Linear Discriminant Model for Information Retrieval[1]

Jianfeng Gao[*], Haoliang Qi[$], Xinsong Xia[#], Jian-Yun Nie[**]

[*]Microsoft Research, Asia, Email: jfgao@microsoft.com; [$]Harbin Institute of Technology, China;
[#]Peking University, China; [**]Université de Montréal, Email: nie@iro.umontreal.ca

ABSTRACT

This paper presents a new discriminative model for information retrieval (IR), referred to as *linear discriminant model* (LDM), which provides a flexible framework to incorporate arbitrary features. LDM is different from most existing models in that it takes into account a variety of linguistic features that are derived from the component models of HMM that is widely used in language modeling approaches to IR. Therefore, LDM is a means of melding discriminative and generative models for IR. We present two algorithms of parameter learning for LDM. One is to optimize the average precision (AP) directly using an iterative procedure. The other is a perceptron-based algorithm that minimizes the number of discordant document-pairs in a rank list. The effectiveness of our approach has been evaluated on the task of ad hoc retrieval using six English and Chinese TREC test sets. Results show that (1) in most test sets, LDM significantly outperforms the state-of-the-art language modeling approaches and the classical probabilistic retrieval model; (2) it is more appropriate to train LDM using a measure of AP rather than likelihood if the IR system is graded on AP; and (3) linguistic features (e.g. phrases and dependences) are effective for IR if they are incorporated properly.

Categories and Subject Descriptors

H.3.3 [**Information Storage and Retrieval**]: *Retrieval models*

General Terms

Design, Algorithms, Theory, Experimentation

Keywords

Language Model, Discriminative Training, Hidden Markov Model, Perceptron, Optimization

1. Introduction

Language modeling (LM) approaches to information retrieval (IR) assume that the relevance of a document, given a query, can be estimated as the generative probability of the query from the document [23]. One of the most appealing properties of this approach is its ability to incorporate linguistic information of language such as phrase and dependences into the retrieval model in a systematic manner. For example, [9] assumes a two-step generation process of a query from a document. First, a dependency structure is generated from a document. Then, a sequence of query terms is generated from the dependency structure. This model is conventionally called Hidden Markov Model (HMM), where the dependency structure can be viewed as the hidden variable. In theory, the hidden variable can represent any internal structures of language, which makes HMM a very powerful modeling framework in many natural language processing (NLP) tasks.

However, the HMM approach may be deficient for realistic applications for two reasons. The first is from a theoretical perspective. The parameters of HMM have been traditionally trained using *maximum likelihood estimation* (MLE), usually with smoothing methods to deal with the sparse data problem. This approach is proved to be optimal in theory under two assumptions: the true distribution of data (documents and queries) on which HMM is based is known; and there are enough training data. Unfortunately, these assumptions rarely hold in IR tasks. The second reason is from a practical perspective. The performance of IR systems is generally measures in terms of precision and recall (i.e. average precision, AP, in this study), which is loosely associated with the optimization criterion that MLE uses (i.e. to maximize the likelihood of training data). Therefore, the traditional HMM approach may not lead to an optimal solution in realistic IR systems.

Recently, discriminative training methods have been found to outperform state-of-the-art HMM approaches in many NLP tasks [2]. The success is mainly attributed to the two properties which would eliminate to some degree the two aforementioned problems of the HMM approach. First, a discriminative model is based on a much weaker assumption that both training data (documents) and test data (queries) are generated from the same distribution but the form of the distribution is unknown. Second, unlike MLE that maximizes the function that is loosely associated with the performance measure of IR, discriminative training methods aim to directly optimize the precision and recall. So they potentially lead to a better solution. The performance of discriminative models depends to a large degree upon the selection of an optimal set of feature functions that can best distinguish desired results from undesired ones. However, there is no systematic selection method. Most previous work simply begins with a list of candidate features defined manually (e.g. by a domain expert), and then construct an optimal set empirically by trial and error.

We see that both HMM and discriminative training methods have their strength and weakness. The HMM approach assumes a generation process of data so that a sequence of models (e.g. each for a generation step) can be derived to capture linguistic information. The discriminative training methods, given a set of feature functions, tend to learn a model di-

[1]The work was done while Haoliang and Xinsong were visiting Microsoft Research Asia. Thank Ken Church for comments on early draft of the paper.

rectly optimized for IR tasks. Our design strategy is to combine the strength of both approaches as follows: We first assume a query generation process, and derive a set of component models within the framework of HMM. We then derive a set of feature functions from the component models, and construct a *linear discriminant model* (LDM) for IR, where the linear interpolation weights are estimated using discriminative training methods so as to optimize the AP on the training set. The effectiveness of this design strategy will be demonstrated on the task of ad hoc retrieval on six English and Chinese TREC test sets. Results show that in most test sets, LDM outperforms significantly the state-of-the-art LM approaches and the classical probabilistic retrieval model. The robustness of the approach is also studied empirically in this paper.

The rest of this paper is structured as follows. Section 2 describes a method of deriving linguistically-motivated feature functions within the HMM framework. Section 3 describes the LDM for IR and discusses in detail two discriminative training methods of estimating parameters. Section 4 presents experimental results. The discussion and related work are presented in Section 5. Finally, the paper is concluded in Section 6.

2. HMM for IR

In LM approaches to IR, a Markov model is trained on each document $\mathbf{d}$ in the collection C to be searched. Then the documents are ranked by the probability that a query $\mathbf{q} = \{q_1,\ldots,q_m\}$ would be generated from the respective document model $P(\mathbf{q}\,|\,\mathbf{d})$. Most state-of-the-art approaches assume a unigram Markov model, where $P(\mathbf{q}\,|\,\mathbf{d})=\prod_{i=1\ldots m}P(q_i\,|\,\mathbf{d})$ [30].

Hidden Markov Model is an extension of the Markov model by introducing hidden variables. In the context of language modeling, the hidden variables can be used to represent any linguistic *concepts* that may improve the performance of IR but are "hidden" in the text. Some representative examples of such concepts are semantic chunks (e.g. named entities like person name, location name, etc.) and syntactic chunks (e.g. noun phrases, verb phrases, etc.). In HMM approaches to IR, documents are ranked by $P(\mathbf{q}\,|\,\mathbf{d})$. But unlike the case of Markov model, a two-stage generation process is assumed when estimating $P(\mathbf{q}\,|\,\mathbf{d})$, as follows.

First, a user chooses a sequence of concepts $\mathbf{c}$ (e.g. person name) to be queried, according to the probability distribution $P(\mathbf{c}\,|\,\mathbf{d})$; Then the user attempts to express each concept by choosing a sequence of terms, according to the probability distribution $P(\mathbf{q}\,|\,\mathbf{c}, \mathbf{d})$. Therefore, $P(\mathbf{q}\,|\,\mathbf{d})$ can be recovered over all possible $\mathbf{c}$ as

$$P(\mathbf{q}\,|\,\mathbf{d}) = \sum_{\mathbf{c}} P(\mathbf{q}\,|\,\mathbf{c},\mathbf{d})P(\mathbf{c}\,|\,\mathbf{d})$$

For efficiency, in practical systems we usually only consider the most likely $\mathbf{c}$, which can be detected by parsing technologies, described in Section 4. We then end up with the basic form of HMM for IR in Equation (1).

$$P(\mathbf{q}\,|\,\mathbf{d}) \propto \max_{\mathbf{c}} P(\mathbf{q}\,|\,\mathbf{c},\mathbf{d})P(\mathbf{c}\,|\,\mathbf{d}) \qquad (1)$$

$P(\mathbf{q}\,|\,\mathbf{c}, \mathbf{d})$ in Equation (1) is referred to as *concept model* afterwards, and $P(\mathbf{c}\,|\,\mathbf{d})$ as *document model*. In our system we use one document model where we assume that each concept c_i is generated depending on its preceding concept c_{i-1}, and each concept is represented by its headword h_i which is detected using rules (e.g. the rightmost noun is the head of a NP, and so on). Therefore, the document model is a headword bigram model $P(h_i\,|\,h_{i-1}, \mathbf{d})$. We use a set of concept models, each of which models a different concept (i.e. estimates the generative probability of a term sequence given a certain type of concept), as shown in Table 1.

Type	**Models**
NP	$P(\mathbf{q}'\,|\,\mathrm{NP})=\mathrm{P}(h\,|\,\mathrm{NP}) \prod_{q\in\mathbf{q}'} P(q\,|\,h, \mathrm{NP})$
VP	$P(\mathbf{q}'\,|\,\mathrm{VP})=\mathrm{P}(h\,|\,\mathrm{VP}) \prod_{q\in\mathbf{q}'} P(q\,|\,h, \mathrm{VP})$
NE	$P(\mathbf{q}'\,|\,\mathrm{NE})=\mathrm{P}(h\,|\,\mathrm{NE}) \prod_{q\in\mathbf{q}'} P(q\,|\,h, \mathrm{NE})$, where there are three NE models, each for one type of NE.
FT	$P(\mathbf{q}'\,|\,\mathrm{FT})=1$ if $\mathbf{q}'$ can be parsed by FT grammar, 0 otherwise; where the FT grammar is a set of Finite-State Machines, each for one type of factoids

Table 1. Linguistic concepts and concept models: NP stands for noun phrase; VP for verb phrase; NE for named entities (i.e. person names, locations, and organizations); FT for factoids (e.g. date, time.); $\mathbf{q}'$ denotes the chunk of query terms, which represent a concept c within the query $\mathbf{q}$; h is the headword of the concept (detected by rules) and q is any other term in the concept.

It should also be noted that different concept models are constructed in different ways (e.g. *person name* models are *n*-gram models trained via MLE on the corpus of person name list whereas *factoid* models use derivation rules and have binary values). The dynamic value ranges of different concept models can be so different that it is inappropriate to combine all models through simple multiplication as in Equation (1).

To remedy this problem, one can introduce a set of weights, one for each concept model, to effectively balance the contribution of each component model to the performance of IR. Intuitively, we would assign a high weight to the component model which is either reliably trained (on enough training data) or represents a salient concept of a query (e.g. proper noun). This motivates the use of the LDM framework, which will be described in the next section.

3. LDM for IR

Linear discriminant model in this study follows the general framework of linear discriminant functions widely used for pattern classification [4], and has been recently introduced into NLP tasks in [2].

In the LDM framework, we assume a set of $N+1$ features $f_i(\mathbf{q}, \mathbf{c}, \mathbf{d})$, for $i = 0, \ldots, N$. The features are arbitrary functions that map $(\mathbf{q}, \mathbf{c}, \mathbf{d})$ to real values. Using vector notation, we have $\mathbf{f}(\mathbf{q},\mathbf{c},\mathbf{d}) \in \Re^{N+1}$, where $\mathbf{f}(\mathbf{q},\mathbf{c},\mathbf{d}) = \{f_0(\mathbf{q},\mathbf{c},\mathbf{d}), f_1(\mathbf{q},\mathbf{c},\mathbf{d}), \ldots, f_N(\mathbf{q},\mathbf{c},\mathbf{d})\}$. The parameters of the model are a vector of $N + 1$ parameters, each for one feature function, $\boldsymbol{\lambda}= \{\lambda_0, \lambda_1, \ldots, \lambda_N\}$. The relevance score of a document $\mathbf{d}$ with respect to a given query $\mathbf{q}$ can be written as

$$Score(\mathbf{q},\mathbf{d},\boldsymbol{\lambda}) = \boldsymbol{\lambda}\mathbf{f}(\mathbf{q},\mathbf{c},\mathbf{d}) = \sum_{i=0}^{N}\lambda_i f_i(\mathbf{q},\mathbf{c},\mathbf{d}) \qquad (2)$$

Equation (2) is essentially of the same form of any linear discriminant functions in [4]. Our method is novel in that most of the feature functions in Equation (2) are derived from the component models in the HMM framework, as shown in Table 1. More specifically,

- $f_0(.)$ is called the base feature and is defined as the logarithm of the unigram probability, i.e. $f_0(\mathbf{q}, \mathbf{d}) = \sum_i \log(P(q_i\,|\,\mathbf{d}))$;

- $f_1(.)$ is defined as the logarithm of the bigram probability, i.e. $f_1(\mathbf{q}, \mathbf{d}) = \sum_i \log(P(q_i \mid q_{i-1}, \mathbf{d}))$;
- $f_2(.)$ is defined as the logarithm of the document model probability, i.e. $f_2(\mathbf{q}, \mathbf{c}, \mathbf{d}) = \sum_i \log(P(h_i \mid h_{i-1}, \mathbf{d}))$.
- $f_i(.)$, for $i = 3,\ldots,N$, are defined for N-2 concepts, respectively (i.e., they are basically derived from the concept models listed in Table 1). Their values are either the negative logarithm of the probabilities of the corresponding probabilistic models, or assigned heuristically (e.g. the value of an FT feature is defined as the count of term sequences of that type of FT in $\mathbf{q}$).

The LDM method described above is expected to be superior to both traditional discriminative training methods and HMM methods. Most discriminative models use only binary-valued features, while the feature functions in LDM are much more "informative" because they are derived from probabilistic models. In HMM approaches, each component model is optimized independently according to a criterion loosely associated with AP, while all feature functions in the LDM framework can be jointly optimized directly toward the maximal AP on training data. In this study, our training set consists of a set of queries, each with a list of documents whose relevance has been judged manually.

We now turn to the description of two methods of estimating $\boldsymbol{\lambda}$ under the framework of gradient descent: an iterative procedure of adjusting the parameters $\boldsymbol{\lambda}$ in the direction that optimizes the objective function. For each of the two algorithms, we will present in turn the objective function and the optimization algorithm.

3.1 Maximum AP Training

The first algorithm is to select an optimal parameter setting so as to directly maximize the average precision (AP) on training data. We call the algorithm *maximum AP* (MaxAP) *training*. AP is the most common performance measure in the IR research community. So MaxAP is intuitively appealing since it optimizes the performance measure directly. We now give the formal definition of AP, and then describe the algorithm.

Let $\mathbf{R_q}$ be the set of relevant documents of the query $\mathbf{q}$, and $\mathbf{Q}$ be the query set in training data. Let Rank($\mathbf{d}_i$, $\mathbf{q}$, $\boldsymbol{\lambda}$) be the rank of the i'th relevant document $\mathbf{d}_i$ (for $i = 1,\ldots, |\mathbf{R_q}|$) appearing in the document list of $\mathbf{q}$, ordered according to the score computed by Equation (2), where $\boldsymbol{\lambda}$ is the current parameter setting of the LDM. Then, the AP of a query $\mathbf{q}$ is defined as

$$\mathrm{AP}(\mathbf{q},\boldsymbol{\lambda}) = \frac{1}{|\mathbf{R_q}|}\sum_{i=1}^{|\mathbf{R_q}|}\frac{i}{\mathrm{Rank}(\mathbf{d}_i,\mathbf{q},\boldsymbol{\lambda})} \tag{3}$$

The MaxAP algorithm is to optimize the empirical AP on training data, and is defined in Equation (4).

$$MaxAP \overset{def}{=} \arg\max_{\boldsymbol{\lambda}} \frac{1}{|\mathbf{Q}|}\sum_{\mathbf{q}\in\mathbf{Q}}\mathrm{AP}(\mathbf{q},\boldsymbol{\lambda}) \tag{4}$$

The MaxAP algorithm can be cast as the multi-dimensional function optimization algorithm (e.g. [24]). Assume that we can maximize AP with respect to one parameter λ using *line search*, which will be described below. The MaxAP algorithm works as follows: Take λ_0, λ_1, …, λ_N as a set of directions. Using line search, move along the first direction so that the objective function, as shown in Equation (4), is maximized; then move from there along the second direction to its maximum, and so on. Cycling through the whole set of directions as many times as necessary, until the object function stops increasing.

We now describe how to implement line search. We notice that regular numeric line search methods [24] cannot be applied directly because the value of a parameter λ versus the objective function AP(.) is not smooth and there are multiple local maxima. Therefore, we use the method proposed in [22]: Let λ be the selected parameter. The line search is to find the optimal value of λ so as to maximize the average precision. By adjusting λ within a bracket (i.e. an interval which is known to contain acceptable points), we obtain for each query in training data an ordered sequence of AP(.) values and a corresponding sequence of λ intervals. By averaging AP(.) values over all queries in training data, we obtained a global sequence of AP(.) and the corresponding global sequence of λ intervals. We can therefore find the optimal λ as well as its corresponding AP(.) by traversing the sequence.

In our experiments, we found that the MaxAP algorithm can converge on different maxima given different starting points. Following [25], we attempt to perform the algorithm multiple times, each from a different, random starting point, and pick the parameter setting that achieves the maximal AP.

3.2 Perceptron-based Training

This section first formulates the ranking problem under the framework of *ordinal regression* [14, 12]; then presents a loss function which is closely associated with AP, and a perceptron-based algorithm to optimize the parameter setting with respect to the loss function.

The ranking problem in IR can be formulated as follows: Given a query $\mathbf{q}$, let $\mathbf{r}^*(\mathbf{q})$ be the target (or optimal) document rank list, usually judged manually, and $\mathbf{r}(\mathbf{q}, \boldsymbol{\lambda})$ the rank list generated by the LDM of Equation (2) with the parameter setting $\boldsymbol{\lambda}$. Then, within the framework of ordinal regression, $\boldsymbol{\lambda}$ is optimized in such a way that $\mathbf{r}(\mathbf{q}, \boldsymbol{\lambda})$ is closest to $\mathbf{r}^*(\mathbf{q})$. As [14] suggested, the similarity between two ranks can be measured in terms of Kendall's τ, which is defined as follows. A document pair $\mathbf{d}_i \neq \mathbf{d}_j$ is concordant, if both $\mathbf{r}(\mathbf{q}, \boldsymbol{\lambda})$ and $\mathbf{r}^*(\mathbf{q})$ agree on how they order $\mathbf{d}_i$ and $\mathbf{d}_j$. It is discordant if they disagree. Let X and Y be the numbers of concordant and discordant document pairs, respectively. Note that if there are M documents in the collection C to be searched, the sum of X and Y is $M(M$-1$)/2$ for strict orderings. Then, Kendall's τ can be defined in Equation (5). See [14] for a full description.

$$\tau(\mathbf{r}^*,\mathbf{r}) = \frac{X-Y}{X+Y} = 1-\frac{4Y}{M(M-1)} \tag{5}$$

The optimization problem can be written as below, where $\mathbf{Q}$ is the query set in training data.

$$\begin{aligned}\boldsymbol{\lambda} &= \arg\max_{\boldsymbol{\lambda}} \sum_{\mathbf{q}\in\mathbf{Q}}\tau(\mathbf{r}^*(\mathbf{q}),\mathbf{r}(\mathbf{q},\boldsymbol{\lambda}))\\ &= \arg\max_{\boldsymbol{\lambda}} \sum_{\mathbf{q}\in\mathbf{Q}}1-\frac{4Y}{M(M-1)}\\ &= \arg\min_{\boldsymbol{\lambda}} \sum_{\mathbf{q}\in\mathbf{Q}}Y \end{aligned} \tag{6}$$

Equation (6) shows that the optimal $\boldsymbol{\lambda}$ is the one that leads to least number of discordant document pairs. It is worth noticing that as proved in [14], the number of inversions Y gives a lower bound on AP. So, optimizing $\boldsymbol{\lambda}$ by minimizing Y is closely associated with increasing AP.

In our experiments, given a query $\mathbf{q}$, each of the documents in C is judged by a binary value: 1 if the document is relevant, 0 otherwise. There is no order among relevant (or irrelevant) documents. Therefore, Y in Equation (6) is reduced to the number of document pairs, where the irrelevant document is ranked higher than the relevant document. Using LDM, all documents are ranked by the score computed by Equation (2). Therefore, Equation (6) can be rewritten as follows

$$\boldsymbol{\lambda} = \arg\min_{\boldsymbol{\lambda}} \sum_{\mathbf{q}\in\mathbf{Q}} \sum_{\mathbf{d}_i,\mathbf{d}_j\in C} I[Score(\mathbf{q},\mathbf{d}_i,\boldsymbol{\lambda}) - Score(\mathbf{q},\mathbf{d}_j,\boldsymbol{\lambda})] \qquad (7)$$

where $I[\pi]$=1 if $\pi \leq 0$, and 0 otherwise; $\mathbf{d}_i$ is any relevant document of $\mathbf{q}$ in C, and $\mathbf{d}_j$ is any irrelevant document.

Given the objective function in Equation (7), we use a perceptron-based algorithm to search for the optimal parameter setting. Please see [2] for the application of the perceptron algorithm in NLP tasks and the theoretical justifications (i.e. proofs of its convergence and bounded generalization errors). It is an incremental, error-correction training procedure. As shown in Figure 1, it starts with an initial parameter setting and adapts it each time a discordant pair is detected.

```
Input: training samples, {$(\mathbf{d}_i, \mathbf{d}_j)_\mathbf{q}$; $\mathbf{d}_i, \mathbf{d}_j \in C$, $\mathbf{q} \in \mathbf{Q}$}
Output: parameter setting $\boldsymbol{\lambda}^T$
1. Initialization: set $\lambda_0 = 1$, $\lambda_i = 0$, for $i=1\ldots N$.
2. For $t = 1$ to $T$
3.   For each training sample $(\mathbf{d}_i, \mathbf{d}_j)_\mathbf{q}$
4.     If $Score(\mathbf{q}, \mathbf{d}_j, \boldsymbol{\lambda}^t) > Score(\mathbf{q}, \mathbf{d}_i, \boldsymbol{\lambda}^t)$ by Equation (4), then
5.       For each $\lambda^t_n$ ($n=1\ldots N$)
6.         $\lambda^{t+1}_n = \lambda^t_n + \eta(f_n(\mathbf{q},\mathbf{c},\mathbf{d}_i) - f_n(\mathbf{q},\mathbf{c},\mathbf{d}_j))$.
```

Figure 1. Perceptron algorithm. In Input: $(\mathbf{d}_i, \mathbf{d}_j)_\mathbf{q}$ represents document pair of query $\mathbf{q}$, where $\mathbf{d}_i$ is any relevant document of $\mathbf{q}$ in C, and $\mathbf{d}_j$ is any irrelevant document; $\mathbf{Q}$ is the query set in training data. In Step 6: η is the learning rate and set to 0.001.

The perceptron algorithm is proved to be robust and guaranteed to converge when the training samples are *separable* (i.e., there is an ideal parameter setting which leads to a LDM that can achieve zero discordant pair). But in IR tasks, such an ideal parameter setting does not exist, and the training samples are not linearly separable. In theory, this may lead the perceptron algorithm unstable, and the error-correction procedure can never cease. As [4] point out, if the correction process is determined at some arbitrary point, the parameter setting may or may not be in a good state. We used two methods to deal with this problem. First, to reduce the risk of obtaining a bad solution by accidentally choosing an unfortunate termination time, we average the parameter settings produced by the correction rule in Step 6 of Figure 1 as follows: Let $\lambda_n^{t,m}$ be the value for the n'th parameter after the m'th training sample has been processed in pass t over the training data. Then the average parameters are defined as

$$\lambda_n = \textstyle\sum_{t=1\ldots T,m=1..M}\lambda_n^{t,m}/MT.$$

where M and T are the number of training samples and the number of learning iterations, respectively. This variant of the perceptron algorithm is called the *averaged perceptron* algorithm, proposed in [2]. Second, we count the number of updates for each training sample. If the number is larger than a preset threshold (meaning that the sample cannot be correctly ordered after many trails and is likely to be a noisy sample), the sample will not be used for training in the consequential iterations. As will be illustrated in Section 4, the two methods lead to a robust algorithm for IR.

4. Experiments

4.1 Settings

We evaluated the LDM approach to IR described in the previous sections using six different TREC test sets, including three English test sets and three Chinese ones. Some statistics are shown in Table 2, where **TX_cn** denotes Chinese collections used in TREC-X. (Note that TREC-5 and 6 use the same collection). The English queries are TREC topics 201 to 250 (description field only) on TREC disks 2 and 3. Those topics are "natural language" queries consisting of one sentence each of length 10 to 15 words. Following [11], for the three English TREC collections, we remove those queries that have no relevant document. The Chinese queries are TREC topics CH1 to CH79. We use long queries that contain the title, description and narrative fields. The average length of these queries is 120 characters.

Coll.	Description	Size (MB)	# Doc.	# Query
WSJ	*Wall Street Journal* (90, 91, 92), Disk 2	248	74,520	45
AP	Associated Press (88 - 90), Disks 2, 3	484	158,240	49
FR	*Federal Register* (88), Disk 2	213	19,860	27
T5_cn	*People's Daily* (91-93) and *Xinhua News*	167	164,789	28
T6_cn	(94-95).			26
T9_cn	*HK Commercial Daily* (98-99), *HK Daily News* (99), *Takungpao* (98-99)	260	126,937	25

Table 2. TREC collections.

To extract the concept sequence **c** for each sentence, which is used for constructing LDM, we process the collections as follows. All Chinese texts have been word-segmented using the word segmentation system **MSRSeg**[2] [8]. The system also identifies factoids and named entities of various types. We then used an in-house HMM chunk parser to detect phrases such as NP and VP, as described in Table 1. Similarly, all English texts have been tokenized and chunked by an in-house HMM parser, which is trained on Penn Treebank.

We compare LDM to both the classical probabilistic model (i.e. the binary independent retrieval (BIR) model [15]) and some state-of-the-art language models proposed for IR in the literature. All models contain free parameters that must be estimated empirically by trial and error. These parameters include feature weights in LDM, smoothing or interpolation parameters in language models and weights or constants in the BIR model. Therefore, we have applied an experimental paradigm called two-fold cross validation. For each of the six query set, we divided it into two subsets, with one used for parameter training and the other for test. The retrieval results reported on each TREC test set (as shown in Tables 3 and 4) combine

[2] The simplified version of **MSRSeg** is accessible at http://research.microsoft.com/~jfgao

two sets of results on two subsets of the query set, respectively. Each set of results on one subset is obtained using the parameter settings optimized on the other subset.

The performance of IR is measured through the precision-recall pair. The main evaluation metric in this study is the non-interpolated average precision (AP). The significance tests are also conducted.

4.2 Results

Tables 3 and 4 present our main experimental results, where we compare LDM with four probabilistic retrieval models, including an implementation of the BIR model and three state-of-the-art language modeling approaches that are based on the framework of either Markov model or HMM.

BIR (Binary Independent Retrieval model) is one of the most representative classical probabilistic retrieval models, and serves as one of the baseline models in our experiments. In particular, we used the Okapi system, which is the best-known implementation of BIR. Among the great number of term weighting functions provided by Okapi, we choose BM2500 for it has achieved good performance in previous experiments[27].

UGM (Unigram Model) is an implementation of the unigram language model approach to IR proposed in [30]. It serves as the baseline LM approach in our experiments. Over all six TREC test sets, UGM achieves the performance similar to, or slightly worse than, that of BIR. It has been observed that in general the classical probabilistic retrieval model and the unigram language model approach perform very similarly if both have been fine-tuned. The slightly worse performance of UGM in our experiment might be due to our "over-tuned" Okapi system, i.e. BM2500 has more weighting parameters tuned empirically. Notice that unlike LDM, whose parameters can be trained using appropriate learning algorithms as described in Section 3, there is no systematic way of tuning free parameters of BIR and UGM (and other language models described below). So, we first use heuristics to find for each parameter a bracket, and perform exhaustive search.

BGM (Bigram Model) is an implementation of the bigram language model approach to IR. The query generation probability is estimated by $P(\mathbf{q}|\mathbf{d}) = P(q_1|\mathbf{d})\prod_{i=2\dots m}P(q_i|q_{i-1}, \mathbf{d})$. It assumes that the query term only depends on its one preceding term. To deal with the sparse data problem, we used two smoothing methods. First, we linearly interpolated the bigram models trained on the document **d** and the entire collection *C*, respectively. Second, for both bigram models, the bigram probability was linearly interpolated with the unigram probability. All *n*-gram (n = 1 or 2 in BGM) probabilities are estimated via MLE with a modified version of the absolute discount smoothing [10]. BGM can be viewed as a special case of HMM described in Section 2, where the concept sequence **c** is a sequence of adjacent word pairs. Results show that BGM substantially outperforms UGM in all English test sets, demonstrating that even the simplest **c** can benefit the IR performance. However, BGM does not outperform UGM on Chinese test sets. A possible reason is that MSRSeg already groups many adjacent short word sequences into long words, such as named entities and compound nouns. Therefore, BGM may not be able to capture additional useful local information compared to UGM. Our speculation has been justified in the pilot study: When using maximal matching to segment words by looking up a small dictionary, BGM outperforms UGM on Chinese test sets.

DLM (Dependence Language Model) is an implementation of the dependence language model described in [9]. It serves in the comparison experiments as an example of the HMM approaches to IR. DLM uses a score function similar to Equation (1), where **c** is defined as a so-called *linkage*. The linkage is detected by a parser and is represented an acyclic, planar, undirected graph where two related query terms are connected by a graph edge. DLM assumes a two stage generation process as described in Section 2: First, the linkage is generated from the document according to the distribution $P(\mathbf{c}|\mathbf{d})$. Second, the query is generated according to the distribution $P(\mathbf{q}|\mathbf{c}, \mathbf{d})$ where each query term is generated depending on the associated term with which it is linked in the linkage. Therefore, DLM can be viewed as an extension of BGM in that **c** is a set of word pairs where the two words are not necessarily adjacent. This advantage is however paid by the complexity of modeling. DLM uses a more sophisticated smoothing method than BGM does, and introduces more free parameters to be optimized heuristically. As shown in Table 3, although DLM outperforms BGM slightly but significantly on two out of three English test sets, considering the modeling cost, the limited performance gain may not be worthwhile in practical systems. More importantly, it is almost impossible to integrate more sophisticated linguistic concepts through the hidden variables.

LDM (Linear Discriminative Model) is the model described in Section 3. **LDM(MaxAP)** is the LDM trained using the MaxAP algorithm described in Section 3.1, and **LDM(Percep)** is the model trained using the perceptron-based algorithm described in Section 3.2. We see that both LDM methods achieve improvements over both BIR and UGM on five out of six query test sets, and also outperform other LM approaches on most of the test sets. As one of the reviewers point out that the comparison may be unfair for the LDM uses more features. We argue that this is the main advantage of LDM to incorporate arbitrary features. When the same set of features are used for both LDM and LMs, similar results were achieved.

We notice that on the T9_cn test set, the LDM methods are worse than other models. The reason is that while our parser is trained on news articles from People's Daily, T9_cn is a collection of news articles using Hong Kong Chinese (a local official language). Thus, the language gap between training and test texts leads to a bad result of concept extraction, and LDM cannot be generated properly. Another point worth noting is that FR query set is "unbalanced" in that a small number of queries have much more relevant documents than the rest. Therefore, the results of LDM are sensitive to the training/test split. We therefore randomly create 10 training/test splits. The results reported in this paper are the average among 10 tests.

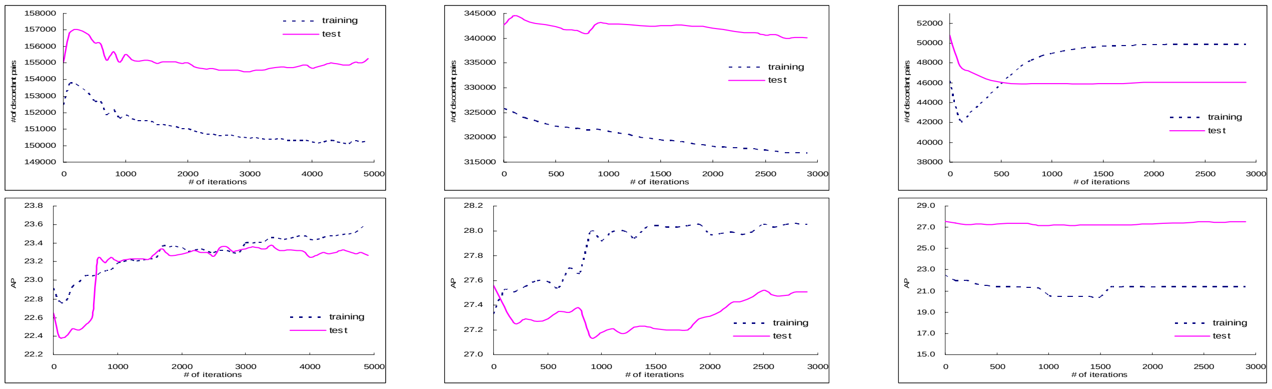
It is also interesting to investigate the robustness of the two training algorithms. We find that MaxAP is robust across all test sets. It generally converges on both training and test sets after four or five iterations. We do not observe a severe overfitting problem. The perceptron-based algorithm, on the other hand, performed differently across different test sets. We can observe the learning curves of this algorithm on six test sets in Figure 2. Recall that the algorithm aims to minimize the number of discordant document pairs (i.e. *Y* in Equation (6)). It turns out that the algorithm achieves its goal quite successfully: In all six test sets, *Y* decreases with the increase of the number of iterations, though the overfitting problem can be observed in some test sets. For example, on FR, T5_cn and T9_cn, *Y*

Models	WSJ			AP			FR		
	AP	% change over BIR	% change over UGM	AP	% change over BIR	% change over UGM	AP	% change over BIR	% change over UGM
BIR	22.30	--	--	25.34	--	--	16.58	--	--
UGM	17.91	-19.69% *	--	24.58	-3.00%	--	14.81	-10.68%	--
DLM	22.41	0.49%	25.13% *	25.87	2.09%	5.25% *	18.51	11.64%	24.98%
BGM	21.46	-3.77%	19.82%	26.24	3.55%	6.75% *	18.03	8.75%	21.74%
LDM(MaxAP)	**23.61**	5.87%	31.83% *	27.33	7.85% *	11.19% *	18.51	11.64%	24.98%
LDM(Percep)	23.34	4.66%	30.32% *	**27.51**	8.56% *	11.92% *	**19.42**	17.13%	31.13%

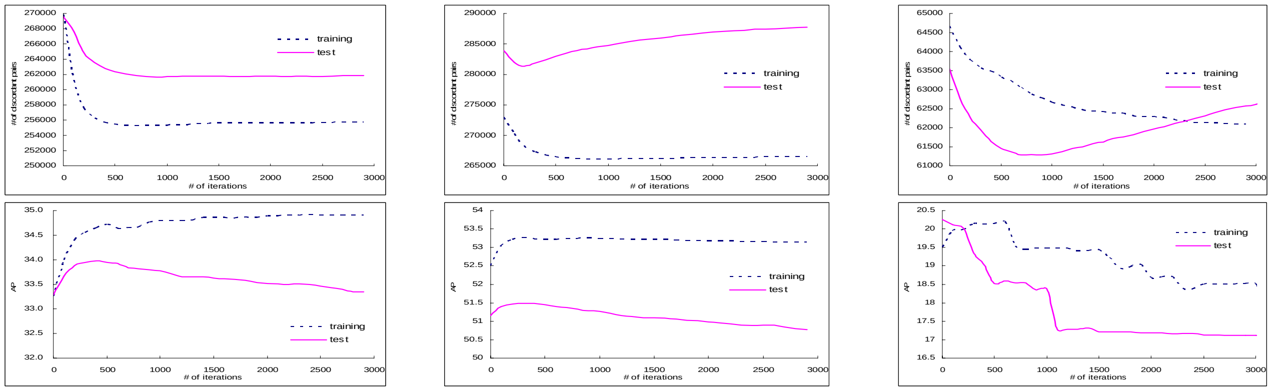
Table 3. Comparison results on **WSJ**, **AP** and **FR** collections. * indicates that the difference is statistically significant.

Models	T5_cn			T6_cn			T9_cn		
	AP	% change over BIR	% change over UGM	AP	% change over BIR	% change over UGM	AP	% change over BIR	% change over UGM
BIR	32.45	--	--	50.28	--	--	**20.79**	--	--
UGM	31.81	-1.97%	--	51.15	1.73%	--	19.93	-4.14%	--
BGM	31.28	-3.61%	-1.67%	50.58	0.60%	-1.11%	19.60	-5.72%	-1.66%
LDM(MaxAP)	33.87	4.38%	6.48% *	**53.06**	5.53% *	3.73% *	19.37	-6.83%	-2.81%
LDM(Percep)	**33.92**	4.53% *	6.63% *	51.40	2.23%	0.49%	18.59	-10.58%	-6.72%

Table 4. Comparison results on **T5_cn**, **T6_cn** and **T9_cn** collections. * indicates that the difference is statistically significant.



(a) **WJS** (b) **AP** (c) **FR**



(d) **T5_cn** (e) **T6_cn** (f) **T9_cn**

Figure 2. Learning curves of the perceptron-based algorithm on six query sets, where the upper figures show the curves of the number of discordant document pairs and lower figures show the curves of average precision (AP).

decreases in the beginning and then increases. We also observe a strong correlation between the reduction of Y and the improvement of AP in half of the test sets (WJS, AP and T6_cn). Globally, the perceptron algorithm is less robust than MaxAP.

In summary, the experimental results show that

(1) LDM significantly outperforms the state-of-the-art LM approaches and the BIR model in most test sets;

(2) It is more appropriate to train whatever IR model to optimize the objective that the IR system is graded on (e.g. AP) rather than likelihood; and

(3) Linguistic features (e.g. phrases and dependences) are effective for IR if they are incorporated properly by for example using LDM.

5. Discussions and Related Work

5.1 Generative and Discriminative Models for IR

The IR problem can be reformulated as a pattern classification problem via many ways, one of which is that, given a query, each document of the collection to be searched is classified into two classes: relevant and non-relevant. Probabilistic classifiers have been typically grouped into two categories: generative and discriminative models. The former learn a model of the joint probability $P(x, y)$ of the input x and the label y, and make predictions by using the Bayes rules to calculate $P(x \mid y)$, and picking the most likely y. The latter model the posterior $P(y \mid x)$ directly. Recently, discriminative classifiers are preferred to generative ones due to several compelling reasons, one of which, as pointed out by Vapnik [29], is that "one should solve a (classification) problem directly and avoid solving a more general problem as an intermediate step (such as modeling $P(x \mid y)$)."

As discussed in [19], most of the existing retrieval models can be viewed as generative models. For example, in the LM approach to IR [17, 30], assume each document is a unique class (y), and the task of IR model is to classify a query (x) into its most likely class as given by the posterior $P(y \mid x)$. Then, language models make their prediction by using the Bayes rules to estimate $P(x \mid y)$. Similarly, [19] shows that the classical probabilistic models including the BIR model [15] and its variant, the two-Poisson model [26], also belong to generative models. Although all these generative models achieve state-of-the-art performance in large scale IR experiments, there are several appealing reasons to explore discriminative models for IR. (Readers can refer to [19] for a detailed discussion.) The first reason is the issue of the modeling assumption in generative models, as discussed in Section 1. Generative models used in IR assume a multinomial distribution where query terms (or term pairs in bigram models) are generated independently by the document model. However, this independence assumption is observed to be false in reality. Moreover, a single document cannot be regarded as a large training set for language model learning. Discriminative models, on the other hand, make very few assumptions on model form. They explore arbitrary features that can differentiate correct labeling versus wrong labeling. Even if a feature function (which for example is derived from a probabilistic model) is poorly estimated due to the sparse data problem, it can still bring some positive impact on the performance of the combined linear discriminative model, provided that the weights are properly learned using an appropriate algorithm.

The second reason concerns the flexibility of incorporating arbitrary features. In generative models, as described earlier, the incorporation can be achieved in two ways (see e.g. [9, 20, 18, 28]). The first approach is to model those features as hidden variables of HMM and integrate them into the generation process. The second approach is to model those features independently, and combine them as a mixture model. The problem of the first approach is its complexity. The parameters of different component models are difficult to learn consistently via MLE. In the second approach, the interpolation weights can only be determined by empirical means and cannot guarantee the optimality. In discriminative models, the features can be arbitrary functions, while the feature weights are learned to optimize an objective function which is defined to be closely related to the evaluation measure of IR systems.

The LDM approach to IR described in this paper can be viewed as a version of discriminative models. As discussed in [20], being a discriminative model, the parameters can be fit either to maximize the conditional likelihood on the training set, or to minimize training error. In LDM, the parameters $\boldsymbol{\lambda}$ are learned either to maximize the AP directly, or to minimize the discordant document pairs in a rank list on the training set. In that sense, LDM belongs to the latter version of discriminative models, which is more truly in the "spirit" of discriminative learning.

Whilst there are many previous attempts similar to the LDM approach, most of them focus on parameter learning algorithms and do not explore thoroughly the problem of how to derive features. They either assume a set of pre-defined features or merge multiple preferences provided by multiple experts directly. In the latter case, each expert can be viewed as a feature function, solely on which a rank (i.e. preference) is based. In our approach, most feature functions are derived from the component models of HMM, and are expected to be more tractable and informative. Below we review some of the parameter learning methods that are closely related to ours.

5.2 Parameter Learning Algorithms

The ranking SVM proposed by Joachims [14] is related to the perceptron-based algorithm described in Section 3.2. If we decompose the sum term in right-hand-side of Equation (7), and rewrite each decomposed term as a constraint of the form

$$\boldsymbol{\lambda}(\mathbf{f}(\mathbf{q}, \mathbf{d}_i) - \mathbf{f}(\mathbf{q}, \mathbf{d}_j)) \geq 1 - \xi_{i,j,\mathbf{q}},$$

where $\xi_{i,j,\mathbf{q}}>0$, the optimization problem appears to be equivalent to that of a classification SVM on pairwise difference vector $\boldsymbol{\lambda}(\mathbf{f}(\mathbf{q}, \mathbf{d}_i) - \mathbf{f}(\mathbf{q}, \mathbf{d}_j))$, and can be solved using decomposition algorithms similar to those used for SVM classification. In our experiments, we have adapted the SVM^{light} algorithm [13]. It achieves similar performance to that of the perceptron-based algorithm.

As described in Section 3.2, the IR problem can be cast as a special case of ordinal regression discussed in [12]. In ordinal regression, all objects are ranked on the same scale, while in IR documents need to be ranked with respect to one query.

Freund et al. [6] proposed a learning approach based on boosting algorithm to linearly combining multiple ranks provided by experts. If we consider each expert as a feature function, it appears to be equivalent to our problem. However, they do not consider explicitly the distribution over queries. In our pilot study that uses the two fold cross-validation method (see Section 4.1), we observe a serious overfitting problem of the boosting algorithm. The optimal parameter setting leaned on one query subset works poorly on the other subset. If we redefine the two fold cross-validation for each query, i.e. half documents for training and the other half for test, then the boosting algorithm works well. However, in realistic IR systems, we have to handle unseen queries which might be very different from previously seen queries. We leave the extension of the method to future work. Earlier work along this line can be found in [1].

Another similar approach is the Pranking algorithm proposed by Crammer and Singer [3]. Unlike the perceptron-based algorithm described in Section 3.2, which reduces the total rank into a set of document pairs, the Pranking algorithm maintains a total ordered set via project, and adjusts the position of documents in the rank directly by adjusting model parameters. Thus, the algorithm is theoretically more efficient. It will be interesting to compare it in the LDM approach in large scale TREC experiments. We leave it to future work.

Finally, it is worth noting that the MaxAP algorithm is a simple example of the non-smooth optimization (NSO) algorithms [5]. Most parameter learning problem in NLP and IR tasks can be considered as a multi-dimensional function optimization problem. However, the objective function, such as the classification error rate of a given finite set of training samples, is usually a non-smooth function (i.e. a piecewise constant function) of the model parameters, and thus cannot be easily optimized using regular gradient-based numerical methods. Therefore, the line search methods of optimizing non-smooth function form a valuable research in IR. Our results show that the line search method (i.e. MaxAP), though simple, achieves even slightly better results than the perceptron-based method in some test sets. This is largely due to its property of directly optimizing the performance measure (i.e. AP). Similar observations have been reported in the experiments on machine translation [22, 25] and LM [7]. Though the method shows empirical benefits, the lack of theoretical underpinnings (such as optimality and stability) is the major concern. We leave it to further study. An alternative approach to the NSO problem is to use an approximated but smoothed objective function that can be easily optimized, such as the one suggested by Juang et al. [16]. The comparison of those NSO methods forms another area of future work.

6. Conclusions

We have presented a discriminative model for IR, referred to as LDM. It provides a flexible framework to incorporate effectively a wide variety of features, including linguistically motivated features. We have also demonstrated that the feature functions that are derived from component model under the framework of HMM provide useful information for IR. When integrated in LDM, they achieve significant improvements over state-of-the-art language models and the classical probabilistic retrieval model on the task of ad hoc retrieval on six English and Chinese TREC test sets. Thus, our method demonstrates an interesting meld of discriminative and generative models for IR. There are plenty of interesting problems left for future work, as described in Section 5. We are particularly interested in exploring the theoretical underpinnings of the learning algorithms presented in Section 3, such as robustness, scalability and stability, without which we cannot prove that our methods always work well.

7. References

[1] Cohen, W. R. Shapire and Y. Singer. 1999. Learning to order things. *Journal of Artificial Intelligence Research*, 10, pp. 243-270.

[2] Collins, Michael. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with the perceptron algorithm. In: *EMNLP*. pp 1-8.

[3] Crammer, K and Y. Singer. 2001. Pranking with ranking. In: *NIPS*.

[4] Duda, Richard O, Hart, Peter E. and Stork, David G. 2001. *Pattern classification*. John Wiley & Sons, Inc.

[5] Fletcher, R. 1987. *Practical methods of optimization*. John Wiley & Sons, Inc.

[6] Freund, Yoav, Raj Iyer, Robert E. Schapire, and Yoram Singer. 1998. An efficient boosting algorithm for combining preferences. In *ICML'98*, pp. 170-178.

[7] Gao, Jianfeng, Hao Yu, Wei Yuan and Peng Xu. 2005. Minimum sample risk methods for language modeling. In *HLT/EMNLP*.

[8] Gao, Jianfeng, Mu Li, Andi Wu and Changning Huang. 2004. A pragmatic approach to Chinese word segmentation. Tech-Report of Microsoft Research. MSR-TR-2004-123.

[9] Gao, Jianfeng, Jian-Yun Nie, Guangyuan Wu and Guihong Cao. 2004. Dependence language model for information retrieval. In: *SIGIR*, pp. 170-177.

[10] Gao, Jianfeng, Joshua Goodman and Jiangbo Miao. 2001. The use of clustering techniques for language model – application to Asian language. *Computational Linguistics and Chinese Language Processing*. Vol. 6, No. 1, pp 27-60.

[11] Harman, D. K. 1995. Overview of the fourth Text REtrieval Conference (TREC-4). In: *TREC-4*, pp 1-24.

[12] Herbrich, R. T. Graepel and K. Obermayer. 2000. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pp. 115-132. MIT Press, Cambridge, MA.

[13] Joachims, T. 1999. Making large-scale SVM learning practical. In B. Scholkopt, C. Burges and A. Smola, editors, *Advances in Kernel Methods – Support Vector Learning*. MIT Press, Cambridge, MA.

[14] Joachims, T. 2002. Optimizing search engines using clickthrough data. In: *SIGKDD*, pp. 133-143.

[15] Jones, K. S., S. Walker and S. Robertson. 1998. *A probabilistic model of information retrieval: development and status*. Technical Report TR-446, Cambridge University Computer Laboratory.

[16] Juang, Biing-Hwang, Wu Chou and Chin-Hui Lee. 1997. Minimum classification error rate methods for speech recognition. *IEEE Tran. Speech and Audio Processing*. Vol. 5, No. 3. pp. 257-265.

[17] Lafferty, John and Chengxiang Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In: *SIGIR*, pp. 111-119.

[18] Miller, D. H., Leek, T. and Schwartz, R. 1999. A hidden Markov model information retrieval system. In: *SIGIR'99*, pp. 214-221.

[19] Nallapati, R. 2004. Discriminative models for information retrieval. In: *SIGIR*, pp. 67-71.

[20] Nallapati, R. and J. Allan. 2002. Capturing term dependencies using a language model based on sentence trees. In: *CIKM*, pp. 383-390.

[21] Ng, A. N. and M. I. Jordan. 2002. On discriminative vs. generative classifiers: a comparison of logistic regression and naïve Bayes. In: *NIPS*, pp. 841-848.

[22] Och, Franz. 2003. Minimum error rate training in statistical machine translation. In: *ACL*, pp. 160-167.

[23] Ponte, J. and W. B. Croft. 1998. A language modeling approach to information retrieval, In: *SIGIR'98*, pp. 275-281.

[24] Press, W. H., S. A. Teukolsky, W. T. Vetterling andB. P. Flannery. 1992. *Numerical Recipes In C: The Art of Scientific Computing*. New York: Cambridge Univ. Press.

[25] Quirk, C., A. Merezes and C. Cherry. 2005. Dependency tree translation: syntactically informed phrasal SMT. To appear.

[26] Robertson, S. E. and S. Walker. 1994. Some simple effective approximations to the 2-Poisson model for probabilistic weighted retrieval. In: *SIGIR*, pp. 232-241.

[27] Robertson, S. E. and Walker, S. 2000. Microsoft Cambridge at TREC-9: Filtering track. In: *TREC-9*, pp. 361-368.

[28] Song, F. and Croft, B. 1999. A general language model for information retrieval. In: *CIKM'99*, pp. 316–321.

[29] Vapnik, V. N. 1999. *The nature of statistical learning theory*. Springer-Verlag, New York.

[30] Zhai, C., and J. Lafferty. 2002. Two-stage language models for information retrieval. In: *SIGIR*, pp. 49-56.