Learning to Rank with Multi-Aspect Relevance for Vertical Search

Changsung Kang, Xuanhui Wang, Yi Chang, Belle Tseng
Yahoo! Labs
Sunnyvale, CA
{ckang,xhwang,yichang,belle}@yahoo-inc.com

ABSTRACT

Many vertical search tasks such as local search focus on specific domains. The meaning of relevance in these verticals is domain-specific and usually consists of multiple well-defined aspects (e.g., text matching and distance in local search). Thus the overall relevance between a query and a document is a tradeoff between multiple relevance aspects. Such a tradeoff can vary for different types of queries or in different contexts. In this paper, we explore these vertical-specific aspects in the learning to rank setting. We propose a novel formulation in which the relevance between a query and a document is assessed with respect to each aspect, forming the multi-aspect relevance. In order to compute a ranking function, we study two types of learning-based approaches to estimate the tradeoff between these relevance aspects: a label aggregation method and a model aggregation method. Since there are only a few aspects, a minimal amount of training data is needed to learn the tradeoff. We conduct both offline and online test experiments on a local search engine and the experimental results show that our proposed multi-aspect relevance formulation is very promising. The two types of aggregation methods perform more effectively than a set of baseline methods including a conventional learning to rank method.

Categories and Subject Descriptors

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

General Terms

Algorithms

Keywords

Web search, multi-aspect relevance, aggregation

WSDM'12, February 8–12, 2012, Seattle, Washington, USA.

1. INTRODUCTION

As the popularity of search engines has grown, the information needs of end users continue being refined. One of the emerging trends is using vertical search intent. For example, a user may want to find a restaurant near her current location; another user may want to follow the recent development of a breaking event such as the earthquake in Japan. Some recent studies show that at least 20% of Web queries have some local intent [26]. As a result, vertical search engines start attracting more and more attention. For example, many search engines provide specialized vertical search results for local search [1, 3] and for real-time search [8]. Furthermore, vertical search results are often slotted into general Web search results [4, 5]. Thus, designing effective ranking functions for vertical search has become practically important to improve users' search experience.

A natural way to build a vertical search engine is to apply the existing ranking techniques on a vertical. In the TREC conference [2], several specialized tracks such as blog and chemical tracks have been introduced to provide a testbed to study retrieval tasks on these special text collections. The main focus of these tracks is on content-based relevance and most participants extend traditional IR techniques to consider a few task-specific ranking signals. Recently, learning to rank approaches [15, 7, 31] have been studied extensively and shown to be effective to combine many useful signals into a ranking function. To adapt such a technique on a vertical, an intuitive approach is to construct a training data set by collecting a set of queries and documents which belong to the vertical and asking human editors to give a single relevance label between a query and a document. A ranking function thus can be learned for the vertical.

However, we observed that in many verticals, the meaning of relevance is domain-specific and usually consists of multiple well-defined aspects. For example, in real-time search, content matching and temporal recency are both important: A stale result which matches a query perfectly is no longer interesting to end users. In local search, as shown in Figure 1, to find a "restaurant" in "Sunnyvale CA", a satisfactory search result is not only about a dining place (text matching aspect), but also requires the place to be close to the search location (distance aspect) and with good user reviews (reputation aspect). Usually, there is no result which is the best in all these aspects. Ranking the results based on a single aspect such as text matching is not optimal. The optimal ranked list of results needs to tradeoff these different aspects appropriately. In such a vertical, blindly applying the conventional learning to rank approaches by ignoring

Figure 1: An example of local search results

vertical-specific domain knowledge may not be cost-effective for collecting training data: (1) Since there are several pertinent aspects in a vertical, human editors naturally need to consider and tradeoff the relevance from different aspects before making the overall relevance judgement. Thus assessing aspect relevance is a necessary step. (2) Trading off multiple aspects is not trivial since such a tradeoff can vary for different queries or in different contexts. For example, in local search, the reputation aspect can be more important for "restaurants" but the distance aspect can be more important for "banking centers". (3) For different verticals, different aspects are involved and the tradeoff among aspects is vertical-dependent. Collecting training data with overall relevance for a new vertical requires human editors learn how to appropriately tradeoff different aspects.

In this paper, we propose a novel multi-aspect relevance formulation to leverage vertical-specific knowledge in a cost-effective way. Instead of asking editors to provide the overall relevance directly, our key idea is to collect aspect relevance labels and *learn* the tradeoff among them in a *quantitative* way. Specifically, in our formulation, the relevance between a query and a document is only judged with respect to individual aspects. Intuitively, the aspect relevance is more finely specified. Thus it is less dependent on other contexts and can be presumably judged by editors with less effort. To learn a ranking function using our multi-aspect relevance formulation, we study two types of learning-based approaches to tradeoff these aspect relevancies: a label aggregation approach and a model aggregation approach.

- In the label aggregation approach, we first learn an aggregation function which predicts the overall relevance given the aspect relevance labels. After we get the overall relevance labels, conventional learning to rank algorithms can be applied to obtain a ranking function.
- In the model aggregation approach, we first train several aspect-specific ranking models based on the aspect relevance labels. The model aggregation function is then learned to combine the output of these aspect ranking models to generate the final overall relevance score.

Compared with the first approach, an advantage of the second one is that we can use different ranking models for different aspects. Since there are only a few aspects in a vertical, a minimal amount of data is needed to learn either the label aggregation function or the model aggregation function. Furthermore, in our aggregation approaches, a mapping function which assigns numerical values to editorial labels (for example, "exact match", "plausible match" and "no match" for the text match aspect) for each aspect is necessary. Such a function can be defined heuristically but we show that we can automatically learn such a function based on training data. Thus our proposed methods are completely free from heuristics.

The main advantage of our learning-based approaches is that they are vertical-independent and can be easily applied to different vertical search tasks. Specifically in this paper, we focus on learning a generic ranking function for each of the two types of queries in local search: business name queries (e.g., walmart) and category queries (e.g., restaurant). We use a training data set with relative preferences to learn the aggregation functions and study several variants on a large data set from local search. The experimental results show that our proposed methods for the multi-aspect relevance formulation is quite promising. The two types of aggregation methods perform more effectively than a set of baseline methods including a conventional learning to rank method.

The rest of the paper is organized as follows. In Section 2, we introduce related work. We define our problem in Section 3 and describe our methods to aggregate aspect relevance in Section 4. We present our experimental results in Section 5 and conclude our paper in Section 6.

2. RELATED WORK

Modeling relevance is the central topic in the information retrieval community and most of the previous work focuses on the overall relevance. In particular, almost all the evaluation methodologies are based on overall relevance. For example, in the TREC conference [2], benchmark data sets are labelled by human editors with overall relevance. The

relevance labels can be either binary or graded [14]. In the past, many models have been proposed to capture the overall relevance [23, 22, 21, 29, 7]. Most of the existing work treats the overall relevance as a unit and does not study at a finer granularity.

Vertical search engines have become popular recently. For example, in the TREC conference, there are multiple tracks and some tracks such as the blog track and the legal track focus on articles from specific domains. Specific characteristics have been explored for vertical search ranking mechanisms. For example, most participants in the TREC conference designed task-specific ranking features. [11] went beyond single blog articles and studied how to rank blog feeds. [9] advocated a comparison-based ranking scheme to rank items in Twitter-like forums. [28] tried to discover implicit geographic local search intents of queries automatically. [19] proposed to use the geographic information to personalize Web search results. On the top of different verticals, [4, 5] studied how to select appropriate verticals by predicting the vertical intents of different queries based on multiple sources of evidence. Our work focuses on the learning to rank approaches for individual verticals and our multi-aspect relevance formulation is novel for vertical search.

Our work is related to multi-faceted search [27, 30]. The goal of multi-faceted search is to use facet metadata of a domain to help users narrow their search results along different dimensions. In the most recent TREC Blog track 2009 [20], a special track of "faceted blog distillation" was initiated and the task of this track is to find results relevant to a single facet of a query in the blog collection. This task is tied with multi-faceted search in that it is aimed at studying how to rank blog articles after a user selects a facet (e.g., "opinionated" or "factual") to refine the original query. While facets are usually categorical and intended to help explore search results, our definition of aspects is closely related to the notion of relevance and intended to capture partial relevance.

We note that our model aggregation technique is closely related to rank aggregation which includes score-based and rank-based methods [18, 10] and model interpolation [12]. We are not contributing any new techniques to this body of work. Rather, we show that supervised score-based aggregation techniques can be used in our multi-aspect relevance formulation. Our work is also different from multi-label learning [13] where the primary focus is to use label correlation to improve learning accuracy.

A recent related work tries to optimize multiple objectives in the learning to rank framework [25]. It considers multiple label sources and design optimization procedures based on the lambdaMART method. In their work, the priority of different label sources and the combination weights are predefined and thus it is not easy to handle a large number of label sources. In our work, the combination weights between different aspects are learned automatically and our formulation is more scalable with respect to the number of aspects. Furthermore, in our paper, we will compare with a rule-based baseline method which is similar to their *graded measure* in the sense that we predefine the aspect priority and use the labels with lower priority to break the ties of labels with a higher priority.

3. PROBLEM FORMULATION

In this section, we formally define our problem. We first describe a conventional learning to rank approach for vertical searches and then propose our multi-aspect relevance formulation.

```
Given a query and a list of documents, answer the the
following questions for each document.

Question 1: How does this document match the query?
[ ] Exact match
[ ] Plausible match
[ ] No match

Question 2: What is the relative distance between the
document location and the query location?
[ ] Same location
[ ] Reasonable location
[ ] Too far

Question 3: How is the rating from different raters?
[ ] Excellent rating
[ ] Good rating
[ ] Bad rating
```

Question 4: Overall Relevance

Matching	Distance	Reputation	Overall
Exact	Same	Excellent	Perfect
Exact	Same	Good	Excellent
Exact	Same	Bad	Good
Exact or Plausible	Same or Reasonable	-	Fair
No	-	-	Bad

Figure 2: An example of an editorial guideline for local search. Note that the overall relevance question is NOT needed in our aggregation methods. The overall relevance is used only for the conventional method.

3.1 Learning to Rank for Vertical Search

Although vertical search involves multiple aspects such as matching, reputation and distance, we can still apply conventional learning to rank methods. Given a query q, let $\mathcal{D}_q = \{(\mathbf{x}_1, z_1), ..., (\mathbf{x}_n, z_n)\}$ be the training data of n documents where $\mathbf{x}_i \in \mathbb{R}^d$ is the feature vector and z_i is the overall relevance label of the i-th document. In a ranking problem, $\mathcal{D}_q$ is given as input and a permutation τ of $\{1, \ldots, n\}$ is returned as output. $\mathbf{x}_i$ is ranked higher than $\mathbf{x}_j$ if $\tau(\mathbf{x}_i) < \tau(\mathbf{x}_j)$ and this means $\mathbf{x}_i$ is more relevant to q than $\mathbf{x}_j$. Typically, a *ranking function* $f : \mathbb{R}^d \to \mathbb{R}$ is trained and applied to $\mathcal{D}_q$. A permutation or ranking τ is generated by ordering the $f(\mathbf{x}_i)$ in the descending order.

The overall relevance label z_i is a discrete label given by human editors. In this work, we follow a common five-grade labeling scheme: {Perfect, Excellent, Good, Fair, Bad}. To reduce disagreement among editors, it is a common practice that an *editorial guideline* is drawn up. An editorial guideline is essentially a set of rules or heuristics that specify a condition for each grade of the overall relevance label.

In this section, we use local search as an example to describe an editorial guideline. A unique characteristic of local search is that the query intents are to find some locations such as restaurants, hotels, or business centers. This also implies that the users intend to use some services provided by the local businesses. Therefore, a user would prefer a location which is close and at the same time whose reputa-

tion of services is good. For example, to find a restaurant in local search, a satisfactory search result is not only about a dining place (matching aspect), but also requires the place to be close to the user (distance aspect) and to have good user reviews (reputation aspect). Overall, we have found that three aspects, i.e., matching, distance and reputation, are the most important aspects for local information needs.

In Figure 2, we show an example of an editorial guideline for local search. We have several questions which are intended to capture the desired properties of local search. Question 1, 2, and 3 are to capture the matching, distance, and reputation aspects respectively. Each question has a graded judgement which is going to be labelled by editors. Finally, there is an aggregation guideline that specifies a rule for each grade of the overall relevance label. An aggregation guideline is important since it defines the learning targets. It is necessary for most conventional learning to rank tasks, especially which involve multiple aspects of relevance. Without a good aggregation guideline, the training data will have too much noise to train a good ranking function due to disagreement among editors regarding the overall relevance.

For example, given the query "Bank of America, Los Angeles, CA 90018", the result "Bank of America, 2907 Crenshaw Blvd, Los Angeles, CA" has a distance of 1 mile and the average rating from 2 people is 3 out of 5. In this case, the labels for these questions are Exact match, Same location, and Good rating. The overall relevance is Excellent according to the aggregation guideline.

Such a rule-based approach is similar to [25] in the sense that we predefine priorities between aspects and use the labels with a lower priority to break the ties for labels with a higher priority. The drawbacks of the conventional rule-based approach are: (1) Defining the right rules needs deep domain knowledge and thus is non-trivial; (2) The rules are very coarse and cannot capture the true preferences at a finer granularity; (3) This method will not scale well since the complexity of defining rules grows exponentially as the number of aspects increases, though it is feasible for the 3 aspects in our work.

3.2 Multi-Aspect Relevance Formulation

In the conventional learning setting, the intermediate questions regarding aspects are only used to help editors reach the final overall relevance and are usually discarded when training a ranking function. How to leverage these aspect relevance labels effectively is not well explored. In this section, we propose a new learning to rank framework for multi-aspect relevance to tackle the drawbacks of the conventional rule-based overall relevance scheme.

First,we define the following concepts:

DEFINITION 1 (ASPECT RELEVANCE). *Given a query q, a document d, and the k-th aspect, the corresponding aspect relevance is the relevance between q and d with respect to this aspect. An aspect relevance label $\hat{l} \in L_k = \{l_{k,1} \prec, ..., \prec l_{k,n_k}\}$ is used to represent the degree of the relevance where $\prec$ ($\succ$) means the left label is less (more) relevant than the right label.*

For example, in the editorial guideline shown in Figure 2, each intermediate question is to assess a single aspect relevance label. An aspect relevance label is independent of other aspect relevance labels.

DEFINITION 2 (MULTI-ASPECT RELEVANCE). *Given a vertical which has m pertinent aspects, the multi-aspect relevance between a query and a document is an m-dimensional vector with each entry corresponding to an aspect relevance label between the query and the document.*

Each entry in a multi-aspect relevance vector corresponds to an aspect relevance label. This label can be mapped to a numerical value by a mapping function as defined below.

DEFINITION 3 (MAPPING FUNCTION). *A mapping function for the k-th aspect $\phi_k : L_k \to \mathbb{R}$ maps an aspect relevance label to a numerical value. For example, ϕ_1(matching = Plausible match) = 0.5. A mapping function is consistent if $\phi_k(l_{k,i}) > \phi_k(l_{k,j})$ for $l_{k,i} \succ l_{k,j}$. We use Φ as the general notation of the m aspect mapping functions.*

A mapping function can be manually defined or learned. In the following, for ease of exposition, we use notation $\mathbf{y}$ to represent a multi-aspect relevance vector of either labels or values unless otherwise specified.

3.3 Label Aggregation

Given the above definitions, the basic idea of label aggregation is to train a function which can *quantitatively* aggregate the multi-aspect relevance values into an overall relevance value.

DEFINITION 4 (LABEL AGGREGATION FUNCTION). *A* label aggregation function $h : \mathbb{R}^m \to \mathbb{R}$ *is a function that maps a multi-aspect relevance vector $\mathbf{y}$ to an absolute overall relevance value z, i.e., $h(\mathbf{y}) = z$.*

To learn an aggregation function h, we need training data with overall relevance signals, either *absolute* relevance labels or *relative* preferences. In this paper, we focus on the relative preferences and use $\mathcal{P} = \{(\mathbf{x}_i, \mathbf{x}_j) \mid \mathbf{x}_i \succ \mathbf{x}_j\}$ to represent the data where $\mathbf{x}_i \succ \mathbf{x}_j$ denotes that $\mathbf{x}_i$ is preferred to $\mathbf{x}_j$. Since there are only a few aspects in a vertical, training the aggregation function needs a minimal amount of data. After learning an aggregation function, we can then apply it to the large amount of multi-aspect relevance labels and thus generate a large amount of training data with overall relevance.

In summary, we have a large data set with ranking features $\mathbf{x}$ and the corresponding multi-aspect relevance vectors $\mathbf{y}$: $\mathcal{F} = \{(\mathbf{x}, \mathbf{y})\}$ and a small set of relative preference data $\mathcal{P}$. Since there is a one-to-one correspondence between $\mathbf{x}$ and $\mathbf{y}$ in our data, we use either $(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}$ or $(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}$. We have the following steps:

- Learn an aggregation function $h(\mathbf{y})$ (and a mapping function Φ if not manually defined) using $\mathcal{P}$.
- Apply $h(\mathbf{y})$ on $\mathcal{F}$ and generate data set $\hat{\mathcal{F}} = \{(\mathbf{x}, h(\mathbf{y})\}$.
- Train a ranking function f_h using $\hat{\mathcal{F}}$ based on a conventional learning to rank method.

3.4 Model Aggregation

The label aggregation method converts the problem of learning from multi-aspect relevance into a conventional learning to ranking problem. All the ranking features related to different aspects are treated uniformly in this method. The idea of model aggregation is to train an individual ranking model for each aspect. In this section, we propose a *model aggregation* method, which aggregates the output of aspect ranking functions to generate the overall relevance score.

Aspect	Label	Score
Matching	Exact Match	$y_1 = 1.0$
	Plausible Match	$y_1 = 0.5$
	No Match	$y_1 = 0$
Distance	Same Location	$y_2 = 1.0$
	Reasonable Location	$y_2 = 0.5$
	Too Far	$y_2 = 0$
Reputation	Excellent Rating	$y_3 = 1.0$
	Good Rating	$y_3 = 0.5$
	Bad Rating	$y_3 = 0$

Table 1: The aspect relevance mapping function for local search.

DEFINITION 5 (ASPECT RANKING FUNCTION). *An aspect ranking function $f_a : \mathbb{R}^k \to \mathbb{R}$ for an aspect a is a function that maps a feature vector $\mathbf{x}$ to an aspect relevance score.*

DEFINITION 6 (MODEL AGGREGATION FUNCTION). *A model aggregation function $h : \mathbb{R}^m \to \mathbb{R}$ is a function that aggregates the estimated aspect relevance scores into the final overall relevance score.*

In summary, we have the following steps for model aggregation:

- For each aspect a_i, learn an aspect ranking function f_{a_i} based on aspect relevance labels and a mapping function.
- For each $\mathbf{x}$ in $\mathcal{P}$, generate an m-dimensional vector $\mathbf{f}(\mathbf{x}) = [f_{a_1}(\mathbf{x}), ..., f_{a_m}(\mathbf{x})]$.
- Train an aggregation function h based on the feature vector $\mathbf{f}(\mathbf{x})$ and the pairs in $\mathcal{P}$.

The final ranking score is computed as $h(\mathbf{f}(\mathbf{x}))$.

Then the central question is how to learn these aggregation functions (and a mapping function Φ). We explore different formulations in the next section.

4. LEARNING AGGREGATION FUNCTIONS

In this section, we propose different methods to learn aggregation functions based on pairwise preferences.

4.1 Learning Label Aggregation

We propose 2 different approaches: a linear aggregation approach and a joint learning approach.

4.1.1 A Linear Aggregation Method

In this section, we explore a linear model for aggregation by assuming that we have a predefined mapping function. A simple mapping function for an aspect label set L_k can be constructed as

$$\phi_k(l_{k,s}) = \frac{s-1}{n_k - 1} \text{ for } s = 1, \ldots, n_k,$$

For example, a fixed mapping function is given in Table 1. We have an unknown parameter vector $\mathbf{w}$ and the linear function takes the form $h(\mathbf{y}) = \mathbf{w}^T\mathbf{y}$. We use the following loss function on the pairwise training data.

$$\mathcal{L} = \frac{1}{2} \sum_{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}} \big(\max(0, 1 - \mathbf{w}^T\mathbf{y}_i + \mathbf{w}^T\mathbf{y}_j) \big)^2$$

Furthermore, to ensure the monotonicity, we have to constrain $\mathbf{w}$ to be non-negative element-wise:

$$\mathbf{w} \succeq 0.$$

We solve the above optimization problem by a simple gradient descent approach in a similar way to the joint learning model in the next section.

4.1.2 A Joint Learning Method

The above method assumes that we have a predefined mapping function and learn the aggregation function directly on the numeric values of aspect relevance. But such a mapping is in an ad-hoc fashion. In this section, we propose a joint learning model which learns a mapping function and aggregation weights simultaneously. Without loss of generality, for each aspect, we assign 0 to its lowest relevance label and 1 to its highest one, i.e., $\phi_k(l_{k,1}) = 0$ and $\phi_k(l_{k,n_k}) = 1$ for $k = 1, \ldots, m$. Our joint learning method will automatically determine the numeric values for the middle labels.

Formally, our goal is to learn the values of all the labels in $L_1, \ldots, L_m$ and weights $\mathbf{w}$ to minimize the following loss function

$$\mathcal{L} = \frac{1}{2} \sum_{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}} \big(\max(0, 1 - \mathbf{w}^T\Phi(\mathbf{y}_i) + \mathbf{w}^T\Phi(\mathbf{y}_j)) \big)^2$$

Here we use $\mathbf{y}$ to specifically denote an aspect label vector. Note the following differences compared to the linear method: (1) $\Phi(\mathbf{y}_i) = [\phi_1(y_{i,1}), ..., \phi_m(y_{i,m})]^T$ is the vector after applying the mapping function. It is also unknown and needs to be optimized; (2) We have the following additional consistency constraint which ensures that a better label gets a higher score:

$$0 = \phi_k(l_{k,1}) \leq \phi_k(l_{k,2}) ... \leq \phi_k(l_{k,n_k}) = 1 \text{ for } k = 1, \ldots, m$$

It easy to verify that the space with the constraints is convex. However, such a problem is not easy to optimize due to the quadratic terms in the objective function. Since the dimensionality of the problem is not high, we thus propose a gradient descent approach with projection to optimize the objective function. Let $\mathcal{A}_{k,s} = \{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P} \mid y_{i,k} = l_{k,s}, y_{j,k} \neq l_{k,s}\}$ and $\mathcal{B}_{k,s} = \{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P} \mid y_{i,k} \neq l_{k,s}, y_{j,k} = l_{k,s}\}$. The gradient for each variable with respect to the objective function is:

$$\frac{\partial \mathcal{L}}{\partial w_k} = \sum_{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}} \max(0, 1 - \mathbf{w}^T\Phi(\mathbf{y}_i) + \mathbf{w}^T\Phi(\mathbf{y}_j)) \cdot (-\phi_k(y_{i,k}) + \phi_k(y_{j,k}))$$

$$\frac{\partial \mathcal{L}}{\partial \phi_k(l_{k,s})} = \sum_{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{A}_{k,s}} \max(0, 1 - \mathbf{w}^T\Phi(\mathbf{y}_i) + \mathbf{w}^T\Phi(\mathbf{y}_j))(-w_k) + \sum_{(\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{B}_{k,s}} \max(0, 1 - \mathbf{w}^T\Phi(\mathbf{y}_i) + \mathbf{w}^T\Phi(\mathbf{y}_j)) \cdot w_k$$

We use t to denote the iteration in the gradient descent. After iteration t, we project the estimated parameters to the convex space defined by the constraints. We use the norm-2 distance for the projection

$$\min_{\mathbf{w}, \phi_k(l_{k,s})} ||\mathbf{w} - \mathbf{w}^{(t)}||^2 + \sum_{k,s} |\phi_k(l_{k,s}) - \phi_k(l_{k,s})^{(t)}|^2$$

$$\text{s.t. } 0 = \phi_k(l_{k,1}) \leq \phi_k(l_{k,2}) ... \leq \phi_k(l_{k,n_k}) = 1$$

$$\mathbf{w} \succeq 0$$

The projection can be efficiently solved since it is a standard quadratic programming problem [24].

4.2 Learning Model Aggregation

In this section, we propose another method for our multi-aspect relevance formulation, the model aggregation method, which is formulated in Section 3.4. In this method, we first learn an aspect ranking function f_{a_i} for each aspect a_i and use a supervised linear model as our aggregation function

$$h(\mathbf{f}(\mathbf{x})) = \mathbf{w}^T \mathbf{f}(\mathbf{x}).$$

To learn an aspect ranking function f_{a_i} using a method such as GBRank [31], we need to assign numerical values to aspect labels, e.g., by a mapping function. We have proposed to automatically learn a mapping function in the joint learning method in the previous section and thus we use the obtained mapping function Φ to convert the aspect labels.

To learn the parameter $\mathbf{w}$, we use the following loss function:

$$\mathcal{L} = \frac{1}{2} \sum_{(\mathbf{x}_i, \mathbf{x}_j) \in \mathcal{P}} \big(\max(0, 1 - \mathbf{w}^T \mathbf{f}(\mathbf{x}_i) + \mathbf{w}^T \mathbf{f}(\mathbf{x}_j)) \big)^2$$

where $\mathbf{f}(\mathbf{x}) = [f_{a_1}(\mathbf{x}), ..., f_{a_m}(\mathbf{x})]$. This model is very similar to the linear model in the label aggregation methods. The difference is that we replace the labels with the output of aspect ranking functions.

Compared to the label aggregation methods, there are two benefits of the model aggregation method. First, we can use a different type of model for each aspect ranking function. This is desirable since aspects are not necessarily homogeneous. For example, the matching aspect can be complex and thus needs a ranking function with high model complexity, but a simple regression model may be good enough for the distance aspect. In particular, we use a gradient-boosted decision tree model GBRank [31] for the matching aspect which shows excellent performance in learning such a function. On the other hand, we use linear regression models for distance and reputation aspects. Hence, the combination of various types of models for different aspects gives great flexibility to the final ranking function.

Also, in the model aggregation method, we can exploit preference data inferred from other sources such as user clicks to learn the aggregation function. Unlike the label aggregation methods, each document in $\mathcal{P}$ does not need aspect relevance labels $\mathbf{y}$ and we only need $\mathbf{f}(\mathbf{x})$, the output of aspect ranking functions, to learn the aggregation function. This provides flexibility to quickly adapt the aggregation function to different contexts. For example, this makes it possible to provide personalized search rankings: We may collect preference data $\mathcal{P}$ for a user u and use it to learn a user-specific aggregated function $\mathbf{w}_u^T \mathbf{f}(\mathbf{x})$. Note that we do not need to learn aspect ranking functions for each user since each aspect ranking function should be common among users but the tradeoff between aspects depends on personal preference. Similarly, we can provide customized search rankings for different search applications. For example, in mobile local search, the distance aspect may be more important than in desktop local search. We can easily build a separate ranking function for mobile local search using the preference data obtained from user clicks in mobile local search.

		#query	#listing
Category Queries	Training	4211	70701
	Test	1055	17675
	Total	5266	88376
Business Name Queries	Training	6966	76343
	Test	1739	18550
	Total	8705	94893

Table 2: Statistics of multi-aspect relevance data.

5. EXPERIMENTS

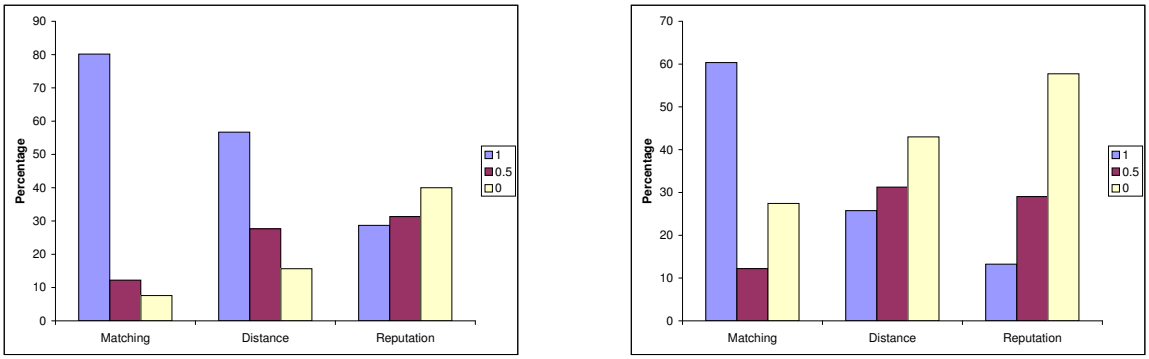
In this section, we present experimental results to validate our approaches. The main objective of our experiments is to study the effectiveness of our multi-aspect relevance formulation and the proposed aggregation methods for local search. We report both offline and online test results in this section.

5.1 Data Sets

The data sets we use are from a commercial local search engine where a document is called "business listing" or "listing" for short. We follow the editorial guideline similar to the guideline in Figure 2 to obtain the training data. Specifically, each query in our data has a location associated (e.g., "Target Sunnyvale CA"). For each (query, listing) pair, we consider 3 aspects: matching, distance and reputation as discussed in Section 3.1 and ask editors to provide 3 aspect relevance labels and an overall relevance label according to the editorial guideline. Note that in our aggregation methods, we do not use the overall relevance labels given by human editors. Table 2 shows the statistics of this data set. In particular, we have two types of queries: category queries and business name queries. A category query such as "Chinese restaurant Los Angeles, CA 90018" is similar to an informational query and can be broadly matched by any Chinese restaurant, while a business name query such as "Bank of America, Los Angeles, CA 90018" is more like a navigational query and can only be matched by Bank of America branches. Intuitively, the relative importance of each aspect for these two types of queries can be potentially different.

In Figure 3, we show the distribution of the labels for different types of queries and different aspects. We can see clear differences between them. The differences of the statistics regarding matching and distance aspects are due to the different densities of businesses for the two query categories: There are more matching businesses for category queries than business name queries (e.g., typically, there are more restaurants than Target stores in a city.). The high percentage of the "Bad rating" label for the reputation aspect for business name queries is due to two reasons: (1) When there is no user review, it is considered as a "Bad rating" in our guideline. (2) About 50% of the business name queries are *chain* queries such as "Walmart" and users typically do not bother review the chain stores. Indeed we will show that the reputation aspect is not particularly important for business name queries.

We obtain the overall relative preference using a side by side comparison as follows: given any query, we randomly sample a pair of documents and put them side by side. The position of the two documents are randomly shuffled to avoid



(a) Category Queries

(b) Business Name Queries

Figure 3: Distribution of aspect relevance labels. 1, 0.5 and 0 correspond to the aspect labels in Table 1.

	#Training $\mathcal{P}_{train}$	#Test $\mathcal{P}_{test}$
Category Queries	549	493
Name Queries	445	457

Table 3: Statistics of overall relative preference data sets obtained by side by side comparison.

any kind of bias. We then ask editors to judge which one is better than the other. The obtained overall training signals are thus relative preferences. Previous studies have shown that relative preferences are more reliable than absolute judgements in many scenarios [6, 16]. This is very critical when the training data is small. However, it is expensive to obtain a large amount of such data. Fortunately, our experiment results show that only a small amount of such data is needed to obtain highly accurate aggregation functions. Table 3 summarizes the statistics of this data set. We split this data into training ($\mathcal{P}_{train}$) and test ($\mathcal{P}_{test}$) to evaluate our aggregation functions.

5.2 Ranking Algorithms

We compare the following ranking algorithms.

- **Rule**: A traditional one overall relevance label scheme described in Section 3.1.
- **Ed-overall**: A ranking function trained directly using editorial pairwise preference data $\mathcal{P}_{train}$.
- **Click**: A ranking function trained using pairwise preference data induced from a click model [17].
- **Linear**: The linear aggregation function described in Section 4.1.1.
- **Joint**: The joint learning model described in Section 4.1.2.
- **ModAgg**: The model aggregation method described in Section 4.2.

Rule, Ed-overall and Click serve as baselines. Rule, a traditional one overall relevance label scheme described in Section 3.1 is similar to the *graded measure* used in [25] in the sense that the secondary labels are used to break the ties of the primary labels. Ed-overall and Click are other baselines to show the benefit of our multi-aspect relevance. They do not use aspect relevance labels but only use pairwise preference data as the training data. Both are learned using the GBRank model [31]. For the Click method, we obtain *SkipAbove* and *SkipNext* [17] pairs from click logs. The click-based training data is easy to obtain but it is noisy. We have 1,441,975 and 58,782 click-based training pairs for category and business name queries respectively. In all our label aggregation methods, the final ranking functions are learned using the GBRank model [31].

5.3 Offline Experimental Results

We report our offline experiment results based on the data sets with editorial labels.

5.3.1 Evaluation Metrics

To evaluate aggregation functions, we consider two types of pair accuracy with respect to the test data set $\mathcal{P}_{test}$. (1) Label aggregation accuracy: how accurate is an aggregation function to aggregate the multi-aspect relevance to generate the overall relevance? This accuracy is only applied to label aggregation methods. (2) Ranking accuracy: how effective is a ranking function trained using either label or model aggregation methods?

Let h be a label aggregation function and f be a final ranking function trained using either label aggregation or model aggregation methods. The label aggregation accuracy of h is:

$$\frac{|\{(\mathbf{y}_i, \mathbf{y}_j) \mid h(\mathbf{y}_i) > h(\mathbf{y}_j), (\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}_{test}\}|}{|\mathcal{P}_{test}|} \quad (1)$$

and the ranking accuracy of f is:

$$\frac{|\{(\mathbf{y}_i, \mathbf{y}_j) \mid f(\mathbf{x}_i) > f(\mathbf{x}_j), (\mathbf{y}_i, \mathbf{y}_j) \in \mathcal{P}_{test}\}|}{|\mathcal{P}_{test}|}. \quad (2)$$

Note that an NDCG-like metric seems to be possible to compare the ranking accuracy of different aggregation methods since a label aggregation function can generate absolute overall relevance values. However, different label aggregation methods can generate different overall values for the same (query, listing) pair and thus we do not have a common

	Category Queries	Business Name Queries
Rule	0.640	0.650
Linear	**0.825**	**0.926**
Joint	**0.825**	**0.932**

Table 4: Evaluation of aggregation functions on label aggregation accuracy. Linear, and Joint are significantly better than Rule (p-value < 0.01).

	Category Queries	Business Name Queries
Rule	0.614	0.788
Ed-overall	0.575	0.841
Click	0.638	0.841
Linear	**0.750**	0.879
Joint	**0.760**	0.871
ModAgg	**0.769**	**0.917**

Table 5: Evaluation of aggregation functions on ranking accuracy. Statistically significant differences (p-value < 0.01) compared to Rule are highlighted in bold.

ground to compare them. Hence, we use the pair accuracy as a more isolated and unbiased metric for evaluation.

5.3.2 Results on Label Aggregation Accuracy

Table 4 shows the comparison of different aggregation functions based on the label aggregation accuracy which is defined in Eq. (1). In this experiment, we only use the data in Table 3. We learn the aggregation functions for each type of queries separately. Table 4 shows the results on category queries and business name queries respectively. From this table, we have the following observation. (1) The Rule method performs much worse than all the learning based methods for both types of queries. For example, for category queries, the rule-based method has about 64% accuracy and all other methods have 82.5% accuracy. This shows that the rules are too coarse to capture the desired tradeoff. (2) The Linear and Joint methods perform equally well on both category and business name queries. The pair accuracy is 83% on category queries and 93% on business name queries. This demonstrates high consistency between the aggregated overall relevance and the true one. (3) Comparing the two types of queries, we can see higher accuracy for business name queries than category queries. This is expected since the relevance for category queries is naturally more complex than business name queries.

5.3.3 Results on Ranking Accuracy

Table 5 shows the comparison of the ranking accuracy for both label and model aggregation methods. For a label aggregation method, we first use the training data in Table 3 to learn the label aggregation function and then apply it to obtain the overall relevance for the set of training queries in Table 2. We then test the ranking function using $\mathcal{P}_{test}$. The pair ranking accuracy is defined in Eq. (2). For the model aggregation method, we use the training data with aspect relevance label data in Table 2 to learn the aspect ranking functions $\mathbf{f}$ and then use the training data in Table 3 to learn the model aggregation function h. We then apply the overall ranking function $h(\mathbf{f}(\mathbf{x}))$ on the test data. We have the following observations: (1) For category queries, all the learning based methods are significantly better than the Rule method. The ModAgg method is slightly better than all other methods. (2) However, for business name queries, only the ModAgg method is significantly better than the Rule method, while the improvement of the Linear and Joint methods over the Rule method is not significant with respect to p-value < 0.01. One possible reason is that business name queries are relative easy and all the methods have achieved high accuracy.

Overall, we can see that our learning methods are quite effective and the model aggregation method is the most stable one for both classes of queries.

Our learning methods improve not only ranking accuracy but also the efficiency of the training procedure compared to the Rule method. In our learning methods, training data collection can be done more efficiently since we do not need overall relevance labels but only need aspect relevance labels and a small amount of data $\mathcal{P}_{train}$.

5.3.4 Benefit of Multi-Aspect Relevance

Table 5 also shows the benefit of our multi-aspect relevance formulation compared to approaches that do not leverage aspect relevance in the learning process. Both Ed-overall and Click only use pairwise preference data as the training data ignoring aspect relevance. We can see that Ed-overall performs much worse compared to the ModAgg method. The reason why Ed-overall performs worse is mainly due to the limited amount of the training data. The Click method performs worse mainly because it is noisy. Our method leverages the aspect relevance and thus can utilize the small amount of Ed-overall data ($\mathcal{P}_{train}$) more effectively.

5.3.5 Comparison of Aspect Importance

The Linear, Joint and ModAgg methods generate a weight vector $(w_{Matching}, w_{Distance}, w_{Reputation})$ for aspects as output. These weights indicate the relative importance of aspects in the overall relevance. For category queries, we have

$$w_{Matching} \gg w_{Reputation} \gg w_{Distance}.$$

For business name queries, we have

$$w_{Matching} \gg w_{Distance} > w_{Reputation}.$$

This result confirms our intuition that the reputation aspect is more important than the distance aspect for category queries and vice versa for business name queries. This shows that different types of queries work best with different aggregation functions.

In our current work, we mainly focus on two broad types of queries. This results show that there is probably a large room for improvement if we can make our aggregation functions query-dependent or personalized. We leave this as future work.

5.4 Online Experimental Results

Ranking functions can be compared in terms of pair accuracy in the offline setting, but they can also be compared on how users interact with the search results in the online setting. In this section, we report our online results.

5.4.1 Experiment Design

In our online experiments, we conduct "bucket tests" during a certain period to compare different ranking algorithms

in a commercial local search engine. The bucket is created based on user cookies. A cookie is assigned to a fixed bucket in our test period. Each bucket corresponds to a small percentage of user population who use the local search engine. In different buckets, we show search results of different ranking algorithms. If a ranking algorithm is better than another, we would expect that the user experience metric is better.

It can be seen that online experiments are expensive and we cannot test many different algorithms. In our experiments, we were able to test only two functions during the same time period in the commercial local search engine. During two weeks, we compared Rule and Linear. During another two weeks, we compared Linear and ModAgg. To compare two functions, we use the click-through rate (CTR) as our user experience metric. Specifically, we use CTR for top i positions and denote it as CTRi:

$$\text{CTR}i = \frac{\sum_{k=1}^{k=i} \text{clicks at position } k}{\sum_{k=1}^{k=i} \text{views at position } k}$$

Due to confidential reasons, we do not report the exact CTR but report a normalized CTR over a fixed number.

5.4.2 Results

In Figure 4, we report the daily trends of the click-through rate (CTR) of ranking functions. The CTRs naturally fluctuate on different days. We can see that Linear is consistently better than Rule and ModAgg outperforms Linear during the whole period. The average CTR5s of Rule and Linear during the first test period are 0.315 and 0.325 respectively. The average CTR5s of Linear and ModAgg during the second test period are 0.308 and 0.314 respectively. These differences are statistically significant (p-value < 0.01).

This result is consistent with our offline experimental results and shows that our multi-aspect relevance framework outperforms the traditional one overall relevance scheme. Also, it demonstrates that the model aggregation method is more effective than the label aggregation methods. Thus, using different ranking models for different aspects is more suitable for multi-aspect relevance aggregation.

6. CONCLUSIONS AND FUTURE WORK

The meaning of relevance in vertical search is domain-specific and usually consists of multiple aspects. In this paper, we proposed a multi-aspect relevance formulation for vertical search in the learning to rank setting. In our formulation, the relevance between a query and a document is assessed with respect to each aspect, forming multi-aspect relevance. In order to learn a ranking function, we studied two types of learning-based approaches to estimate the tradeoff between the multiple aspects, a label aggregation method and a model aggregation method. Then a ranking function is learned based on the multi-aspect relevance formulation. Since there are only a few aspects, a minimal amount of training data is needed to learn the tradeoff. We studied several methods to learn aggregation functions and conducted experiments on local search engine data. Our experimental results show our multi-aspect relevance formulation is promising. The proposed aggregation approaches are very effective to learn the tradeoff between different aspects.

Our work can be extended as follows. First, we proposed to learn the ranking functions and aggregation functions separately in this paper. How to learn them jointly is a promising direction. Second, our methods rely on manually identified aspects and thus how to automatically discover all important aspects given a vertical is worth studying. Third, our multi-aspect relevance formulation provides a new base for studying how to learn context sensitive or personalized ranking functions. For example, it is possible to dynamically update our aggregation function based on the short-term user interaction history.

7. REFERENCES

[1] Google Local. http://local.google.com/.

[2] TREC. http://trec.nist.gov/.

[3] Yahoo Local. http://local.yahoo.com/.

[4] J. Arguello, F. Diaz, J. Callan, and J.-F. Crespo. Sources of evidence for vertical selection. In *SIGIR*, pages 315–322, 2009.

[5] J. Arguello, F. Diaz, and J.-F. Paiement. Vertical selection in the presence of unlabeled verticals. In *SIGIR*, pages 691–698, 2010.

[6] R. K. Belew. *Finding Out About*. Cambridge Univ. Press, 2000.

[7] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd international conference on Machine learning*, pages 89–96, 2005.

[8] N. Carr. Real-time search. *Communication of ACM*, 2010. http://www.technologyreview.com/computing/25079/.

[9] A. Das Sarma, A. Das Sarma, S. Gollapudi, and R. Panigrahy. Ranking mechanisms in twitter-like forums. In *Proceedings of the third ACM international conference on Web search and data mining*, WSDM, pages 21–30, 2010.

[10] C. Dwork, R. Kumar, M. Naor, and D. Sivakumar. Rank aggregation methods for the web. In *Proceedings of the 10th international conference on World Wide Web*, WWW '01, pages 613–622, 2001.

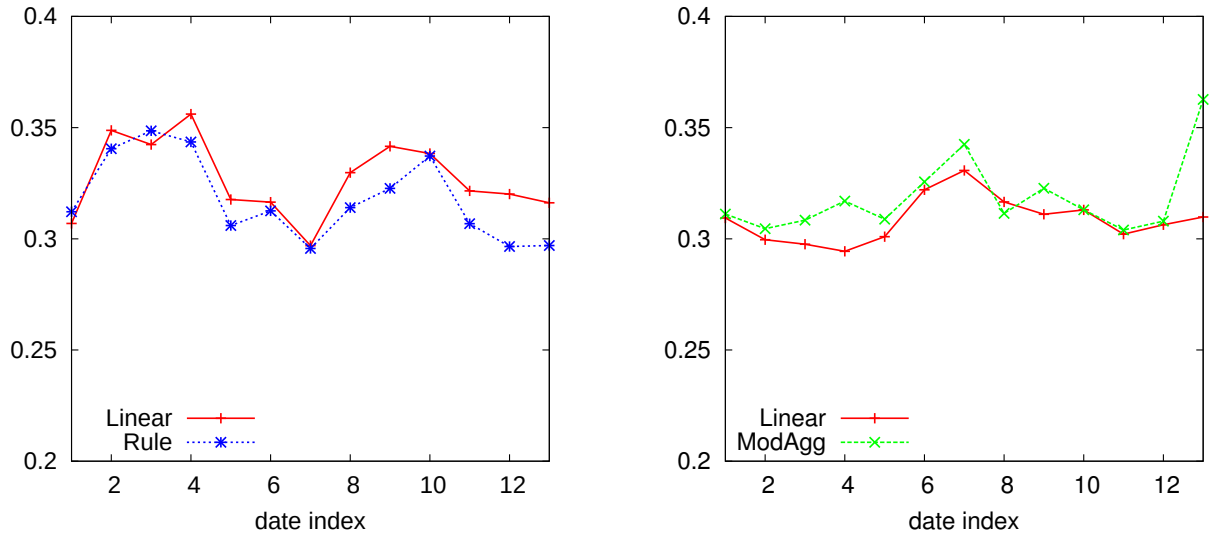
[11] J. L. Elsas, J. Arguello, J. Callan, and J. G. Carbonell. Retrieval and feedback models for blog feed search. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 347–354, 2008.

[12] J. Gao, Q. Wu, C. Burges, K. Svore, Y. Su, N. Khan, S. Shah, and H. Zhou. Model adaptation via model interpolation and boosting for Web search ranking. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 505–513, August 2009.

[13] N. Ghamrawi and A. McCallum. Collective multi-label classification. In *CIKM*, pages 195–200, 2005.

[14] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20:422–446, October 2002.

[15] T. Joachims. Optimizing search engines using clickthrough data. In *KDD '02: Proceedings of the eighth ACM SIGKDD*, pages 133–142, New York, NY, USA, 2002. ACM Press.



(a) CTR5 comparison of Rule and Linear

(b) CTR5 comparison of Linear and ModAgg

Figure 4: The click-through rate (CTR) comparisons based on online experiments for a commercial local search engine. CTRs are normalized not to show absolute values. Each comparison is done for a different test period due to online experiment constraints.

[16] T. Joachims, L. Granka, B. Pan, H. Hembrooke, and G. Gay. Accurately interpreting clickthrough data as implicit feedback. In *Proceedings of ACM SIGIR 2005*, pages 154–161, New York, NY, USA, 2005. ACM Press.

[17] T. Joachims, L. Granka, B. Pan, H. Hembrooke, F. Radlinski, and G. Gay. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2), 2007.

[18] Y.-T. Liu, T.-Y. Liu, T. Qin, Z.-M. Ma, and H. Li. Supervised rank aggregation. In *Proceedings of the 16th international conference on World Wide Web*, WWW, pages 481–490, 2007.

[19] Y. Lu, F. Peng, X. Wei, and B. Dumoulin. Personalize web search results with user's location. In *SIGIR*, pages 763–764, 2010.

[20] C. Macdonald, I. Ounis, and I. Soboroff. Overview of the trec 2009 blog track. 2009.

[21] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 275–281, 1998.

[22] S. E. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *SIGIR*, pages 232–241, 1994.

[23] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[24] S.Boyd and L.Vandenberghe. *Convex Optimization*. Cambridge University Press, 2004.

[25] K. M. Svore, M. Volkovs, and C. J. C. Burges. Learning to rank with multiple objective functions. In *WWW*, pages 367–376, 2011.

[26] P. Venetis, H. Gonzalez, C. S. Jensen, and A. Y. Halevy. Hyper-local, directions-based ranking of places. *PVLDB*, 4(5):290–301, 2011.

[27] K.-P. Yee, K. Swearingen, K. Li, and M. Hearst. Faceted metadata for image search and browsing. In *CHI*, pages 401–408, 2003.

[28] X. Yi, H. Raghavan, and C. Leggetter. Discovering users' specific geo intention in web search. In *WWW*, pages 481–490, 2009.

[29] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of ACM SIGIR 2001*, pages 334–342, 2001.

[30] L. Zhang and Y. Zhang. Interactive retrieval based on faceted feedback. In *Proceeding of the 33rd international ACM SIGIR*, pages 363–370, 2010.

[31] Z. Zheng, H. Zha, T. Zhang, O. Chapelle, K. Chen, and G. Sun. A general boosting method and its application to learning ranking functions for web search. In *Advances in Neural Information Processing Systems 20*, pages 1697–1704. MIT Press, 2008.