Incorporating Query Difference for Learning Retrieval Functions in World Wide Web Search

Hongyuan Zha
College of Computing
Georgia Institute of Technology
Atlanta, GA 30332
zha@cc.gatech.edu

Zhaohui Zheng, Haoying Fu
Gordon Sun
Yahoo! Inc.
701 First Avenue
Sunnyvale, CA 94089
{zhaohui, haoying, gzsun}@yahoo-inc.com

ABSTRACT

We discuss information retrieval methods that aim at serving a diverse stream of user queries such as those submitted to commercial search engines. We propose methods that emphasize the importance of taking into consideration of query difference in learning effective retrieval functions. We formulate the problem as a multi-task learning problem using a risk minimization framework. In particular, we show how to calibrate the empirical risk to incorporate query difference in terms of introducing nuisance parameters in the statistical models, and we also propose an alternating optimization method to simultaneously learn the retrieval function and the nuisance parameters. We work out the details for both L_1 and L_2 regularization cases, and provide convergence analysis for the alternating optimization method for the special case when the retrieval functions belong to a reproducing kernel Hilbert space. We illustrate the effectiveness of the proposed methods using modeling data extracted from a commercial search engine. We also point out how the current framework can be extended in future research.

Categories and Subject Descriptors

H.3.3 [**Information Systems**]: Information Search and Retrieval—*Retrieval functions*; H.4.m [**Information Systems**]: Miscellaneous—*Machine learning*

General Terms

Algorithms, Experimentation, Theory

Keywords

relevance, relevance judgment, retrieval function, WWW search, discounted cumulative gain, machine learning, gradient boosting, risk minimization, query document feature, query dependence, query specific feature, alternating optiminization, least-squares regression, quadratic programming, regularization

1. INTRODUCTION

Designing effective and efficient retrieval functions[1] is central to information retrieval, especially in the context of serving many user queries submitted to commercial search engines. Many retrieval models and methods have been proposed in the past, including vector space models, probabilistic models and the more recently developed language modeling-based methods, with varying degree of success [28, 29, 27, 1, 24, 25, 31, 16]. In particular, algorithms and techniques from machine learning have been applied to information retrieval long before the recent advances of the World Wide Web [13, 14, 6, 15, 4, 17]. One key issue in designing retrieval functions is to find efficient and effective representations of query-document relations. In this regard, Fuhr and coworkers considered dividing the design space for query-document representations into query-oriented, document-oriented and feature-oriented schemes [13, 14, 15]. They advocated the use of *feature-oriented* method for probabilistic indexing and retrieval whereby features (relevance descriptions) of query document pairs such as the number of query terms, length of the document text, term frequencies for the terms in the query, etc. are extracted, and (least-squares) regression methods and decision-trees are used for learning the retrieval functions based on a set of query-document pairs represented as feature vectors with relevance assessment [13, 14, 15]. In a related work, Cooper and coworkers have developed similar approaches and used logistic regression to build the retrieval functions and experimented with several retrieval tasks in TREC [6, 17] (see also [16]).

Compared with the query-oriented method and document-oriented method, the greatest advantage of feature-oriented method is its capacity at maintaining large effective sample sizes. This is because the sample space for the feature-oriented method is the collection of all the feature vectors and two different query-document pairs can generate the same or similar feature vectors. The query-oriented method seems to be quite adequate or even advantageous especially in traditional information retrieval where retrieval functions

[1]By retrieval functions we mean functions used to rank documents in response in user queries.

are usually designed to work for small and/or homogeneous text corpora. For more diverse corpora such as the World Wide Web and with the introduction of non-textual features including the information from hyperlinks such as anchor texts and user click-through data, it is more likely that similar feature vectors corresponding to different queries are labeled very differently in terms of relevance assessment. Falling back on the query-oriented method or document-oriented method is not an option because the labeled data with relevance assessment will become even more sparse in the enlarged sample space under those methods, considering that it is usually very expensive to acquire data with high-quality relevance assessment [14, 15]. To overcome this difficulty, we propose to incorporate query difference to sufficiently disambiguate feature vectors while maintaining the general framework of the feature-oriented method. The starting point for our work is the formulation of the information retrieval problem as a multi-task learning problem where the retrieval problem for each query is considered as a separate classification/ranking problem. The goal is to minimize the expected retrieval risk over the whole set of user queries (Section 2). The retrieval problems for the individual queries are closely related to each other, and in fact, in feature-oriented method, the individual retrieval problems are not sufficiently distinguished from each other at all. In section 3, we develop some novel schemes that can incorporate query difference in the feature-oriented framework. This is done by introducing query dependent parameters in the empirical risk which are separated from the retrieval function. We discuss the details of solving the optimization problem and issues on regularizations to control the size of the query dependent parameters. A convergence analysis is also given for the case when the retrieval functions are from a reproducing kernel Hilbert space. In section 4, we present several experimental results using data extracted from a commercial search engine. In section 5, we provide some concluding remarks and give pointers to topics for future research.

2. PROBLEM FORMULATION

In this section we present a general formulation of the information retrieval problem using the risk minimization framework (a related work using risk minimization but emphasizing document generation models and queries from individual users was presented in [32]). We consider $\mathcal{D}$, the set of all the documents in consideration, $\mathcal{L}$ the set of labels which can be either a finite or infinite set, and $\mathcal{Q}$ the set of all potential user queries.[2] We model each query $q \in \mathcal{Q}$ as a probabilistic distribution P_q over $\mathcal{D} \times \mathcal{L}$,

$$P_q(d, \ell), \quad d \in \mathcal{D}, \ \ \ell \in \mathcal{L}$$

which specifies the probability of document d being labeled as ℓ under query q. This probabilistic setting also includes the case where there is a deterministic function

$$f_q : \mathcal{D} \mapsto \mathcal{L}$$

which specifies the "correct" label $f_q(d)$ for each document $d \in \mathcal{D}$.

[2]To be more realistic, the model should also contain a temporal component to account for the time evolution of the document collection and user query stream.

Remark. With labels associated with documents, we are using the absolute relevance framework where judgments are made with respect to the fact that whether a document is or is not relevant to a query. We mention that there is also the possibility of using relative relevance judgment framework where judgments are in the form of whether a document is more relevant than other documents [20, 21, 22]. In certain situations, we may encounter judgments of both forms, absolute relevance judgments from human judgment process and relative relevance judgments extracted from click-through data.

Now we define a loss function L over the set $\mathcal{L} \times \mathcal{L}$,

$$L : \mathcal{L} \times \mathcal{L} \mapsto \mathcal{R}^1_+,$$

the set of nonnegative real numbers, and we also specify a class of functions $\mathcal{H}$ from which the retrieval function will be extracted, where for $h \in \mathcal{H}$,

$$h : \mathcal{Q} \times \mathcal{D} \mapsto \mathcal{L}.$$

For each query q, we can then specify a learning problem (classification or regression problem): find $h^*_q \in \mathcal{H}$ such that

$$h^*_q = \arg\min_{h \in \mathcal{H}} \mathcal{E}_{P_q} L(\ell, h(q, d)),$$

where the expectation $\mathcal{E}_{P_q}$ is with respect to the probability distribution P_q corresponding to the query q.

The goal of information retrieval, however, is not just to learn a retrieval function h^*_q for some individual query q, but rather to learn a retrieval function h^* that will be good for all the queries $q \in \mathcal{Q}$. Therefore, we need to deal with potentially *infinite* number of *related* learning problems, each for one of the query $q \in \mathcal{Q}$. To this end, we specify a distribution over $\mathcal{Q}$: $P_Q(q)$ can indicate, for example, the probability that a specific query q is issued to the information retrieval system which can be approximated, for example, by its frequency in the query logs of queries submitted to the information retrieval system in the past. Then the optimization problem we need to solve is for the combined risk,

$$\min_{h \in \mathcal{H}} \mathcal{E}_{P_Q} \mathcal{E}_{P_q} L(\ell, h(q, d)) \qquad (1)$$

In practice, we sample a set of queries $\{q_i\}_{i=1}^{Q}$ from the distribution P_Q, and for each query q, we also sample a set of documents from $\mathcal{D}$ for labeling to obtain

$$\{d_{qj}, l_{qj}\}, \quad q = 1, \dots, Q, \ j = 1, \dots, n_q$$

where $l_{qj} \in \mathcal{L}$ are labels obtained from human judges for example after relevance assessment. Ideally, the sampling of the queries should be according to P_Q and the sampling of the documents for each query q should be according to P_q, the former is relatively easy to do while the later is a much more difficult issue. The optimization problem for the empirical counterpart of (1) is

$$\min_{h \in \mathcal{H}} \sum_{q=1}^{Q} \sum_{j=1}^{n_q} L(\ell_{qj}, h(q, d_{qj})).$$

In general we also add a regularization term to control the complexity of h with $\Omega(h)$ measuring the complexity of h, and the optimization problem becomes

$$\min_{h \in \mathcal{H}} \sum_{q=1}^{Q} \sum_{j=1}^{n_q} L(\ell_{qj}, h(q, d_{qj})) + \lambda\, \Omega(h),$$

here λ is the regularization parameter that balances the fit of the model in terms of the empirical risk and the complexity of the model [9].

An important consequence of the above framework is that we are simultaneously dealing with multiple learning tasks each for an individual query, and there are substantial correlations and overlaps among those learning tasks and it is important to explore the solution of (1) in the context of multi-task learning. The feature-oriented method where the feature vector does not contain any query features can be considered as an extreme form of multi-task learning for (1) where the individual learning problems loss their identity and become a single learning problem. The main objective of this paper is to work within the multi-task learning by extending the feature-oriented method.

REMARK. Many loss functions have been proposed for both classification and regression problems. For example, for the binary classification problem, we can use the exponential loss used in Adaboost [10]

$$L(\ell, h(q,d)) = \exp(-\ell h(q,d)),$$

where $\ell \in \{-1,1\}$ denotes the class labels, and for the regression problem, we can use the squared-error loss,

$$L(\ell, h(q,d)) = (\ell - h(q,d))^2,$$

where ℓ is a real number.

REMARK. Instead of considering classification and regression problems, we can also consider directly the ranking problem for information retrieval: Each query corresponds to an ideal ranking function R_q defined on $\mathcal{D}$. Let $\mathcal{R}$ be a class of ranking functions, we define a loss function L that measures the difference between two ranking functions, then the optimization we need to solve is

$$\min_{R \in \mathcal{R}} \mathcal{E}_{P_Q} L(R_q, R).$$

A recent work along those lines can be found in [3].

3. INCORPORATING QUERY DIFFERENCE IN THE EMPIRICAL RISK

To discuss the subtle issue of query difference, we first consider a simple illustrative example. Consider two queries q_1: "harvard university" and q_2: "college of san mateo". q_1 is more popular than q_2 generating about 13 million search results while results for q_2 are two orders of magnitude less. For simplicity we consider a retrieval function $h(x)$ using a single feature x which counts the number of inbound links to a document. Now let us examine the top three results $d_{i1}, d_{i2}, d_{i3}, i = 1, 2$, for each of the query. Assume d_{i1} is ranked perfect, d_{i2} is ranked excellent, and d_{i3} is ranked good, and we convert the labels to numerical values as follows,

$$0 \Leftrightarrow \text{perfect},\ 1 \Leftrightarrow \text{excellent},\ 2 \Leftrightarrow \text{good}.$$

Since q_1 is very popular, each of the top three results generate high feature values, say,

$$x = 100000,\ 80000,\ 50000$$

while for q_2 the corresponding feature values are

$$x = 1000,\ 800,\ 500.$$

Assume x is negatively correlated with the label, i.e., small x values tend to indicate better relevance, then we will need to find a monotonically decreasing function h such that for q_1

$$h(100000) \approx 0,\ h(80000) \approx 1,\ h(50000) \approx 2$$

and for q_2,

$$h(1000) \approx 0,\ h(800) \approx 1,\ h(500) \approx 2.$$

This is, however, impossible to do. As we can see the major issue comes from the difference of popularity of the queries and we need a framework for designing retrieval functions that can take this difference into account. The above is a simple illustrative example and for a single query we can use simple scaling to achieve the same effects. But in reality, a retrieval function involves several thousands or more features and simple scaling will not work. To this end, we discuss the issue in more formal terms.

To be concrete, we consider the risk minimization problem in the context of regression, i.e., we assume the labels are real numbers, and to be consistent with convention, we will use y_{qj} to denote the label ℓ_{qj}. We also use the squared-error loss function. Since we operate under the feature-oriented framework, we assume the labeled set is represented as

$$\{x_{qj}, y_{qj}\}, \quad q = 1, \ldots, Q,\ j = 1, \ldots, n_q,$$

here x_{qj} denote the feature vector for the query-document pair $\{q, d_{qj}\}$. Before we proceed to formally discuss the problem, we want to address an important and subtle issue. For a query q, we say a feature is query-dependent if it has the same value across all the documents $d \in \mathcal{D}$. For example, the number of terms in q is a query-dependent feature. Other features are called document-dependent and query-document-dependent. So we can split the feature vector x_{qj} into two parts

$$x_{qj} = [x_{qj}^{Q},\ x_{qj}^{D},\ x_{qj}^{QD}]$$

with the first component indicates the set of query-dependent features (more details on those three types of features can be found in Section 4.1). One extreme form of a retrieval function is a function that only depends on x_{qj}^{D}, x_{qj}^{QD}, i.e.,

$$\{h([x_{qj}^{D},\ x_{qj}^{QD}])\}$$

is used for ranking documents for all the queries. At the other end of the spectrum is to have one function

$$h([x_{qj}^{D},\ x_{qj}^{QD}])$$

for each query $q \in \mathcal{Q}$. On the other hand, to have a single function

$$h([x_{qj}^{Q},\ x_{qj}^{D},\ x_{qj}^{QD}])$$

for all queries and using x_{qj}^{Q} to disambiguate the queries seems to be quite natural. But the problem with this approach is that it is not known *a-priori* what set of query-dependent features are adequate for this purpose; for example, just using the number of query terms in a query is certainly too coarse. Our philosophy instead is to let the data implicitly capture this set of adequate query-dependent features and completely bypass its explicit construction. In particular, we allow the feature vector to contain any number of query-dependent features or none at all.

To this end, we seek to find a function $h \in \mathcal{H}$ to minimize the following empirical risk,

$$L(h) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} (y_{qj} - h(x_{qj}))^2.$$

We will deal with regularization issue momentarily. To incorporate query-dependent effects, we can consider the following modified empirical risk, and we seek to find h and $g_q, q = 1, \ldots, Q$, to minimize

$$L(h, g) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - g_q(h(x_{qj}))]^2, \tag{2}$$

where $g_q(\cdot)$ is a general monotonically increasing function, and $g = [g_1, \ldots, g_Q]$, i.e., we seek

$$\{h^*, g^*\} = \text{argmin}_{\{h \in \mathcal{H}, g \text{ mono increasing}\}} L(h, g).$$

The intention is that the function g_q incorporate the difference for queries when using $h(x)$ to predict the label y. From another viewpoint, for suitably chosen g_q, we seek to find $h(x)$ to match $g_q^{-1}(y)$. This has some flavor of response transformation used in general regression analysis [5]. However, in our case, the response transformation $g_q^{-1}(\cdot)$ is not fixed in advance but is learned from the data, and it is also query dependent.

Now for a new query q_n which is not in the given labeled set, there is also a corresponding function g_{q_n}. However, the retrieval function is used for ranking the documents and since g_{q_n} is monotonically increasing, rankings based on $g_{q_n}(h)$ and h are exactly the same, and for using the latter, there is no need to know g_{q_n}.

In this work we focus on the simple case where $g_q(\cdot)$ is a linear function,[3] i.e.,

$$g_q(x) = \beta_q + \alpha_q x, \quad q = 1, \ldots, Q$$

with $\alpha_q \geq 0$. The query dependent parameters β_q and α_q are *nuisance parameters* because they do not appear in the retrieval function h. Interestingly, even though we incorporate query dependent parameters in learning the retrieval function, to assess the relevance for a document $d \in \mathcal{D}$, we simply look at the learned retrieval function h. For example, to rank a list of documents $d_i \in \mathcal{D}, i = 1, \ldots, n$, for a query q, we can simply sort them according to the values $h(x_i), i = 1, \ldots, n$, where x_i is the feature vector for the query document pair $\{q, d_i\}$.

As we mentioned before, to control the size of the parameters β_q, α_q and the complexity of h, we also need to add regularization terms to the modified empirical risk (2) to obtain the regularized empirical risk

$$L(h, \beta, \alpha) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 +$$

$$+ \lambda_\beta \|\beta\|_p^p + \lambda_\alpha \|\alpha\|_p^p + \lambda_h \Omega(h),$$

where $\beta = [\beta_1, \ldots, \beta_Q]$ and $\alpha = [\alpha_1, \ldots, \alpha_Q]$, and $\lambda_\beta, \lambda_\alpha$ and λ_h are regularization parameters, and $\| \cdot \|_p$ is the p norm of a vector. We will only consider the case for $p = 1$ or $p = 2$. In summary, we seek to find

$$\{h^*, \beta^*, \alpha^*\} = \text{argmin}_{\, h \in \mathcal{H}, \beta, \alpha \geq 0} L(h, \beta, \alpha). \tag{3}$$

In general, we will not impose a parametric form for the function h, and we will employ the methodology of coordinate descent (alternating optimization) to solve the optimization problem (3) [2]. Specifically, we will alternate between optimizing against h and optimizing against β and α. The regularization parameter λ_h will be determined during the nonlinear regression process for finding h discussed below while regularization parameters λ_β and λ_α will be determined by cross-validation (see section 4.4 for more details).

[3]We can use more complicated monotone functions by using more parameters or even monotone functions in nonparametric form.

3.1 Nonlinear regression

For fixed β and α, and for simplicity, we assume $\alpha_q > 0$, define the modified residuals

$$\hat{y}_{qj} = (y_{qj} - \beta_q)/\alpha_q, \quad q = 1, \ldots, Q, \; j = 1, \ldots, n_q.$$

Then we seek to find h that solves the following *weighted* nonlinear regression problem

$$\sum_{q=1}^{Q} \sum_{j=1}^{n_q} \alpha_q^2 [\hat{y}_{qj} - h(x_{qj})]^2 + \lambda_h \Omega(h).$$

Many nonparametric fitting algorithms can be used to find h including MARS, the recently proposed boosting methods, and the general kernel-based methods [10, 11, 12]. For more detailed discussion in the context of learning retrieval function see [7, 8]. In section 3.3, we present a detailed discussion for the kernel-based methods. Here we briefly describe the general framework of gradient boosting, assuming we are given a training set $\{x_i, y_i\}_{i=1}^N$ with a loss function $(y - h(x))^2$ [12]. The gradient boosting algorithm consists of the following steps.

1. Initialize $h_0(x) = \sum_{i=1}^N y_i/N$.
2. For i=1,. . . , M: (number of trees in gradient boosting)
   (a) For i=1,. . . , N, compute the negative gradient
   $$r_{im} = -(y_i - f_{m-1}(x_i))$$
   (b) Fit a regression tree to $\{r_{im}\}_{i=1,\ldots,N}$ giving terminal regions $R_{jm}, j = 1, \ldots, J_m$.
   (c) For $j = 1, \ldots, J_m$, compute
   $$\gamma_{jm} = \sum_{x_i \in R_{jm}} (y_i - f_{m-1}(x_i))/|\{i : \; x_i \in R_{jm}\}|$$
   (d) Update
   $$f_m(x) = f_{m-1}(x) + \eta(\sum_{j=1}^{J_m} \gamma_{jm} I(x \in R_{jm})),$$
   where η is the *shrinkage factor.*

There are two parameters M, the number of regression trees and η, the shrinkage factor that need to be chosen by the user. We use cross-validation for choosing the two parameters.

3.2 Computing α and β

Next we consider for fixed h, how to optimize against β and α. We need to solve the optimization problem

$$\min_{\beta, \alpha} \sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 + \lambda_\beta \|\beta\|_p^p + \lambda_\alpha \|\alpha\|_p^p.$$

It is easy to see that we can break the above into Q separate optimization problems, for $q = 1, \ldots, Q$,

$$\min_{\beta_q, \alpha_q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 + \lambda_\beta |\beta_q|^p + \lambda_\alpha |\alpha_q|^p.$$

We distinguish two cases in what follows.

3.2.1 The L_2 case

For $p = 2$, the component β_q and α_q of β and α can be found by solving the following least squares problem,

$$\min_{\{\beta_q,\alpha_q\}} \sum_{j=1}^{n_q} (y_{qj} - \beta_q - \alpha_q h(x_{qj}))^2 + \lambda_\beta \beta_q^2 + \lambda_\alpha(\alpha_q - 1)^2,$$

where to force nonnegativity on α_q, we control the size of $\hat{\alpha}_q = \alpha_q - 1$ instead of α_q and consequently, α_q will be around 1. Let the data matrix for query q and the response be

$$X_q = \begin{bmatrix} 1 & h(x_{q1}) \\ 1 & h(x_{q2}) \\ \vdots & \vdots \\ 1 & h(x_{q,n_q}) \end{bmatrix} \in R^{n_q \times 2}, \quad \hat{y}_q = \begin{bmatrix} y_{q1} - h(x_{q1}) \\ y_{q2} - h(x_{q2}) \\ \vdots \\ y_{n_q} - h(x_{qn}) \end{bmatrix}.$$

The optimal β_q and α_q can be obtained by solving the following normal equation,

$$\left(X_q^T X_q + \begin{bmatrix} \lambda_\beta & 0 \\ 0 & \lambda_\alpha \end{bmatrix} \right) \begin{bmatrix} \beta_q \\ \hat{\alpha}_q \end{bmatrix} = X_q^T \hat{y}_q,$$

here superscript T denotes matrix transpose.

3.2.2 The L_1 case

For $p = 1$, we need to solve the following optimization problem,

$$\min_{\{\beta_q,\alpha_q\}} \sum_{j=1}^{n_q} (y_{qj} - \beta_q - \alpha_q h(x_{qj}))^2 + \lambda_\beta |\beta_q| + \lambda_\alpha |\alpha_q - 1|. \quad (4)$$

We can turn the above into a quadratic programming problem. To this end, we introduce the following notation, for a real number r, we define

$$r^+ = \max\{0, r\}, \quad r^- = \max\{0, -r\}.$$

Then it is easy to see that $r = r^+ - r^-$ and $|r| = r^+ + r^-$. Let $\hat{\alpha}_q = \alpha_q - 1$, and we rewrite (4) as

$$\min \sum_{j=1}^{n_q} [y_{qj} - (\beta_q^+ - \beta_q^-) - (\hat{\alpha}_q^+ - \hat{\alpha}_q^- + 1)h(x_{qj})]^2 +$$

$$+ \lambda_\beta(\beta_q^+ + \beta_q^-) + \lambda_\alpha(\hat{\alpha}_q^+ + \hat{\alpha}_q^-),$$

subject to the constraints

$$\beta_q^+ \geq 0, \ \beta_q^- \geq 0, \ \hat{\alpha}_q^+ \geq 0, \ \hat{\alpha}_q^- \geq 0.$$

The above optimization problem is in the standard form of convex quadratic programming and can be solved by the available methods discussed in [26].

3.3 Convergence and stopping criteria

In this section, we present a convergence analysis of the above alternating optimization method for optimization problem (3). We specialize to the case when $\mathcal{H}$ the set of functions in (3) is restricted to a reproducing kernel Hilbert space (RKHS) [30]. We first give a very brief description of RKHS just to introduce the relevant notations. A RKHS is characterized by a continuous, symmetric, positive definite kernel function $K(\cdot,\cdot)$,

$$K : \mathcal{X} \times \mathcal{X} \mapsto \mathcal{R}_+^1,$$

where $\mathcal{X}$ is the domain of the functions, and K satisfies

$$K(x, y) = K(y, x), \quad \sum_{i,j=1}^{n} a_i a_j K(x_i, x_j) \geq 0.$$

For a fixed $x \in \mathcal{X}$, $K(x,\cdot)$ is a function on $\mathcal{X}$. Consider the linear space $\mathcal{H}_0$ spanned by $\{K(x,\cdot), x \in \mathcal{X}\}$, i.e., $\mathcal{H}_0$ consists of finite linear combination of the form $f(x) = \sum_{i=1} c_i K(x_i,\cdot)$. For any two functions f and g in $\mathcal{H}_0$,

$$f(x) = \sum_{i=1} c_i K(x_i, x), \quad g(x) = \sum_{i=1} d_i K(y_i, x)$$

Define an inner-product on $\mathcal{H}_0$ by

$$< f, g >_K = \sum_{i,j=1}^{n} c_i d_j K(x_i, y_j).$$

It is easy to see that

$$< f, K(x,\cdot) >_K = f(x)$$

and this is where the name reproducing kernel comes from. Now we can define $\mathcal{H}_K$ the RKHS uniquely associated with K as the completion of $\mathcal{H}_0$ under the norm $\|f\|_K^2 =< f, f >_K$.

For a fixed kernel function K, we replace $\mathcal{H}$ in (3) by the RKHS $\mathcal{H}_K$ and the complexity measure $\Omega(h)$ by $\|h\|_K^2$. One advantage of using RKHS is the following representer theorem [30] which we rewrite in the context of the optimization problem (3).

Theorem 1. *The optimal function h^* for the optimization problem (3) has the following form,*

$$h^*(x) = \sum_{q=1}^{Q} \sum_{j=1}^{n_q} c_{qj} K(x_{qj}, x),$$

where $c_{qj}, q = 1, \ldots, Q, j = 1, \ldots, n_q$, are real numbers.

In essence, Theorem 1 reduces the search over $\mathcal{H}_K$ which is an infinite dimensional space to the search for c_{qj} belonging to a finite dimensional Euclidean space, and we can write the emipirical risk $L(h, \beta, \alpha) \equiv L(c, \beta, \alpha)$ with $c = [c_{qj}]$. With the above discussion, the alternating optimization algorithm discussed before can be more concisely described as

Algorithm. (Alternating optimization)

- Start with an initial guess β^0 and α^0.
- For $k = 1, 2, \cdots$ until convergence, compute

$$c^k = \text{argmin}_{\ c}\ L(c, \beta^k, \alpha^k), \quad (5)$$

$$\{\beta^{k+1}, \alpha^{k+1}\} = \text{argmin}_{\ \beta,\alpha\geq 0}\ L(c^k, \beta, \alpha). \quad (6)$$

We now present a convergence result using the general convergence theorem for block coordinate descent in Section 2.7 of [2].

Theorem 2. *Every limit point of $\{c^k, \beta^k, \alpha^k\}_{k=1}^\infty$ is a stationary point of $L(c, \beta, \alpha)$.*

Proof. According to proposition 2.7.1 in [2], we need to check that both the optimization problems (5) and (6) have unique solutions, and the feasible regions for both are convex. Convexity for the feasible regions for both (5) and (6) are obvious. For fixed c, it is easy to see that the objective function

$$\sum_{q=1}^{Q} \sum_{j=1}^{n_q} [y_{qj} - \beta_q - \alpha_q h(x_{qj})]^2 + \lambda_\beta |\beta|^p + \lambda_\alpha |\alpha|^p$$

Table 1: Number of queries and query-url pairs on US and CN datasets

	# queries	# total query-document pairs
Chinese	2649	78188
English	910	70742

is strictly convex for $p > 1$, hence minimizer is unique. For fixed β and α, using the representer theorem of Theorem 1, the objective function for the optimization problem (5) can be written as

$$\sum_{q=1}^{Q}\sum_{j=1}^{n_q}[y_{qj} - \beta_q - \alpha_q \sum_{p=1}^{Q}\sum_{i=1}^{n_q} c_{pi}K(x_{pi}, x_{qj})]^2 +$$

$$+\lambda_h \sum_{p=1}^{Q}\sum_{i=1}^{n_q}\sum_{q=1}^{Q}\sum_{j=1}^{n_q} c_{pi}c_{qj}K(x_{pi}, x_{qj})$$

which is also seen to be strictly convex in c.

Remark. One alternative to the Alternating Optimization method for the RKHS case is to directly optimize the function $L(c, \beta, \alpha)$.

Stopping criteria. In practice, we iterate the Alternating Optimization Algorithm until β and α do not change very much, for example, we specify $\eta > 0$, and terminate the iterations once $\|\beta^{k+1} - \beta^k\| + \|\alpha^{k+1} - \alpha^k\| \le \eta$.

4. EXPERIMENTAL RESULTS

In this section we report some experimental results on data generated from a commercial search engine. We focus on data for two popular languages English and chinese, respectively.

4.1 Data collection

In our experiments, a set of queries are sampled from query logs, and a certain number of query-document pairs are labeled according to their perceived relevance judged by human editors. The labels are mapped to numerical values and the goal is to learn a retrieval function that can best mimic the human judgment process. In Table 1, we list some basic statistics for the two data sets.

The construction of the English and Chinese training sets mainly consists of the following steps.

- randomly sample a certainly number of queries from the query logs.
- for each query, we sample certain number of documents for judgments.
- human judges assign a $0-4$ grade to each query-url pair based on the degree of relevance (perfect match, excellent match, etc). The grades will be used as target values for regression.
- each query-url pair is then represented with a feature vector as we detail below.

For each query-document pair (q, d) with $q \in \mathcal{Q}$ and $d \in \mathcal{D}$, a feature vector $x = [x^Q, x^D, x^{QD}]$ is generated and the features generally fall into the following three categories:

- Query-feature vector x^Q which comprises features dependent on the query q only and have constant values across all the documents $d \in \mathcal{D}$, for example, the number of terms in the query, whether or not the query is a person name, etc.
- Document-feature vector x^D which comprises features dependent on the document d only and have constant values across all the queries $q \in \mathcal{Q}$, for example, the number of inbound links pointing to the document, the amount of anchor-texts in bytes for the document, and the language identity of the document, etc.
- Query-document feature vector x^{QD} which comprises features dependent on the relation of the query q with respect to the document d, for example, the number of times each term in the query q appears in the document d, the number of times each term in the query q appears in the anchor-texts of the document d, etc.

We use two data sets in our experiments, one mainly consists of English queries and the other mainly chinese queries, and the two sets are labeled as English and Chinese, repectively. In Table 1 we list the number of queries and query-document pairs for the two data sets. The numbers of features for Chinese and US datasets vary from a few hundreds to more than a thousand. The difference between the numbers of features are mainly due to the different maximum query lengthes for the two datasets.

4.2 Evaluation metrics

Many evaluation metrics have been proposed to assess the effectiveness of information retrieval systems including the popular precision-recall method. In this work, we use the recently popularized Discounted Cumulative Gain (DCG) methodology which seems to be more appropriate for assess relevance in the context of search engines [19]. For a ranked list of N documents(N is set to be 5 in our experiments), we use the following variation of DCG,

$$\text{DCG}_N = \sum_{i=1}^{N} \frac{G_i}{\log_2(i+1)},$$

where G_i represents the weights assigned to the label of the document at position i. Higher degree of relevance corresponds to higher value of the weight. We will use the symbol *dcg* to indicate the average of this value over a set of queries in our experiments.

4.3 Experiment methodology

In this section, we provide some more details on the learning and evaluation methods used in our experiments.

4.3.1 *Adaptive target value transformation* (aTVT)

In section 3 we mentioned that our proposed methods can also be considered as an adaptive way to adjust the response values. The alternating optimization process consists of fitting the α and β values and the fitting of the nonlinear function h which amounts to solving a *weighted* nonlinear least squares problem. For situations where it is not convenient to incorporate weights, we can slightly modify the empirical loss to be (cf. section 3)

$$\min_{\{\alpha_q, \beta_q\}} \sum_{j=1}^{n_q}(\alpha_q y_{qj} + \beta_q - h(x_{qj}))^2 + \lambda_\alpha|(\alpha_q - 1)|_p^p + \lambda_\beta|\beta_q|_p^p.$$

For each choice of regularization parameters λ_α and λ_β we optimize the regression function and the parameters α_q and β_q as follows:

1) initialize y^0_{qj} to the assigned numerical values for each query-document pair (q, d);

2) iterate until the α^k_q and β^k_q do not change much (cf. section 3.3) or $k, k = 1, 2, ...,$ reaches the maximum number of iterations K ($K = 10$, in this work), do the following,

   a) fit a nonlinear function on $\{y^{k-1}_{qj}\}$ using the general methodology discussed in [7, 8, 12].

   b) obtain optimal values for the α^k_q and β^k_q as in section 3.2.1 or 3.2.2.

   c) $y^k_{qj} \leftarrow \alpha^k_q y^{k-1}_{qj} + \beta^k_q$.

Then $\alpha_q = \prod_{k=1}^K \alpha^k_q$, and $\beta_q = \sum_{k=1}^K (\beta^k_q \prod_{i=k+1}^K \alpha^i_q)$.

In what follows, we will refer the above algorithm as *adaptive Target Value Tranformation* (aTVT). We will compare aTVT against a particular system that uses nonlinear regression to learn a retrieval function based on the gradient boosting methods [7, 8, 12]. It is the same fitting method used in Step 2/a) in the above. In our experiments, we will refer the retrieval function of this system as the retrieval function without aTVT. We should mention that many methods for nonlinear regression can be used for fitting the h function, we adopt the methods in [7, 8] so that we can provide a consistent comparison.

4.3.2 *Using cross validation for comparison*

The main objective of this work is to demonstrate the potential advantage of incorporating query difference in learning retrieval functions. In general, the idea can be applied to many existing information retrieval systems. We will only focus on the comparison between systems using aTVT and the same system without using aTVT. A more comprehensive comparison of the existing information retrieval systems is beyond the scope of this work.

The comparison is carried out in the following way, and in essence we seek to assess the significance of the observed difference between systems with aTVT and systems without aTVT in terms of the difference in the DCG values.

- Randomly split the set of queries in data set into 10 folds and obtain 10 9-vs-1 combinations.
- For each 9-vs-1 combination, do the following:
  - learn a retrieval function using the data in the 9 folds as training data with and without aTVT, respectively.
  - test the learned retrieval function on the remaining one fold by computing the DCG values for the queries in the one fold, and thus obtain two lists of query-*dcg* pairs corresponding to retrieval function with and without aTVT, respectively.
- For each of the two methods, we concatenate the above lists from the 10 combinations together to obtain a full list of query-*dcg* pairs for all the queries in the data set. Notice that the orders of queries in the two full lists are the same.

Table 5: *dcg* gains and percentage of gains for retrieval functions with aTVT over without aTVT at different percentiles on English data for the L_2 case with $\lambda_\alpha = 10$ and $\lambda_\beta = 1$

percentile	Δ_{dcg}	$\%\Delta_{dcg}$
0%	11.35	1454.84
10%	2.51	29.91
20%	1.44	15.23
30%	0.79	7.88
40%	0.19	1.80
50%	0.00	0.00
60%	0.00	0.00
70%	-0.28	-2.84
80%	-1.08	-9.48
90%	-1.76	-18.02
100%	-6.98	-92.54

- Finally, we conduct Wilcoxon signed rank tests on the two lists and obtain the p-values for the average dcg gains for systems with aTVT over systems without aTVT [18].

4.4 Comparison results

Before we present our results we want to emphasize that the baseline result is based on a state of the art commercial search engine, and for all the top search engines the difference in dcg values is below 5%.

Table 2 and Table 3 list the *dcg* for retrieval function with aTVT as compared to retrieval function without aTVT in the L_2 case (cf. 3.2.1) for the English and Chinese data sets, respectively. The percentage dcg gains and the p-values from Wilcoxon signed rank tests are also presented. From the two tables, we can see aTVT gives statistically significant dcg gains for both English and Chinese data sets. The optimal regularization parameter combinations give about 2% *dcg* gain for the English data and slightly more than 1% *dcg* gain for the Chinese data.

Table 4 also shows the means and variances of the optimal α_q and β_q values for the queries in English data set. As is expected, increasing λ_α and λ_β will move α and β closer toward to 1 and 0, respectively. Retrieval function without aTVT corresponds to $\alpha = 1$ and $\beta = 0$, where $\lambda_\alpha \to +\infty$ and $\lambda_\beta \to +\infty$. Our experimental results also indicate that regularization with respect to α and β is very important to prevent overfitting. Compared with the dcg values for retrieval function without aTVT, aTVT without regularization on α and β, e.g., $\lambda_\alpha \to 0$ and $\lambda_\beta \to 0$, degrade *dcg*.

Table 5 shows the absolute and percentage of dcg gains at different percentiles of queries for the English data form the L_2 case and the fixed regularization parameters: $\lambda_\alpha = 10$ and $\lambda_\beta = 1$.

Due to the limit of time, we only tried a few combinations of regularization parameters on English data for the L_1 case. The *dcg* for aTVT with one iteration $K = 1$, $\lambda_\alpha = 100$ and $\lambda_\beta = 0.2$ are 11.45(+1.75%, $p-$value=0.001). The mean and variance of α_q are 0.92 and 0.014 respectively, and those for β_q are 0.22 and 0.179, respectively.

In Table 6, we list percentage *dcg* gains at different query percent intervals for the English data. We can see that for those queries where the *dcg* does not change much, the av-

Table 2: The dcgs and percentage dcg increases of retrieval function with aTVT over without aTVT on the English data set, and p values for different regularization parameters: λ_α and λ_β in the L_2 case. Notice that the *dcg* for retrieval function without aTVT are 11.30.

		λ_β=1	10	50	100
λ_α=1	*dcg*	11.48(+1.55%,p=0.01)	11.49(+1.68%,p=0.006)	11.51(+1.83%,p=0.005)	11.46(+1.43%,p=0.02)
10	*dcg*	**11.53(+1.98%,p=0.002)**	11.52(+1.88%,p=0.002)	11.47(+1.44%,p=0.02)	11.47(+1.49%,p=0.02)
50	*dcg*	11.50(+1.75%,p=0.002)	11.51(+1.79%,p=0.008)	11.45(+1.27%,p=0.03)	11.45(+1.31%,p=0.03)
100	*dcg*	11.51(+1.83%,p=0.007)	11.49(+1.65%,p=0.003)	11.45(+1.31%,p=0.07)	11.48(+1.52%,p=0.01)

Table 3: Same as table 2, but for the Chinese data set. Notice that the *dcg* for retrieval function without aTVT is 7.87.

		λ_β=1	10	50	100
λ_α=1	*dcg*	7.81(-.80%,p=0.02)	7.90(+.36%,p=0.43)	7.89(+.24%,p=0.62)	7.93(+.72%,p=0.03)
10	*dcg*	7.84(-.36%,p=0.42)	7.87(-.02%,p=0.88)	7.93(+.69%,p=0.03)	**7.96(+1.14%,p=0.004)**
50	*dcg*	7.87(+.08%,p=0.80)	7.91(+.45%,p=0.21)	7.94(+.87%,p=0.01)	7.96(+1.12%,p=0.003)
100	*dcg*	7.89(+.28%,p=0.25)	7.91(+.49%,p=0.07)	7.94(+.86%,p=0.009)	7.96(+1.10%,p=0.01)

Table 6: The average α, β and percentage of gains for retrieval functions with aTVT at different percentage intervals on English data for the L_2 case with $\lambda_\alpha = 10$ and $\lambda_\beta = 1$

interval	average α	average β	average $\%\Delta_{dcg}$
0-10%	0.1436	2.4803	118.56
10-20%	0.1450	2.4385	21.98
20-30%	0.1461	2.4134	11.53
30-40%	0.1571	2.3953	4.41
40-50%	0.1682	2.3033	0.55
50-60%	0.1838	2.2995	0.00
60-70%	0.1539	2.3674	-1.04
70-80%	0.1523	2.3753	-6.54
80-90%	0.1456	2.4196	-13.65
90-100%	0.1425	2.4428	-33.58

erage α tends to be bigger while the average β tends to be smaller, for those queries where the *dcg* changes significantly, the opposite is true. To gain better insights into the question—for what kind of queries, aTVT is more effective—we notice that the average total number of anchor text lines for the top 10% and bottom 10% queries are about 15.42 and 10.83, respectively which is significantly different from the average for all queries (22.89). As a comparison, the average total number of anchor text lines for the queries with small *dcg* change is about 24.71 which is very close to the average for all the queries. Certainly more work needs to be done to understand the effectiveness of aTVT at the query level.

A more telling story is what are presented in Table 7. Here we sort all the 910 English queries according to the corresponding values of $\delta \equiv |\alpha^{0.1} - 1.0| + |\beta/10|$ which measures how far the α and β deviate from their nominal values of one and zero. Those queries with larger δ have larger adjustment to compensate query difference, and the corresponding improvement in DCG is also larger. The adjustment for query

Table 7: dcg gains and corresponding p-values for queries sorted according to $|\alpha^{0.1} - 1.0| + |\beta/10|$

# top queries	dcg gain of aTVT	p-value
200	5.33%	0.002
300	4.21%	0.002
400	4.01%	0.0004
500	3.42%	0.0006
600	2.75%	0.001
700	2.57%	0.0006
800	2.34%	0.0005
910	1.98%	0.002

difference is purely data-driven and it is not easy to attribute the adjustments to some specific aspects of the query such as length and language identity etc. One the other hand, it is also possible to devise more targeted adjustments using query classes for example. We will mention this issue again in the conclusion remarks.

5. CONCLUDING REMARKS

We cast the information retrieval problem as a multi-task learning problem which presents a natural setting for discussing the important issues of query difference and its impact in learning effective retrieval functions. There are many ways to incorporate query difference in learning retrieval functions, the approaches of constructing appropriate query features being one of them even though it is usually not looked at from this viewpoint. In this paper, we present an approach through modifications of the empirical risk using nuisance parameters to accommodate the effects of query difference. The approach can be used even when there are query features contained in the feature vector of query-document pairs. In this work we have also developed numerical algorithms for solving the resulted optimization problems for minimizing the modified empirical risk. Based on data sets extracted from a commercial search engine, we

Table 4: The means and variances of α and β for different regularization parameters on English data. where: Case $p = 2, \lambda_\alpha = 10$ and $\lambda_\beta = 1$

		λ_β=1	10	50	100
λ_α=1	mean(α),var(α)	0.11, 0.002	0.27, 0.006	0.41, 0.006	0.45, 0.006
	mean(β),var(β)	2.5116, 0.031	1.73, 0.061	0.86, 0.048	0.55, 0.029
10	mean(α),var(α)	0.15, 0.003	0.29, 0.006	0.42, 0.006	0.46, 0.006
	mean(β),var(β)	2.39, 0.041	1.69, 0.066	0.85, 0.049	0.54, 0.03
50	mean(α),var(α)	0.26, 0.006	0.35, 0.007	0.46, 0.007	0.5, 0.006
	mean(β),var(β)	2.09, 0.073	1.56, 0.082	0.80, 0.055	0.52, 0.033
100	mean(α),var(α)	0.35, 0.008	0.41, 0.008	0.50, 0.007	0.53, 0.006
	mean(β),var(β)	1.86, 0.095	1.43, 0.095	0.76, 0.06	0.49, 0.036

also demonstrate that the proposed method gives good improvements in terms of DCG gains. We believe our approach represents an initial step towards better understanding the issues of query difference and learning retrieval functions and there are many topics that deserve further investigation including 1) explore the relation between constructing retrieval functions using query-dependent features and the approaches we used in this work, especially elucidate the concept of adequate query-dependent features; 2) prove generalization bounds for the multi-task risk minimization problem; and 3) explore query difference in other information retrieval models. In particular, the aTVT framework can be combined with other prior information, for example, we can devise a set of class structures for queries and use a set of α and β for each class of the queries instead of one set for each query, many sort of variations on this idea can be further investigated.

6. ACKNOWLEDGMENTS

The work of the first author was supported in part by Yahoo!.

7. REFERENCES

[1] A. Berger. *Statistical machine learning for information retrieval.* Ph.D. Thesis, School of Computer Science, Carnegie Mellon University, 2001.

[2] D. Bertsekas. *Nonlinear programming.* Athena Scientific, second edition, 1999.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *Proceedings of international conference on Machine learning*, 89–96, 2005.

[4] H. Chen. Machine Learning for information retrieval: Neural networks, symbolic learning and genetic algorithms. *JASIS*, 46:194-216, 1995.

[5] R. D. Cook and S. Weisberg. *Residuals and influence in regression.* Chapman & Hall, 1982.

[6] W. Cooper, F. Gey and A. Chen. Probabilistic retrieval in the TIPSTER collections: an application of staged logistic regression. *Proceedings of TREC*, 73-88, 1992.

[7] D. Cossock. Method and apparatus for machine learning a document relevance function. US patent application, 20040215606, 2003.

[8] D. Cossock and T. Zhang. Subset ranking using regression. Technical Report, Yahoo! Research Laboratory, 2006.

[9] F. Cucker and S. Smale. On the mathematical foundations of learning. *Bull. Amer. Math. Soc.*, 39:1–49, 2002.

[10] Y. Freund and R. Schapire. Experiments with a new boosting algorithm. *Proceedings of the international conference on Machine learning*, 148–156, 1996.

[11] J. Friedman. Multivariate adaptive regression splines (with discussion). *Ann. Statist.*, 19:1-141, 1991.

[12] J. Friedman. Greedy function approximation: a gradient boosting machine. *Ann. Statist.*, 29:1189-1232, 2001.

[13] N. Fuhr. Optimum polynomial retrieval functions based on probability ranking principle. *ACM Transactions on Information Systems*, 7:183-204, 1989.

[14] N. Fuhr and C. Buckley. A probabilistic learning approach for document indexing. *ACM Transactions on Information Systems*, 9:223-248, 1991.

[15] N. Fuhr and U. Pfeifer. Probabilistic information retrieval as a combination of abstraction, inductive learning, and probablistic assumptions. *ACM Transactions on Information Systems*, 12:92-115, 1994.

[16] J. Gao, H. Qi, X. Xia and J. Nie. Linear discriminant model for information retrieval. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 290–297, 2005.

[17] F. Gey, A. Chen, J. He and J. Meggs. Logistic regression at TREC4: probabilistic retrieval from full text document collections. *Proceedings of TREC*, 65-72, 1995.

[18] M. Hollander and D. A. Wolfe. *Nonparametric statistical methods.* Wiley-Interscience, 2nd edition, 1999.

[19] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of IR techniques. *ACM Transactions on Information Systems*, 20:422-446, 2002.

[20] T. Joachims. Optimizing search engines using clickthrough data. *Proceedings of the ACM Conference on Knowledge Discovery and Data Mining*, 2002.

[21] T. Joachims. Evaluating retrieval performance using clickthrough data. *Proceedings of the SIGIR Workshop on Mathematical/Formal Methods in Information Retrieval*, 2002.

[22] T. Joachims, L. Granka, B. Pang, H. Hembrooke, and G. Gay. Accurately Interpreting Clickthrough Data as Implicit Feedback. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2005.

[23] K. L. Kwok. A neural network for probablistic information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 21-30, 1989.

[24] X. Liu and W. B. Croft. Cluster-based retrieval using language models. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 186 – 193, 2004.

[25] R. Nallapati. Discriminative models for information retrieval. *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 64– 71, 2004.

[26] J. Nocedal and S. Wright. *Numerical Optimization.* Springer, 1999.

[27] J. Ponte and W. Croft. A language modeling approach to information retrieval. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval*, 1998.

[28] G. Salton. *Automatic Text Processing.* Addison Wesley, Reading, MA, 1989.

[29] H. Turtle and W. B. Croft. Inference networks for document retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, 1-24, 1990.

[30] G. Wahba. *Spline models for observational data.* SIAM press, 1990.

[31] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems*, 22, 2004.

[32] C. Zhai and J. Lafferty. A risk minimization framework for information retrieval , *Information Processing and Management*, 42:31–55, 2006.