Linear Feature-Based Models for Information Retrieval

Donald Metzler and W. Bruce Croft
University of Massachusetts, Amherst

Abstract. There have been a number of linear, feature-based models proposed by the information retrieval community recently. Although each model is presented differently, they all share a common underlying framework. In this paper we explore and discuss the theoretical issues of this framework, including a novel look at the parameter space. We then detail supervised training algorithms that directly maximize the evaluation metric under consideration, such as mean average precision. We present results that show training models in this way can lead to significantly better test set performance compared to other training methods that do not directly maximize the metric. Finally, we show that linear feature-based models can consistently and significantly outperform current state of the art retrieval models with the correct choice of features.

Keywords: Retrieval models, linear models, features, direct maximization

1. Introduction

Features lie at the very heart of information retrieval. The two features *term frequency* and *inverse document frequency* form the core of most modern retrieval models, including BM25 [29] and language modeling [27]. In most cases, the only difference between any two models is how these elementary features are combined. The resulting scoring functions are often non-linear and contain one or more free parameters that can be tuned in various ways.

Since most models are built upon some underlying theoretical framework, it is not always possible to easily incorporate new features. For example, it proved non-trivial to include query-independent features such as PageRank and inlink count into the BM25 ranking formula [6]. Therefore, a model that can handle arbitrary query-dependent and query-independent features is desirable.

In this paper we describe the theory behind a class of models we call *linear feature-based models*. As its name implies, the model's scoring function is comprised of a linear combination of features. One of the main benefits of such models is their ability to combine many different kinds of features in a straightforward manner. There have been many models proposed that fall under this general framework. The models include, but are not limited to, Gey's logistic regression model [9], Nallapati's discriminative model [24], Gao et. al.'s linear discriminate

model [8], and Metzler and Croft's dependence model [18]. More details of these models are given in Section 3.

The goal of this paper, therefore, is to bring together the ideas of these papers and present details of the general framework they all fall under. This includes investigating what scoring functions fit into the framework, the characteristics of the underlying parameter space, and how the model parameters can be effectively estimated. Empirical results are given that show that a linear feature-based model using simple bag of words features and trained using the methods discussed in this paper is capable of outperforming a state of the art retrieval model. Finally, results from a model using a more complex set of features based on term proximity information are presented that show consistent and significant improvements over a strong language modeling baseline, which shows the true power of the model.

The remainder of this paper proceeds as follows. In Section 2 the theoretical details of the framework are given, followed by a look at a range of related work in Section 3. Experimental results using both simple and more complex features are provided in Section 4. Finally, Section 5 concludes and summarizes the work presented in the paper.

2. Linear Feature-Based Models

This section provides a general description of linear feature-based models, discusses the parameter space, and describes supervised methods for estimating the model parameters.

2.1. Description

Suppose we are given a set of documents $\mathcal{D}$, queries $\mathcal{Q} = \{Q_i\}_{i=1}^{N}$, and training data $\mathcal{T}$. In addition, we are given a real-valued scoring function $S_\Lambda(D;Q)$ parameterized by Λ, a vector of parameters. Given a query Q_i, the scoring function $S_\Lambda(D;Q_i)$ is computed for each $D \in \mathcal{D}$ and documents are then ranked in descending order according to their score.

The scoring function induces a total ordering[1] (ranking) $R(\mathcal{D}, Q_i, S_\Lambda)$ on $\mathcal{D}$ for each query Q_i. For simplicity, we rewrite $R(\mathcal{D}, Q_i, S_\Lambda)$ as $R_i(\Lambda)$ and let $\mathcal{R}_\Lambda = \{R_i(\Lambda)\}_{i=1}^{N}$ be the set of rankings induced over all of the queries.

Finally, in order to evaluate a parameter setting, we need an evaluation function $E(\mathcal{R}_\Lambda;\mathcal{T})$ that produces real valued output given a set of ranked lists and the training data. It should be noted that we require E

[1] We assume ties are broken by document id.

to only consider the document *rankings* and not the document scores. The scores are only used to rank the documents and not used to evaluate the ranking. This is a standard characteristic among information retrieval evaluation metrics such as mean average precision, precision at 10, among others.

Therefore, our goal is to find the parameter setting Λ that maximizes the evaluation metric E over the parameter space. Formally, this can be stated as:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} E(\mathcal{R}_\Lambda; \mathcal{T}) \\ s.t. \quad & \mathcal{R}_\Lambda \sim S_\Lambda(D; Q) \\ & \Lambda \in M_\Lambda \end{aligned}$$

where $\mathcal{R}_\Lambda \sim S_\Lambda(D;Q)$ denotes that the orderings in $\mathcal{R}_\Lambda$ are induced using scoring function S, and M_Λ is the parameter space over Λ.

Rather than tackle the general optimization problem proposed above, we aim to solve a more constrained version. We restrict our focus to scoring functions from the following family:

$$\begin{aligned} \mathcal{S} = \{S_\Lambda(D;Q) \ : \ & \exists l(\cdot) \text{ s.t. } l \text{ is strictly monotonically increasing and} \\ & l(S_\Lambda(D;Q)) = \Lambda^T f(D,Q) + Z\} \end{aligned}$$

where $f(\cdot,\cdot)$ is a *feature function* that maps query/document pairs to real-valued vectors in $\mathbb{R}^d$, Z is a constant that does not depend on D (but may depend on Λ or Q). That is, we require there to exist some strictly monotonically increasing function l that, when applied to S, yields a function that is linear in our parameters Λ. The ranking functions in $\mathcal{S}$ define the universe of linear feature-based models.

Examples of functions within this family include linear discriminants, such as those used with perceptrons, Support Vector Machines (SVMs) [4], and the so-called maximum entropy distribution [26]. In addition, many information retrieval ranking functions proposed in the past, at their very core, also live within this family (e.g. [8, 9, 18, 24]).

By definition, every $S \in \mathcal{S}$ can can be reduced to a linear form via a strictly monotonically increasing function. Since such functions are rank preserving and subsequently evaluation metric preserving, we can always write the optimization problem for any scoring function in $\mathcal{S}$ as:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} E(\mathcal{R}_\Lambda; \mathcal{T}) \\ s.t. \quad & \mathcal{R}_\Lambda \sim \Lambda^T f(D,Q) + Z \\ & \Lambda \in M_\Lambda \end{aligned}$$

This general optimization problem fully describes how documents are ranked and how the parameters are estimated. We see that linear

feature-based models are instantiated by choosing an evaluation function E, training data $\mathcal{T}$, features f, and a parameter space M_Λ. We now describe the details and theory underlying each of these aspects.

2.2. Parameter Space

Thus far we have only talked abstractly about the parameter space M_Λ. There are many potential ways to choose a parameter space, each with its advantages and disadvantages. The most obvious choice, to not constrain the parameter space, occurs when $M_\Lambda = \mathbb{R}^d$. The advantage of this model is that the parameter weights can be either negative or positive. This allows the model to include features that convey both negative and positive evidence. On the downside, the search space is somewhat daunting, although not unmanageable.

Another option is to restrict the parameter values to be non-negative. Although this may seem too strict, there are several reasons why it is an acceptable assumption in most cases. In information retrieval, a majority of the features commonly used provide positive evidence. For example, large tf or idf values are evidence in support of relevance. If a feature is known *a priori* to provide negative evidence, then the feature can still be used, with its value negated. If we do not know if a feature provides positive or negative evidence and we 'guess' incorrectly, then the trained model will simply assign that feature a weight of 0. If this occurs, the feature value can be adjusted accordingly and the model retrained.

Therefore, the positivity constraint will typically have only a minor impact on the model. As we will now show, several nice results can be shown to hold under this assumption. Hence, for the remainder of this section, we assume that $M_\Lambda = \mathbb{R}^d_+ \stackrel{def}{=} \left\{\Lambda \in \mathbb{R}^d : \lambda_i \geq 0\right\}$.

2.2.1. *Reduction to Multinomial Manifold*

Only considering positive parameter values allows us to map our problem onto a more intuitively appealing space with several nice characteristics. We will now show that the parameter estimation problem previously described, under the positivity constraint, is equivalent to the following constrained optimization problem:

$$\begin{aligned} \hat{\Lambda} &= \arg\max_{\Lambda} E(\mathcal{R}_\Lambda; \mathcal{T}) \\ s.t. \quad & \mathcal{R}_\Lambda \sim \Lambda^T f(D, Q) + Z \\ & \Lambda \in \mathbb{P}^{d-1} \end{aligned}$$

where $\mathbb{P}^k$ is a multinomial manifold (also known as a k-simplex) described by:

$$\mathbb{P}^k \;=\; \left\{ \Theta \in \mathbb{R}^{k+1} : \forall j \;\theta_j \geq 0, \sum_{i=1}^{k+1} \theta_i = 1 \right\}$$

The multinomial manifold $\mathbb{P}^k$ can be intuitively thought of as the space of all multinomial distributions over $k+1$ potential outcomes. We now give proof of this equivalence.

Theorem. Any solution to the optimization problem over $\mathbb{R}_+^d$ has a rank-equivalent solution to the optimization over $\mathbb{P}^{d-1}$.

Proof. Suppose that $\hat{\Lambda} \in \mathbb{R}_+^d$ is the solution to the original optimization problem. Now, consider the following transformation of $\hat{\Lambda}$ to $\hat{\Theta}$:

$$\hat{\theta}_i \;=\; \frac{\hat{\lambda}_i}{W}$$

where $W = \sum_i \hat{\lambda}_i$. If $W = 0$, then λ_i is mapped to $\theta_i = \frac{1}{d}$ for all i, which is the uniform distribution.

It is easy to see that the transformed parameter setting $\hat{\Theta}$ is in $\mathbb{P}^{d-1}$, and thus the transformation maps the original point onto the manifold. We must now show this transformation preserves rank-equivalence. Under $\hat{\Theta}$, the scoring function becomes:

$$\begin{aligned} S_{\hat{\Theta}}(D;Q) &= \sum_i \hat{\theta}_i f(D,Q)_i + Z \\ &= \sum_i \left(\frac{\hat{\lambda}_i}{W} \right) f(D,Q)_i + Z \\ &= \frac{1}{W} \sum_i \hat{\lambda}_i f(D,Q)_i + Z \\ &\overset{rank}{=} \sum_i \hat{\lambda}_i f(D,Q)_i \end{aligned}$$

where the last step follow from the fact that scaling (by $\frac{1}{W}$) and translating (by Z) all scores in the same way has no effect on ranking. Thus, the model under the transformed parameter $\hat{\Theta}$ ranks documents exactly the same as using $\hat{\Lambda}$, the original parameter value. We have therefore shown that any solution to the problem over $\mathbb{R}_+^d$ can be transformed into a rank-equivalent solution over $\mathbb{P}^d$, thus completing the proof $\square$

It can be shown that this is a many-to-one, onto mapping. This suggests that our original parameter space is inefficient, in that many (in fact, infinitely many) points within the space produce the same

ranking. Within the new space, all of these redundant points are conflated to a single point. Although not explored here, this knowledge can potentially be used to implement more efficient or intelligent parameter estimation techniques.

2.2.2. *Rank Equivalence*
We now show that the reduction to the multinomial manifold has an interesting connection with the notion of rank equivalence. Given two parameter settings of a linear, feature-based model, Λ_1 and Λ_2, with a fixed set of features, we define the binary relation "$\sim$" as: $\Lambda_1 \sim \Lambda_2$ if and only if Λ_1 and Λ_2, under the model, are guaranteed to produce the same ranking for all queries. That is, "$\sim$" is the binary relation corresponding to rank equivalence between two parameter settings.

It is easy to see that "$\sim$" is an equivalence relation as it is reflexive ($\Lambda \sim \Lambda$), symmetric ($\Lambda_1 \sim \Lambda_2 \Rightarrow \Lambda_2 \sim \Lambda_1$), and transitive ($\Lambda_1 \sim \Lambda_2 \wedge \Lambda_2 \sim \Lambda_3 \Rightarrow \Lambda_1 \sim \Lambda_3$). It therefore induces equivalence classes over the original Euclidean parameter space. In fact, every parameter on the multinomial manifold corresponds to a *unique* equivalence class. Therefore, the set of parameters on the multinomial manifold can be thought of as *canonical* parameters that can be used to describe any possible parameter setting.

2.2.3. *Distance Between Models*
Our reduction provides another unique mechanism that is not available in most other retrieval models. The reduction allows us, for a fixed set of features, to quantitatively measure the *distance* between two models (i.e. two parameter settings). For the BM25 retrieval model, there is no straightforward way of measuring the distance between two parameter settings. In language modeling, there exists the notion of distance in terms of KL-divergence between two models [15], but it is not the same as the distance we can compute here. Instead, we can compute the distance between the actual scoring functions themselves.

Within this framework, how can we compute the distance between two parameter vectors which live in $\mathbb{R}^d$? The naive solution is to use the Euclidean distance between the two vectors. However, this leads to unappealing results. For example, consider the following two parameter vectors in $\mathbb{R}^2$:

$$\Lambda_1 = \begin{bmatrix} 1 \\ 3 \end{bmatrix} \ , \ \Lambda_2 = \begin{bmatrix} 2 \\ 6 \end{bmatrix}$$

The Euclidean distance between these two vectors is $\sqrt{10}$, despite the fact that the two parameter settings produce precisely the same ranking. Therefore, the intuitive distance between these two parameter

settings is 0. Now, suppose we apply the mapping to the multinomial manifold ($\mathbb{P}^1$), that was defined above, to these two points. The mapped points can be shown to be:

$$\Theta_1 = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix} \ , \ \Theta_2 = \begin{bmatrix} 0.25 \\ 0.75 \end{bmatrix}$$

Both of the original parameters are mapped to the same point in $\mathbb{P}^1$, and thus they have no distance between them. Now that it is clear the manifold better captures the *intrinsic* properties of the parameters, we still have to answer the question of how to properly measure the distance between arbitrary points on the manifold. Results from information geometry tell us that the multinomial manifold follows a non-Euclidean geometry, and therefore the Euclidean distance does not hold. Thus, we must use a more appropriate distance metric, known as the geodesic distance, which is a generalization of the Euclidean distance to non-Euclidean geometries. The geodesic distance between two points in $\mathbb{P}^d$ is computed as:

$$d(\Theta, \Theta') = 2 \arccos \left(\sum_{i}^{d+1} \sqrt{\theta_i \theta'_i} \right)$$

where $d(\cdot, \cdot)$ ranges from 0 to π. Although we do not explore specific applications of this distance in this paper, a great deal of work considering various properties of the multinomial manifold exists [16, 34]. Future applications may find uses for this unique property.

We also note that the cosine distance is an equally valid measure of the distance between two linear feature-based parameter settings. However, it is less theoretically motivated in terms of the underlying intrinsic geometry of the parameter space.

Finally, it is important to state again that the properties discussed in this section are only valid if the parameters are constrained to be non-negative. We feel the theoretical benefits gained by imposing the constraint outweigh any potential disadvantages.

2.3. Parameter Estimation

Ranking large sets of documents is inherently important in information retrieval. Most state of the art machine learning approaches cannot efficiently handle the large data sets used, the highly unbalanced nature of the training data, or the non-standard evaluation metrics. Since the end goal of most systems is to maximize some evaluation metric based on the rankings produced, we focus on approaches that directly maximize the evaluation metric by solving the optimization presented in Section 2.

Several models proposed in the past have performed parameter estimation via maximization of likelihood or maximization of margin. However, these approaches are inherently maximizing the incorrect metric. Information retrieval is typically not concerned with likelihood, nor classification accuracy. Instead it is entirely concerned with how well a model ranks documents. It can be argued that estimating parameters by maximizing the likelihood of some training data or minimizing classification error is optimizing a function that is correlated with the underlying retrieval metric, such as mean average precision. However, this has been shown experimentally to be invalid and it can also be shown theoretically to be invalid, as well. This is known as metric divergence [22].

The remainder of this section describes approaches for directly maximizing evaluation metrics. These techniques can easily handle large training sets, such as those that typically arise in information retrieval. They can also handle the highly unbalanced nature of the training data. Although these techniques have nice properties, they also have several pitfalls, as we will show.

2.3.1. *Grid Search*

The most naive approach to solving the optimization problem is to perform an exhaustive grid search over the parameter space. That is, we place a grid over the parameter space and evaluate $E(\mathcal{R}_\Lambda; \mathcal{T})$ at every grid intersection, outputting the parameter setting that yields the maximum at the end.

A grid search over $\mathbb{R}^d$ is unbounded and ill-defined. For this reason, we restrict our discussion to the case where our parameter space is the multinomial manifold. For this case, the grid search is bounded and can be easily implemented.

Given a parameter $\epsilon = \frac{1}{K}$ for $K \in \mathbb{Z}^+$ that controls how fine grained our grid is, we define:

$$\begin{aligned} \mathcal{G} &= \left\{ \Lambda = (k_1\epsilon \ldots k_d\epsilon) : \sum_i k_i\epsilon = 1, k_i \in \mathbb{N} \right\} \\ &= \left\{ \Lambda = (k_1\epsilon \ldots k_d\epsilon) : \sum_i k_i = K, k_i \in \mathbb{N} \right\} \end{aligned}$$

As we see, $|\mathcal{G}|$, the number of parameter values we must evaluate E at, depends both on d (the number of parameters) and K (how fine grained our grid is). A grid search is feasible only if both d and K are relatively small. For larger values we must turn to more sophisticated training methods. However, we should note that the grid search method has the nice property that it is guaranteed to find a global maximum

as K gets large. This allows exact global convergence to be traded off for faster training time.

2.3.2. *Coordinate Ascent*

Coordinate ascent is a commonly used optimization technique for unconstrained optimization problems. The algorithm iteratively optimizes a multivariate objective function by solving a series of one dimensional searches. It repeatedly cycles through each parameter, holding all other parameters fixed, and optimizes over the free parameter. The technique is known to converge slowly on objective functions with long ridges. Variations of the method, including Powell's method, have been proposed to overcome this issue [28].

Coordinate ascent can be applied to the optimization problem under consideration regardless of whether we choose to optimize in the original Euclidean parameter space ($\mathbb{R}^d$) or the mapped multinomial parameter space ($\mathbb{P}^{d-1}$). Optimizing over the manifold may be beneficial due to the reduction in the number of repeated local extrema. This may be particularly useful when estimating gradients using finite differences while performing coordinate ascent over a non-differentiable metric surface.

If coordinate ascent is performed over the multinomial manifold, then only a minor modification to the original algorithm is necessary. All one dimensional searches done by the algorithm will be performed as if they were being done in $\mathbb{R}^d$. However, this does not ensure that the updated parameter estimate will be a point on the manifold. Therefore, after a step is taken in $\mathbb{R}^d$, we project the point back onto the manifold, which we showed is always possible. Note that this projection preserves the function value since the unnormalized and projected parameter estimates lead to equivalent rankings. Therefore, the optimization is implicitly being done in a space that we know how to optimize over ($\mathbb{R}^d$), but is continually being projected back onto to the manifold.

More concretely, suppose that λ_i is the current free parameter and all other parameters are held fixed. Then, the update rule is given by:

$$\lambda_i' \;=\; \arg\max_{\lambda_i} E(\mathcal{R}_\Lambda; \mathcal{T})$$

After λ_i' is updated, the entire parameter vector is then projected back onto the manifold. This process is performed iteratively over all parameters until some convergence criteria is met.

If the parameter space is not constrained to be non-negative, then search via coordinate ascent is still possible. However, the entire search must be done in $\mathbb{R}^d$. The algorithm is applied exactly as described, except the parameter setting no longer has to be projected.

Finally, we note that if E is partially differentiable with respect to each parameter then the update rule is straightforward. For those functions where E is not partially differentiable, such as the ones considered in the remainder of this paper, a line search must be done to find the arg max.

2.3.3. *Other Methods*

There also exist a number of other training techniques that attempt to directly maximize metrics related to information retrieval. One method proposed to optimize for mean average precision uses a perceptron-based algorithm [8]. This approach, like the coordinate ascent approach, is not guaranteed to find a global maxima. Instead, it can be shown to be optimizing a lower bound on mean average precision.

Another approach, based on neural networks, called RankNet, has recently been proposed [3]. A RankNet is trained using gradient descent over a differentiable cost function. This allows gradients to be computed easily. However, the model suffers from standard neural network training issues, such as local minima. In addition, the cost function is general and does not correspond to any specific information retrieval metric. Therefore, it is not clear how to train the model to maximize different metrics. Recent work has looked at addressing some of the issues involved in training a RankNet [17].

Finally, Joachims has developed large margin training techniques for multivariate performance measures [12]. Using these techniques it is possible to directly maximize a variety of information retrieval metrics, such as precision at k, precision-recall breakeven, and area under the ROC curve. In fact, any metric that can be computed based solely on a contingency table can be efficiently maximized. However, metrics such as mean average precision, which cannot be computed using a contingency table, cannot be used.

The downside of most of these approaches is that they specifically work for one metric or a family of metrics, and not for arbitrary metrics. The grid search and coordinate ascent algorithms, however, do not have this problem.

There currently is no well developed understanding of best practice training techniques for linear feature-based information retrieval models. Most studies in the area have looked at traditional machine learning problems, which typically differ from information retrieval tasks. Therefore, an interesting, and necessary, direction of future work is to undertake a comprehensive evaluation of these techniques, in terms of how effective they are across a wide range of retrieval data sets and metrics, how well they generalize, and how efficient they are.

2.3.4. *Discussion*

Finding the maximum of an arbitrary evaluation function E can be very difficult, especially in high-dimensional space. Only a grid search method, with a suitably chosen granularity, is guaranteed to find a global maxima. Coordinate ascent is a local search technique that is only guaranteed to find a global maxima if the evaluation function E is concave. Our experiments using this approach, as will be discussed in Section 4, show that, for a certain set of term and phrase features, mean average precision is approximately concave over a wide range of collections. This may be the case for many related applications and feature sets, but is not true in general, as was pointed out in [8]. For functions with many local maxima, a multiple random restart strategy can be used to increase the chances of finding a global solution. Throughout the remainder of this work, all optimization is carried out using coordinate ascent with 10 random restarts.

2.4. Training Data

We have thus far glossed over the form of $\mathcal{T}$, the training data. The general framework allows $\mathcal{T}$ to be any type of data that can be used to compute the evaluation metric E over a set of ranked lists. For example, this data may come in the form of TREC relevance judgments or web clickthrough data [11, 13]. To estimate the parameters we only need to evaluate E, so models may even use abstract concepts in place of $\mathcal{T}$, such as novelty [10] or aspect precision/recall [32], as long as E remains independent of the document scores.

One disadvantage of the models presented here is that they require some form of training data to get a parameter estimate. If little or no training data exists, then unsupervised and active learning [30] techniques from machine learning can potentially be employed. However, such methods are out of the scope of the current work.

2.5. Features

In this section we take a brief look at features. Although not the focus of this paper, we feel it is important to provide a sketch of the types of features that may be useful in a linear feature-based model.

After decades of research there are a surprisingly small number of features that have proven to be useful as the basis of retrieval models. The following are the most commonly used features:

- **Term occurrence / non-occurrence** – whether or not a term occurs within a document

- **Term frequency** – the number of times a term occurs within a document
- **Inverse document frequency** – inverse of the proportion of documents that contain a given term
- **Document length** – number terms within the document
- **Term proximity** – occurrence patterns of terms within a document

Almost every major retrieval model that has been developed has been based on one or more of these features. We refer to these as *primitive textual features*. They can be thought of as building blocks that can be used to construct more complex ranking functions.

In addition, there are *high level textual features* that are combinations of the primitive textual feature. Examples of these features are the term weighting functions used by BM25, language modeling, and most other popular retrieval models.

Some feature functions may have hyperparameters. For example, the BM25 model is typically parameterized to have two hyperparameters, k_1 and b. In linear feature-based models, hyperparameters are not part of the model, but of the features themselves. Therefore, hyperparameters must either be fixed before training or adapted in some way during training.

Finally, there exist *non-textual features*. These features are typically task-specific and often exploit knowledge about the domain or problem structure. Examples of these features are PageRank [2], URL depth [14], document quality [35], readability [31], sentiment [25], and query clarity [7].

Of course, the list of features presented here is by no means complete. As new and useful features are discovered, linear feature-based models provide a simple, convenient framework for combining them.

3. Related Work

Many models proposed in the past, at their very core, are linear feature-based models [8, 9, 18, 24]. The models typically differ in their formulation, features, or training. This section briefly summarizes several of these models.

In 1994, Gey proposed a logistic regression model for information retrieval [9]. In terms of our discussion above, the scoring function is in $\mathcal{S}$ after application of the rank-preserving logit transformation and thus

Table I. Features used in the bag of words experiments. $tf_{w,D}$ is the number of times term w occurs in document D, cf_w is the number of times term w occurs in the entire collection, df_w is the number of documents term w occurs in, $|D|$ is the length (in terms) of document D, $|C|$ is the length (in terms) of the collection, and N is the number of documents in the collection.

	Feature		Feature
1	$\sum_{w \in Q \cap D} \log(tf_{w,D})$	4	$\sum_{w \in Q \cap D} \log(\frac{\|C\|}{cf_w})$
2	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|})$	5	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|} \frac{N}{df_w})$
3	$\sum_{w \in Q \cap D} \log(\frac{N}{df_w})$	6	$\sum_{w \in Q \cap D} \log(1 + \frac{tf_{w,D}}{\|D\|} \frac{\|C\|}{cf_w})$

is a linear feature-based model. In the model, six features were used. The features were query absolute frequency, query relative frequency, document absolute frequency, document relative frequency, *idf*, and relative frequency in all documents. The maximum likelihood estimate was used for the parameters. Results showed mixed improvements over a vector space baseline when trained on one collection and tested on another.

In [24], Nallapati argued for a discriminative model for information retrieval, focusing in particular on a SVM formulation. Like Gey, Nallapati also made use of six features. Table I shows the six features considered. In this case, the parameter vector is estimated by training a linear SVM, with relevant documents considered the "positive class" and non-relevant documents the "negative class". Therefore, the ranking task is treated as a classification problem. Results were mixed when compared against a language modeling baseline.

When training using maximum likelihood or SVMs, it is often important to have balanced training data. However, in information retrieval it is very often the case that there are many more relevant documents compared to non-relevant documents for a given query. For this reason, the training data is very unbalanced. Nallapati found that the data needed to be balanced in order to achieve good generalization performance. Balancing was done by undersampling the majority (non-relevant) class. Although this led to improved performance over the unbalanced case, it had the negative effect of throwing away valuable training data. Other solutions to the unbalanced data problem for SVMs exist that do not require training data to be compromised, such as allowing separate costs for training errors in the positive and negative classes [23]. We note that the coordinate ascent method discussed in

this paper does not suffer from this problem and can use the training data in its entirety.

Nearly simultaneously, Gao, et. al. [8] and Metzler and Croft [18] described ranking functions that are linear feature-based models similar in spirit to those of Gey and Nallapati, but trained their models by directly maximizing mean average precision. Although the two models are presented differently, at their very core they are very similar. The main difference being the features used. Both models went beyond the simplistic bag of words features used by Gey and Nallapati and used powerful phrase-based features which led to significant improvements in effectiveness over baseline systems. Experimental results and details of the features used in [18] are given in the next section. The success of these models, compared to the mixed results of Gey and Nallapati, suggest the importance of proper training and feature selection.

4. Experiments

This section describes experiments carried out using the framework described in this work on a number of *ad hoc* retrieval experiments. These particular experiments were chosen to empirically show that training by directly maximizing mean average precision is superior to a SVM trained model and that combining richer features within a linear feature-based model can consistently and significantly outperform traditional retrieval models.

The goal of *ad hoc* retrieval is to retrieve as many relevant documents as high in the ranked list as possible. Relevance is binary (relevant/non-relevant) and is assessed according to whether the document is topically related to the query or not. In all of our experiments, we make use of TREC data, which consists of a set of topics (queries) and a set of human relevance judgments for each topic. Given a ranking of documents in response to a query, we compute the average precision for each query to evaluate our ranking [1]. As is typically done, we compute the mean of the average precisions over all of the queries (mean average precision) and use this as our primary evaluation metric.

4.1. Bag of Words Features

We first compare a linear feature-based model trained by directly maximizing mean average precision, Nallapati's SVM model [24], and a language modeling system [27]. Three standard newswire TREC collections are used. For each collection, 50 topics (queries) are used for training and 50 for testing. Only the title portion of the TREC topics

Table II. Summary of TREC collections used in bag of words experiments.

	Disks 1,2	Disk 3	Disks 4,5
Num. Docs	741,856	336,310	556,077
Training topics	101-150	51-100	301-350
Test topics	151-200	101-150	401-450

are used. All documents are stemmed using Krovetz Stemmer and stopped using a standard list of common terms. A summary of the collections used and the training and test topics are given in Table II.

For these experiments, E = mean average precision, $\mathcal{T}$ = TREC relevance judgments, f = bag of words features (see Table I), and $M_\Lambda = \mathbb{R}^d$. Each model is trained using only the data in the relevance judgments. That is, when the model is being trained it only "knows" about the documents contained in the relevance judgments and not about any of the unjudged documents in the collection. However, when being tested, *all* documents in the collection are ranked. In the case of the balanced SVM model, the non-relevant judgments from the relevance file were undersampled. The feature-based model is trained using the same features as the SVM, and therefore has no additional power. The language modeling system ranks documents via query likelihood, with document models estimated using Bayesian (Dirichlet) smoothing [33] and is trained by finding the smoothing parameter that maximizes the mean average precision on the training data.

The results of the experiments are given in Table III. As we see from the results, our parameter estimation technique based on coordinate ascent consistently leads to statistically significant improvements over the SVM estimates. Furthermore, it significantly outperforms language modeling on 4 out of 6 runs. Language modeling, on the other hand, significantly outperforms the SVM model on 4 out of the 6 runs.

The results indicate that language modeling, despite its simplicity, stands up very well compared to sophisticated feature-based machine learning techniques. The results also provide empirical proof that SVM parameter estimation is simply not the correct paradigm here, mainly because it is optimizing the wrong objective function. Our estimation technique, however, is directly maximizing the evaluation metric under consideration and results in stable, effective parameter estimates across the collections.

When it comes to implementation, our method is considerably easier to implement than SVMs, but more complex than language modeling.

Table III. Training and test set mean average precision values for various *ad hoc* retrieval data sets and training methods. The † represents a statistically significant improvement over language modeling and ‡ denotes significant improvement over the balanced SVM model. Tests done using a one tailed paired t-test at the 95% confidence level.

	Disks 1,2		Disk 3		Disks 4,5	
	Train	Test	Train	Test	Train	Test
Unbalanced SVM	0.0955	0.1091	0.1501	0.1336	0.1421	0.1434
Balanced SVM	0.1577	0.1849	0.1615	0.1361	0.1671	0.1897
Language modeling	0.1883‡	0.2155‡	0.1875‡	0.1642‡	0.1819	0.1995
Feature Based	0.1955‡	0.2327†‡	0.2080†‡	0.1773‡	0.2238†‡	0.2328†‡

For this reason, a number of issues should be considered before choosing a retrieval model. We feel that for this simple case, using simple term statistic features, language modeling (or BM25) is very likely the best practical choice. The real power of feature-based methods comes when more complex features, such as those that bag of words models fail to handle directly, are used. A good example of the power of feature-based models is given in the next section, where we consider such features.

4.2. Term Proximity Features

In this section we consider a richer set of features beyond the ones considered in the previous section. Here, we investigate features that make use of term proximity information. We choose this set of features because there has been recent evidence that using term proximity information can provide significant improvements in effectiveness over models that only consider terms to be a simple bag of words [8, 21]. We present results for *ad hoc* retrieval experiments carried out on two medium sized newswire collections and two large TREC web collections. Table IV provides a summary of the data sets. For this set of experiments, E = mean average precision, $\mathcal{T}$ = TREC relevance judgments, f = term proximity features, and $M_\Lambda = \mathbb{P}^{d-1}$.

We consider an extremely simple set of features that account for different kinds of proximity between terms within the query. Table V explains the three features used. As we see, there are three features —

Table IV. Overview of TREC collections and topics used in term proximity experiments.

Name	Description	Size	# Docs	Topics
WSJ	Wall St. Journal '87–'92	510 MB	173,252	50–200
AP	Associated Press '88–'90	730 MB	242,918	50–200
WT10g	TREC Web collection	11 GB	1,692,096	451–550
GOV2	2004 crawl of .gov domain	427 GB	25,205,179	701–750

Table V. Features used in the term proximity retrieval experiments. The term sum is over every query term, the ordered phrase sum is over every contiguous subset of query terms of length two or more within the query, and the unordered phrase sum is over every subset of two or more query terms. $tf_{\#1}$ is the count of the number of times the expression occurs as an exact phrase within D, and $tf_{\#uw8}$ is the count of the number of times the terms within the expression appear ordered or unordered within a window of length 8 within D. In addition, α_d is a hyperparameter that is fixed *a priori*. See [18] for more details.

Type	Feature
Term	$\sum_{q_i} \log \left[(1-\alpha_d) \frac{tf_{q_i,D}}{\|D\|} + \alpha_d \frac{cf_{q_i}}{\|C\|} \right]$
Ordered Phrase	$\sum_{q_i,\ldots,q_{i+k}} \log \left[(1-\alpha_d) \frac{tf_{\#1(q_i \ldots q_{i+k}),D}}{\|D\|} + \alpha_d \frac{cf_{\#1(q_i \ldots q_{i+k})}}{\|C\|} \right]$
Unordered Phrase	$\sum_{q_i,\ldots,q_j} \log \left[(1-\alpha_d) \frac{tf_{\#uw8(q_i \ldots q_j),D}}{\|D\|} + \alpha_d \frac{cf_{\#uw8(q_i \ldots q_j)}}{\|C\|} \right]$

a single term feature, an exact phrase feature, and an unordered phrase feature. These features are meant to capture the fact that query term order provides important information. For example, the queries "white house rose garden" and "white rose house garden" seek completely different pieces of information, yet are viewed as the same query in the bag of words representation. The features also attempt to capture the fact that most web queries are made up of one or more implicit phrases, such as "white house" and "rose garden" in our example. Since we consider all subphrases, we are likely to pick up on such phrases and retrieve more relevant documents.

In our first experiment, we find the optimal parameters for our model and compare against a simple bag of words model (language modeling) baseline. As for the bag of words experiments, the language model is

Table VI. Mean average precision using optimal parameter settings for each model. Values in parenthesis denote percentage improvement over language modeling. The † symbol indicates an improvement that is statistical significance ($p < 0.05$ with a one-tailed paired t-test).

	Language Modeling	Feature-Based
AP	0.1775	0.1866† (+5.1%)
WSJ	0.2592	0.2738† (+5.6%)
WT10g	0.2032	0.2231† (+9.8%)
GOV2	0.2502	0.2844† (+13.7%)

trained by setting the smoothing parameter to the value that maximizes mean average precision.

The results of the experiments are given in Table VI. As the results show, the models learned achieve statistically significant improvements over the baseline language modeling system. These results provide evidence that both term proximity and the training method developed in this work can be leveraged to significantly improve effectiveness. As further evidence of the power of both term proximity for web retrieval and the proposed training method, a model similarly trained achieved the best title-only run at the 2004 and 2005 TREC Terabyte Tracks [5, 19, 20].

Another important aspect of these models is how well a set of parameters estimated on one collection generalizes to other collections. As we saw in the previous section, the feature-based model generalized well on the same collection across different query sets. However, this does not mean it will generalize well when tested on a different test collection. To test this, we trained and tested our model on every possible combination of collections. This allows us to see, for example, how a model trained on a newswire collection generalizes to a web collection, and vice versa. The results are reported in Table VII.

The results show that the parameters generalize well across collections. Somewhat surprisingly, parameters estimated on newswire collections generalize sufficiently well to large web collections. The converse is true, as well. These results show that the model parameters are very stable across collections and that the model, using these features, does not seem to suffer from overfitting.

Table VII. Mean average precision using term proximity features in feature-based model for various training/test set splits.

Train \ Test	AP	WSJ	WT10g	GOV2
AP	0.1866	0.2716	0.2226	0.2839
WSJ	0.1841	0.2738	0.2195	0.2694
WT10g	0.1865	0.2719	0.2231	0.2783
GOV2	0.1852	0.2709	0.2201	0.2844

Finally, Figure 1 illustrates the well behaved, nearly concave surfaces that arise by imposing the mean average precision metric over the multinomial simplex of ranking function parameters for each of the collections used in this section using term proximity features. Each of the surfaces has the same general form, indicating that the features capture an inherent property that persists across different types of collections. Although there is no guarantee that such a nicely concave surface will exist for all features and all evaluation metrics, it provides some evidence that the functions we are maximizing over the simplex are not too difficult to optimize using simple algorithms, such as coordinate ascent.

5. Conclusions

This paper presented a detailed overview of linear feature-based models. Although such models have been looked at many times in the past, there has been no single unifying investigation of the theory at the heart of the framework. Here, we have presented a general description of the theory underlying the models, in terms of the family of ranking functions $\mathcal{S}$ and showed that the effective parameter space of such models, under a mild assumption, is the multinomial manifold. We also described a novel method for computing the distance between two models (parameter settings) that is not possible in most, if not all, of the commonly used retrieval models.

Furthermore, we described methods for estimating the model parameters by directly maximizing the evaluation metric under consideration, such as mean average precision. It was argued, as well as shown empirically, that this method for estimating the model parameters is more effective than methods proposed in the past, such as maximum

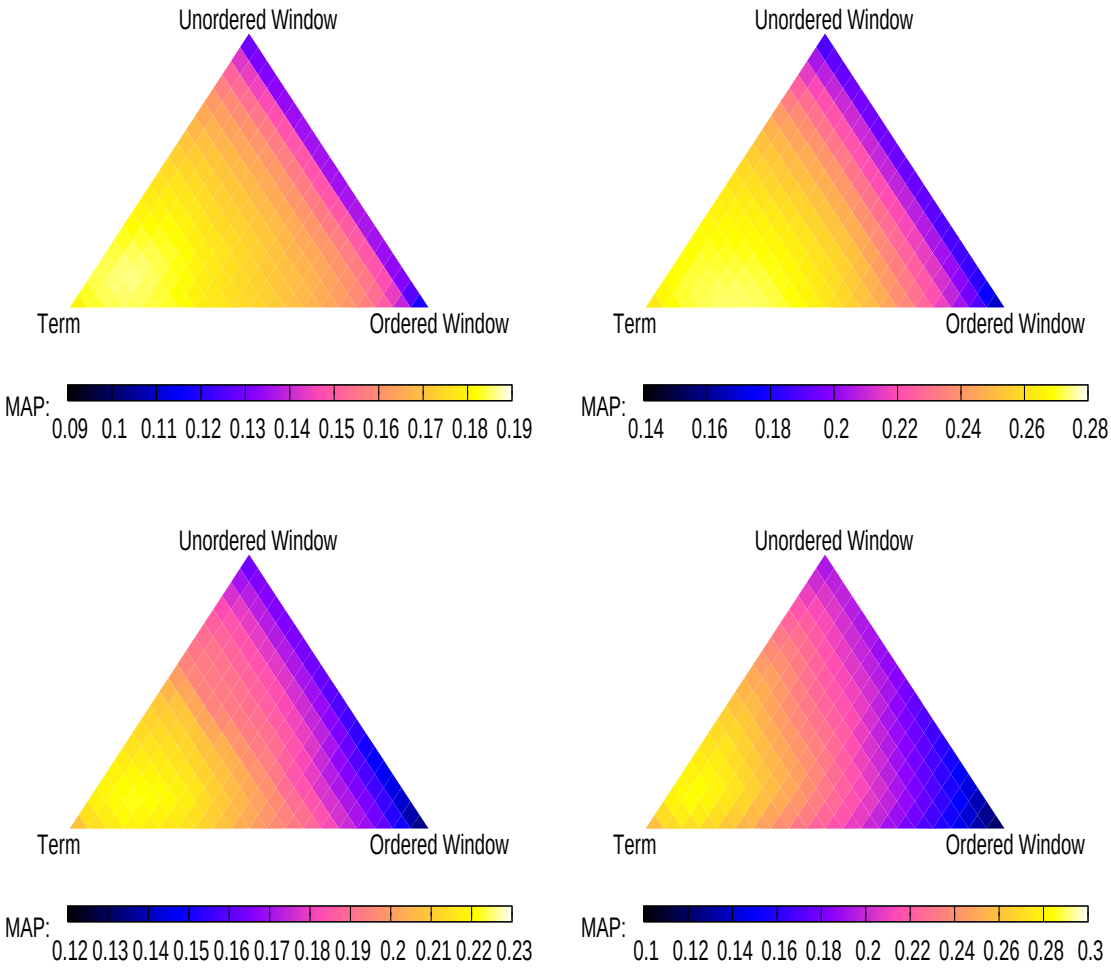

Figure 1. Mean average precision values plotted over multinomial parameter simplex for AP, WSJ, WT10g, and GOV2 collections using term proximity features (left to right, top to bottom). Intensity corresponds to mean average precision, as indicated by the legend.

likelihood or maximum margin estimation. Directly maximizing the evaluation metric results in maximizing the correct objective function and avoids the problem of metric divergence.

Finally, we showed empirically that the performance of linear feature-based model using a simple set of features was capable of outperforming a strong language modeling baseline. This demonstrates the practicality of the model. Furthermore, experiments were carried out on a model

that was constructed using more complex term proximity features. This model was shown to significantly outperform the language modeling baseline on a wide range of test collections. These results bring to light the true power of the model in in terms of effectiveness and suggest that combining even richer sets of features can lead to more improvements in retrieval effectiveness.

Therefore, linear feature-based models, when formulated using a rich set of features and trained appropriately, can achieve better than state of the art performance. This, combined with straightforward implementation, makes such models an attractive choice.

Acknowledgments

This work was supported in part by the Center for Intelligent Information Retrieval, in part by NSF grant #CNS-0454018, in part by NSF grant #IIS-0527159, and in part by Advanced Research and Development Activity and NSF grant #CCF-0205575. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

References

1. Baeza-Yates, R. and G. Navarro: 1999, *Modern Information Retrieval*. Addison-Wesley.
2. Brin, S. and L. Page: 1998, 'The anatomy of a large-scale hypertextual Web search engine'. *Computer Networks and ISDN Systems* **30**(1–7), 107–117.
3. Burges, C., T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender: 2005, 'Learning to rank using gradient descent'. In: *ICML '05: Proceedings of the 22nd international conference on Machine learning*. pp. 89–96.
4. Burges, C. J. C.: 1998, 'A Tutorial on Support Vector Machines for Pattern Recognition'. *Data Mining and Knowledge Discovery* **2**(2), 121–167.
5. Clarke, C., N. Craswell, and I. Soboroff: 2004, 'Overview of the TREC 2004 Terabyte Track'. In: *Online proceedings of the 2004 Text Retrieval Conf.*
6. Craswell, N., S. Robertson, H. Zaragoza, and M. Taylor: 2005, 'Relevance Weighting for Query Independent Evidence'. In: *Proceedings of the 28th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 416–423.
7. Cronen-Townsend, S., Y. Zhou, and W. B. Croft: 2002, 'Predicting query performance'. In: *SIGIR '02: Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. pp. 299–306.
8. Gao, J., H. Qi, X. Xia, and J.-Y. Nie: 2005, 'Linear discriminant model for Information Retrieval'. In: *Proceedings of the 28th Annual International ACM*

SIGIR Conf. on Research and Development in Information Retrieval. pp. 290–297.
9. Gey, F.: 1994, 'Inferring probability of relevance using the method of logistic regression'. In: *Proceedings of the 17th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*.
10. Harman, D.: 2004, 'Overview of the TREC 2002 Novelty Track'. In: *Proceedings of the 2002 Text Retrieval Conf.*
11. Joachims, T.: 2002, 'Optimizing search engines using clickthrough Data'. In: *Proceedings of the eighth ACM SIGKDD International Conf. on Knowledge discovery and Data Mining*. pp. 133–142.
12. Joachims, T.: 2005, 'A Support Vector Method for Multivariate Performance Measures'. In: *Proceedings of the International Conference on Machine Learning*. pp. 377–384.
13. Joachims, T., L. Granka, B. Pan, H. Hembrooke, and G. Gay: 2005, 'Accurately interpreting clickthrough Data as implicit feedback'. In: *Proceedings of the 28th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 154–161.
14. Kraaij, W., T. Westerveld, and D. Hiemstra: 2002, 'The Importance of Prior Probabilities for Entry Page Search'. In: *Proceedings of SIGIR 2002*. pp. 27–34.
15. Lafferty, J. and C. Zhai: 2001, 'Document language models, query models, and risk minimization for Information Retrieval'. In: *Proceedings of the 24th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 111–119.
16. Lebanon, G. and J. Lafferty: 2004, 'Hyperplane margin classifiers on the multinomial manifold'. In: *Proceedings of the twenty-first International Conf. on Machine learning*. pp. 66–71.
17. Matveeva, I., C. Burges, T. Burkard, A. Laucius, and L. Wong: 2006, 'High accuracy retrieval with multiple nested ranker'. In: *Proceedings of the 29th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 437–444.
18. Metzler, D. and W. B. Croft: 2005, 'A Markov Random Field Model for Term Dependencies'. In: *Proceedings of the 28th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 472–479.
19. Metzler, D., T. Strohman, H. Turtle, and W. B. Croft: 2004, 'Indri at Terabyte Track 2004'. In: *Online proceedings of the 2004 Text Retrieval Conf.*
20. Metzler, D., T. Strohman, Y. Zhou, and W. B. Croft: 2005, 'Indri at Terabyte Track 2005'. In: *Online proceedings of the 2005 Text Retrieval Conf.*
21. Mishne, G. and M. de Rijke: 2005, 'Boosting Web Retrieval through Query Operators'. In: *Proceedings of the 27th European Conf. on Information Retrieval*.
22. Morgan, W., W. Greiff, and J. Henderson: 2004, 'Direct Maximization of Average Precision by Hill-Climbing with a Comparison to a Maximum Entropy Approach'. Technical report, MITRE.
23. Morik, K., P. Brockhausen, and T. Joachims: 1999, 'Combining statistical learning with a knowledge-based approach - A case study in intensive care monitoring'. In: *Proceedings of the 16th International Conf. on Machine Learning*.
24. Nallapati, R.: 2004, 'Discriminative models for Information Retrieval.'. In: *Proceedings of the 27th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 64–71.

25. Pang, B., L. Lee, and S. Vaithyanathan: 2002, 'Thumbs up? Sentiment Classification using Machine Learning Techniques'. In: *Proceedings of the 2002 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. pp. 79–86.
26. Pietra, S. D., V. D. Pietra, and J. Lafferty: 1997, 'Inducing Features of Random Fields'. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **19**(4), 380–393.
27. Ponte, J. M. and W. B. Croft: 1998, 'A Language Modeling Approach to Information Retrieval'. In: *Proceedings of the 21st Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 275–281.
28. Press, W. H., S. A. Teukolsky, W. T. Vetterling, and B. P. Flannery: 1992, *Numerical Recipes in C: The Art of Scientific Computing*. Cambridge University Press.
29. Robertson, S., S. Walker, M. M. Beaulieu, and M. Gatford: 1995, 'Okapi at TREC-4'. In: *Online proceedings of the Fourth Text Retrieval Conf.* pp. 73–96.
30. Shen, X. and C. Zhai: 2005, 'Active feedback in ad hoc Information Retrieval'. In: *Proceedings of the 28th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 59–66.
31. Si, L. and J. Callan: 2001, 'A statistical model for scientific readability'. In: *CIKM '01: Proceedings of the tenth international conference on Information and knowledge management*. pp. 574–576.
32. Zhai, C.: 2002, 'Risk Minimization and Language Modeling in Information Retrieval'. Ph.D. thesis, Carnegie Mellon University.
33. Zhai, C. and J. Lafferty: 2001, 'A study of smoothing methods for language models applied to Ad-Hoc Information Retrieval'. In: *Proceedings of the 24th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 334–342.
34. Zhang, D., X. Chen, and W. S. Lee: 2005, 'Text classification with kernels on the multinomial manifold'. In: *Proceedings of the 28th Annual International ACM SIGIR Conf. on Research and Development in Information Retrieval*. pp. 266–273.
35. Zhou, Y. and W. B. Croft: 2005, 'Document quality models for web ad hoc retrieval'. In: *CIKM '05: Proceedings of the 14th ACM international conference on Information and knowledge management*. pp. 331–332.