \qquad \text{df effect}$$
$$\frac{\partial g}{\partial l_{\mathbf{d}}} < 0 \qquad \text{doc-len effect}$$

From the definition of x (Eq. 5.3.1), taking derivative with respect to $t^d_{\mathbf{w}}$ we have:

$$\frac{\partial x}{\partial t^d_{\mathbf{w}}} = \log\left(1 + c\frac{l_{avg}}{l_{\mathbf{d}}}\right)$$

Here $c, l_{avg}, l_{\mathbf{d}} > 0$, thus $\left(1 + c\frac{l_{avg}}{l_{\mathbf{d}}}\right) > 1$ giving right hand side of the equation always positive. That is $\frac{\partial x}{\partial t^d_{\mathbf{w}}} > 0$. So starting from the *tf effect* given above, using chain rule for derivatives we have the following:

$$\frac{\partial g}{\partial t^d_{\mathbf{w}}} = \frac{\partial g}{\partial x}.\frac{\partial x}{\partial t^d_{\mathbf{w}}}$$

Since $\frac{\partial x}{\partial t^d_{\mathbf{w}}} > 0$, hence to make $\frac{\partial g}{\partial t^d_{\mathbf{w}}} > 0$ we need:

$$\frac{\partial g}{\partial x} > 0 \tag{C1}$$

Applying *Faà di Bruno's formula* for chain rule to the second order derivative, from the *concavity effect* we have:

$$\frac{\partial^2 g}{\partial (t^d_{\mathbf{w}})^2} = \frac{\partial^2 g}{\partial x^2}.\left(\frac{\partial g}{\partial x}\right)^2 + \frac{\partial g}{\partial x}.\frac{\partial^2 x}{\partial (t^d_{\mathbf{w}})^2}$$

From the definition of x (Eq. 5.3.1), $\frac{\partial^2 x}{\partial (t^d_{\mathbf{w}})^2} = 0$. Hence the above derivative is reduced to:

$$\frac{\partial^2 g}{\partial (t^d_{\mathbf{w}})^2} = \frac{\partial^2 g}{\partial x^2}.\left(\frac{\partial g}{\partial x}\right)^2$$

But if g satisfies C1, we also have $\left(\frac{\partial g}{\partial x}\right)^2 > 0$. Thus the derivative $\frac{\partial^2 g}{\partial (t^d_{\mathbf{w}})^2}$ will be negative if and only if:

$$\frac{\partial^2 g}{\partial x^2} < 0 \tag{C2}$$

[1] for the ease of readability we have used g for the function $g(x, y)$

Now from defintion of y, taking derivative with respect to $\mathcal{N}_{\mathbf{w}}$: $\frac{\partial y}{\partial \mathcal{N}_{\mathbf{w}}} = \frac{1}{\mathcal{N}}$. Since $\mathcal{N} > 0$, we always have $\frac{\partial y}{\partial \mathcal{N}_{\mathbf{w}}} > 0$. Starting form the *df effect* above:

$$\frac{\partial g}{\partial \mathcal{N}_{\mathbf{w}}} = \frac{\partial g(x,y)}{\partial y} . \frac{\partial y}{\partial \mathcal{N}_{\mathbf{w}}}$$

It has been shown that $\frac{\partial y}{\partial \mathcal{N}_{\mathbf{w}}} > 0$, hence to have left hand side of the equation to be negative it must be ensured that:

$$\frac{\partial g}{\partial y} < 0 \tag{C3}$$

Again from definition of x taking derivative with respect to $l_{\mathbf{d}}$:

$$\frac{\partial x}{\partial l_{\mathbf{d}}} = -t^d_{\mathbf{w}} \frac{1}{\left(1 + c\frac{l_{avg}}{l_{\mathbf{d}}}\right)} \left(c\frac{l_{avg}}{l_{\mathbf{d}}}\right)$$

which is always negative as $t^d_{\mathbf{w}}, c, l_{avg}, l_{\mathbf{d}} > 0$. Now starting from the *doc-len effect* we have:

$$\frac{\partial g}{\partial l_{\mathbf{d}}} = \frac{\partial g}{\partial x} . \frac{\partial x}{\partial l_{\mathbf{d}}}$$

We already proved that $\frac{\partial x}{\partial l_{\mathbf{d}}} < 0$. So to have $\frac{\partial g}{\partial l_{\mathbf{d}}} < 0$ we must ensure that $\frac{\partial g}{\partial x} > 0$ which is already ensured by C1.

To summarize, it is sufficient to check if g satisfies the following three constraints (denoted by C1, C2 and C3):

$$\underbrace{\frac{\partial g}{\partial x} > 0}_{\text{C1}}, \quad \underbrace{\frac{\partial^2 g}{\partial x^2} < 0}_{\text{C2}}, \quad \underbrace{\frac{\partial g}{\partial y} < 0}_{\text{C3}}$$

For every generated function g it is mandatory to pass all four validity verification steps. Then only it is considered well defined and eligible to be used as an IR scoring function. Even if the function fails to satisfy only one step, it is considered non-valid and is not considered in any later steps.

5.3.5 *Generating Candidate Functions*

Before going into the details of the function generation procedure, two concepts need to be defined:

Definition. *The length of a function is defined as the number of symbols or operators present in that function.*

As for example the function $sqrt(x/y)$ has a length 4, where $sqrt()$ and the *division* are two operators and x, y are two symbols present. Similarly $sqrt(x) * exp(-y)$ has a length 6. One can note that the length of a function loosely reflects the complexity of the function. More length it has, more complex it is.

Definition. *A function generated by grammar $\mathcal{G}$ is said to be a* candidate scoring function *if it survives all the validity verification steps described earlier.*

ALGORITHM 5.3.1: Generating candidate scoring functions

Input : maximum length l_{max}, the grammar $\mathcal{G}$

Output :
- set of candidate functions $\mathcal{C}_V$ till l_{max}
- set of functions which do not satisfy *heuristic IR constraints* but pass other two validity tests, $\mathcal{C}_N$

Initialization : $\mathcal{C}_V \leftarrow \{\ \}$, $\mathcal{C}_N \leftarrow \{\ \}$, $\mathcal{S} \leftarrow \{\ \}$

for $l_{curr} \in \{1, 2, \ldots, l_{max}\}$ **do**
  repeat
    A new function $g_{|\mathcal{S}|+1}$ is created by any of the following rules:
      - Create a symbol (variable or constant): $g_{|\mathcal{S}|+1} = x, y$ or k
      - Take unary operation: $g_{|\mathcal{S}|+1} = \mathbb{U}(g_i)$, $i \in [1, |\mathcal{S}|]$
      - Take a binary operation: $g_{|\mathcal{S}|+1} = \mathbb{B}(g_i, g_j)$, $i, j \in [1, |\mathcal{S}|]$

    $\mathcal{S} \leftarrow \mathcal{S} \cup \{g_{|\mathcal{S}|+1}\}$

    if $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* presence of both x and y *test* ***AND*** $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* domain of definition *test* ***AND*** $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* positiveness **then**
      if $g_{|\mathcal{S}|+1}(x, y)$ *satisfies* heuristic IR constraints **then**
        $\mathcal{C}_V \leftarrow \mathcal{C}_V \cup \{g_{|\mathcal{S}|+1}\}$
      else
        $\mathcal{C}_N \leftarrow \mathcal{C}_N \cup \{g_{|\mathcal{S}|+1}\}$
      end
    end
  until *all the functions till l_{curr} in $\mathcal{S}_\mathcal{G}$ are generated*;
end

A iterative length-limited strategy is used here to generate the set of all candidate scoring functions till length l_{max}. The set is denoted by $\mathcal{C}_V$. Complete procedure is summarized in Algorithm 5.3.1. The algorithm works as follows. Suppose $\mathcal{S}_\mathcal{G}$ is the set of all possible functions generated by grammar $\mathcal{G}$. In a particular iteration $\mathcal{S} \subset \mathcal{S}_\mathcal{G}$ is the set of already generated functions with a length less than or equal to l_{curr} where $l_{curr} < l_{max}$ and $\mathcal{S} = \{g_1, g_2, \ldots, g_{|\mathcal{S}|}\}$. Next iteration expands the set $\mathcal{S}$ by creating new functions. A new function $g_{|\mathcal{S}|+1}$ is created either by starting with a symbol that is S, or taking any unary operation of any function $g_i \in \mathcal{S}$ that is $\mathbb{U}(g_i)$, or by taking any binary operation of two functions $g_i, g_j \in \mathcal{S}$ that is $\mathbb{B}(g_i, g_j)$. As for example, starting from an initial empty set $\mathcal{S} = \{\}$, the function $sqrt(x/y)$ is generated by the following steps:

$$(g_1 = x) \rightarrow (g_2 = y) \rightarrow (g_3 = g_1/g_2) \rightarrow (g_4 = sqrt(g_3))$$

Once a new function $g_{|\mathcal{S}|+1}$ is generated its validity is checked. If it passes all four steps, it is included in the set of generated candidate scoring functions $\mathcal{C}_V$, otherwise it is rejected. Note that if $g_{|\mathcal{S}|+1}$ is rejected it is still kept in the set $\mathcal{S}$. Because $g_{|\mathcal{S}|+1}$ itself may not be valid but later in the procedure it can contribute to generate a valid candidate function. As for example, $-x$ does not satisfies "presence of both x and y" step as well as "positiveness" step, but $log(-x + \frac{x+y}{y})$ is a valid candidate function which contains $-x$. A pictorial representation of the algorithm is presented in Figure 15.

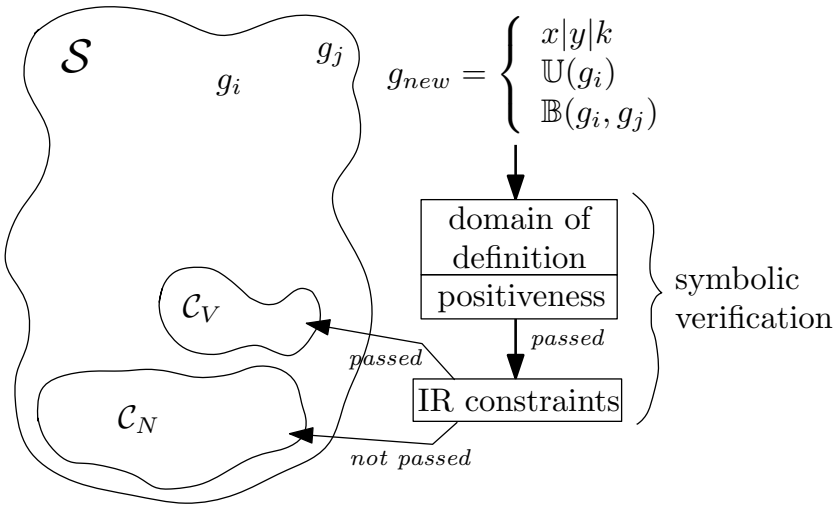


Figure 15: Method to generate the set of candidate scoring functions using the grammar $\mathcal{G}$.

5.3.6 *Generating Functions Based on IR Constraint Satisfaction*

One can note that for the purpose of our experimental study, Algorithm 5.3.1 also stores the functions until length l_{max} which do not satisfy heuristic IR constraints but otherwise are valid (denoted by $\mathcal{C}_N$). By adopting some simple changes to the algorithm it is not difficult to get sets of functions satisfying only one of the IR constraints among three. This will help study the effect of individual constraints in more detail. Let us assume that the functions in set $\mathcal{C}_N^i$ satisfies the i^{th} constraint only.

Algorithm 5.3.2 shows the necessary modifications. If a newly generated function $g_{|\mathcal{S}|+1}$ successfully passes through first three validity verification steps and all three IR constraints (C1, C2 and C3), then $g_{|\mathcal{S}|+1}$ is included in $\mathcal{C}_V$. Otherwise the Algorithm 5.3.2 considers each of the IR constraints separately instead of treating them as a single module like Algorithm 5.3.1. As for example if $g_{|\mathcal{S}|+1}$ satisfy constraint C1 only it is included $\mathcal{C}_N^1$. One can note that the sets $\mathcal{C}_N^1$, $\mathcal{C}_N^2$ and $\mathcal{C}_N^3$ may not be disjoint, as a function may satisfy two

constraints but not all three. Finally, if $g_{|\mathcal{S}|+1}$ does not satisfy either C1 or C2 or C3, is it included in $\mathcal{C}_N$. The rest of the algorithm remains same.

ALGORITHM 5.3.2: Constraintwise Function Generation

if $g_{|\mathcal{S}|+1}(x,y)$ *satisfies* presence of both x and y *test* ***AND*** $g_{|\mathcal{S}|+1}(x,y)$ *satisfies* domain of definition *test* ***AND*** $g_{|\mathcal{S}|+1}(x,y)$ *satisfies* positiveness **then**

  if $g_{|\mathcal{S}|+1}(x,y)$ *satisfies all three* heuristic IR constraints **then**

    $\mathcal{C}_V \leftarrow \mathcal{C}_V \cup \{g_{|\mathcal{S}|+1}\}$

  end

  if $g_{|\mathcal{S}|+1}(x,y)$ *satisfies only the constraint C1:* $\frac{\partial g(x,y)}{\partial x} > 0$ **then**

    $\mathcal{C}_N^1 \leftarrow \mathcal{C}_N^1 \cup \{g_{|\mathcal{S}|+1}\}$

  end

  if $g_{|\mathcal{S}|+1}(x,y)$ *satisfies only the constraint C2:* $\frac{\partial^2 g(x,y)}{\partial x^2} < 0$ **then**

    $\mathcal{C}_N^2 \leftarrow \mathcal{C}_N^2 \cup \{g_{|\mathcal{S}|+1}\}$

  end

  if $g_{|\mathcal{S}|+1}(x,y)$ *satisfies only the constraint C3:* $\frac{\partial g(x,y)}{\partial y} < 0$ **then**

    $\mathcal{C}_N^3 \leftarrow \mathcal{C}_N^3 \cup \{g_{|\mathcal{S}|+1}\}$

  end

  if $g_{|\mathcal{S}|+1}(x,y)$ *does not satisfies any of the* heuristic IR constraints **then**

    $\mathcal{C}_N \leftarrow \mathcal{C}_N \cup \{g_{|\mathcal{S}|+1}\}$

  end

end

5.4 FUNCTION SELECTION

Algorithm 5.3.1 has been applied to generate functions till length 8. Thus in this work we assume $l_{max} = 8$. Table 28 shows the number of candidate functions available at each length till length 8 and also the corresponding generation time. The total numbers of valid and non-valid functions (i.e. $|\mathcal{C}_V \cup \mathcal{C}_N|$) are also shown. One can note that a large number of functions (up to 97%) got discarded for not satisfying IR constraints. These numbers suggest how effective the constraints are to prune the search space. Another point to note is that functions with length less than 4 did not pass the four steps of validity verification.

There are a total of 5407 valid functions from length 4 to length 8. Testing all these functions and getting the best performing functions on all standard IR collections is very time consuming, especially on large collections like `WT10G` and `GOV2` (details in Section 5.5.1). Hence we chose a simple strategy. `CLEF-3` is the smallest among the collections used

length	# of valid and non-valid functions $\lvert\mathcal{C}_V \cup \mathcal{C}_N\rvert$	# of candidate functions $\lvert\mathcal{C}_V\rvert$	generation time
4	42	2	≈1 sec
5	328	10	≈1 min
6	2378	100	≈5 min
7	16447	638	≈30 min
8	49989	4657	≈1 day

Table 28: Number of candidate functions and generation times for different lengths.

here (Table 29). Initially all 5407 functions are tested on CLEF-3. The 500 best performing functions, among 5407, on CLEF-3 are selected with respect to the MAP measure. These 500 functions are further tested on the remaining collections. This strategy is justified because it is very much unlikely that a function appearing lower than 500 top positions with respect to CLEF-3 will come up to a top position in other collections. These experiments are performed under the default parameter value settings.

After the initial run on CLEF-3, the top 500 functions are analyzed over the TREC-3,5,6,7,8 collections to obtain the best performing ones. Further experiments are done on WT10G, GOV2 and with default as well as tuned parameter setting (explained in Section 5.5.2). Details of these findings are discussed in Section 5.6.

In the classification task in machine learning, while a classifier is learned, at every step the model is made a bit more detailed and more training cases covered. That is the model is refined incrementally using the training data as the source of reference. However, there is a possibility that the algorithm reaches such a level of refinement where it simply describes specific random features of the data (or noise) instead of the actual underlying relationship. Thus the model becomes sensitive to the training data and performs poorly on any unseen data. This problem is known as *overfitting*. When the model is a complex one it has a greater chance of refinement and hence is more susceptible to overfitting.

The selection strategy proposed above selects the functions by optimizing their performance in terms of MAP on CLEF-3. One can thus wonder whether there is a risk of overfitting the CLEF-3 collection by selecting the top 500 functions on it. The risk increases even more as the lengths of the functions are increased. Because functions with higher length are more complex and hence more prone to overfitting.

From the machine learning theory, we know that the risk of overfitting is present as soon as one is dealing with infinite sets of functions. We are here considering a finite set of functions, consisting of 5407 different functions, thus limiting overfitting on any collection. Furthermore, the capacity of a classification model is defined using *Vapnik-Chervonenkis dimension* or *VC dimension* and it is related to how complex the model can be. The VC dimension of any finite set of functions is always finite and the empirical risk minimization for families of functions with finite VC dimension is consistent [Vapnik, 1995,

Vapnik, 2000]. This suggests that the behavior of a function, from the finite set retained, on a collection with enough queries is a good indicator of the behavior in general.

One can note that, irrespective of the considered length, set of candidate functions $\mathcal{C}_V$ is always finite. Thus the selection strategy we proposed here does not suffer from the problem of overfitting.

5.5 EXPERIMENTAL SETUP

We conducted a number of experiments aimed at validating the discovered candidate functions. We looked into the behaviour of IR constraints by studying the functions which do or do not respect those constraints. We also compare these functions with respect to classical IR models.

5.5.1 *Collections*

The candidate functions are tested using seven standard IR collections which includes one from CLEF (www.clef-campaign.org) and six from TREC (trec.nist.gov) collections. Basic statistics of the collections used are provided in Table 29.

We appended TREC-9 Web and TREC-10 Web tracks and used the combined track to experiment with WT10G. Similarly TREC-2004 Terabyte and TREC-2005 Terabyte tracks are combined and used for experimenting with GOV2. These collections are indexed using Terrier IR Platform v3.5 (terrier.org). Preprocessing steps in creating an index include stemming using Porter stemmer and removing stop-words using the stopword list provided by Terrier 3.5.

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
TREC-3	741,856	261	427.7 MB	50
TREC-5	524,929	339	378.0 MB	50
TREC-6	528,155	296	373.0 MB	50
TREC-7				50
TREC-8				50
CLEF-3	169,477	301	126.2 MB	60

Table 29: Statistics of various collections used in our experiments, sorted by size.

5.5.2 *Standard IR Models and Parameter Values*

For comparison purpose three standard IR models are used, namely Okapi BM25 (denoted by BM25) [Robertson and Walker, 1994], language model with Dirichlet prior (denoted by LM) [Jelinek and Marcer, 1980] and log-logistic model of information model family [Clinchant and Gaussier, 2010]. These models are used with two different settings for the values of the free parameters.

- In *default parameters* setting, values of the free parameters are the values provided as default in Terrier. That is $b = 0.75, k_1 = 1.2, k_3 = 8.0$ for BM25, $\mu = 2500$ for LM and $c = 1.0$ for LGD.
- For *tuned parameters* version, the free parameters are tuned from a set of values using 5 fold cross validation. The sets of values used in this tuning are given by Table 30. The query set is sequentially partitioned into 5 subsets. Of the 5 subsets, a single subset is retained for testing the model, and the remaining four subsets are used as training the parameters of each model (b and k_1 for BM25, μ for LM and c for LGD). The cross-validation process is then repeated 5 times, with each of the 5 subsets used exactly once for testing. Each time a query-wise average precision and precision at 10 documents is calculated for each set. After 5 folds, average precision of all the queries are obtained and Mean Average Precision (MAP) is calculated. Similarly average of precision at 10 documents for each query is obtained and average of the quantity (P@10) is reported.

b	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
c	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
μ	10, 25, 50, 75, 100, 200, 300, 400, 500, 600, 700, 800, 900, 1000, 1500, 2000, 2500, 3000, 4000, 5000, 10000

Table 30: Set of values of different parameters used for tuning standard IR models through 5 fold cross validation.

5.5.3 *Implementing Function Generation*

Algorithm 5.3.1 has been implemented using *python* (www.python.org). The symbolic mathematics library of python, *SymPy* (sympy.org) combined with some other simulations is used to symbolically verify *presence of both x and y*, *domain of definition*, *positiveness* and *heuristic IR constraints*. For testing candidate functions, retrieval simulations are performed on Terrier IR platform v3.5 (terrier.org) as all standard modules are integrated. We implemented our functions inside this framework and used other necessary standard modules by Terrier, mainly the indexing and the evaluation components. *Mean average precision* (MAP) and *presicion at 10 documents* (P@10) measures are used for evaluation.

To check the signs of a given expression 1000 random combinations of non-negative values of x and y are used. The functions and their first and second derivatives are evaluated with these random values using *SymPy*. If for all the combinations of the values of x and y an expression evaluates to be positive then the expression is considered to be positive valued. On the other hand, if the expression evaluates to be negative for all combinations, then it is considered to be negative valued. This technique is used to verify the *positiveness* step as well as the signs of the derivatives in the *heuristic IR constraints* verification step.

For the discovered functions, we assume $k = 1.0$. One can note that in the definion of x (Section 5.3.1) a real valued free parameter c is included. This is the same c used in the term frequency normalization of log-logistic model. Hence for both the versions of parameter settings value of c is fixed according to the rules used for LGD. That is for the *default parameters* version we have taken $c = 1.0$, and for tuned version same 5 fold cross validation method is used over the set of values of c as given in Table 30. Initial selection of functions (Section 5.4) on CLEF-3 is performed with default parameter value settings. 5 fold cross validation method is used to compare the position of the discovered functions with respect to other standard IR models when the parameters are optimized (Section 5.6.3).

As symbolic libraries are used for generating functions, sometimes two generated functions may be symbolically different but analytically identical. As for exmple, following two valid functions generated by Algorithm 5.3.1 are analytically identical but symbolically different:

$$\sqrt{\frac{\sqrt{xy}}{y}} \quad \text{and} \quad \sqrt{\sqrt{\frac{x}{y}}}$$

Primarily these type of functions can be detected if they yield exactly same MAP and P@10 in the initial run on CLEF-3. Note that functions producing same MAP but different P@10 (or other way round) are not considered as identical. But having same MAP and P@10 is a necessary condition for two functions to be identical, not sufficient, as there may exist non-identical functions having same MAP and P@10. Once two possible identical functions are detected, again 1000 random combinations of non-negative values of x and y are used. If for both the functions in question yield exactly same value for all 1000 combinations then they are declared identical. In that case only one of the functions is preserved and the other one is removed from consideration.

5.6 RESULTS

We first begin our investigation over the comparison of functions which satisfy IR constraints and functions which do not (Section 5.6.1). This will help us understand the effectiveness of the heuristic IR constraints. Then we compare valid candidate functions with classical IR functions with default parameter values (Section 5.6.2) as well as tuned

parameter values (Section 5.6.3). Finally we compare some best discovered functions with the genetic algorithm based approaches (Section 5.6.5).

5.6.1 *Effectiveness of IR Constraints*

From Algorithm 5.3.1, two sets of functions are produced, namely the set of all valid candidate functions ($\mathcal{C}_V$) and the set of functions which do not satisfy heuristic IR constraints, but are valid otherwise ($\mathcal{C}_N$) (Section 5.3.5). These sets are used to emperically justify the usefullness of heuristic IR constraints by comparing the performances of functions from $\mathcal{C}_V$ and functions from $\mathcal{C}_N$.

From each of the sets $\mathcal{C}_V$ and $\mathcal{C}_N$, 10 subsets are created. Each subset contains 100 randomly selected sample functions chosen from the initial set ($\mathcal{C}_V$ or $\mathcal{C}_N$). When creating a subset, 100 functions are selected without replacement. When creating another different subset, again all functions are considered for selection. That is a function may repeat itself in a different subset but never within the same subset. These samples are tested on `CLEF-3` and `TREC-3,5,6,7,8`. For each function `MAP` and `P@10` are noted and they are averaged over all 100 functions within a single sample set. Finally, average performance over 10 sample sets are reported.

Datasets	`MAP`		`P@10`	
	$\mathcal{C}_N$	$\mathcal{C}_V$	$\mathcal{C}_N$	$\mathcal{C}_V$
CLEF-3	$0.162^{\downarrow}$	**0.307**	$0.131^{\downarrow}$	**0.238**
TREC-3	$0.045^{\downarrow}$	**0.151**	$0.093^{\downarrow}$	**0.303**
TREC-5	$0.019^{\downarrow}$	**0.077**	$0.036^{\downarrow}$	**0.141**
TREC-6	$0.104^{\downarrow}$	**0.163**	$0.144^{\downarrow}$	**0.272**
TREC-7	$0.063^{\downarrow}$	**0.123**	$0.139^{\downarrow}$	**0.269**
TREC-8	$0.083^{\downarrow}$	**0.164**	$0.152^{\downarrow}$	**0.295**

Table 31: Average `MAP` and `P@10` of the set of valid of $\mathcal{C}_V$ and non-valid $\mathcal{C}_N$ functions. The best results are in bold, and a $^{\downarrow}$ indicates a result that is statistically significantly worse than the best result according to a Wilcoxon rank sum test used at a p-value threshold of 0.05.

Table 31 shows the average `MAP` and `P@10` of 10 sample sets drawn from both $\mathcal{C}_V$ and $\mathcal{C}_N$ over these collections. As it can be seen the average `MAP` measure of functions in $\mathcal{C}_V$ are 6% to 14% higher than the `MAP` measures of functions in $\mathcal{C}_N$, and the average `P@10` measures of $\mathcal{C}_V$ are 10% to 21% higher than that of the set $\mathcal{C}_N$. These results empirically validate the effectiveness of IR constraints, i.e. IR scoring functions following these constraints yields better performance. This finding is in line with other empirical studies which aimed to test the validity of these constraints [Cummins and O'Riordan, 2009b, Cummins and O'Riordan, 2009a].

Above experiments analyze all the constraints as a single module. Performances are compared between funtions which satisfy *all* the constraints against functions which satisfy *none* of them. Now we move to examine the effect of each constraint separately. Let us assume the set $\mathcal{C}_N^i$ contains functions which satisfy only i^{th} constraint. Algorithm 5.3.2 generates three separate sets of functions: $\mathcal{C}_N^1$, $\mathcal{C}_N^2$ and $\mathcal{C}_N^3$ which satisfy either C1 or C2 or C3 respectively.

An initial intuition can be made by the sizes of the sets $\mathcal{C}_N^1$, $\mathcal{C}_N^2$ and $\mathcal{C}_N^3$. Figure 16 shows the number of functions in each of these sets till length 8. It is clearly visible from the plot that constraint C3 is the harshest one, where C1 is the loosest one. Because the number of functions satisfying C3 is the minimum among the three, whereas the number of functions satisfying C1 is the maximum. Another very trivial yet interesting observation is that $\mathcal{C}_N$ is the biggest set and $\mathcal{C}_V$ is the smallest one among all five sets.

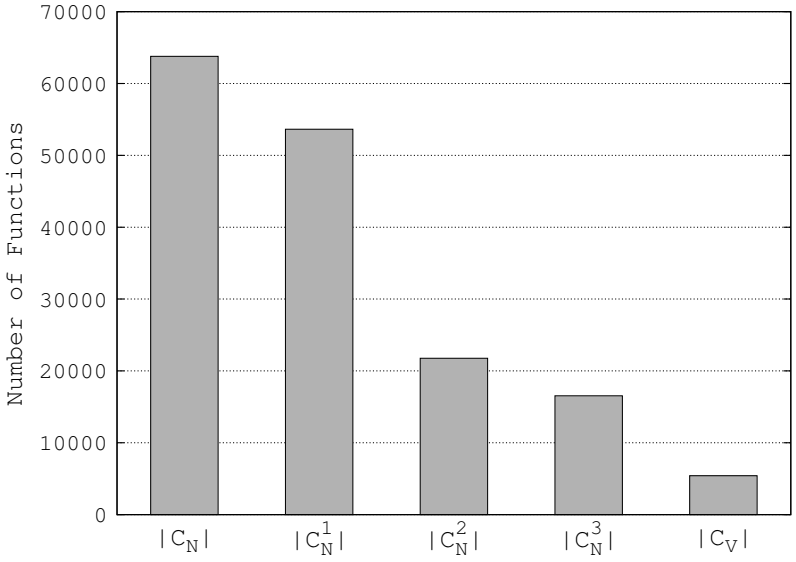
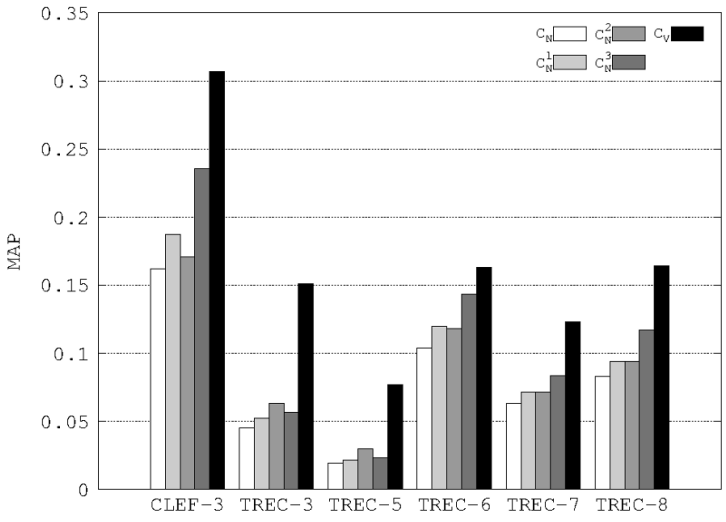


Figure 16: Number of functions in the sets $\mathcal{C}_N$, $\mathcal{C}_N^1$, $\mathcal{C}_N^2$, $\mathcal{C}_N^3$ and $\mathcal{C}_V$ till length 8.

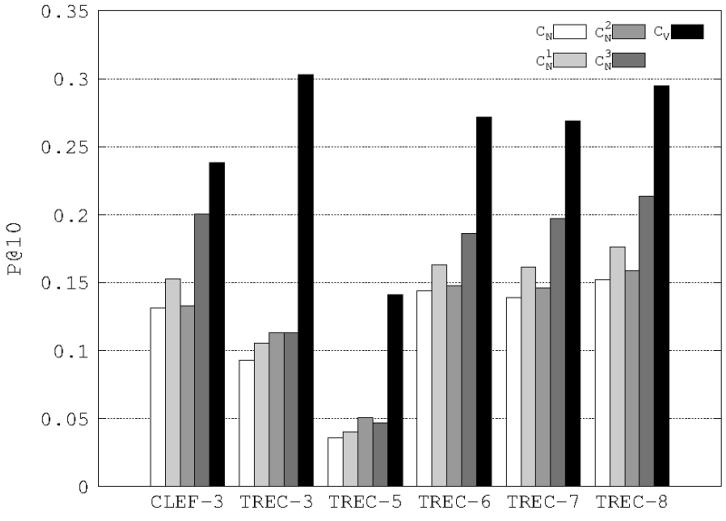
To compare the performances of $\mathcal{C}_N^1$, $\mathcal{C}_N^2$ and $\mathcal{C}_N^3$, a similar experimental strategy is adopted like the previous experiment. For every set $\mathcal{C}_N$, $\mathcal{C}_N^1$, $\mathcal{C}_N^2$, $\mathcal{C}_N^3$ and $\mathcal{C}_V$ 10 sample sets are created each containing 100 random functions selected without replacement. These functions are again tested on `CLEF-3` and `TREC-3,5,6,7,8`. `MAP` and `P@10` measures are averaged over 100 functions in a sample set and average of all 10 sample sets are reported.

Figure 17 shows a plot of average `MAP` and `P@10` of 10 sample sets from all five sets $\mathcal{C}_N$, $\mathcal{C}_N^1$, $\mathcal{C}_N^2$, $\mathcal{C}_N^3$ and $\mathcal{C}_V$. As expected $\mathcal{C}_V$ is always best and $\mathcal{C}_N$ is always worst among the five sets. Performance of other three sets $\mathcal{C}_N^1$, $\mathcal{C}_N^2$ and $\mathcal{C}_N^3$ are in between $\mathcal{C}_V$ and $\mathcal{C}_N$. Both for average `MAP` and `P@10`, C3 is best performing on 4 out of 6 collections. But for `TREC-3` and `TREC-5` C2 is either better or equal to C3. There is no deterministic comparative pattern between C1 and C2. All possible relative orders in terms of performance between C1 and C2 is visible. As for example in case of average `MAP` (Figure 5.17(a)) C1>C2 on `CLEF-3`,

C1<C2 on TREC-3,5 and C1≈C2 on TREC-6,7,8. Same erratic behaviour is visible for average P@10 (Figure 5.17(b)). In summary the general trend is that C3 is the most effective among three constraints although the plots display an inconclusive pattern. Thus it can be said that the relative effectiveness of the constraints is highly dependent on the collection in hand.



(a) MAP



(b) P@10

Figure 17: Average MAP and P@10 of the sets $\mathcal{C}_N$ (□), $\mathcal{C}_N^1$ (▒), $\mathcal{C}_N^2$ (▒), $\mathcal{C}_N^3$ (▒) and $\mathcal{C}_V$ (■) till length 8.

Above experiments are performed to study the effects of each constraint. But these experiments also revealed that combination of all the constraints (i.e. set $\mathcal{C}_V$) always

performs best. Hence for all practical purposes it is always better to utilize all the constraints together as done in Algorithm 5.3.1.

5.6.2 *Function Validation for Default Parameter Setting*

Intially top 500 functions are identified by the strategy explained in Section 5.4 and further experiments are performed on these functions. We first limit our analysis over the TREC-3,5,6,7,8 collections. We tested these 500 functions on all five TREC collections. Functions are ranked based on MAP and P@10 within each collection. Average rank of a function is estimated by the average of all the ranks it got on all the collections. Note that functions with lower average ranks are better performing. Average ranks of standard models, BM25, LM and LGD with respect to these 500 functions over the test collections are also considered. Two separate average ranks are calculated: one based on MAP based rankings and one on P@10 based rankings. For this phase of experiment, we consider the default parameter value settings for all the functions and standard models.

Table 32 shows the top 7 functions along with three standard IR models with respect to the average rank over 5 test collections, TREC-3,5,6,7,8. In the tables we replaced k with 1 and presented the simplified functions. We denote these functions by using an exponent 'M' or 'P' for whether they are the best performing functions with respect to MAP or P@10, and 'd' to indicate that their default parameter values are used.

on MAP			on P@10		
functions	denoted by	rank_{avg}	functions	denoted by	rank_{avg}
$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	$f_1^{\text{M-d}}$	4.8	$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	$f_1^{\text{P-d}}$	19.8
$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	$f_2^{\text{M-d}}$	14.8	$\log\left(-x+\frac{x+y}{y}\right)$	$f_2^{\text{P-d}}$	24.6
$\sqrt{\frac{\sqrt{xy}}{y}}$	$f_3^{\text{M-d}}$	15.6	$\sqrt{x+\sqrt{\frac{x}{y}}}$	$f_3^{\text{P-d}}$	25.2
$\sqrt{y+\sqrt{\frac{x}{y}}}$	$f_4^{\text{M-d}}$	17.8	$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	$f_4^{\text{P-d}}$	27.6
$\sqrt{\sqrt{\frac{x}{y}}.e^{-y}}$	$f_5^{\text{M-d}}$	18.8	$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	$f_5^{\text{P-d}}$	27.8
$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	$f_6^{\text{M-d}}$	19.0	$\log\left(\frac{x+y}{y}\right)$	$f_6^{\text{P-d}}$	30.0
$\log\left(-x+\frac{x+y}{y}\right)$	$f_7^{\text{M-d}}$	20.0	$\log\left(\frac{x}{y}+\sqrt{e}\right)$	$f_7^{\text{P-d}}$	34.2
LGD	LGD_d	26.0	LGD	LGD_d	36.8
BM25	BM25_d	108.8	BM25	BM25_d	43.6
LM_{Dir}	LM_d	129.6	LM_{Dir}	LM_d	208.4

Table 32: Best functions with default parameter values based on their average ranks on TREC-3,5,6,7,8.

We note that all 7 best ranked functions over the 5 TREC collections are better ranked than all three classical IR models. The same function is the best one for both the lists, that is $f_1^{\text{M-d}} = f_1^{\text{P-d}}$. This first ranked function $(x, y) \mapsto e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$ is 2 to 6 times better ranked (with respect to P@10 and MAP) than the best standard IR model. Interestingly this first ranked function is basically the exponential of square root of LGD.

In tables 33 and 34 we show the ranks each function got and respectively the MAP and the P@10 measures on TREC-3,5,6,7,8 collections. In each case we statistically compare the performance of standard models with first ranked function in terms of MAP using a paired two sided t-test at 0.05 level. A $^{\downarrow}$ indicates that the corresponding model is statistically significantly worse than the first ranked function.

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
BM25_d(1; .252)	$f_1^{\text{M-d}}$(1; .140)	$f_1^{\text{M-d}}$(1; .249)	$f_4^{\text{M-d}}$(5; .194)	$f_1^{\text{M-d}}$(1; .256)
$f_5^{\text{M-d}}$(2; .252)	$f_2^{\text{M-d}}$(3; .139)	$f_6^{\text{M-d}}$(2; .248)	$f_3^{\text{M-d}}$(6; .194)	$f_7^{\text{M-d}}$(6; .255)
$f_3^{\text{M-d}}$(3; .250)	$f_5^{\text{M-d}}$(4; .138)	$f_4^{\text{M-d}}$(4; .247)	$f_5^{\text{M-d}}$(8; .194)	LGD_d(11; .255)
$f_2^{\text{M-d}}$(4; .249)	BM25_d(5; .138)	$f_3^{\text{M-d}}$(5; .247)	$f_6^{\text{M-d}}$(9; 194)	$f_6^{\text{M-d}}$(30; .252)
$f_1^{\text{M-d}}$(5; .249)	LM_d(10; .137)	$f_2^{\text{M-d}}$(7; .246)	$f_2^{\text{M-d}}$(13; .193)	f_{d4}^{M}(46; .250)
$f_4^{\text{M-d}}$(7; .249)	$f_3^{\text{M-d}}$(13; .137)	LGD_d(16; .245)	$f_1^{\text{M-d}}$(16; .192)	$f_2^{\text{M-d}}$(47; .249)
LM_d(8; .249)	$f_7^{\text{M-d}}$(14; .137)	$f_5^{\text{M-d}}$(19; .244)	$f_7^{\text{M-d}}$(41; .189)	$f_3^{\text{M-d}}$(51; .249)
$f_7^{\text{M-d}}$(9; .248)	$f_4^{\text{M-d}}$(27; .136)	$f_7^{\text{M-d}}$(30; .244)	LGD_d(49; .188)	$f_5^{\text{M-d}}$(61; .248)
$f_6^{\text{M-d}}$(12; .246)	LGD_d(40; .135)	BM25_d(252; .232)$^{\uparrow}$	LM_d(64; .186)	BM25_d(181; .241)$^{\uparrow}$
LGD_d(14; .245)	$f_6^{\text{M-d}}$(42; .134)	LM_d(381; .222)$^{\uparrow}$	BM25_d(105; .183)	LM_d(185; 0.240)$^{\uparrow}$

Table 33: MAP based ranks of functions with default parameter values (first value in the parenthesis) and their corresponding MAP (second value in the parenthesis).

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
LM_d(1; .532)	LM_d(1; .276)	$f_5^{\text{P-d}}$(1; .418)	$f_1^{\text{P-d}}$(6; .432)	$f_6^{\text{P-d}}$(4; .474)
$f_3^{\text{P-d}}$(6; .516)	$f_3^{\text{P-d}}$(6; .248)	BM25_d(2; .414)	$f_5^{\text{P-d}}$(11; .430)	$f_2^{\text{P-d}}$(5; .474)
BM25_d(9; .514)	$f_1^{\text{P-d}}$(47; .236)	$f_4^{\text{P-d}}$(7; .402)	$f_6^{\text{P-d}}$(16; .428)	LGD_d(7; .474)
$f_4^{\text{P-d}}$(12; .506)	$f_4^{\text{P-d}}$(49; .234)	$f_1^{\text{P-d}}$(12; .400)	$f_7^{\text{P-d}}$(17; .428)	$f_7^{\text{P-d}}$(8; .472)
$f_2^{\text{P-d}}$(13; .504)	$f_2^{\text{P-d}}$(53; .234)	$f_3^{\text{P-d}}$(15; .398)	$f_2^{\text{P-d}}$(18; .428)	BM25_d(14; .472)
$f_1^{\text{P-d}}$(15; .504)	BM25_d(63; .232)	$f_6^{\text{P-d}}$(29; .396)	LGD_d(26; .428)	$f_1^{\text{P-d}}$(19; .468)
$f_5^{\text{P-d}}$(25; .496)	$f_5^{\text{P-d}}$(73; .228)	$f_7^{\text{P-d}}$(33; .396)	$f_4^{\text{P-d}}$(27; .426)	$f_5^{\text{P-d}}$(29; .460)
$f_6^{\text{P-d}}$(27; .496)	$f_6^{\text{P-d}}$(74; .228)	$f_2^{\text{P-d}}$(34; .396)	$f_3^{\text{P-d}}$(58; .422)	$f_3^{\text{P-d}}$(41; .458)
$f_7^{\text{P-d}}$(28; .496)	LGD_d(75; .228)	LGD_d(47; .396)	BM25_d(130; .418)	$f_4^{\text{P-d}}$(43; .458)
LGD_d(29; .496)	$f_7^{\text{P-d}}$(85; .226)	LM_d(483; .346)	LM_d(285; .392)	LM_d(272; .432)

Table 34: P@10 based ranks of functions with default parameter values (first value in the parenthesis) and their corresponding P@10 (second value in the parenthesis).

Here we see that the ranks of the 7 best functions over different collections stay approximately on the same range, while the ranks of IR models may vary a lot. For example, considering the MAP, BM25 with its default values is ranked first on TREC-3 while it is ranked 252^{nd} on TREC-6 (Table 33). We find the same result by looking at the ranks of different models with respect to their P@10 measure on different TREC collections (Table 34). As for example the language model is ranked first on TREC-3 and TREC-5 whereas it has the lowest rank over the three other TREC collections. In the beginning (Section 5.1) it is mentioned that one of the goal of this study is to find something which will work well on almost all collections. It is evident here that these discovered functions behave more or less consistently on all the collections, whereas rankings state-of-the-art IR models varies a lot.

5.6.3 *Function Validation for Tuned Parameter Setting*

We now consider the top 25 functions among the best performing functions under default parameter setting based on average ranks they have, and optimize their hyper-parameter c using 5 fold cross validation (Section 5.5.3). In order to maintain our comparisons, we also consider the optimized versions of the standard models, BM25, LM and LGD using again 5 fold cross validation (Section 5.5.2). Table 35 shows the top optimized functions along with optimized standard IR models with respect to the average rank over TREC-3,5,6,7,8. Like previous tables we replaced k with 1 and presented the simplified functions. The exponent 'M' or 'P' denotes whether they are the best performing functions with respect to MAP or P@10, and the index o indicates that the function or the standard IR models are used with their optimized (or tuned) parameter values.

Here again 7 best functions with respect to MAP are better ranked than all classical IR models. Moreover, the best function of Table 32 has been placed second in terms of MAP. But on the basis of average ranking on P@10 standard IR models are better placed.

One can note that the top two functions with respect to MAP based ranking in Table 35 are also in top two positions in Table 32. Hence it is evident these two functions are worth looking into. These functions are further tested in Section 5.6.5.

In tables 36 and 37 we show detailed ranks of the optimized versions of different functions with respect to their MAP and P@10 measures on different TREC-3,5,6,7,8 collections. A $^{\downarrow}$ indicates that the corresponding model is statistically significantly worse than the first ranked function with respect to a paired two sided t-test at 0.05 level.

In these tables also the earlier trend is clearly visible. Though it may seem that the difference between the average ranks of the optimized IR models and the top 7 generated functions have decreased, but we did this optimization on total 28 models, 25 generated functions and 3 standard models. So this ranking has been done among 28 models instead of 503 in previous section. Taken that into consideration here also BM25 and LM varies more in comparison to others.

on MAP			on P@10		
functions	denoted by	rank_{avg}	functions	denoted by	rank_{avg}
$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	$f_1^{\text{M-o}}$	3.0	BM25	$\texttt{BM25}_o$	9.0
$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	$f_2^{\text{M-o}}$	8.4	$\log\left(-x+\frac{x+y}{y}\right)$	$f_1^{\text{P-o}}$	9.2
$\sqrt{y+\sqrt{\frac{x}{y}}}$	$f_3^{\text{M-o}}$	8.8	$\log\left(\frac{x}{y}+\sqrt{e}\right)$	$f_2^{\text{P-o}}$	10.0
$\sqrt{\frac{\sqrt{xy}}{y}}$	$f_4^{\text{M-o}}$	9.0	$\log\left(\frac{x+y}{y}\right)$	$f_3^{\text{P-o}}$	10.4
$\sqrt{\sqrt{\frac{x}{y}}.e^{-y}}$	$f_5^{\text{M-o}}$	10.0	$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	$f_4^{\text{P-o}}$	11.0
$\sqrt{1+\sqrt{\frac{x}{y}}}$	$f_6^{\text{M-o}}$	10.4	LM_{Dir}	$\texttt{LM}_o$	12.2
$\sqrt{\sqrt{x}+\sqrt{\frac{x}{y}}}$	$f_7^{\text{M-o}}$	11.0	$\sqrt{x+\sqrt{\frac{x}{y}}}$	$f_6^{\text{P-o}}$	12.8
LM_{Dir}	$\texttt{LM}_o$	16.4	LGD	$\texttt{LGD}_o$	13.0
BM25	$\texttt{BM25}_o$	16.6	$\log\left(\frac{x+2y}{y}\right)$	$f_6^{\text{P-o}}$	13.4
LGD	$\texttt{LGD}_o$	17.6	$\sqrt{1+\sqrt{\frac{x}{y}}}$	$f_7^{\text{P-o}}$	13.6

Table 35: Best functions with optimized parameter values based on average rank on TREC-3,5,6,7,8.

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\texttt{BM25}_o(1; .273)^{\downarrow}$	$f_1^{\text{M-o}}(1; .141)$	$f_1^{\text{M-o}}(1; .255)$	$f_5^{\text{M-o}}(4; .195)$	$f_1^{\text{M-o}}(1; .262)$
$\texttt{LM}_o(2; .269)^{\downarrow}$	$\texttt{LM}_o(2; .141)$	$f_7^{\text{M-o}}(2; .250)$	$f_1^{\text{M-o}}(6; .194)$	$f_2^{\text{M-o}}(2; .261)$
$f_5^{\text{M-o}}(3; .260)$	$f_4^{\text{M-o}}(3; .139)$	$f_6^{\text{M-o}}(4; .249)$	$f_6^{\text{M-o}}(7; .193)$	$\texttt{BM25}_o(10; .259)$
$f_4^{\text{M-o}}(4; .258)$	$f_5^{\text{M-o}}(4; .138)$	$f_3^{\text{M-o}}(7; .248)$	$f_3^{\text{M-o}}(9; .193)$	$\texttt{LGD}_o(15; .258)$
$f_3^{\text{M-o}}(5; .257)$	$f_3^{\text{M-o}}(5; .138)$	$f_4^{\text{M-o}}(9; .248)$	$f_4^{\text{M-o}}(10; .193)$	$f_3^{\text{M-o}}(18; .257)$
$f_1^{\text{M-o}}(6; .256)$	$f_2^{\text{M-o}}(6; .138)$	$f_2^{\text{M-o}}(12; .248)$	$f_2^{\text{M-o}}(12; .192)$	$f_4^{\text{M-o}}(19; .257)$
$f_7^{\text{M-o}}(7; .253)$	$f_6^{\text{M-o}}(8; .137)$	$f_5^{\text{M-o}}(15; .247)$	$f_7^{\text{M-o}}(15; .191)$	$f_7^{\text{M-o}}(22; .256)$
$f_6^{\text{M-o}}(8; .252)$	$f_7^{\text{M-o}}(9; .135)$	$\texttt{LGD}_o(20; .245)$	$\texttt{BM25}_o(16; .191)$	$f_5^{\text{M-o}}(24; .256)$
$f_2^{\text{M-o}}(10; .251)$	$\texttt{LGD}_o(20; .133)$	$\texttt{LM}_o(27; .243)^{\uparrow}$	$\texttt{LGD}_o(20; .190)$	$f_6^{\text{M-o}}(25; .255)$
$\texttt{LGD}_o(13; .246)^{\uparrow}$	$\texttt{BM25}_o(28; .126)^{\uparrow}$	$\texttt{BM25}_o(28; .232)^{\uparrow}$	$\texttt{LM}_o(25; .189)$	$\texttt{LM}_o(26; .254)$

Table 36: MAP based ranks of functions with optimized parameter values (first value in the parenthesis) and their corresponding MAP (second value in the parenthesis).

5.6.4 *Performance of Functions on Larger Collections*

We performed experiments with the discovered functions on WT10G and GOV2 collections. Under default parameter settings top 7 functions of Table 32 both in are tested on WT10G

TREC-3	TREC-5	TREC-6	TREC-7	TREC-8
$\mathtt{BM25}_o$(1; .562)	$\mathtt{LM}_o$(1; .274)	$f_4^{\text{P-o}}$(1; .410)	$f_1^{\text{P-o}}$(1; .444)	$\mathtt{BM25}_o$(1; .464)
$f_5^{\text{P-o}}$(2; .560)	$f_5^{\text{P-o}}$(2; .258)	$f_1^{\text{P-o}}$(2; .406)	$f_3^{\text{P-o}}$(3; .442)	$f_2^{\text{P-o}}$(3; .458)
$\mathtt{LM}_o$(3; .558)	$\mathtt{BM25}_o$(4; .250)	$f_3^{\text{P-o}}$(6; .404)	$\mathtt{LGD}_o$(4; .442)	$f_7^{\text{P-o}}$(4; .458)
$f_4^{\text{P-o}}$(7; .538)	$f_4^{\text{P-o}}$(9; .240)	$f_2^{\text{P-o}}$(8; .404)	$f_2^{\text{P-o}}$(5; .440)	$f_3^{\text{P-o}}$(5; .456)
$f_1^{\text{P-o}}$(17; .490)	$f_2^{\text{P-o}}$(13; .238)	$f_6^{\text{P-o}}$(9; .404)	$f_7^{\text{P-o}}$(6; .440)	$f_6^{\text{P-o}}$(7; .456)
$f_3^{\text{P-o}}$(19; .490)	$f_7^{\text{P-o}}$(14; .238)	$\mathtt{LGD}_o$(11; .404)	$f_6^{\text{P-o}}$(7; .438)	$\mathtt{LGD}_o$(9; .456)
$\mathtt{LGD}_o$(20; .484)	$f_1^{\text{P-o}}$(15; .238)	$f_5^{\text{P-o}}$(15; .400)	$\mathtt{BM25}_o$(13; .430)	$f_1^{\text{P-o}}$(11; .452)
$f_2^{\text{P-o}}$(21; .482)	$f_3^{\text{P-o}}$(19; .236)	$f_7^{\text{P-o}}$(18; .400)	$\mathtt{LM}_o$(17; .424)	$\mathtt{LM}_o$(13; .452)
$f_6^{\text{P-o}}$(22; .476)	$\mathtt{LGD}_o$(21; .236)	$\mathtt{BM25}_o$(26; .394)	$f_5^{\text{P-o}}$(18; .416)	$f_4^{\text{P-o}}$(19; .444)
$f_7^{\text{P-o}}$(25; .472)	$f_6^{\text{P-o}}$(22; .234)	$\mathtt{LM}_o$(27; .390)	$f_4^{\text{P-o}}$(19; .412)	$f_5^{\text{P-o}}$(27; .438)

Table 37: P@10 based ranks of functions with optimized parameter values (first value in the parenthesis) and their corresponding P@10 (second value in the parenthesis).

and GOV2 along with standard IR models. Top functions obtained from MAP is used to compare in terms of MAP. These results are given in Table 38. The models are sorted in descending order of MAP or P@10. Similarly for optimized parameter settings top 7 functions from Table 35 are used. Table 39 shows the performance of the models when parameters are optimized. Here also models are sorted in descending order of MAP or P@10.

WT10G		GOV2	
MAP	P@10	MAP	P@10
$\mathtt{LM}_d$(.204)	$f_5^{\text{P-d}}$(.300)	$f_1^{\text{M-d}}$(.291)	$\mathtt{LM}_d$(.555)
$f_7^{\text{M-d}}$(.196)	$f_1^{\text{P-d}}$(.299)	$f_7^{\text{M-d}}$(.289)	$f_7^{\text{P-d}}$(.544)
$\mathtt{LGD}_d$(.194)	$\mathtt{LM}_d$(.293)	$\mathtt{LGD}_d$(.288)	$f_1^{\text{P-d}}$(.543)
$f_1^{\text{M-d}}$(.194)	$f_4^{\text{P-d}}$(.292)	$\mathtt{LM}_d$(.280)	$f_2^{\text{P-d}}$(.542)
$f_4^{\text{M-d}}$(.187)	$\mathtt{BM25}_d$(.291)	$f_2^{\text{M-d}}$(.274)	$f_6^{\text{P-d}}$(.541)
$f_3^{\text{M-d}}$(.187)	$f_6^{\text{P-d}}$(.287)	$\mathtt{BM25}_d$(.274)	$\mathtt{LGD}_d$(.541)
$f_5^{\text{M-d}}$(.187)	$\mathtt{LGD}_d$(.287)	$f_6^{\text{M-d}}$(.265)	$\mathtt{BM25}_d$(.538)
$f_6^{\text{M-d}}$(.186)	$f_2^{\text{P-d}}$(.284)	$f_3^{\text{M-d}}$(.262)	$f_5^{\text{P-d}}$(.535)
$f_2^{\text{M-d}}$(.186)	$f_7^{\text{P-d}}$(.284)	$f_4^{\text{M-d}}$(.261)	$f_4^{\text{P-d}}$(.525)
$\mathtt{BM25}_d$(.184)	$f_3^{\text{P-d}}$(.256)	$f_5^{\text{M-d}}$(.260)	$f_3^{\text{P-d}}$(.471)

Table 38: MAP and P@10 measures of different functions and IR models with their default values on WT10G and GOV2 datasets.

On GOV2, function 2 behaves well with a difference of 5% in MAP with the second best model when optimizing the parameters. However on WT10G, though function 1 and 3 are best models but the difference is not significant with the performances of other models. It seems that the simple strategy of selecting the first 500 functions over CLEF-3 has its limits

WT10G		GOV2	
MAP	P@10	MAP	P@10
$f_1^{\text{M-o}}(.209)$	$\text{BM25}_o(.320)$	$f_2^{\text{M-o}}(.302)$	$f_2^{\text{P-o}}(.557)$
$f_3^{\text{M-o}}(.209)$	$\text{LM}_o(.310)$	$f_1^{\text{M-o}}(.295)$	$f_6^{\text{P-o}}(.553)$
$\text{LM}_o(.208)$	$f_2^{\text{P-o}}(.300)$	$\text{LM}_o(.294)$	$f_1^{\text{P-o}}(.551)$
$f_7^{\text{M-o}}(.208)$	$f_6^{\text{P-o}}(.299)$	$\text{LGD}_o(.288)$	$\text{LM}_o(.551)$
$f_5^{\text{M-o}}(.284)$	$f_1^{\text{P-o}}(.298)$	$\text{BM25}_o(.284)$	$f_3^{\text{P-o}}(.550)$
$f_4^{\text{M-o}}(.284)$	$f_3^{\text{P-o}}(.297)$	$f_7^{\text{M-o}}(.274)$	$\text{LGD}_o(.541)$
$f_6^{\text{M-o}}(.207)$	$\text{LGD}_o(.297)$	$f_3^{\text{M-o}}(.271)$	$f_7^{\text{P-o}}(.539)$
$f_2^{\text{M-o}}(.207)$	$f_4^{\text{P-o}}(.296)$	$f_6^{\text{M-o}}(.271)$	$\text{BM25}_o(.531)$
$\text{BM25}_o(.206)$	$f_7^{\text{P-o}}(.293)$	$f_4^{\text{M-o}}(.270)$	$f_4^{\text{P-o}}(.505)$
$\text{LGD}_o(.203)$	$f_5^{\text{P-o}}(.258)$	$f_5^{\text{M-o}}(.269)$	$f_5^{\text{P-o}}(.469)$

Table 39: MAP and P@10 measures of different functions and IR models with their optimized values on WT10G and GOV2 datasets.

on larger collections, but the selected functions are still competitive on MAP and P@10 with respect to standard models over these large datasets.

5.6.5 *Comparison with Genetic Programming based Approaches*

Several genetic programming based approaches are mention in Section 5.2. In this section we compare scoring functions generated by two state-of-the-art genetic programming based methods against two best functions discovered by the method proposed here.

The two genetic programming based scoring functions considered here are the following:

1. [Fan et al., 2004] reported the following consensus scoring function (denoted by GP_1) for a TREC ad-hoc task:

$$GP_1(d,w) = \frac{\log\left(t_{\mathbf{w}}^d \times \left(t_{\mathbf{w}}^{avg} + \frac{t_{\mathbf{w}}^d}{\log((t_{\mathbf{w}}^d)^2 \times t_{\mathbf{w}}^{avg})} + \frac{t_{\mathbf{w}}^d \times N}{\mathcal{N}_{\mathbf{w}}} \times \frac{t_{\mathbf{w}}^{avg} \times (t_{\mathbf{w}}^{max} + uq^d)}{\mathcal{N}_{\mathbf{w}}}\right)\right)}{uq^d + 2 \times t_{\mathbf{w}}^{max} + 0.373}$$

   Here, $t_{\mathbf{w}}^d$ is term frequency, $\mathcal{N}_{\mathbf{w}}$ is document frequency, $t_{\mathbf{w}}^{avg}$ is the average term frequency within the document, $t_{\mathbf{w}}^{max}$ is the maximum term frequency in the entire collection, N is the number of documents in the collection and uq^d is the number of unique terms in the document.

2. [Cummins and O'Riordan, 2006a] divides a term weighting scheme in three parts: a global part (gw_w), a term-frequency part (ntf_w^d) and a normalization part (n). The final scoring function is defined as a product of these three components:

$$GP_2(d,w) = gw_w \times ntf_w^d \times n$$

The best evolved parts reported by [Cummins and O'Riordan, 2006a] are as follows:

$$gw_w = \frac{(cf_w)^2\sqrt{cf_w}}{(\mathcal{N}_{\mathbf{w}})^3}$$

$$ntf_w^d = \log\Big(\sqrt{\frac{200.\frac{t_{\mathbf{w}}^d}{n}}{1+\frac{t_{\mathbf{w}}^d}{n}}}\Big)$$

$$n = \sqrt{\log(|\mathbf{q}|)} \times \log(|\mathbf{q}|) \times \frac{uq^d}{l_{avg}}$$

Here $t_{\mathbf{w}}^d$ is term frequency, $\mathcal{N}_{\mathbf{w}}$ is document frequency, cf_w is the collection frequency, $|\mathbf{q}|$ is the total length of the query, uq^d is the number of unique term in the document and l_{avg} is the average document length of the collection. Note that $t_{\mathbf{w}}^d$ in ntf_w^d is normalized using the normalization term n.

We compare these state-of-the-art genetic programming approaches with top two discovered scoring functions (Table 32 and 35). The performance of the models are given in Table 40 over TREC-7,8, WT10G and GOV2. The best results are in bold, and a result with a $^{\downarrow}$ indicates that it is statistically significantly worse than the best result according to a Wilcoxon rank sum test used at a p-value threshold of 0.05.

	TREC-7		TREC-8		WT10G		GOV2	
	MAP	P@10	MAP	P@10	MAP	P@10	MAP	P@10
GP_1	0.157$^{\downarrow}$	0.330$^{\downarrow}$	0.207$^{\downarrow}$	0.350$^{\downarrow}$	0.119$^{\downarrow}$	0.192$^{\downarrow}$	0.143$^{\downarrow}$	0.363$^{\downarrow}$
GP_2	0.165$^{\downarrow}$	0.330$^{\downarrow}$	0.216$^{\downarrow}$	0.342$^{\downarrow}$	0.142$^{\downarrow}$	0.217$^{\downarrow}$	0.162$^{\downarrow}$	0.394$^{\downarrow}$
$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$	0.192	**0.432**	**0.256**	**0.468**	**0.194**	0.299	**0.291**	**0.543**
$\sqrt{\frac{\log(1+x)}{\sqrt{y}}}$	**0.193**	0.430	0.249	0.460	0.186	**0.300**	0.274	0.535

Table 40: Comparison of MAP and P@10 based performances between top discovered functions against two genetic algorithm based approaches denoted by GP_1 [Fan et al., 2004] and GP_2 [Cummins and O'Riordan, 2006a]. The best results are in bold, and a $^{\downarrow}$ indicates a result that is statistically significantly worse than the best result according to a Wilcoxon rank sum test used at a p-value threshold of 0.05.

Clearly on all occasions functions discovered by the strategy proposed here significantly outperform the functions evolved by genetic programming based approaches.

5.7 CONCLUSION

In this chapter we have addressed the problem of exploring the space of simple IR functions with the goal to discover some promising IR scoring functions. To do so, we

proposed a systematic iterative approach to explore the search space till some given length and to identify the set of candidate scoring functions which are mathematically valid and satisfy heuristic IR constraints. We tested the functions obtained on a variety of standard IR test collections.

With our results, we empirically validated that the scoring functions satisfying IR constraints perform significantly better than valid scoring functions which do not satisfy them. Secondly, extensive testing of the discovered scoring functions shows that these functions are highly competitive with respect to standard IR models. We found that these functions behave almost similarly on different collection, while, standard IR models have more unstable behavior over different collections. Moreover best of the discovered functions outperformed functions evolved by state-of-the-art genetic programming methods. Ultimately, we showed that it is possible to search the space of simple closed-form IR scoring functions exhaustively and the discovered functions can perform consistently well on all the collections.

Our results show that, if one wants to make use of an efficient IR scoring function without tuning parameters on a collection, then one should use:

$$\texttt{ES} - \texttt{LGD}(x, y) = e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$$

where the name ES-LGD derives from the fact that this function consists in the exponential of the square root of the log-logistic function. ES-LGD is consistently above (for the MAP) all other ones; furthermore, the difference with standard IR functions is significant in several cases. This result is all the more interesting that the "complexity" of this function (measured by its length) is smaller than the one of $\texttt{BM25}_d$ and $\texttt{LM}_d$, which can, in theory, be generated by our method using a different term frequency normalization but nevertheless require additional computing resources as they involve more operators. In the situation where it is possible to optimize the value of some parameters (i.e. when relevance judgments are readily available), our results are more contrasted: the difference in ranks between the best functions and the standard ones is not as important as before and if in several cases the difference is significant in favor of the discovered functions, it is, in other cases, in favor of standard functions. All in all, there is no real difference in this case. As ES-LGD is ranked second in this setting, we recommend its use in all cases if one is interested in the MAP. It can of course be used as an additional feature in learning to rank approaches.

6

CONCLUSION AND PERSPECTIVES

The goal of the thesis was to explore the areas of performance improvement of standard IR models by proper tuning of the associated parameters and to learn new IR functions. To do so we studied unsupervised and supervised methods of parameter tuning and learning to rank based algorithms to learn IR functions and exhaustive exploration techniques to discover new IR functions.

Firstly we investigated various possibilities to estimate the collection parameter of information based models. The collection parameter controls the behavior of a term in the collection and thus needs proper estimation, but in earlier studies it was assigned to the average number of documents where the term appears, without a full explanation. After exploring various estimation techniques including maximum likelihood estimation and Kaplan-Meier estimation, we have shown that generalized method of moments is able to provide valid estimate for the collection parameter under both log-logistic and smoothed power law distributions and we have also proved that this method complies with the IDF criterion. Aim was to investigate the question:

How to set the value of the collection parameter of information-based models?

Through various experiments we have shown that with new estimates information models yield state-of-the-art results, improving over the original setting. The improvement is significant for smoothed power law model. Unlike the original version, the new version of this model regularly outperforms other standard IR models in most cases.

As we saw in Chapter 2, log-logistic and smoothed power law distributions with their original collection parameter or estimated collection parameter performs very well in practice. But there exist other distributions with same power law based features like these ones, as for example *Pareto Distribution*. This distribution was applied earlier to derive normalization of term frequency which was in turn applied to a model based on Bose-Einstein statistics [Amati and van Rijsbergen, 2002b]. It would be interesting to see the application of these distributions applied in information based models, as well as how they are impacted by the estimation of the collection parameter.

Even if the above estimation method was able to yield a state-of-the-art IR model, but like other unsupervised methods it is unable to estimate parameter values for any other parameter. This is why we turned to supervised methods and we also ventured the possibilities to apply these methods on the collections where relevance judgments are not readily available. Here the goal was to investigate the question:

> Is it possible to transfer parameter values of standard IR models tuned on past labeled collections to a new collection without any relevance judgments?

We have proposed two learning strategies which exactly do that. Instead of commonly used tuning methods, our approach predict the parameters at query level, that is a single parameter value for each query. To do so, we first introduced a 4-dimensional vector which summarizes a word in the collection and used these vectors to represent each word in a query, thus allowing the query to be represented as a set of these vectors. Then we build two collection and language independent vector spaces, one based on the feature space of positive definite kernels between queries, and one based on similarity scores between the query under consideration and a set of *representational* queries. We introduced a simple similarity measure between queries which turns out to be a positive definite kernel as well. Moreover for the kernel induced space we considered other traditional positive definite kernels namely polynomial and Gaussian. We learn the association between the parameter values tuned in the source collection and the queries represented in either of the vector spaces through a regression function (kernel regression function for the feature space and potentially any regression function for similarity space). In our experiments we used support vector regression on feature space and random forest regression on similarity space over several source and target collections. These experiments revealed that the proposed methods (a) significantly outperform the models which uses default parameters (Section 1.3.1), (b) either outperform or is at par with commonly used tuning methods (Section 1.3.2) which uses relevance information on the target collection, (c) are significantly faster and incurs almost no extra time burden, (d) requires little training data (150 queries are sufficient) (e) independent of languages and regressor trained in any language (they can even be applied to collections written in different languages).

The proposed method, at its current stage, can predict the value of single parameter. What if the model has more than one parameter, like `BM25` (which originally has three free parameters, b, k_1, and k_3)? One obvious way out is to train separate regressors for each of the parameters. This works fine as long as there is no dependency between the parameters, meaning that changing one will not need the changing of the other(s) to optimize the model performance. But the inter-dependency among the parameters depends completely on the IR model and it cannot be safely assumed that the parameters of all IR models have no such dependencies. So the immediate possible future study could be to concentrate on this problem so that the method can predict values for multiple parameters. 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. Furthermore, we plan to investigate new query representation spaces, not necessarily based on PDS kernels, with the hope to better capture the relation between queries and optimal parameter values.

Successful application of transfer learning to transfer parameter values creates a possibility to learn entire ranking functions through learning to rank framework but using relevance

information from a source collection. In this context we explored the solution to the question:

> Is it possible to learn new IR functions on collections without any relevance judgments by transferring the relevance information from past labeled collections?

To answer this we introduced a transfer learning method which uses absolute relevance judgments available in the source collection and tries to infer relative "pseudo-relevance" judgments between the documents in the target collection. We first develop a data structure, called grid, that associates a relevance score with basic quantities, normalized term and document frequency values of a term in a query-document pair. These grids are build on source collections and can predict relevance score for a target query term in a target query-document pair with normalized term and document frequency values. This predicted relevance score is then combined over all query terms. Then a learning to rank algorithm is deployed on the obtained relative pseudo-relevance judgments which is further improved using an iterative self-learning mechanism. In our experimentation we used ranking `SVM` as the learning algorithm. Through the experiments we have conducted, it can indeed be said that the ranking functions learned through our approach significantly outperforms standard IR functions in almost all cases. Moreover, as our approach learns the ranking function directly on the target it is robust against the choice of source, unlike previous approaches which learned the ranking function on re-weighted version of the source collection. We also attempted a simple source selection procedure where the algorithm is given an option to select most appropriate source for a single query from a pool of available sources. But, not much improvement was gain over the original version, enforcing the fact that the algorithm is robust to the choice of source collection.

The transfer learning approach to predict parameter values proposed in Chapter 3 works well even when the transfer is done between collections written in different languages. It would also be interesting to study whether this holds for our transfer learning to rank framework. We conjecture it does, because the experiments have shown that the proposed transfer technique through grids is robust to the choice of source. The proposed transfer learning to approach used underlying kernels of the employed learning algorithm to represent feature vectors associated with query-document pairs. However, in Chapter 3 we showed that queries represented in a common vector space enhanced the learning providing promising results. Similarly, if an embedded representation can be developed for query-document pairs in some common vector space, then there is a broader chance to capture the behaviors of the query-document pairs through this representation and their association with the relevance judgments. Moreover, like Chapter 3, this will enable a collection and language independent representation of the query-document pairs and the transfer may become simpler and more effective.

Learning to rank algorithms attempt to learn a ranking function by optimized weights of different variables inside a predefined function. Though these algorithms are very

effective in practice, still one can ask how the functions behave when one assumes no particular intuitionistic forms. This motivates us to study IR functions which can be learned without standard restrictions on the family of the functions considered. As any restriction on the form is relieved, development of ranking function from intuitions and theoretical angle becomes difficult, which turns the attention to explore the space of simple IR functions with the goal to discover some promising IR scoring functions. Different heuristic search techniques like genetic programming have been applied to explore the space of IR functions. But we wish to do it deterministically and exhaustively. So one possible rephrasing of the question is:

> Is it possible to exhaustively search the function space in order to discover new effective IR functions?

We proposed a systematic iterative approach to explore the search space till some given length. While exploring we utilized simple tools like mathematical validity of a function or heuristic IR constraints (Section 1.4). We searched for functions which are mathematically valid and satisfy heuristic IR constraints. This helped to prune the search space deterministically. We performed extensive experiments to test the functions obtained though which (a) we empirically validated heuristic IR constraints by comparing the performances of the functions which satisfy these constraints and which do not, (b) we discovered some highly competitive functions with respect to standard IR models, and which also outperformed functions evolved by state-of-the-art genetic programming methods, and (c) we found that these functions performs almost similarly on different collections, whereas standard IR models have more unstable behavior. As an example, one of the promising functions that the approach discovered is:

$$e^{\sqrt{\log\left(\frac{x+y}{y}\right)}}$$

where x is the normalized term frequency and y is the normalized document frequency. Interestingly this function is the exponential of the square root of the log-logistic function. Clearly this function is obtained because there was no restriction of form and it consistently outperformed other standard IR models.Thus it can be concluded that it is indeed possible to discover good functions through an exhaustive search of simple closed-form IR function space.

This explorative strategy have already discovered promising functions till length 8. Immediate goal is to explore functions with higher lengths. But generating candidate functions till length 8 took almost a day, and for higher lengths it can take weeks or even months. Considering that, here we wish to assume that good candidate functions of a higher length can be found by combining good candidate functions of lower lengths. Number of functions obtained at a lower depth are reasonably low and it is feasible to test them all on several collections to get top performing functions. Then these good functions can be combined to yield functions with higher length. But first the basic assumption needs experimental validation. If that is a valid assumption, it can be helpful in two ways:

firstly, it can take into consideration several collections while selecting top performing functions (opposed to the the strategy proposed in Section 5.4 which uses only CLEF-3 for this purpose), secondly it enables to explore functions of higher depth which with current approach is very time and resource consuming. Note that, if successful, this assumption acts as another filtering step just like four other mentioned in Section 5.3.4 which prune the search space so that an exhaustive search becomes feasible.

To summarize, in this thesis we proposed:

1. a new unsupervised method to estimate the collection parameter of information models, which improved the performance of these models over their original versions,
2. two supervised transfer learning strategies to predict values of any free parameter of any IR model which do not need any relevance judgment on the collection it is working on, instead it make use of past labeled collections,
3. a transfer learning to rank algorithm to learn IR functions on a collection without any relevance judgments by using the same from a different labeled collection,
4. a explorative discovery approach which can learn new IR function without restriction on its form, thus allowing to obtain functions which are not intuitive and difficult to comprehend theoretically.

The perspectives we envisioned from this thesis can be summarized as:

1. Application of different power law distributions (for example Pareto distribution) in information based models and the effect of the estimation of the collection parameter on these distributions.
2. Modification of the parameter learning approach proposed in Chapter 3 so that it can predict values for multiple parameters by taking into consideration their inter-dependency in the model.
3. Study of other possible query representation spaces similar to one used in Chapter 3 which can better capture the relation between queries and optimal parameter values.
4. Development of a embedded representation of the query-document pairs in a common space which can provide a collection and language independent representation of the query-document pairs and which can enhance the effectiveness of the transfer learning to rank approach proposed in Chapter 4.
5. Development and validation of explorative strategy based on the assumption that good candidate functions of higher lengths can be discovered by combining good candidate functions of lower lengths.

PUBLICATIONS

[Goswami et al., 2013] Goswami, P., Amini, M.-R., and Gaussier, E. (2013). 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)*, pages 2315–2320. ACM.

[Goswami et al., 2014a] Goswami, P., Amini, M.-R., and Gaussier, E. (2014a). Knowledge transfer in IR: Learning IR functions on unlabeled collections.

[Goswami et al., 2014b] Goswami, P., Amini, M.-R., and Gaussier, E. (2014b). A novel dicovery approach to explore the space of IR functions.

[Goswami and Gaussier, 2013a] Goswami, P. and Gaussier, E. (2013a). Estimation du paramètre de collection des modèles d'information pour la RI. In *Conférence en Recherche d'Infomations et Applications - 10th French Information Retrieval Conference (CORIA)*, pages 201–216. UNINE.

[Goswami and Gaussier, 2013b] Goswami, P. and Gaussier, E. (2013b). Estimation of the collection parameter of information models for IR. In *Proceedings of the* 35^{th} *European Conference on Information Retrieval (ECIR)*, pages 459–570. Springer.

[Goswami et al., 2014c] Goswami, P., Moura, S., Gaussier, E., Amini, M.-R., and Maes, F. (2014c). Exploring the space of ir functions. In *Proceedings of the* 36^{th} *European Conference on Information Retrieval (ECIR)*, pages 372–384. Springer.

REFERENCES

[Amati and van Rijsbergen, 2002a] Amati, G. and van Rijsbergen, C. (2002a). Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems (TOIS)*, 20(4):357–389.

[Amati and van Rijsbergen, 2002b] Amati, G. and van Rijsbergen, C. (2002b). Term frequency normalization via pareto distributions. In *Proceedings of the* 24^{th} *European Conference on Information Retrieval (ECIR)*, pages 183–192. Springer.

[Amini and Gallinari, 2002] Amini, M.-R. and Gallinari, P. (2002). The use of unlabeled data to improve supervised learning for text summarization. In *Proceedings of the* 25^{th}

annual international ACM SIGIR conference on research and development in information retrieval, pages 105–112. ACM.

[Amini and Gaussier, 2013] Amini, M.-R. and Gaussier, E. (2013). *Recherche d'Information - applications, modèles et algorithmes*. Eyrolles.

[Arnold et al., 2007] Arnold, A., Nallapati, R., and Cohen, W. (2007). A comparative study of methods for transductive transfer learning. In *Workshops Proceedings of the 7th IEEE International Conference on Data Mining (ICDM)*, pages 77–82. IEEE Computer Society.

[Balakrishnan and Rao, 2004] Balakrishnan, N. and Rao, C. (2004). *Advances in Survival Analysis*, volume 23 of *Handbook of Statistics*, chapter 5, page 96. North Holland.

[Bennett et al., 2008] Bennett, G., Scholer, F., and Uitdenbogerd, A. (2008). A comparative study of probabalistic and language models for information retrieval. In *Proceedings of the 19th Australasian Database Conference*, pages 65–74. Australian Computer Society.

[Billhardt et al., 2002] Billhardt, H., Borrajo, D., and Maojo, V. (2002). Using genetic algorithms to find suboptimal retrieval expert combinations. In *Proceedings of the 2002 ACM Symposium on Applied Computing (SAC)*, pages 657–662. ACM.

[Blair, 1990] Blair, D. (1990). *Language and Representation in Information Retrieval*. Elsevier.

[Blitzer et al., 2008] Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2008). Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems (NIPS 20)*, pages 129–136. Curran Associates Inc.

[Burges, 2010] Burges, C. (2010). From ranknet to lambdarank to lambdamart : An overview. Technical Report MSR-TR-2010-82, Microsoft Research.

[Cai et al., 2011a] Cai, P., Gao, W., Wong, K., and Zhou, A. (2011a). Weight-based boosting model for cross-domain relevance ranking adaptation. In *Proceedings of the 33rd European Conference on Information Retrieval (ECIR)*, pages 562–567. Springer.

[Cai et al., 2011b] Cai, P., Gao, W., Zhou, A., and Wong, K. (2011b). Query weighting for ranking model adaptation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 112–122. ACM.

[Calauzènes et al., 2012] Calauzènes, C., Usunier, N., and Gallinari, P. (2012). On the (non-)existence of convex, calibrated surrogate losses for ranking. In *Advances in Neural Information Processing Systems (NIPS 25)*, pages 197–205. Curran Associates Inc.

[Cao et al., 2006] Cao, Y., Xu, J., Liu, T., Li, H., Huang, Y., and Hon, H. (2006). Adapting ranking SVM to document retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 186–193. ACM.

[Carterette, 2007] Carterette, B. (2007). Robust test collections for retrieval evaluation. In *Proceedings of the* 30^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 55–62. ACM.

[Castronovo et al., 2012] Castronovo, M., Maes, F., Fonteneau, R., and Ernst, D. (2012). Learning exploration/exploitation strategies for single trajectory reinforcement learning. In *Proceedings of the* 10^{th} *European Workshop on Reinforcement Learning (EWRL)*, pages 1–10. JMLR.org.

[Chapelle et al., 2011] Chapelle, O., Chang, Y., and Liu, T. (2011). Future directions in learning to rank. *Journal of Machine Learning Research*, 14:91–100.

[Chen et al., 2010] Chen, D., Xiong, Y., Yan, J., Xue, G., Wang, G., and Chen, Z. (2010). Knowledge transfer for cross domain learning to rank. *Information Retrieval*, 13(3):236–253.

[Chen et al., 2008a] Chen, D., Yan, J., Wang, G., Xiong, Y., Fan, W., and Chen, Z. (2008a). Transrank: A novel algorithm for transfer of rank learning. In *IEEE International Conference on Data Mining Workshops (ICDMW)*, pages 106–115. IEEE Xplore.

[Chen, 1995] Chen, H. (1995). Machine learning for information retrieval: Neural networks, symbolic learning, and genetic algorithms. *Journal of the American Society for Information Science (JASIS)*, 46(3):194–216.

[Chen et al., 2008b] Chen, K., Lu, R., Wong, C., Sun, G., Heck, L., and Tseng, B. (2008b). Trada: Tree based ranking function adaptation. In *Proceedings of the* 17^{th} *ACM International Conference on Information & Knowledge Management (CIKM)*, pages 1143–1152. ACM.

[Chen et al., 2009] Chen, W., Liu, T., Lan, Y., Ma, Z., and Li, H. (2009). Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems (NIPS 22)*, pages 315–323. Curran Associates, Inc.

[Church and Gale, 1995] Church, K. and Gale, W. (1995). Poisson mixtures. *Natural Language Engineering*, 1:163–190.

[Clinchant and Gaussier, 2010] Clinchant, S. and Gaussier, E. (2010). 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*, pages 234–241. ACM.

[Clinchant and Gaussier, 2011] Clinchant, S. and Gaussier, E. (2011). Retrieval constraints and word frequency distributions a log-logistic model for ir. *Information Retrieval*, 14(1):5–25.

[Cohen et al., 1999] Cohen, W., Schapire, R., and Singer, Y. (1999). Learning to order things. *Journal of Artificial Intelligence Research*, 10(1):243–270.

[Crammer and Singer, 2001] Crammer, K. and Singer, Y. (2001). Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS 14)*, pages 641–647. Curran Associates, Inc.

[Cummins and O'Riordan, 2005] Cummins, R. and O'Riordan, C. (2005). Evolving general term-weighting schemes for information retrieval: Tests on larger collections. *Artificial Intelligence Review*, 24(3–4):277–299.

[Cummins and O'Riordan, 2006a] Cummins, R. and O'Riordan, C. (2006a). Evolved term-weighting schemes in information retrieval: an analysis of the solution space. *Artificial Intelligence Review*, 26(1-2):35–47.

[Cummins and O'Riordan, 2006b] Cummins, R. and O'Riordan, C. (2006b). Evolving local and global weighting schemes in information retrieval. *Information Retrieval*, 9(3):311–330.

[Cummins and O'Riordan, 2009a] Cummins, R. and O'Riordan, C. (2009a). *Information extraction from the Internet*, chapter Analysing Ranking Functions in Information Retrieval Using Constraints. CreateSpace Independent Publishing Platform.

[Cummins and O'Riordan, 2009b] Cummins, R. and O'Riordan, C. (2009b). Measuring constraint violations in information retrieval. In *Proceedings of the* 32^{nd} *annual international ACM SIGIR conference on research and development in information retrieval*, pages 722–723. ACM.

[Dai et al., 2007] Dai, W., Xue, G., Yang, Q., and Yu, Y. (2007). Co-clustering based classification for out-of-domain documents. In *Proceedings of the* 13^{th} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 210–219. ACM.

[Fan et al., 2000] Fan, W., Gordon, M., and Pathak, P. (2000). Personalization of search engine services for effective retrieval and knowledge management. In *Proceedings of the* 21^{st} *International Conference on Information Systems (ICIS)*, pages 20–34. Association for Information Systems.

[Fan et al., 2004] Fan, W., Gordon, M., and Pathak, P. (2004). A generic ranking function discovery framework by genetic programming for information retrieval. *Information Processing and Management*, 40(4):587–602.

[Fang et al., 2004] Fang, H., Tao, T., and Zhai, C. (2004). A formal study of information retrieval heuristics. In *Proceedings of the* 27^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 49–56. ACM.

[Fang and Zhai, 2011] Fang, H. and Zhai, T. T. C. (2011). Diagnostic evaluation of information retrieval models. *ACM Transactions on Information Systems (TOIS)*, 29(2):7:1–7:42.

[Freund et al., 2003] Freund, Y., Iyer, R., Schapire, R., and Singer, Y. (2003). An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969.

[Gao et al., 2010] Gao, W., Cai, P., Wong, K., and Zhou, A. (2010). 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*, pages 162–169. ACM.

[Goldberg, 1989] Goldberg, D. (1989). *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley Longman Publishing Co., Inc.

[Gordon, 1988] Gordon, M. (1988). Probabilistic and genetic algorithms in document retrieval. *Communications of the ACM*, 31(10):1208–1218.

[Gordon, 1991] Gordon, M. (1991). User-based document clustering by redescribing subject descriptions with a genetic algorithm. *Journal of the American Society for Information Science (JASIS)*, 42(5):311–322.

[Harrington, 2003] Harrington, E. (2003). Online ranking/collaborative filtering using the perceptron algorithm. In *Proceedings of the* 20^{th} *International Conference on Machine Learning (ICML)*, pages 250–257. ACM.

[Herbrich et al., 1999] Herbrich, R., Graepel, T., and Obermayer, K. (1999). Support vector learning for ordinal regression. In 9^{th} *International Conference on Artificial Neural Networks (ICANN)*, pages 97–102. IEEE Xplore.

[Huang et al., 2009] Huang, J., Smola, A., Gretton, A., Borgwardt, K., and Schölkopf, B. (2009). Correcting sample selection bias by unlabeled data. In *Advances in Neural Information Processing Systems (NIPS 19)*, pages 601–608. Curran Associates, Inc.

[Jelinek and Marcer, 1980] Jelinek, F. and Marcer, R. (1980). Interpolated estimation of markov source parameters from sparse. In *Proceedings of the Workshop on Pattern Recognition in Practice*.

[Joachims, 1999] Joachims, T. (1999). Making large-scale support vector machine learning practical. In Schölkopf, B., Burges, C., and Smola, A., editors, *Advances in Kernel Methods*, pages 169–184. MIT Press.

[Joachims, 2002] Joachims, T. (2002). Optimizing search engines using clickthrough data. In *Proceedings of the* 8^{th} *ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 133–142. ACM.

[Johnson et al., 1993] Johnson, N., Kemp, A., and Kotz, S. (1993). *Univariate Discrete Distributions*. John Wiley & Sons, Inc.

[Kaplan and Meier, 1958] Kaplan, E. L. and Meier, P. (1958). Nonparametric estimation from incomplete observations. *Journal of the American Statistical Association*, 53(282):457–481.

[Koza, 1992] Koza, J. (1992). *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press.

[Li et al., 2008] Li, P., Burges, C., and Wu, Q. (2008). Mcrank: Learning to rank using multiple classification and gradient boosting. In *Advances in Neural Information Processing Systems (NIPS 20)*, pages 897–904. Curran Associates, Inc.

[Liu, 2009] Liu, T. (2009). Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331.

[Lv and Zhai, 2009] Lv, Y. and Zhai, C. (2009). Adaptive relevance feedback in information retrieval. In *Proceedings of the* 18^{th} *ACM Conference on Information and Knowledge Management (CIKM)*, pages 255–264. ACM.

[Lv and Zhai, 2012] Lv, Y. and Zhai, C. (2012). A log-logistic model-based interpretation of TF normalization of BM25. In *Proceedings of the* 34^{th} *European Conference on Information Retrieval (ECIR)*, pages 244–255. Springer.

[Maes et al., 2012] Maes, F., Fonteneau, R., Wehenkel, L., and Ernst, D. (2012). Policy search in a simple closed-form formulas: Towards interpretability of reinforcement learning. In *Proceedings of* 15^{th} *International Conference on Discovery Science*, pages 37–51. Springer.

[Maes et al., 2011] Maes, F., Wehenkel, L., and Ernst, D. (2011). Automatic discovery of ranking formulas for playing with multi-armed bandits. In *Proceedings* 9^{th} *European Workshop on Recent Advances in Reinforcement Learning (EWRL)*, pages 5–17. Springer.

[Manning et al., 2008] Manning, C. D., Raghavan, P., and Schütze, H. (2008). *Introduction to information retrieval*. Cambridge University Press.

[McAllester et al., 2010] McAllester, D., Hazan, T., and Keshet, J. (2010). Direct loss minimization for structured prediction. In *Advances in Neural Information Processing Systems (NIPS 23)*, pages 1594–1602. Curran Associates, Inc.

[McLachlan, 1992] McLachlan, G. (1992). *Discriminant Analysis and Statistical Pattern Recognition*. John Wiley & Sons, Inc.

[Metzler and Croft, 2005] Metzler, D. and Croft, W. (2005). A markov random field model for term dependencies. In *Proceedings of the* 28^{th} *annual international ACM SIGIR conference on research and development in information retrieval*, pages 472–479. ACM.

[Mohri et al., 2012] Mohri, M., Rostamizadeh, A., and Talwalkar, A. (2012). *Foundations of Machine Learning*. The MIT Press.

[Nallapati, 2004] Nallapati, R. (2004). Discriminative models for information retrieval. In *Proceedings of the* 27^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 64–71. ACM.

[Oren, 2002] Oren, N. (2002). Reexamining tf.idf based information retrieval with genetic programming. In *Proceedings of the 2002 Annual Research Conference of the South African Institute of Computer Scientists and Information Technologists on Enablement Through Technology (SAICSIT)*, pages 224–234. South African Institute for Computer Scientists and Information Technologists.

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

[Pan and Yang, 2010] Pan, S. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

[Pathak et al., 2000] Pathak, P., Gordon, M., and Fan, W. (2000). Effective information retrieval using genetic algorithms based matching functions adaptation. In *Proceedings of the* 33^{rd} *Annual Hawaii International Conference on System Sciences (HICSS)*. IEEE Xplore.

[Petry et al., 1994] Petry, F., Buckles, B. P., Sadasivan, T., and Kraft, D. (1994). The use of genetic programming to build queries for information retrieval. In *International Conference on Evolutionary Computation*, pages 468–473.

[Ponte and Croft, 1998] Ponte, J. and Croft, W. (1998). 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*, pages 275–281. ACM.

[Raghavan and Agarwal, 1987] Raghavan, V. and Agarwal, B. (1987). Optimal determination of user-oriented clusters: An application for the reproductive plan. In *Proceedings of the* 2^{nd} *International Conference on Genetic Algorithms (ICGA)*, pages 241–246. Lawrence Erlbaum Associates.

[Rice, 2006] Rice, J. (2006). *Mathematical Statistics and Data Analysis*. Duxbury Advanced.

[Robertson and Walker, 1994] Robertson, S. and Walker, S. (1994). 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*, pages 232–241. ACM.

[Robertson and Zaragoza, 2009] Robertson, S. E. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389.

[Saenko et al., 2010] Saenko, K., Kulis, B., Fritz, M., and Darrell, T. (2010). Adapting visual category models to new domains. In *Proceeding of* 11^{th} *European Conference on Computer Vision (ECCV)*, pages 213–226. Springer.

[Salton and McGill, 1983] Salton, G. and McGill, J. (1983). *Introduction to Modern Information Retrieval*. McGraw-Hill.

[Saunders et al., 1998] Saunders, C., Gammerman, A., and Vovk, V. (1998). Ridge regression learning algorithm in dual variables. In *Proceedings of the* 15^{th} *International Conference on Machine Learning (ICML)*, pages 515–521. Morgan Kaufmann.

[Singhal, 2001] Singhal, A. (2001). Modern information retrieval: A brief overview. *IEEE Data Engineering Bulletin*, 24(4):35–43.

[Sugiyama et al., 2008] Sugiyama, M., Nakajima, S., Kashima, H., Buenau, P., and Kawanabe, M. (2008). Direct importance estimation with model selection and its application to covariate shift adaptation. In *Advances in Neural Information Processing Systems (NIPS 20)*, pages 1433–1440. Curran Associates, Inc.

[Tao and Zhai, 2006] Tao, T. and Zhai, C. (2006). 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*, pages 162–169. ACM.

[Taylor et al., 2006] Taylor, M., Zaragoza, H., Craswell, N., Robertson, S., and Burges, C. (2006). Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the* 15^{th} *ACM International Conference on Information and Knowledge Management*, pages 585–593. ACM.

[Valizadegan et al., 2009] Valizadegan, H., Jin, R., Zhang, R., and Mao, J. (2009). Learning to rank by optimizing ndcg measure. In *Advances in Neural Information Processing Systems (NIPS 22)*, pages 1883–1891. Curran Associates, Inc.

[Vapnik, 1995] Vapnik, V. (1995). Estimation of dependencies based on small number of observations. In *Proceedings of the IEEE/IAFE Computational Intelligence for Financial Engineering (CIFEr)*, page 41. IEEE Xplore.

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

[Vrajitoru, 1998] Vrajitoru, D. (1998). Crossover improvement for the genetic algorithm in information retrieval. *Information Processing and Management*, 34(4):405–415.

[Xing et al., 2008] Xing, D., Dai, W., Xue, G., and Yu, Y. (2008). Bridged refinement for transfer learning. In *Proceedings of the* 11^{th} *European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD)*, pages 488–496. Springer.

[Xu and Li, 2007] Xu, J. and Li, H. (2007). Adarank: A boosting algorithm for information retrieval. In *Proceedings of the* 30^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 391–398. ACM.

[Xu et al., 2008] Xu, J., Liu, T., Lu, M., Li, H., and Ma, W. (2008). Directly optimizing evaluation measures in learning to rank. In *Proceedings of the* 31^{st} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 107–114. ACM.

[Yang et al., 1992] Yang, J., Korfhage, R., and Rasmussen, E. (1992). Query improvement in information retrieval using genetic algorithms - a report on the experiments of the trec project. In *Proceedings of First Text Retrieval Conference (Trec-1)*, pages 31–58. DIANE Publishing.

[Yeh et al., 2007] Yeh, J., Lin, J., Ke, H., and Yang, W. (2007). Learning to rank for information retrieval using genetic programming. In *Proceedings of the* 30^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM.

[Zaragoza et al., 2004] Zaragoza, H., Craswell, N., Taylor, M., Saria, S., and Robertson, S. (2004). Microsoft cambridge at trec 13: Web and hard tracks. In *Proceedings of the* 13^{th} *Text REtrieval Conference (TREC)*. National Institute of Standards and Technology (NIST).

[Zhai and Lafferty, 2001] Zhai, C. and Lafferty, J. (2001). A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the* 24^{th} *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 334–342. ACM.

[Zhai and Lafferty, 2004] Zhai, C. and Lafferty, J. (2004). A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22(2):179–214.

[Zheng and Fang, 2009] Zheng, W. and Fang, H. (2009). Axiomatic approaches to information retrieval – university of delaware at trec 2009 million query and web tracks. In *Proceedings of The Eighteenth Text REtrieval Conference, TREC 2009*. National Institute of Standards and Technology (NIST).