Learning to Rank for Information Retrieval

Learning to Rank for Information Retrieval

Tie-Yan Liu

Microsoft Research Asia, Sigma Center
Beijing, 100190
P. R. China
Tie-Yan.Liu@microsoft.com

now
the essence of knowledge
Boston – Delft

Foundations and Trends$^{®}$ in Information Retrieval

Published, sold and distributed by:
now Publishers Inc.
PO Box 1024
Hanover, MA 02339
USA
Tel. +1-781-985-4510
www.nowpublishers.com
sales@nowpublishers.com

Outside North America:
now Publishers Inc.
PO Box 179
2600 AD Delft
The Netherlands
Tel. +31-6-51115274

The preferred citation for this publication is T.-Y. Liu, Learning to Rank for Information Retrieval, Foundation and Trends$^{®}$ in Information Retrieval, vol 3, no 3, pp 225–331, 2009

ISBN: 978-1-60198-244-5

Foundations and Trends® in Information Retrieval

Volume 3 Issue 3, 2009

Editorial Scope

Foundations and Trends® in Information Retrieval will publish survey and tutorial articles in the following topics:

- Applications of IR
- Architectures for IR
- Collaborative filtering and recommender systems
- Cross-lingual and multilingual IR
- Distributed IR and federated search
- Evaluation issues and test collections for IR
- Formal models and language models for IR
- IR on mobile platforms
- Indexing and retrieval of structured documents
- Information categorization and clustering
- Information extraction
- Information filtering and routing
- Metasearch, rank aggregation and data fusion
- Natural language processing for IR
- Performance issues for IR systems, including algorithms, data structures, optimization techniques, and scalability
- Question answering
- Summarization of single documents, multiple documents, and corpora
- Text mining
- Topic detection and tracking
- Usability, interactivity, and visualization issues in IR
- User modelling and user studies for IR
- Web search

Information for Librarians

Foundations and Trends® in Information Retrieval, 2009, Volume 3, 4 issues. ISSN paper version 1554-0669. ISSN online version 1554-0677. Also available as a combined paper and online subscription.

Foundations and Trends® in
Information Retrieval
Vol. 3, No. 3 (2009) 225–331
© 2009 T.-Y. Liu
DOI: 10.1561/1500000016

Learning to Rank for Information Retrieval

Tie-Yan Liu

Microsoft Research Asia, Sigma Center, No. 49, Zhichun Road, Haidian District, Beijing, 100190, P. R. China, Tie-Yan.Liu@microsoft.com

Abstract

Learning to rank for Information Retrieval (IR) is a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance. Many IR problems are by nature ranking problems, and many IR technologies can be potentially enhanced by using learning-to-rank techniques. The objective of this tutorial is to give an introduction to this research direction. Specifically, the existing learning-to-rank algorithms are reviewed and categorized into three approaches: the pointwise, pairwise, and listwise approaches. The advantages and disadvantages with each approach are analyzed, and the relationships between the loss functions used in these approaches and IR evaluation measures are discussed. Then the empirical evaluations on typical learning-to-rank methods are shown, with the LETOR collection as a benchmark dataset, which seems to suggest that the listwise approach be the most effective one among all the approaches. After that, a statistical ranking theory is introduced, which can describe different learning-to-rank algorithms, and be used to analyze their query-level generalization abilities. At the end of the tutorial, we provide a summary and discuss potential future work on learning to rank.

Contents

1

Introduction

With the fast development of the Web, every one of us is experiencing a flood of information. It was estimated that there are about 25 billion pages on the Web as of October 2008,[1] which makes it generally impossible for common users to locate desired information by browsing the Web. As a consequence, efficient and effective Information Retrieval (IR) has become more important than ever, and search engines (or IR systems) have become an essential tool for many people.

Ranking is a central problem in IR. Many IR problems are by nature ranking problems, such as document retrieval, collaborative filtering [58], key term extraction [30], definition finding [130], important email routing [23], sentiment analysis [94], product rating [36], and anti Web spam [56]. In this tutorial, we will mainly take document retrieval as an example. Note that document retrieval is not a narrow task. Web pages, emails, academic papers, books, and news articles are just a few of the many examples of documents. There are also many different ranking scenarios for document retrieval of our interest.

[1] http://www.worldwidewebsize.com/

Scenario 1: Rank the documents purely according to their relevance with regards to the query.

Scenario 2: Consider the relationships of similarity [117], website structure [35], and diversity [139] between documents in the ranking process. This is also referred to as relational ranking [103].

Scenario 3: Aggregate several candidate ranked lists to get a better ranked list. This scenario is also referred to as meta search [7]. The candidate ranked lists may come from different index servers or different vertical search engines, and the target ranked list is the final result presented to users.

Scenario 4: Find whether and to what degree a property of a webpage influences the ranking result. This is referred to as "reverse engineering" in search engine optimization (SEO).[2]

To tackle the problem of document retrieval, many heuristic ranking models have been proposed and used in IR literature. Recently, given the amount of potential training data available, it has become possible to leverage Machine Learning (ML) technologies to build effective ranking models. Specifically, we call those methods that learn how to combine predefined features for ranking by means of discriminative learning "learning-to-rank" methods.

In recent years, learning to rank has become a very hot research direction in IR, and a large number of learning-to-rank algorithms have been proposed, such as [9, 13, 14, 16, 17, 26, 29, 33, 34, 47, 49, 59, 63, 73, 78, 90, 97, 99, 102, 114, 119, 122, 129, 134, 136]. We foresee that learning to rank will have an even bigger impact on IR in the future.

When a research area comes to this stage, several questions naturally arise.

- To what respect are these learning-to-rank algorithms similar and in which aspects do they differ? What are the strengths and weaknesses of each algorithm?
- Empirically, which of those many learning-to-rank algorithms perform the best? What kind of datasets can be used to make fair comparison among different learning-to-rank algorithms?

[2] http://www.search-marketing.info/newsletter/reverse-engineering.htm

- Theoretically, is ranking a new ML problem, or can it be simply reduced to existing ML problems? What are the unique theoretical issues for ranking that should be investigated?
- Are there many remaining issues regarding learning to rank to study in the future? What are they?

The above questions have been brought to the attention of the IR and ML communities in a variety of contexts, especially during recent years. The aim of this tutorial is to review the recent work that attempts to answer these questions. Needless to say, the comprehensive understanding of the task of ranking in IR is the key to finding the right answers. Therefore, we will first give a brief introduction of ranking in IR, and then formalize the problem of learning to rank so as to set the stage for the upcoming detailed reviews.

1.1 Ranking in IR

In this subsection, we briefly review representative ranking models in IR literature, and introduce how these models are evaluated.

1.1.1 Conventional Ranking Models for IR

In IR literature, many ranking models have been proposed [8]; they can be roughly categorized as query-dependent models and query-independent models.

Query-dependent models

The early models retrieve documents based on the occurrences of the query terms in the documents. Examples include the *Boolean model* [8]. Basically these models can only predict whether a document is relevant to the query or not, but cannot predict the degree of relevance.

To further model the relevance degree, the *Vector Space model* (VSM) was proposed [8]. Both documents and queries are represented as vectors in a Euclidean space, in which the inner product of two vectors can be used to measure their similarities. To get an effective vector representation of queries and documents, TF–IDF weighting has been widely used.[3] The TF of term t in a vector is defined as the normalized number of its occurrences in the document, and the IDF of it is defined as follows:

$$\mathrm{IDF}(t) = \log \frac{N}{n(t)}, \tag{1.1}$$

where N is the total number of documents in the collection, and $n(t)$ is the number of documents containing term t.

While VSM implies the assumption on the independence between terms, *Latent Semantic Indexing* (LSI) [37] tries to avoid this assumption. In particular, Singular Value Decomposition (SVD) is used to linearly transform the feature space and thus a "latent semantic space" is generated. Similarity in this new space is then used to define the relevance between queries and documents.

As compared with the above, models based on the probabilistic ranking principle [83] garnered more attention and achieved more success in past decades. The famous ranking models like *BM25* [111][4] and *language model for IR* can both be categorized as probabilistic ranking models.

The basic idea of BM25 is to rank documents by the log-odds of their relevance. Actually BM25 is not a single model, but it defines a whole family of ranking models, with slightly different components and parameters. One of the popular instantiations of the model is as follows.

Given query q, containing terms $t_1,\ldots,t_M$, the BM25 score of document d is computed as below:

$$\mathrm{BM25}(d,q) = \sum_{i=1}^{M} \frac{\mathrm{IDF}(t_i) \cdot \mathrm{TF}(t_i,d) \cdot (k_1+1)}{\mathrm{TF}(t_i,d) + k_1 \cdot \left(1 - b + b \cdot \frac{\mathrm{LEN}(d)}{\mathrm{avdl}}\right)}, \tag{1.2}$$

where $\mathrm{TF}(t,d)$ is the term frequency of t in document d; $\mathrm{LEN}(d)$ is the length (number of words) of document d; avdl is the average document length in the text collection from which documents are drawn; k_1 and

[3] Note that there are many different definitions of TF and IDF in IR literature. Some are purely based on the frequency and the others include smoothing or normalization [116]. Here we just give some simple examples to illustrate the main idea.

[4] The name of the actual model is BM25. However, it is usually referred to as "OKapi BM25", since the OKapi system was the first system to implement this model.

b are free parameters; IDF(t) is the IDF weight of term t, computed by using Equation (1.1), for example.

Language model for IR [96] is an application of the statistical language model on IR. A statistical language model assigns a probability to a sequence of terms. When used in IR, a language model is associated with a document. With query q as input, documents are ranked based on the query likelihood, or the probability that the document's language model would generate the terms in the query (i.e., $P(q\,|\,d)$). By further assuming the independence among terms, one has $P(q\,|\,d) = \prod_{i=1}^{M} P(t_i\,|\,d)$, if query q contains terms $t_1,\ldots,t_M$.

To learn the document's language model, a maximum likelihood method is used. As in many maximum likelihood methods, the issue of smoothing the estimate is critical. Usually a background language model estimated using the entire collection is used for this purpose. Then, the document's language model can be constructed as follows:

$$p(t_i\,|\,d) = (1-\lambda)\frac{\mathrm{TF}(t_i,d)}{\mathrm{LEN}(d)} + \lambda p(t_i\,|\,C), \tag{1.3}$$

where $p(t_i\,|\,C)$ is the background language model for term t_i, and $\lambda \in [0,1]$ is a smoothing factor.

There are many variants of language model for IR, some of them even go beyond the query likelihood retrieval model (e.g., the models based on K–L divergence [140]). We will not introduce more about them, and readers are encouraged to read the tutorial authored by Zhai [138].

In addition to the above examples, many other models have also been proposed to compute the relevance between a query and a document. Some of them [118] take the proximity of the query terms into consideration, and some others consider the relationship between documents in terms of content similarity [117], hyperlink structure [113], website structure [101], and topic diversity [139].

Query-independent models

In IR literature, there are also many models that rank documents based on their own importance. We will take PageRank [92] as an example for illustration. This model is particularly applicable to Web search because it makes use of the hyperlink structure of the Web for ranking.

PageRank uses the probability that a surfer randomly clicking on links will arrive at a particular webpage to rank the web pages. In the general case, the PageRank value for any page d_u can be expressed as:

$$\mathrm{PR}(d_u) = \sum_{d_v \in B_u} \frac{\mathrm{PR}(d_v)}{U(d_v)}. \tag{1.4}$$

That is, the PageRank value for page d_u is dependent on the PageRank values for each page d_v out of the set B_u (containing all pages linking to page d_u), divided by $U(d_v)$, the number of outlinks from page d_v.

To get a meaningful solution to Equation (1.4), a smoothing term is introduced. When the random surfer walks on the link graph, she/he does not necessarily always follow the existing hyperlinks. There is a small probability that she/he will jump to any other page uniformly. This small probability can be represented by $(1 - \alpha)$, where α is called the damping factor. Accordingly, PageRank is refined as follows:

$$\mathrm{PR}(d_u) = \alpha \sum_{d_v \in B_u} \frac{\mathrm{PR}(d_v)}{U(d_v)} + \frac{(1-\alpha)}{N}, \tag{1.5}$$

where N is the total number of pages on the Web.

There is much work discussing the theoretical properties, variations, and efficient implementations of PageRank. Furthermore, there are also many other link analysis algorithms, such as Hyperlink Induced Topic Search (HITS) [72] and TrustRank [57]. Some of these algorithms even leverage the content or topic information in the process of link analysis [91].

1.1.2 Query-level Position-based Evaluations in IR

Given the large number of ranking models as introduced in the previous subsection, a standard evaluation mechanism is needed to select the most effective model. For this purpose, one usually proceeds as follows:

- Collect a large number of (randomly sampled) queries to form a test set.

- For each query q,
  - Collect documents $\{d_j\}_{j=1}^m$ associated with the query.
  - Get the relevance judgment for each document by human assessment.
  - Use a given ranking model to rank the documents.
  - Measure the difference between the ranking results and the relevance judgment using an evaluation measure.
- Use the average measure on all the queries in the test set to evaluate the performance of the ranking model.

As for collecting the documents associated with a query, a number of strategies can be used. For example, one can simply collect all the documents containing the query word. One can also choose to use some predefined rankers to get documents that are more likely to be relevant. A popular strategy is the pooling method used in TREC.[5] In this method a pool of possibly relevant documents is created by taking a sample of documents selected by various participating systems. In particular, the top 100 documents retrieved in each submitted run for a given query are selected and merged into the pool for human assessment. On average, an assessor judges the relevance of approximately 1500 documents per query.

As for the relevance judgment, three strategies were used in the literature.

(1) Specifying whether a document is relevant or not to the query (i.e., binary judgment 1 or 0), or further specifying the degree of relevance (i.e., multiple ordered categories, e.g., Perfect, Excellent, Good, Fair, or Bad). Suppose for document d_j associated with query q, we get its relevance judgment as l_j. Then for two documents d_u and d_v, if $l_u \succ l_v$, we say that document d_u is more relevant than document d_v, with regards to query q, according to the relevance judgment.

[5] http://trec.nist.gov/

(2) Specifying whether a document is more relevant than another with regards to a query. For example, if document d_u is judged to be more relevant than document d_v with regards to query q, we give the judgment $l_{u,v} = 1$; otherwise, $l_{u,v} = -1$. That is, this kind of judgment captures the relative preference between documents.[6]

(3) Specifying the partial order or even total order of the documents with respect to a query. For the group of documents $\{d_j\}_{j=1}^m$ associated with query q, this kind of judgment is usually represented as a certain permutation of these documents, denoted as π_l, or a set of such permutations.

Given the vital role that relevance judgments play in a test collection, it is important to assess the quality of the judgments. In previous practices like TREC, both the completeness and the consistency of relevance judgments are of interest. Completeness measures the degree to which all the relevant documents for a topic have been found; consistency measures the degree to which the assessor has marked all the "truly" relevant documents relevant and the "truly" irrelevant documents irrelevant.

Since manual judgment is time consuming, it is almost impossible to judge all the documents with regards to a query. Consequently, there are always unjudged documents returned by the ranking model. As a common practice, one regards the unjudged documents as irrelevant in the evaluation process.[7]

With the relevance judgment, several evaluation measures have been proposed and used in IR literature. It is clear that understanding these measures will be very important for learning to rank, since to some extent they define the "true" objective function of ranking. Below we list some popularly used measures.

Mean reciprocal rank (MRR): For query q, the rank position of its first relevant document is denoted as $r(q)$. Then $\frac{1}{r(q)}$ is defined as MRR for

[6] This kind of judgment can also be mined from click-through logs of search engines [68, 69, 105].

[7] In recent years, several new evaluation mechanisms [18] that consider the relevance probability of an unjudged document have also been proposed.

query q. It is clear that documents ranked below $r(q)$ are not considered in MRR.

Mean average precision (MAP): To define MAP [8], one needs to define Precision at position k ($P@k$) first,

$$P@k(q) = \frac{\#\{\text{relevant documents in the top } k \text{ positions}\}}{k}. \tag{1.6}$$

Then, the Average Precision (AP) is defined below:

$$\mathrm{AP}(q) = \frac{\sum_{k=1}^{m} P@k(q) \cdot l_k}{\#\{\text{relevant documents}\}}, \tag{1.7}$$

where m is the total number of documents associated with query q, and l_k is the binary judgment on the relevance of the document at the k-th position. The mean value of AP over all the test queries is named MAP.

Discounted cumulative gain (DCG): While the aforementioned measures are mainly designed for binary judgments, DCG [65, 66] can leverage the relevance judgment in terms of multiple ordered categories, and has an explicit position discount factor in its definition. More formally, suppose the ranked list for query q is π, then DCG at position k is defined as follows:

$$\mathrm{DCG}@k(q) = \sum_{r=1}^{k} G(\pi^{-1}(r))\eta(r), \tag{1.8}$$

where $\pi^{-1}(r)$ denotes the document ranked at position r of the list π, $G(\cdot)$ is the rating of a document (one usually sets $G(\pi^{-1}(r)) = (2^{l_{\pi^{-1}(r)}} - 1)$), and $\eta(r)$ is a position discount factor (one usually sets $\eta(r) = 1/\log_2(r+1)$).

By normalizing DCG@k with the maximum value of it (denoted as Z_k), we will get another measure named Normalized DCG (NDCG). That is:

$$\mathrm{NDCG}@k(q) = \frac{1}{Z_k}\sum_{r=1}^{k} G(\pi^{-1}(r))\eta(r). \tag{1.9}$$

It is clear that NDCG takes values from 0 to 1.

Rank correlation (RC): The correlation between the ranked list given by the model (denoted as π) and the relevance judgment (denoted as π_l) can be used to define a measure. For example, when the weighted Kendall's τ is used, the RC measures the weighted pairwise inconsistency between two lists. Its definition is given below:

$$\tau_K(q) = \frac{\sum_{u<v} w_{u,v}(1 + \mathrm{sgn}((\pi(u) - \pi(v))(\pi_l(u) - \pi_l(v))))}{2\sum_{u<v} w_{u,v}}, \quad (1.10)$$

where $w_{u,v}$ is the weight, and $\pi(u)$ means the rank position of document d_u in list π.

To summarize, there are some common properties in these evaluation measures.

(1) All these evaluation measures are calculated at the *query level*. That is, first the measure is computed for each query, and then averaged over all queries in the test set. No matter how poorly the documents associated with a particular query are ranked, it will not dominate the evaluation process since each query contributes similarly to the average measure.

(2) All these measures are *position based*. That is, rank position is explicitly used. Considering that with small changes in the scores given by a ranking model the rank positions will not change until one document's score passes another, the position-based measures are usually non-continuous and non-differentiable with regards to the scores. This makes the optimization of these measures quite difficult. We will conduct more discussions on this in Section 4.1.

1.2 Learning to Rank

Many ranking models have been introduced in the previous subsection, most of which contain parameters. For example, there are parameters k_1 and b in BM25 (see Equation (1.2)), parameter λ in language model for IR (see Equation (1.3)), and parameter α in PageRank (see Equation (1.5)). In order to get a reasonably good ranking performance (in terms of IR evaluation measures), one needs to tune these parameters using a validation set. Nevertheless, the parameter tuning is far from trivial, especially considering that IR evaluation measures are non-continuous and non-differentiable with respect to the parameters.

In addition, a model perfectly tuned on the validation set sometimes performs poorly on unseen test queries. This is usually called over-fitting. Another issue is about the combination of these ranking models. Given that many models have been proposed in the literature, it is natural to investigate how to combine these models and create an even more effective new model. This is, however, not straightforward either.

While IR researchers were facing these problems, machine learning has been demonstrating its effectiveness in automatically tuning parameters, combining multiple evidences, and avoiding over-fitting. Therefore, it seems quite promising to adopt ML technologies to solve the aforementioned problems.

1.2.1 ML Framework

In much ML research (especially discriminative learning), attention has been paid to the following key components.[8]

(1) The *input space*, which contains the objects under investigation: Usually objects are represented by feature vectors, extracted according to different applications.
(2) The *output space*, which contains the learning target with respect to the input objects: There are two related but different definitions of the output space in ML.[9] The first is the output space of the task, which is highly dependent on the application. For example, in the regression problem the output space is the space of real numbers $\mathbb{R}$; in classification, it is the set of discrete categories $\{0, 1, \ldots, K-1\}$. The second is the output space to facilitate the learning process. This may differ from the output space of the task. For example, one can use regression algorithms to solve the problem of classification. In this case, the output space that facilitates learning is the space of real numbers but not discrete categories.
(3) The *hypothesis* space, which defines the class of functions mapping the input space to the output space: The functions

[8] For a more comprehensive introduction to the ML literature, please refer to [89].

[9] In this tutorial, when we mention the output space, we mainly refer to the second type.

operate on the feature vectors of the input objects, and make predictions according to the format of the output space.

(4) In order to learn the optimal hypothesis, a training set is usually used, which contains a number of independent and identically distributed (i.i.d.) objects and their ground truth labels, sampled from the product of the input and output spaces. The *loss function* measures to what degree the prediction generated by the hypothesis is in accordance with the ground truth label. For example, widely used loss functions for classification include the exponential loss, the hinge loss, and the logistic loss. It is clear that the loss function plays a central role in ML, since it encodes the understanding of the target application (i.e., what prediction is correct and what is not). With the loss function, an empirical risk can be defined on the training set, and the optimal hypothesis is usually (but not always) learned by means of empirical risk minimization.

1.2.2 Learning-to-Rank Framework

In recent years, more and more ML technologies have been used to train the ranking model, and a new research area named "learning to rank" has gradually emerged. Especially in the past several years, learning to rank has become one of the most active research areas in IR.

In general, we can call all those methods that use ML technologies to solve the problem of ranking "learning-to-rank" methods,[10] such as the work on relevance feedback[11] [39, 112] and automatically tuning the parameters of existing IR models [60, 120]. However, most of the state-of-the-art learning-to-rank algorithms learn the optimal way of combining features extracted from query–document pairs through discriminative training. Therefore, in this tutorial we define learning to rank in a more specific way to better summarize these algorithms.

[10] In ML literature, there is a topic named label ranking. It is to predict the ranking of multiple class labels for an individual document, but not to predict the ranking of documents. In this regard, it is largely different from the task of ranking for IR.

[11] We will make further discussions on the relationship between relevance feedback and learning to rank in Section 2.

We call those ranking methods that have the following two properties learning-to-rank methods.

Feature based: All the documents under investigation are represented by feature vectors,[12] reflecting the relevance of the documents to the query. That is, for a given query q, its associated document d can be represented by a vector $x = \Phi(d,q)$, where Φ is a feature extractor. Typical features used in learning to rank include the frequencies of the query terms in the document, the BM25 and PageRank scores, and the relationship between this document and other documents. If one wants to know more about widely used features, please refer to Tables 6.2 and 6.3 in Section 6.

Even if a feature is the output of an existing retrieval model, in the context of learning to rank, one assumes that the parameter in the model is fixed, and only the optimal way of combining these features is learned. In this sense, the previous work on automatically tuning the parameters of existing models [60, 120] is not categorized as "learning-to-rank" methods.

The capability of combining a large number of features is a very important advantage of learning to rank. It is easy to incorporate any new progress on the retrieval model by including the output of the model as one dimension of the features. Such a capability is highly demanding for real search engines, since it is almost impossible to use only a few factors to satisfy complex information needs of Web users.

Discriminative training: The learning process can be well described by the four components of discriminative learning as mentioned in the previous subsection. That is, a learning-to-rank method has its specific input space, output space, hypothesis space, and loss function.

In ML literature, discriminative methods have been widely used to combine different kinds of features, without the necessity of defining a probabilistic framework to represent the objects and the correctness of prediction. In this sense, previous works that train generative ranking

[12] Note that, hereafter in this tutorial, when we refer to a document, we will not use d any longer. Instead, we will directly use its feature representation x. Furthermore, since our discussions will focus more on the learning process, we will always assume the features are pre-specified, and will not purposely discuss how to extract them.

models are not categorized as "learning-to-rank" methods in this tutorial. If one has interest in such work, please refer to [74, 85, 141], etc.

Discriminative training is an automatic learning process based on the training data. This is also highly demanding for real search engines, because everyday these search engines will receive a lot of user feedback and usage logs indicating poor ranking for some queries or documents. It is very important to automatically learn from feedback and constantly improve the ranking mechanism.

Due to the aforementioned two characteristics, learning to rank has been widely used in commercial search engines,[13] and has also attracted great attention from the academic research community.

Figure 1.1 shows the typical "learning-to-rank" flow. From the figure we can see that since learning to rank is a kind of supervised learning, a training set is needed. The creation of a training set is very similar to

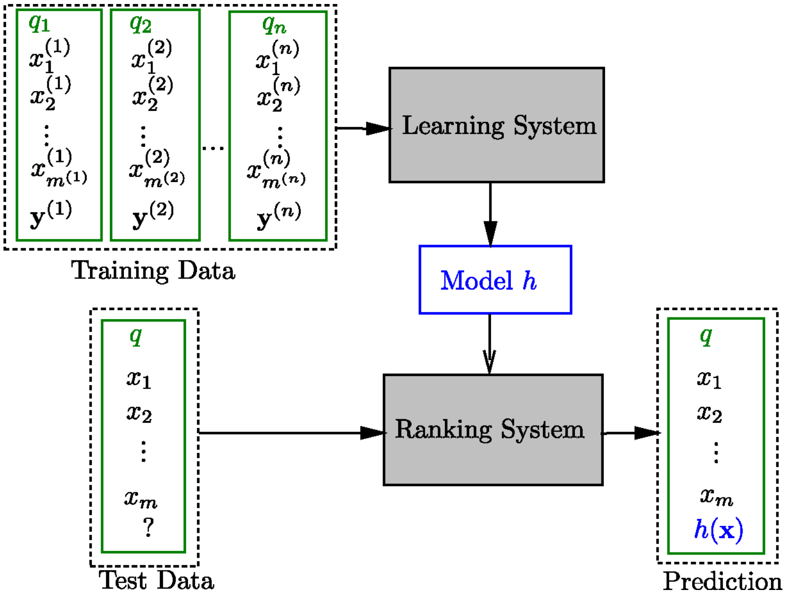

Fig. 1.1 Learning-to-rank framework.

[13] See http://blog.searchenginewatch.com/050622-082709, http://blogs.msdn.com/msnsearch/archive/2005/06/21/431288.aspx, and http://glinden.blogspot.com/2005/06/msn-search-and-learning-to-rank.html.

the creation of the test set for evaluation. For example, a typical training set consists of n training queries $q_i (i = 1, \ldots, n)$, their associated documents represented by feature vectors $\mathbf{x}^{(i)} = \{x_j^{(i)}\}_{j=1}^{m^{(i)}}$ (where $m^{(i)}$ is the number of documents associated with query q_i), and the corresponding relevance judgments.[14] Then a specific learning algorithm is employed to learn the ranking model (i.e., the way of combining the features), such that the output of the ranking model can predict the ground truth label in the training set[15] as accurately as possible, in terms of a loss function. In the test phase, when a new query comes in, the model learned in the training phase is applied to sort the documents according to their relevance to the query, and return the corresponding ranked list to the user as the response to her/his query.

1.2.3 Approaches to Learning to Rank

Many learning-to-rank algorithms can fit into the above framework. In order to better understand them, we perform a categorization on these algorithms. In particular, we group the algorithms, according to the four pillars of ML, into three approaches: the pointwise approach, the pairwise approach, and the listwise approach. Note that different approaches model the process of learning to rank in different ways. That is, they define different input and output spaces, use different hypotheses, and employ different loss functions. Note that the output space is used to facilitate the learning process, which can be different from the relevance judgments on the documents. That is, even if provided with the same format of judgments, one can derive different ground truth labels from it, and use them for different approaches.

The pointwise approach

The *input space* of the pointwise approach contains the feature vector of each single document.

[14] Please distinguish between the judgment for evaluation and the judgment for constructing the training set, although the processes of obtaining them may be very similar.

[15] Hereafter, when we mention the *ground truth labels* in the remainder of the tutorial, we will mainly refer to the ground truth labels in the training set, although we assume every document has its intrinsic label no matter whether it is judged or not.

The *output space* contains the relevance degree of each single document. The ground truth label in the output space is usually defined in the following way. If the judgment is directly given as relevance degree l_j, the ground truth label for document x_j is defined as $y_j = l_j$. If the judgment is given as total order π_l, one can get the ground truth label by using a mapping function.[16] However, if the judgment is given as pairwise preference $l_{u,v}$, it is not straightforward to make use of it to generate the ground truth label.

The *hypothesis* space contains functions that take the feature vector of a document as the input and predict the relevance degree of the document. We usually call such a function f the scoring function. Note that, based on the scoring function, one can sort all the documents and produce the final ranked list.

The *loss function* examines the accurate prediction of the ground truth label for each single document. In different pointwise ranking algorithms, ranking is modeled as regression, classification, and ordinal regression (see Section 2). Therefore the corresponding regression loss, classification loss, and ordinal regression loss are used as the loss function. Note that the pointwise approach does not consider the interdependency among documents, and thus the position of a document in the final ranked list is invisible to its loss function. Furthermore, the approach does not make use of the fact that some documents are actually associated with the same query. Considering that most IR evaluation measures are query-level and position-based, intuitively speaking, the pointwise approach has its limitations.

Example algorithms belonging to the pointwise approach include [24, 25, 26, 31, 33, 34, 49, 53, 73, 78, 90, 114]. We will introduce some of them in Section 2.

The pairwise approach

The *input space* of the pairwise approach contains a pair of documents, both represented as feature vectors.

The *output space* contains the pairwise preference (which takes values from $\{1, -1\}$) between each pair of documents. The ground truth

[16] For example, the position of the document in π_l can be used to define the relevance degree.

label in the output space is usually defined in the following way. If the judgment is given as relevance degree l_j, then the order for document pair (x_u, x_v) can be defined as $y_{u,v} = 2 \cdot I_{\{l_u \succ l_v\}} - 1$. Here $I_{\{A\}}$ is an indicator function, which is defined to be 1 if predicate A holds and 0 otherwise. If the judgment is given directly as pairwise preference $l_{u,v}$, then it is straightforward to set $y_{u,v} = l_{u,v}$. If the judgment is given as total order π_l, one can define $y_{u,v} = 2 \cdot I_{\{\pi_l(u) < \pi_l(v)\}} - 1$.

The *hypothesis* space contains bi-variate functions h that take a pair of documents as the input and output the relative order between them. Some pairwise ranking algorithms directly define their hypotheses as such [29], however, in more algorithms, the hypothesis is still defined with a scoring function f for simplicity, i.e., $h(x_u, x_v) = 2 \cdot I_{\{f(x_u) > f(x_v)\}} - 1$.

The *loss function* measures the inconsistency between $h(x_u, x_v)$ and the ground truth label $y_{u,v}$. For example, in some algorithms, ranking is modeled as a pairwise classification, and the corresponding classification loss on a pair of documents is used as the loss function. Note that the loss function used in the pairwise approach only considers the relative order between two documents. When one looks at only a pair of documents, however, the position of the documents in the final ranked list can hardly be derived. Furthermore, the approach ignores the fact that some pairs are generated from the documents associated with the same query. Considering that most IR evaluation measures are query-level and position-based, intuitively speaking, there is still a gap between this approach and ranking for IR.

Example algorithms belonging to the pairwise approach include [9, 14, 16, 29, 47, 63, 97, 122]. We will introduce some of them in Section 3.

The listwise approach

The *input space* of the listwise approach contains the entire group of documents associated with query q, e.g., $\mathbf{x} = \{x_j\}_{j=1}^m$.

There are two types of *output spaces* used in the listwise approach. For some listwise ranking algorithms, the *output space* contains the relevance degrees of all the documents associated with a query. In this case, the ground truth label $\mathbf{y} = \{y_j\}_{j=1}^m$ can be derived from the judgment in terms of the relevance degree or total order, in a similar manner to that of the pointwise approach. For some other listwise ranking algorithms, the *output space* contains the ranked list (or permutation) of the documents. In this case, the ground truth label, denoted as π_y, can be generated in the following way. When the judgment is given as total order π_l, we can define $\pi_y = \pi_l$. Otherwise, we can derive π_y by using the concept of the *equivalent permutation set* (see Section 4). When π_y is given as the ground truth label, the output space that facilitates the learning process is exactly the output space of the ranking task. Therefore, the theoretical analysis on the listwise approach has a more direct value where understanding the real ranking problem than the other approaches where there are mismatches between the output space that facilitates learning and the real output space of the task.

The *hypothesis* space contains multivariate functions h that operate on a group of documents, and predict their relevance degrees or their permutation. For practical reasons, the hypothesis h is also usually implemented with scoring function f. When the relevance degree comprises the output space, $h(\mathbf{x}) = f(\mathbf{x})$. When the ranked list (permutation) comprises the output space, h is defined as a compound function $h(\mathbf{x}) = \text{sort} \circ f(\mathbf{x})$. That is, first scoring function f is used to give a score to each document, and then these documents are sorted in the descending order of the scores to produce the desired ranked list.

There are also two types of *loss functions*, corresponding to the two types of output spaces. When the ground truth label is given as $\mathbf{y}$, the loss function is usually defined on the basis of the approximation or bound of widely used IR evaluation measures. When the ground truth label is given as π_y, the loss function measures the difference between the ranked list given by the hypothesis and the ground truth list. As compared to the pointwise and pairwise approaches, the advantage of the listwise approach lies in that its loss function can naturally consider the positions of documents in the ranked list of all the documents associated with the same query.

Example algorithms that belong to the listwise approach include [13, 17, 99, 102, 119, 129, 134, 136]. We will introduce some of them in Section 4.

It is noted that different loss functions are used in different approaches, while the same IR evaluation measures are used for testing their performances. A natural question that arises concerns the relationship between these loss functions and IR evaluation measures. The investigation on this issue can help us explain the empirical results of learning-to-rank algorithms. We will introduce some such investigations in Section 5. In addition, in Section 6, we will introduce a benchmark dataset for the research on learning to rank, named LETOR, and report some empirical results of representative learning-to-rank algorithms on the dataset.

Furthermore, one may have noticed that the scoring function, which is widely used in defining the hypotheses of different approaches, is a kind of "pointwise" function. However, it is not to say that all the approaches are in nature pointwise approaches. The categorization of the aforementioned three approaches is based on the four pillars of ML. That is, different approaches regard the same training data as in different input and output spaces, and define different loss functions and hypotheses accordingly. From the ML point of view, they have different assumptions on the i.i.d. distribution of the data and therefore the theoretical properties (e.g., generalization ability) of their corresponding algorithms will be largely different. We will further discuss this in Section 7, with the introduction of a new theory, which we call the statistical ranking theory.

1.3 About this Tutorial

As for the writing of the tutorial, we do not aim to be fully rigorous. Instead, we try to provide insights into the basic ideas. However, it is still unavoidable that we will use mathematics for better illustration of the problem, especially when we jump into the theoretical discussions on learning to rank. We will have to assume familiarity with basic concepts of probability theory and statistical learning in the corresponding discussions.

Furthermore, we will use the notation rules as listed in Table 1.1 throughout the tutorial. Here we would like to add one more note. Since in practice the hypothesis h is usually defined with scoring function f,

Table 1.1 Notation rules.

Meaning	Notation
Query	q, or q_i
A quantity z for query q_i	$z^{(i)}$
Number of training queries	n
Number of documents associated with query q	m
Number of document pairs associated with query q	$\tilde{m}$
Feature vector of a document associated with query q	x
Feature vectors of documents associated with query q	$\mathbf{x} = \{x_j\}_{j=1}^{m}$
Term frequency of query q in document d	$\mathrm{TF}(q,d)$
Inverse document frequency of query q	$\mathrm{IDF}(q)$
Length of document d	$\mathrm{LEN}(d)$
Hypothesis	$h(\cdot)$
Scoring function	$f(\cdot)$
Loss function	$L(\cdot)$
Expected risk	$R(\cdot)$
Empirical risk	$\hat{R}(\cdot)$
Relevance degree for document x_j	l_j
Document x_u is more relevant than document x_v	$l_u \succ l_v$
Pairwise preference between documents x_u and x_v	$l_{u,v}$
Total order of document associated with the same query	π_l
Ground truth label for document x_j	y_j
Ground truth label for document pair (x_u, x_v)	$y_{u,v}$
Ground truth list for documents associated with query q	π_y
Ground truth permutation set for documents associated with query q	Ω_y
Original document index of the j-th element in permutation π	$\pi^{-1}(j)$
Rank position of document j in permutation π	$\pi(j)$
Number of classes	K
Index of class	k
VC dimension of a function class	V
Indicator function	$I_{\{\cdot\}}$
Gain function	$G(\cdot)$
Position discount function	$\eta(\cdot)$

we sometimes use $L(h)$ and $L(f)$ interchangeably to represent the loss function. When we need to emphasize the parameter in the scoring function, we will use $f(w,x)$ instead of $f(x)$ in the discussion, although they actually mean the same thing.

References

[1] S. Agarwal, T. Graepel, R. Herbrich, S. Har-Peled, and D. Roth, "Generalization bounds for the area under the ROC curve," *Journal of Machine Learning*, vol. 6, pp. 393–425, 2005.

[2] S. Agarwal and P. Niyogi, "Stability and generalization of bipartite ranking algorithms," in *COLT 2005*, pp. 32–47, 2005.

[3] E. Agichtein, E. Brill, S. T. Dumais, and R. Ragno, "Learning user interaction models for predicting web search result preferences," in *SIGIR 2006*, pp. 3–10, 2006.

[4] H. Almeida, M. Goncalves, M. Cristo, and P. Calado, "A combined component approach for finding collection-adapted ranking functions based on genetic programming," in *SIGIR 2007*, pp. 399–406, 2007.

[5] M.-R. Amini, T.-V. Truong, and C. Goutte, "A boosting algorithm for learning bipartite ranking functions with partially labeled data," in *SIGIR 2008*, pp. 99–106, 2008.

[6] S. Andrews, I. Tsochantaridis, and T. Hofmann, "Support vector machines for multiple-instance learning," in *NIPS 2003*, pp. 561–568, 2003.

[7] J. A. Aslam and M. Montague, "Models for metasearch," in *SIGIR 2001*, pp. 276–284, 2001.

[8] R. Baeza-Yates and B. Ribeiro-Neto, *Modern Information Retrieval.* Addison Wesley, May 1999.

[9] B. Bartell, G. W. Cottrell, and R. Belew, "Learning to retrieve information," in *SCC 1995*, 1995.

[10] P. L. Bartlett and S. Mendelson, "Rademacher and Gaussian complexities risk bounds and structural results," *Journal of Machine Learning*, pp. 463–482, 2002.

[11] O. Bousquet, S. Boucheron, and G. Lugosi, "Introduction to statistical learning theory," in *Advanced Lectures on Machine Learning*, pp. 169–207, Berlin/Heidelberg: Springer, 2004.

[12] O. Bousquet and A. Elisseeff, "Stability and generalization," *The Journal of Machine Learning Research*, vol. 2, pp. 449–526, 2002.

[13] C. J. Burges, R. Ragno, and Q. V. Le, "Learning to rank with nonsmooth cost functions," in *NIPS 2006*, pp. 395–402, 2006.

[14] C. J. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *ICML 2005*, pp. 89–96, 2005.

[15] G. Cao, J. Nie, L. Si, and J. Bai, "Learning to rank documents for ad-hoc retrieval with regularized models," in *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[16] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon, "Adapting ranking SVM to document retrieval," in *SIGIR 2006*, pp. 186–193, 2006.

[17] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *ICML 2007*, pp. 129–136, 2007.

[18] B. Carterette, V. Pavlu, E. Kanoulas, J. A. Aslam, and J. Allan, "Evaluation over thousands of queries," in *SIGIR 2008*, pp. 651–658, 2008.

[19] V. R. Carvalho, J. L. Elsas, W. W. Cohen, and J. G. Carbonell, "A meta-learning approach for robust rank learning," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[20] S. Chakrabarti, R. Khanna, U. Sawant, and C. Bhattacharyya, "Structured learning for non-smooth ranking losses," in *SIGKDD 2008*, pp. 88–96, 2008.

[21] O. Chapelle, Q. Le, and A. Smola, "Large margin optimization of ranking measures," in *NIPS Workshop on Machine Learning for Web Search 2007*, 2007.

[22] W. Chen, Y. Lan, T.-Y. Liu, and H. Li, "A unified view on loss functions in learning to rank," Technical Report, Microsoft Research, MSR-TR-2009-39, 2009.

[23] P. Chirita, J. Diederich, and W. Nejdl, "MailRank: Using ranking for spam detection," in *CIKM 2005*, pp. 373–380, New York, NY, USA: ACM, 2005.

[24] W. Chu and Z. Ghahramani, "Gaussian processes for ordinal regression," *Journal of Machine Learning Research*, vol. 6, pp. 1019–1041, 2005.

[25] W. Chu and Z. Ghahramani, "Preference learning with Gaussian processes," in *ICML 2005*, pp. 137–144, 2005.

[26] W. Chu and S. S. Keerthi, "New approaches to support vector ordinal regression," in *ICML 2005*, pp. 145–152, 2005.

[27] S. Clemencon, G. Lugosi, and N. Vayatis, "Ranking and empirical minimization of U-statistics," *The Annals of Statistics*, vol. 36, pp. 844–874, 2008.

[28] S. Clemencon and N. Vayatis, "Ranking the best instances," *Journal of Machine Learning Research*, vol. 8, pp. 2671–2699, December 2007.

[29] W. W. Cohen, R. E. Schapire, and Y. Singer, "Learning to order things," in *NIPS 1998*, Vol. 10, pp. 243–270, 1998.

[30] M. Collins, "Ranking algorithms for named-entity extraction: Boosting and the voted perceptron," in *ACL 2002*, pp. 7–12, 2002.

[31] W. S. Cooper, F. C. Gey, and D. P. Dabney, "Probabilistic retrieval based on staged logistic regression," in *SIGIR 1992*, pp. 198–210, 1992.

[32] C. Cortes, M. Mohri, and A. Rastogi, "Magnitude-preserving ranking algorithms," in *ICML 2007*, pp. 169–176, 2007.

[33] D. Cossock and T. Zhang, "Subset ranking using regression," in *COLT 2006*, pp. 605–619, 2006.

[34] K. Crammer and Y. Singer, "Pranking with ranking," in *NIPS 2002*, pp. 641–647, 2002.

[35] N. Craswell, D. Hawking, R. Wilkinson, and M. Wu, "Overview of the TREC 2003 Web track," in *TREC 2003*, pp. 78–92, 2003.

[36] K. Dave, S. Lawrence, and D. Pennock, "Mining the peanut gallery: Opinion extraction and semantic classification of product reviews," in *WWW 2003*, pp. 519–528, New York, NY, USA: ACM Press, 2003.

[37] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by latent semantic analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, 1990.

[38] F. Diaz, "Regularizing query-based retrieval scores," *Information Retrieval*, vol. 10, pp. 531–562, 2007.

[39] H. Drucker, B. Shahrary, and D. C. Gibbon, "Support vector machines: Relevance feedback and information retrieval," *Information Processing and Management*, vol. 38, pp. 305–323, 2002.

[40] K. Duh and K. Kirchhoff, "Learning to rank with partially-labeled data," in *SIGIR 2008*, pp. 251–258, 2008.

[41] W. Fan, E. A. Fox, P. Pathak, and H. Wu, "The effects of fitness functions on genetic programming based ranking discovery for web search," *Journal of American Society for Information Science and Technology*, vol. 55, pp. 628–636, 2004.

[42] W. Fan, M. Gordon, and P. Pathak, "Discovery of context-specific ranking functions for effective information retrieval using genetic programming," *IEEE Transactions on Knowledge and Data Engineering*, vol. 16, pp. 523–527, 2004.

[43] W. Fan, M. Gordon, and P. Pathak, "A generic ranking function discovery framework by genetic programming for information retrieval," *Information Processing and Management*, vol. 40, pp. 587–602, 2004.

[44] W. Fan, M. Gordon, and P. Pathak, "Genetic programming-based discovery of ranking functions for effective web search," *Journal of Management of Information Systems*, vol. 21, pp. 37–56, 2005.

[45] W. Fan, M. Gordon, and P. Pathak, "On linear mixture of expert approaches to information retrieval," *Decision Support System*, vol. 42, pp. 975–987, 2006.

[46] W. Fan, M. D. Gordon, W. Xi, and E. A. Fox, "Ranking function optimization for effective web search by genetic programming: An empirical study," in *HICSS 2004,* pp. 40105, 2004.

[47] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer, "An efficient boosting algorithm for combining preferences," *Journal of Machine Learning Research*, vol. 4, pp. 933–969, 2003.

[48] Y. Freund and R. E. Schapire, "A decision-theoretic generalization of online learning and an application to boosting," *Journal of Computer and System Sciences*, vol. 55, pp. 119–139, 1995.

[49] N. Fuhr, "Optimum polynomial retrieval functions based on the probability ranking principle," *ACM Transactions on Information Systems*, vol. 7, pp. 183–204, 1989.

[50] G. Fung, R. Rosales, and B. Krishnapuram, "Learning rankings via convex hull separation," in *NIPS 2005 Workshop on Learning to Rank*, 2005.

[51] X.-B. Geng, T.-Y. Liu, and T. Qin, "Feature selection for ranking," in *SIGIR 2007*, pp. 407–414, 2007.

[52] X.-B. Geng, T.-Y. Liu, T. Qin, H. Li, and H.-Y. Shum, "Query-dependent ranking using k-nearest neighbor," in *SIGIR 2008*, pp. 115–122, 2008.

[53] F. C. Gey, "Inferring probability of relevance using the method of logistic regression," in *SIGIR 1994*, pp. 222–231, 1994.

[54] S. Guiasu and A. Shenitzer, "The principle of maximum entropy," *The Mathematical Intelligencer*, vol. 7, pp. 42–48, 1985.

[55] J. Guiver and E. Snelson, "Learning to rank with softrank and Gaussian processes," in *SIGIR 2008*, pp. 259–266, 2008.

[56] Z. Gyöngyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB 2004*, pp. 576–587, VLDB Endowment, 2004.

[57] Z. Gyongyi, H. Garcia-Molina, and J. Pedersen, "Combating web spam with trustrank," in *VLDB 2004*, pp. 576–587, 2004.

[58] E. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *ICML 2003*, Vol. 20, pp. 250–257, 2003.

[59] E. F. Harrington, "Online ranking/collaborative filtering using the perceptron algorithm," in *ICML 2003*, pp. 250–257, 2003.

[60] B. He and I. Ounis, "A study of parameter tuning for term frequency normalization," in *CIKM 2003*, pp. 10–16, 2003.

[61] Y. He and T.-Y. Liu, "Are algorithms directly optimizing IR measures really direct?," Technical Report, Microsoft Research, MSR-TR-2008-154, 2008.

[62] R. Herbrich, T. Graepel, and C. Campbell, "Bayes point machines," *Journal of Machine Learning Research*, vol. 1, pp. 245–279, 2001.

[63] R. Herbrich, K. Obermayer, and T. Graepel, "Large margin rank boundaries for ordinal regression," in *Advances in Large Margin Classifiers*, pp. 115–132, 2000.

[64] W. Hersh, C. Buckley, T. J. Leone, and D. Hickam, "OHSUMED: An interactive retrieval evaluation and new large test collection for research," in *SIGIR 1994*, pp. 192–201, 1994.

[65] K. Järvelin and J. Kekäläinen, "IR evaluation methods for retrieving highly relevant documents," in *SIGIR 2000*, pp. 41–48, 2000.

[66] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Transactions on Information Systems*, vol. 20, pp. 422–446, 2002.

[67] R. Jin, H. Valizadegan, and H. Li, "Ranking refinement and its application to information retrieval," in *WWW 2008*, pp. 397–406, 2008.

[68] T. Joachims, "Optimizing search engines using clickthrough data," in *KDD 2002*, pp. 133–142, 2002.

[69] T. Joachims, "Evaluating retrieval performance using clickthrough data," *Text Mining*, pp. 79–96, 2003.

[70] T. Joachims, "A support vector method for multivariate performance measures," in *CML 2005*, pp. 377–384, 2005.

[71] I. Kang and G. Kim, "Query type classification for web document retrieval," in *SIGIR 2003*, pp. 64–71, 2003.

[72] J. M. Kleinberg, "Authoritative sources in a hyperlinked environment," *Journal of ACM*, vol. 46, pp. 604–632, 1999.

[73] S. Kramer, G. Widmer, B. Pfahringer, and M. D. Groeve, "Prediction of ordinal classes using regression trees," *Funfamenta Informaticae*, vol. 34, pp. 1–15, 2000.

[74] J. Lafferty and C. Zhai, "Document language models, query models and risk minimization for information retrieval," in *SIGIR 2001*, pp. 111–119, 2001.

[75] Y. Lan and T.-Y. Liu, "Generalization analysis of listwise learning-to-rank algorithms," in *ICML 2009*, 2009.

[76] Y. Lan, T.-Y. Liu, T. Qin, Z. Ma, and H. Li, "Query-level stability and generalization in learning to rank," in *ICML 2008*, pp. 512–519, 2008.

[77] F. Li and Y. Yang, "A loss function analysis for classification methods in text categorization," in *ICML 2003*, pp. 472–479, 2003.

[78] P. Li, C. Burges, and Q. Wu, "McRank: Learning to rank using multiple classification and gradient boosting," in *NIPS 2007*, pp. 845–852, 2007.

[79] T.-Y. Liu, J. Xu, T. Qin, W.-Y. Xiong, and H. Li, "LETOR: Benchmark dataset for research on learning to rank for information retrieval," in *SIGIR '07 Workshop on Learning to Rank for Information Retrieval*, 2007.

[80] Y. Liu, T.-Y. Liu, T. Qin, Z. Ma, and H. Li, "Supervised rank aggregation," in *WWW 2007*, pp. 481–490, 2007.

[81] R. D. Luce, *Individual Choice Behavior.* New York: Wiley, 1959.

[82] C. L. Mallows, "Non-null ranking models," *Biometrika*, vol. 44, pp. 114–130, 1975.

[83] M. E. Maron and J. L. Kuhns, "On relevance, probabilistic indexing and information retrieval," *Journal of the ACM*, vol. 7, pp. 216–244, 1960.

[84] I. Matveeva, C. Burges, T. Burkard, A. Laucius, and L. Wong, "High accuracy retrieval with multiple nested ranker," in *SIGIR 2006*, pp. 437–444, 2006.

[85] D. A. Metzler and W. B. Croft, "A Markov random field model for term dependencies," in *SIGIR 2005*, pp. 472–479, 2005.

[86] D. A. Metzler, W. B. Croft, and A. McCallum, "Direct maximization of rank-based metrics for information retrieval," in *CIIR Technical Report*, 2005.

[87] D. A. Metzler and T. Kanungo, "Machine learned sentence selection strategies for query-biased summarization," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[88] T. Minka and S. Robertson, "Selection bias in the LETOR datasets," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[89] T. Mitchell, *Machine Learning.* McGraw Hill, 1997.

[90] R. Nallapati, "Discriminative models for information retrieval," in *SIGIR 2004*, pp. 64–71, 2004.

[91] L. Nie, B. D. Davison, and X. Qi, "Topical link analysis for web search," in *SIGIR 2006*, pp. 91–98, 2006.

[92] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: Bringing order to the web," Technical Report, Stanford Digital Library Technologies Project, 1998.

[93] T. Pahikkala, E. Tsivtsivadze, A. Airola, J. Boberg, and T. Salakoski, "Learning to rank with pairwise regularized least-squares," in *SIGIR 2007 Workshop on Learning to Rank for Information Retrieval*, 2007.

[94] B. Pang and L. Lee, "Seeing stars: Exploiting class relationships for sentiment categorization with respect to rating scales," in *ACL 2005*, pp. 115–124, NJ, USA: Association for Computational Linguistics Morristown, 2005.

[95] R. L. Plackett, "The analysis of permutations," *Applied Statistics*, vol. 24, pp. 193–202, 1975.

[96] J. M. Ponte and W. B. Croft, "A language modeling approach to information retrieval," in *SIGIR 1998*, pp. 275–281, 1998.

[97] T. Qin, T.-Y. Liu, W. Lai, X.-D. Zhang, D.-S. Wang, and H. Li, "Ranking with multiple hyperplanes," in *SIGIR 2007*, pp. 279–286, 2007.

[98] T. Qin, T.-Y. Liu, and H. Li, "A general approximation framework for direct optimization of information retrieval measures," Technical Report, Microsoft Research, MSR-TR-2008-164, 2008.

[99] T. Qin, T.-Y. Liu, M.-F. Tsai, X.-D. Zhang, and H. Li, "Learning to search web pages with query-level loss functions," Technical Report, Microsoft Research, MSR-TR-2006-156, 2006.

[100] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "How to make LETOR more useful and reliable," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[101] T. Qin, T.-Y. Liu, X.-D. Zhang, Z. Chen, and W.-Y. Ma, "A study of relevance propagation for web search," in *SIGIR 2005*, pp. 408–415, 2005.

[102] T. Qin, T.-Y. Liu, X.-D. Zhang, M.-F. Tsai, D.-S. Wang, and H. Li, "Query-level loss functions for information retrieval," *Information Processing and Management*, vol. 44, pp. 838–855, 2007.

[103] T. Qin, T.-Y. Liu, X.-D. Zhang, D. Wang, and H. Li, "Learning to rank relational objects and its application to web search," in *WWW 2008*, pp. 407–416, 2008.

[104] T. Qin, T.-Y. Liu, X.-D. Zhang, D.-S. Wang, and H. Li, "Global ranking using continuous conditional random fields," in *NIPS 2008*, pp. 1281–1288, 2008.

[105] F. Radlinski and T. Joachims, "Query chain: Learning to rank from implicit feedback," in *KDD 2005*, pp. 239–248, 2005.

[106] F. Radlinski and T. Joachims, "Active exploration for learning rankings from clickthrough data," in *KDD 2007*, 2007.

[107] F. Radlinski, R. Kleinberg, and T. Joachims, "Learning diverse rankings with multi-armed bandits," in *ICML 2008*, pp. 784–791, 2008.

[108] S. Rajaram and S. Agarwal, "Generalization bounds for k-partite ranking," in *NIPS 2005 WorkShop on Learning to Rank*, 2005.

[109] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "Learning to rank by a neural-based sorting algorithm," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[110] S. Robertson and H. Zaragoza, "On rank-based effectiveness measures and optimization," *Information Retrieval*, vol. 10, pp. 321–339, 2007.

[111] S. E. Robertson, "Overview of the okapi projects," *Journal of Documentation*, vol. 53, pp. 3–7, 1997.

[112] J. J. Rochhio, "Relevance feedback in information retrieval," *The SMART Retrieval System — Experiments in Automatic Document Processing*, pp. 313–323, 1971.

[113] A. Shakery and C. Zhai, "A probabilistic relevance propagation model for hypertext retrieval," in *CIKM 2006*, pp. 550–558, 2006.

[114] A. Shashua and A. Levin, "Ranking with large margin principles: Two approaches," in *NIPS 2002*, pp. 937–944, 2002.

[115] J. Shawe-Taylor and N. Cristianini, *Kernel Methods for Pattern Analysis*. Cambridge University Press, 2004.

[116] A. Singhal, "Modern information retrieval: A brief overview," *IEEE Data Engineering Bulletin*, vol. 24, pp. 35–43, 2001.

[117] T. Tao and C. Zhai, "Regularized estimation of mixture models for robust pseudo-relevance feedback," in *SIGIR 2006*, pp. 162–169, 2006.

[118] T. Tao and C. Zhai, "An exploration of proximity measures in information retrieval," in *SIGIR 2007*, pp. 295–302, 2007.

[119] M. Taylor, J. Guiver, S. Robertson, and T. Minka, "SoftRank: Optimising non-smooth rank metrics," in *WSDM 2008*, pp. 77–86, 2008.

[120] M. Taylor, H. Zaragoza, N. Craswell, S. Robertson, and C. J. Burges, "Optimisation methods for ranking functions with multiple parameters," in *CIKM 2006*, pp. 585–593, 2006.

[121] A. Trotman, "Learning to rank," *Information Retrieval*, vol. 8, pp. 359–381, 2005.

[122] M.-F. Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma, "FRank: A ranking method with fidelity loss," in *SIGIR 2007*, pp. 383–390, 2007.

[123] I. Tsochantaridis, T. Hofmann, T. Joachims, and Y. Altun, "Support vector machine learning for interdependent and structured output space," in *ICML 2004*, pp. 104–111, 2004.

[124] N. Usunier, M.-R. Amini, and P. Gallinari, "Generalization error bounds for classifiers trained with interdependent data," in *NIPS 2005*, pp. 1369–1376, 2005.

[125] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[126] V. N. Vapnik, *Statistical Learning Theory*. Wiley-Interscience, 1998.

[127] A. Veloso, H. M. de Almeida, M. A. Gonçalves, and W. Meira, Jr., "Learning to rank at query-time using association rules," in *SIGIR 2008*, pp. 267–274, 2008.

[128] W. Xi, J. Lind, and E. Brill, "Learning effective ranking functions for newsgroup search," in *SIGIR 2004*, pp. 394–401, 2004.

[129] F. Xia, T.-Y. Liu, J. Wang, W. Zhang, and H. Li, "Listwise approach to learning to rank — Theorem and algorithm," in *ICML 2008*, pp. 1192–1199, 2008.

[130] J. Xu, Y. Cao, H. Li, and M. Zhao, "Ranking definitions with supervised learning methods," in *WWW 2005*, pp. 811–819, New York, NY, USA: ACM Press, 2005.

[131] J. Xu and H. Li, "AdaRank: A boosting algorithm for information retrieval," in *SIGIR 2007*, pp. 391–398, 2007.

[132] J. Xu, T.-Y. Liu, M. Lu, H. Li, and W.-Y. Ma, "Directly optimizing IR evaluation measures in learning to rank," in *SIGIR 2008*, pp. 107–114, 2008.

[133] G.-R. Xue, Q. Yang, H.-J. Zeng, Y. Yu, and Z. Chen, "Exploiting the hierarchical structure for link analysis," in *SIGIR 2005*, pp. 186–193, 2005.

[134] J.-Y. Yeh and J.-Y. Lin, and etc, "Learning to rank for information retrieval using genetic programming," in *SIGIR 2007 Workshop in Learning to Rank for Information Retrieval*, 2007.

[135] H. Yu, "SVM selective sampling for ranking with application to data retrieval," in *KDD 2005*, pp. 354–363, 2005.

[136] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *SIGIR 2007*, pp. 271–278, 2007.

[137] Y. Yue and T. Joachims, "Predicting diverse subsets using structural SVM," in *ICML 2008*, pp. 1224–1231, 2008.

[138] C. Zhai, "Language models," *Foundations and Trends in Information Retrieval*, 2008.

[139] C. Zhai, W. W. Cohen, and J. Lafferty, "Beyond independent relevance: Methods and evaluation metrics for subtopic retrieval," in *SIGIR 2003*, pp. 10–17, 2003.

[140] C. Zhai and J. Lafferty, "Model-based feedback in the language modeling approach to information retrieval," in *CIKM 2001*, pp. 403–410, 2001.

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

[142] Z. Zheng, H. Zha, and G. Sun, "Query-level learning to rank using isotonic regression," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.

[143] K. Zhou, G.-R. Xue, H. Zha, and Y. Yu, "Learning to rank with ties," in *SIGIR 2008*, pp. 275–282, 2008.

[144] O. Zoeter, M. Taylor, E. Snelson, J. Guiver, N. Craswell, and M. Szummer, "A decision theoretic framework for ranking using implicit feedback," in *SIGIR 2008 Workshop on Learning to Rank for Information Retrieval*, 2008.