Chapter 1
Introduction

Abstract In this chapter, we give a brief introduction to learning to rank for information retrieval. Specifically, we first introduce the ranking problem by taking document retrieval as an example. Second, conventional ranking models proposed in the literature of information retrieval are reviewed, and widely used evaluation measures for ranking are mentioned. Third, the motivation of using machine learning technology to solve the problem of ranking is given, and existing learning-to-rank algorithms are categorized and briefly depicted.

1.1 Overview

With the fast development of the Web, every one of us is experiencing a flood of information. A study[1] conducted in 2005 estimated the World Wide Web to contain 11.5 billion pages by January 2005. In the same year, Yahoo![2] announced that its search engine index contained more than 19.2 billion documents. It was estimated by http://www.worldwidewebsize.com/ that there were about 25 billion pages indexed by major search engines as of October 2008. Recently, the Google blog[3] reported that about one trillion web pages have been seen during their crawling and indexing. According to the above information, we can see that the number of webpages is growing very fast. Actually, the same story also happens to the number of websites. According to a report,[4] the evolution of websites from 2000 to 2007 is shown in Fig. 1.1.

The extremely large size of the Web makes it generally impossible for common users to locate their desired information by browsing the Web. As a consequence, efficient and effective information retrieval has become more important than ever, and the search engine (or information retrieval system) has become an essential tool for many people.

A typical search engine architecture is shown in Fig. 1.2. As can be seen from the figure, there are in general six major components in a search engine: crawler, parser,

[1] http://www.cs.uiowa.edu/~asignori/web-size/.

[2] http://www.iht.com/articles/2005/08/15/business/web.php.

[3] http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html.

[4] http://therawfeed.com/.

T.-Y. Liu, *Learning to Rank for Information Retrieval*,
DOI 10.1007/978-3-642-14267-3_1,

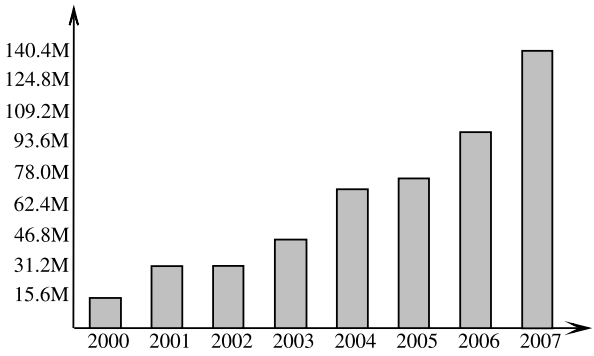

Fig. 1.1 Number of websites

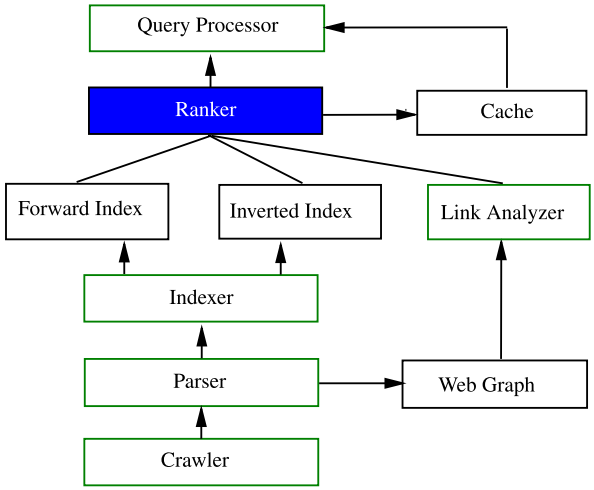

Fig. 1.2 Search engine overview

indexer, link analyzer, query processor, and ranker. The crawler collects webpages and other documents from the Web, according to some prioritization strategies. The parser analyzes these documents and generates index terms and a hyperlink graph for them. The indexer takes the output of the parser and creates the indexes or data structures that enable fast search of the documents. The link analyzer takes the Web graph as input, and determines the importance of each page. This importance can be used to prioritize the recrawling of a page, to determine the tiering, and to serve as a feature for ranking. The query processor provides the interface between users and search engines. The input queries are processed (e.g., removing stop words, stemming, etc.) and transformed to index terms that are understandable by search engines. The ranker, which is a central component, is responsible for the matching between processed queries and indexed documents. The ranker can directly take the queries and documents as inputs and compute a matching score using some heuristic formulas, and can also extract some features for each query-document pair and combine these features to produce the matching score.

Partly because of the central role of ranker in search engines, great attention has been paid to the research and development of ranking technologies. Note that ranking is also the central problem in many other information retrieval applications, such as collaborative filtering [30], question answering [3, 71, 79, 83], multimedia retrieval [86, 87], text summarization [53], and online advertising [16, 49]. In this book, we will mainly take document retrieval in search as an example. Note that even document retrieval is not a narrow task. There are many different ranking scenarios of interest for document retrieval. For example, sometimes we need to rank documents purely according to their relevance with regards to the query. In some other cases, we need to consider the relationships of similarity [73], website structure [22], and diversity [91] between documents in the ranking process. This is also referred to as relational ranking [61].

To tackle the problem of document retrieval, many heuristic ranking models have been proposed and used in the literature of information retrieval. Recently, given the amount of potential training data available, it has become possible to leverage machine learning 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 information retrieval. First, a large number of learning-to-rank papers have been published at top conferences on machine learning and information retrieval. Examples include [4, 7–10, 15, 18, 20, 21, 25, 26, 31, 37, 44, 47, 55, 58, 62, 68, 72, 76, 82, 88, 89]. There have been multiple sessions in recent SIGIR conferences dedicated for the topic on learning to rank. Second, benchmark datasets like LETOR [48] have been released to facilitate the research on learning to rank.[5] Many research papers on learning to rank have used these datasets for their experiments, which make their results easy to compare. Third, several activities regarding learning to rank have been organized. For example, the workshop series on Learning to Rank for Information Retrieval (2007–2009), the workshop on Beyond Binary Relevance: Preferences, Diversity, and Set-Level Judgments (2008), and the workshop on Redundancy, Diversity, and Interdependent Document Relevance (2009) have been organized at SIGIR. A special issue on learning to rank has been organized at Information Retrieval Journal (2009). In Table 1.1, we have listed the major activities regarding learning to rank in recent years. Active participation of researchers in these activities has demonstrated the continued interest from the research community on the topic of learning to rank. Fourth, learning to rank has also become a key technology in the industry. Several major search engine companies are using learning-to-rank technologies to train their ranking models.[6]

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

[5] http://research.microsoft.com/~LETOR/.

[6] http://glinden.blogspot.com/2005/06/msn-search-and-learning-to-rank.html, http://www.ysearchblog.com/2008/07/09/boss-the-next-step-in-our-open-search-ecosystem/.

Table 1.1 Recent events on learning to rank

Year	Event
2010	SIGIR Tutorial on Learning to Rank for Information Retrieval *Given by Tie-Yan Liu* Learning to Rank Challenge *Organized by Yahoo! Labs* Microsoft Learning to Rank Dataset *Released by Microsoft Research Asia* ICML Workshop on Learning to Rank *Organized by Olivier Chapelle, Yi Chang, and Tie-Yan Liu*
2009	Special Issue on Learning to Rank at Information Retrieval Journal *Edited by Tie-Yan Liu, Thorsten Joachims, Hang Li, and Chengxiang Zhai* NIPS Workshop on Learning with Orderings *Organized by Tiberio Caetano, Carlos Guestrin, Jonathan Huang, Guy Lebanon, Risi Kondor, and Marina Meila* NIPS Workshop on Advances in Ranking *Organized by Shivani Agarwal, Chris J.C. Burges, and Koby Crammer* SIGIR Workshop on Redundancy, Diversity, and Interdependent Document Relevance *Organized by Paul N. Bennett, Ben Carterette, Thorsten Joachims, and Filip Radlinski* SIGIR Workshop on Learning to Rank for Information Retrieval *Organized by Hang Li, Tie-Yan Liu, and ChengXiang Zhai* LETOR Dataset 4.0 *Released by Microsoft Research Asia* ACL-IJNLP Tutorial on Learning to Rank *Given by Hang Li* WWW Tutorial on Learning to Rank for Information Retrieval *Given by Tie-Yan Liu* ICML Tutorial on Machine Learning in Information Retrieval: Recent Successes and New Opportunities *Given by Paul Bennett, Misha Bilenko, and Kevyn Collins-Thompson*
2008	SIGIR Workshop on Learning to Rank for Information Retrieval *Organized by Hang Li, Tie-Yan Liu, and ChengXiang Zhai* LETOR Dataset 3.0 *Released by Microsoft Research Asia* SIGIR Tutorial on Learning to Rank for Information Retrieval *Given by Tie-Yan Liu* WWW Tutorial on Learning to Rank for Information Retrieval *Given by Tie-Yan Liu*
2007	SIGIR Workshop on Learning to Rank for Information Retrieval *Organized by Thorsten Joachims, Hang Li, Tie-Yan Liu, and ChengXiang Zhai* WWW Tutorial on Learning to Rank in Vector Spaces and Social Networks *Given by Soumen Chakrabarti* LETOR Dataset 1.0 and 2.0 *Released by Microsoft Research Asia*
2006	ICML Workshop on Learning in Structured Output Space *Organized by Ulf Brefeld, Thorsten Joachims, Ben Taskar, and Eric P. Xing*
2005	NIPS Workshop on Learning to Rank *Organized by Shivani Agarwal, Corinna Cortes, and Ralf Herbrich*

- In 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 speaking, which of those many learning-to-rank algorithms performs the best?
- Theoretically speaking, is ranking a new machine learning problem, or can it be simply reduced to existing machine learning 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?

The above questions have been brought to the attention of the information retrieval and machine learning communities in a variety of contexts, especially during recent years. The aim of this book is to answer these questions. Needless to say, the comprehensive understanding of the task of ranking in information retrieval is the first step to find the right answers. Therefore, we will first give a brief review of ranking in information retrieval, and then formalize the problem of learning to rank so as to set the stage for the upcoming detailed discussions.

1.2 Ranking in Information Retrieval

In this section, we will briefly review representative ranking models in the literature of information retrieval, and introduce how these models are evaluated.

1.2.1 Conventional Ranking Models

In the literature of information retrieval, many ranking models have been proposed [2]. They can be roughly categorized as relevance ranking models and importance ranking models.

1.2.1.1 Relevance Ranking Models

The goal of a relevance ranking model is to produce a ranked list of documents according to the relevance between these documents and the query. Although not necessary, for ease of implementation, the relevance ranking model usually takes each individual document as an input, and computes a score measuring the matching between the document and the query. Then all the documents are sorted in descending order of their scores.

The early relevance ranking models retrieve documents based on the occurrences of the query terms in the documents. Examples include the *Boolean model* [2]. Basically these models can 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 [2]. 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 the query and the documents, TF-IDF weighting has been widely used.[7] The TF of a 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:

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

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) [23] tries to avoid this assumption. In particular, Singular Value Decomposition (SVD) is used to linearly transform the original feature space to a "latent semantic space". Similarity in this new space is then used to define the relevance between the query and the documents.

As compared with the above, models based on the probabilistic ranking principle [50] have garnered more attention and achieved more success in past decades. The famous ranking models like the *BM25 model*[8] [65] and the *language model for information retrieval (LMIR)*, 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 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 a query q, containing terms $t_1, \ldots, t_M$, the BM25 score of a document d is computed as

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

where $TF(t,d)$ is the term frequency of t in document d, $LEN(d)$ is the length (number of words) of document d, and $avdl$ is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, $IDF(t)$ is the IDF weight of the term t, computed by using (1.1), for example.

LMIR [57] is an application of the statistical language model on information retrieval. A statistical language model assigns a probability to a sequence of terms. When used in information retrieval, a language model is associated with a document.

[7] Note that there are many different definitions of TF and IDF in the literature. Some are purely based on the frequency and the others include smoothing or normalization [70]. Here we just give some simple examples to illustrate the main idea.

[8] The name of the actual model is BM25. In the right context, however, it is usually referred to as "OKapi BM25", since the OKapi system was the first system to implement this model.

With query q as input, documents are ranked based on the query likelihood, or the probability that the document's language model will generate the terms in the query (i.e., $P(q|d)$). By further assuming the independence between 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:

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

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 LMIR, some of them even go beyond the query likelihood retrieval model (e.g., the models based on K-L divergence [92]). We would not like to introduce more about it, and readers are encouraged to read the tutorial authored by Zhai [90].

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 [74] take the proximity of the query terms into consideration, and some others consider the relationship between documents in terms of content similarity [73], hyperlink structure [67], website structure [60], and topic diversity [91].

1.2.1.2 Importance Ranking Models

In the literature of information retrieval, there are also many models that rank documents based on their own importance. We will take PageRank 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 webpages. In the general case, the PageRank value for any page d_u can be expressed as

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

That is, the PageRank value for a 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 (1.4), a smoothing term is introduced. When a 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

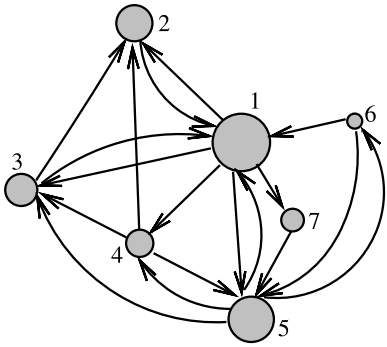

Fig. 1.3 Illustration of PageRank

page uniformly. This small probability can be represented by $(1-\alpha)$, where α is called the damping factor. Accordingly, the PageRank model is refined as follows[9]:

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

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

The above process of computing PageRank can be vividly illustrated by Fig. 1.3. By using the above formula, it is not difficult to discover that the PageRank values of the seven nodes in the graph are 0.304, 0.166, 0.141, 0.105, 0.179, 0.045, and 0.060, respectively. The sizes of the circles in the figure are used to represent the PageRank values of the nodes.

Many algorithms have been developed in order to further improve the accuracy and efficiency of PageRank. Some work focuses on the speed-up of the computation [1, 32, 51], while others focus on the refinement and enrichment of the model. For example, topic-sensitive PageRank [35] and query-dependent PageRank [64] introduce topics and assume that the endorsement from a page belonging to the same topic is larger than that from a page belonging to a different topic. Other variations of PageRank include those modifying the 'personalized vector' [34], changing the 'damping factor' [6], and introducing inter-domain and intra-domain link weights [46]. Besides, there is also some work on the theoretic issues of PageRank [5, 33]. Langville et al. [46] provide a good survey on PageRank and its related work.

Algorithms that can generate robust importance ranking against link spam have also been proposed. For example, TrustRank [29] is an importance ranking algorithm that takes into consideration the reliability of web pages when calculating the importance of pages. In TrustRank, a set of reliable pages are first identified as seed pages. Then the *trust* of a seed page is propagated to other pages on the web link graph. Since the propagation in TrustRank starts from reliable pages, TrustRank can be more spam-resistant than PageRank.

[9] If there are web pages without any inlinks (which is usually referred to as dangling nodes in the graph), some additional heuristics is needed to avoid rank leak.

1.2.2 Query-Level Position-Based Evaluations

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.

Actually, evaluation has played a very important role in the history of information retrieval. Information retrieval is an empirical science; and it has been a leader in computer science in understanding the importance of evaluation and benchmarking. Information retrieval has been well served by the Cranfield experimental methodology [81], which is based on sharable document collections, information needs (queries), and relevance assessments. By applying the Cranfield paradigm to document retrieval, the corresponding evaluation process can be described 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.[10] In this method a pool of possibly relevant documents is created by taking a sample of documents selected by the 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.

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

1. *Relevance degree*: Human annotators specify whether a document is relevant or not to the query (i.e., binary judgment), or further specify 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.
2. *Pairwise preference*: Human annotators specify whether a document is more relevant than the other with regards to a query. For example, if document d_u is judged to be more relevant than document d_v, we give the judgment $l_{u,v} = 1$;

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

otherwise, $l_{u,v} = -1$. That is, this kind of judgment captures the relative preference between documents.[11]

3. *Total order*: Human annotators specify the total order of the documents with respect to a query. For the set 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.

Among the aforementioned three kinds of judgments, the first kind is the most popularly used judgment. This is partially because this kind of judgment is easy to obtain. Human assessors only need to look at each individual document to produce the judgment. Comparatively, obtaining the third kind of judgment is the most costly. Therefore, in this book, we will mostly use the first kind of judgment as an example to perform the discussions.

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 the 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 as relevant and the "truly" irrelevant documents as irrelevant.

Since manual judgment is always 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.

With the relevance judgment, several evaluation measures have been proposed and used in the literature of information retrieval. 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. In order to better understand these measures, we use the example shown in Fig. 1.4 to perform some quantitative calculation with respect to each measure. In the example, there are three documents retrieved for the query "learning to rank", and binary judgment on the relevance of each document is provided.

Most of the evaluation measures are defined first for each query, as a function of the ranked list π given by the ranking model and the relevance judgment. Then the measures are averaged over all the queries in the test set.

As will be seen below, the maximum values for some evaluation measures, such as MRR, MAP, and NDCG are one. Therefore, we can consider one minus these measures (e.g., $(1 - \text{MRR})$, $(1 - \text{NDCG})$, and $(1 - \text{MAP})$) as ranking errors. For ease of reference, we call them *measure-based ranking errors*.

Mean Reciprocal Rank (MRR) For query q, the rank position of its first relevant document is denoted as r_1. Then $\frac{1}{r_1}$ is defined as MRR for query q. It is clear that documents ranked below r_1 are not considered in MRR.

[11] This kind of judgment can also be mined from click-through logs of search engines [41, 42, 63].

Fig. 1.4 Retrieval result for query "learning to rank"

Consider the example as shown in Fig. 1.4. Since the first document in the retrieval result is relevant, $r_1 = 1$. Therefore, MRR for this query equals 1.

Mean Average Precision (MAP) To define MAP [2], one needs to define Precision at position k ($P@k$) first. Suppose we have binary judgment for the documents, i.e., the label is one for relevant documents and zero for irrelevant documents. Then $P@k$ is defined as

$$P@k(\pi, l) = \frac{\sum_{t \le k} I_{\{l_{\pi^{-1}(t)}=1\}}}{k}, \tag{1.6}$$

where $I_{\{\cdot\}}$ is the indicator function, and $\pi^{-1}(j)$ denotes the document ranked at position j of the list π.

Then the Average Precision (AP) is defined by

$$\text{AP}(\pi, l) = \frac{\sum_{k=1}^{m} P@k \cdot I_{\{l_{\pi^{-1}(k)}=1\}}}{m_1}, \tag{1.7}$$

where m is the total number of documents associated with query q, and m_1 is the number of documents with label one.

The mean value of AP over all the test queries is called mean average precision (MAP).

Consider the example as shown in Fig. 1.4. Since the first document in the retrieval result is relevant, it is clear $P@1 = 1$. Because the second document is irrelevant, we have $P@2 = \frac{1}{2}$. Then for $P@3$, since the third document is relevant, we obtain $P@3 = \frac{2}{3}$. Then $AP = \frac{1}{2}(1 + \frac{2}{3}) = \frac{5}{6}$.

Discounted Cumulative Gain (DCG) DCG [39, 40] is an evaluation measure that 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:

$$\text{DCG@}k(\pi, l) = \sum_{j=1}^{k} G(l_{\pi^{-1}(j)})\eta(j), \tag{1.8}$$

where $G(\cdot)$ is the rating of a document (one usually sets $G(z) = (2^z - 1)$), and $\eta(j)$ is a position discount factor (one usually sets $\eta(j) = 1/\log(j + 1)$).

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

$$\text{NDCG@}k(\pi, l) = \frac{1}{Z_k} \sum_{j=1}^{k} G(l_{\pi^{-1}(j)})\eta(j). \tag{1.9}$$

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

If we want to consider all the labeled documents (whose number is m) to compute NDCG, we will get NDCG@m (and sometimes NDCG for short).

Consider the example as shown in Fig. 1.4. It is easy to see that DCG@3 = 1.5, and $Z_3 = 1.63$. Correspondingly, NDCG = NDCG@3 = $\frac{1.5}{1.63} = 0.92$.

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 as a measure. For example, when the weighted Kendall's τ [43] is used, the rank correlation measures the weighted pairwise inconsistency between two lists. Its definition is given by

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

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

Note that the above rank correlation is defined with the assumption that the judgment is given as total order π_l. When the judgments are in other forms, we need to introduce the concept of an equivalent permutation set to generalize the above definition. That is, there will be multiple permutations that are consistent with the judgment in terms of relevance degrees or pairwise preferences. The set of such permutations is called the equivalent permutation set, denoted as Ω_l.

Specifically, for the judgment in terms of relevance degree, the equivalent permutation set is defined as follows.

$$\Omega_l = \{\pi_l | u < v, \text{ if } l_{\pi_l^{-1}(u)} \succ l_{\pi_l^{-1}(v)}\}.$$

Similarly, for the judgment in terms of pairwise preferences, Ω_l is defined as below.

$$\Omega_l = \{\pi_l | u < v, \text{ if } l_{\pi_l^{-1}(u), \pi_l^{-1}(v)} = 1\}.$$

Then, we can refine the definition of weighted Kendall's τ as follows.

$$\tau_K(\pi, \Omega_l) = \max_{\pi_l \in \Omega_l} \tau_K(\pi, \pi_l). \tag{1.11}$$

Consider the example as shown in Fig. 1.4. It is clear that there are multiple permutations consistent with the labels: (1, 3, 2) and (3, 1, 2). Since the ranking result is (1, 2, 3), it is not difficult to compute that the τ_K between (1, 2, 3) and

(1, 3, 2) is $\frac{2}{3}$ and the τ_K between (1, 2, 3) and (3, 1, 2) is $\frac{1}{3}$. Therefore, we can obtain that $\tau_K(\pi, \Omega_l) = \frac{2}{3}$ in this case.

To summarize, there are some common properties in these evaluation measures.[12]

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 discontinuous 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 Sect. 4.2.

Note that although when designing ranking models, many researchers have taken the assumption that the ranking models can assign a score to each query-document pair independently of other documents; when performing evaluation, all the documents associated with a query are considered together. Otherwise, one cannot determine the rank position of a document and the aforementioned measures cannot be defined.

1.3 Learning to Rank

Many ranking models have been introduced in the previous section, most of which contain parameters. For example, there are parameters k_1 and b in BM25 (see (1.2)), parameter λ in LMIR (see (1.3)), and parameter α in PageRank (see (1.5)). In order to get a reasonably good ranking performance (in terms of evaluation measures), one needs to tune these parameters using a validation set. Nevertheless, parameter tuning is far from trivial, especially considering that evaluation measures are discontinuous 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 regarding 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.

[12]Note that this is not a complete introduction of evaluation measures for information retrieval. There are several other measures proposed in the literature, some of which even consider the novelty and diversity in the search results in addition to the relevance. One may want to refer to [2, 17, 56, 91] for more information.

While information retrieval researchers were suffering from these problems, machine learning has been demonstrating its effectiveness in automatically tuning parameters, in combining multiple pieces of evidence, and in avoiding over-fitting. Therefore, it seems quite promising to adopt machine learning technologies to solve the aforementioned problems in ranking.

1.3.1 Machine Learning Framework

In many machine learning researches (especially discriminative learning), attention has been paid to the following key components.[13]

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 machine learning.[14] The first is the output space of the task, which is highly dependent on the application. For example, in regression, the output space is the space of real numbers $\mathcal{R}$; in classification it is the set of discrete categories $\{1, 2, \ldots, K\}$. The second is the output space to facilitate the learning process. This may differ from the output space of the task. For example, when one uses regression technologies to solve the problem of classification, 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. That is, the functions 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 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 machine learning, 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.

We plot the relationship between these four components in Fig. 1.5 for ease of understanding.

[13] For a more comprehensive introduction to the machine learning literature, please refer to [54].

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

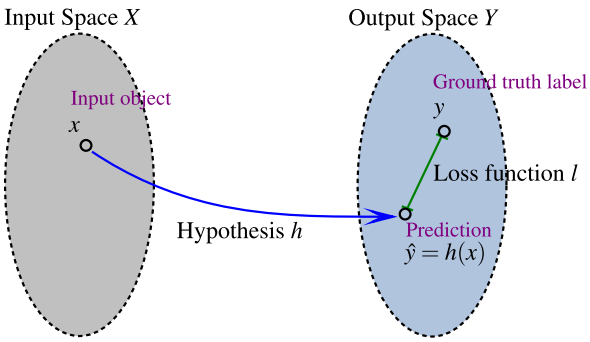

Fig. 1.5 Machine learning framework

1.3.2 Definition of Learning to Rank

In recent years, more and more machine learning 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 information retrieval.

In general, we call all those methods that use machine learning technologies to solve the problem of ranking "learning-to-rank" methods.[15] Examples include the work on relevance feedback[16] [24, 66] and automatically tuning the parameters of existing information retrieval models [36, 75]. However, most of the state-of-the-art learning-to-rank algorithms learn the optimal way of combining features exacted from query-document pairs through discriminative training. Therefore, in this book we define learning to rank in a more narrow and specific way to better summarize these algorithms. That is, we call those ranking methods that have the following two properties learning-to-rank methods.

Feature Based "*Feature based*" means that all the documents under investigation are represented by feature vectors,[17] 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 outputs of the BM25 model and the PageRank model, and even the relationship between this document and other documents. These features can be extracted from the index of a

[15] In the literature of machine learning, there is a topic named label ranking. It predicts the ranking of multiple class labels for an individual document, but not the ranking of documents. In this regard, it is largely different from the task of ranking for information retrieval.

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

[17] Note that, in this book, 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.

search engine (see Fig. 1.2). If one wants to know more about widely used features, please refer to Tables 10.2 and 10.3 in Chap. 10.

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 works on automatically tuning the parameters of existing models [36, 75] are not categorized as "learning-to-rank" methods.

The capability of combining a large number of features is an advantage of learning-to-rank methods. It is easy to incorporate any new progress on a 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 the complex information needs of Web users.

Discriminative Training "*Discriminative training*" means that 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 own input space, output space, hypothesis space, and loss function.

In the literature of machine learning, discriminative methods have been widely used to combine different kinds of features, without the necessity of defining a probabilistic framework to represent the generation of objects and the correctness of prediction. In this sense, previous works that train generative ranking models are not categorized as "learning-to-rank" methods in this book. If one has interest in such works, please refer to [45, 52, 93], 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. It is very important to automatically learn from the feedback and constantly improve the ranking mechanism.

Due to the aforementioned two characteristics, learning to rank has been widely used in commercial search engines,[18] and has also attracted great attention from the academic research community.

1.3.3 Learning-to-Rank Framework

Figure 1.6 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 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

[18] 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.

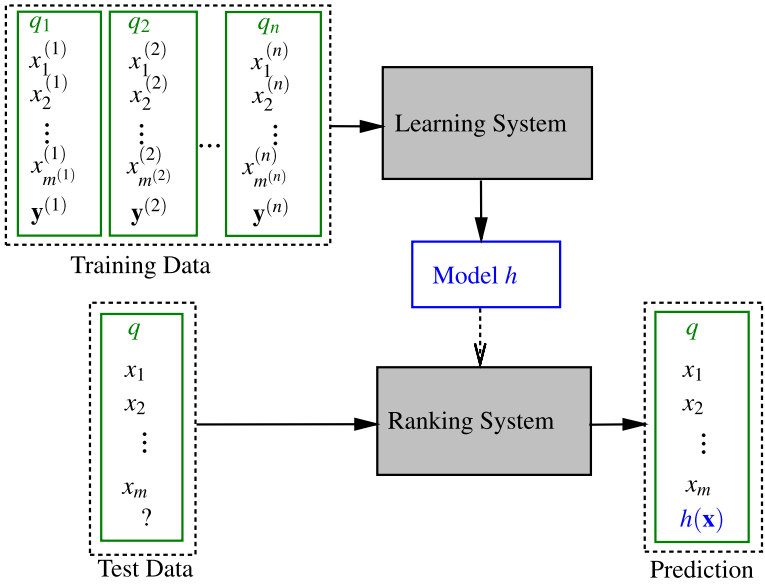

Fig. 1.6 Learning-to-rank framework

$m^{(i)}$ is the number of documents associated with query q_i), and the corresponding relevance judgments.[19] 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[20] 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 and return the corresponding ranked list to the user as the response to her/his query.

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 machine learning, into three approaches: the pointwise approach, the pairwise approach, and the listwise approach. Different approaches model the process of learning to rank in different ways. That is, they may define different input and output spaces, use different hypotheses, and employ different loss functions.

[19] Please distinguish the judgment for evaluation and the judgment for constructing the training set, although the process may be very similar.

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

1.3.3.1 The Pointwise Approach

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

The *output space* contains the relevance degree of each single document. Different kinds of judgments can be converted to ground truth labels in terms of relevance degree:

- 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 pairwise preference $l_{u,v}$, one can get the ground truth label by counting the frequency of a document beating other documents.
- If the judgment is given as the total order π_l, one can get the ground truth label by using a mapping function. For example, the position of the document in π_l can be used as the ground truth.

The *hypothesis* space contains functions that take the feature vector of a document as input and predict the relevance degree of the document. We usually call such a function f the scoring function. 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, respectively (see Chap. 2). Therefore the corresponding regression loss, classification loss, and ordinal regression loss are used as the loss functions.

Example algorithms belonging to the pointwise approach include [13–15, 19–21, 26, 28, 44, 47, 55, 68, 78]. We will introduce some of them in Chap. 2.

Note that the pointwise approach does not consider the inter-dependency between 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 evaluation measures for information retrieval are query level and position based, the pointwise approach has its limitations.

1.3.3.2 The Pairwise Approach

The *input space* of the pairwise approach contains pairs of documents, both represented by feature vectors.

The *output space* contains the pairwise preference (which takes values from $\{+1, -1\}$) between each pair of documents. Different kinds of judgments can be converted to ground truth labels in terms of pairwise preferences:

- If the judgment is given as relevance degree l_j, then the pairwise preference for (x_u, x_v) can be defined as $y_{u,v} = 2 \cdot I_{\{l_u \succ l_v\}} - 1$.
- If the judgment is given directly as pairwise preference, then it is straightforward to set $y_{u,v} = l_{u,v}$.

- If the judgment is given as the 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 input and output the relative order between them. While some pairwise ranking algorithms directly define their hypotheses as such [18], in some other algorithms, the hypothesis is 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 $y_{u,v}$. In many pairwise ranking algorithms, ranking is modeled as a pairwise classification, and the corresponding classification loss on a pair of documents is used as the loss function. When scoring function f is used, the classification loss is usually expressed in terms of the difference $(f(x_u) - f(x_v))$, rather than the thresholded quantity $h(x_u, x_v)$.

Example algorithms belonging to the pairwise approach include [8, 18, 25, 27, 37, 41, 69, 76, 94, 95]. We will introduce some of them in Chap. 3.

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 evaluation measures for information retrieval are query level and position based, we can see a gap between this approach and ranking for information retrieval.

1.3.3.3 The Listwise Approach

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

The *output space* of the listwise approach contains the ranked list (or permutation) of the documents. Different kinds of judgments can be converted to ground truth labels in terms of a ranked list:

- If the judgment is given as relevance degree l_j, then all the permutations that are consistent with the judgment are ground truth permutations. Here we define a permutation π_y as consistent with relevance degree l_j, if $\forall u, v$ satisfying $l_u > l_v$, we always have $\pi_y(u) < \pi_y(v)$. There might be multiple ground truth permutations in this case. We use Ω_y to represent the set of all such permutations.[21]
- If the judgment is given as pairwise preferences, then once again all the permutations that are consistent with the pairwise preferences are ground truth permutations. Here we define a permutation π_y as consistent with preference $l_{u,v}$, if $\forall u, v$ satisfying $l_{u,v} = +1$, we always have $\pi_y(u) < \pi_y(v)$. Again, there might also be multiple ground truth permutations in this case, and we use Ω_y to represent the set of all such permutations.

[21] Similar treatment can be found in the definition of Rank Correlation in Sect. 1.2.2.

- If the judgment is given as the total order π_l, one can straightforwardly define $\pi_y = \pi_l$.

Note that for the listwise approach, the output space that facilitates the learning process is exactly the same as the output space of the task. In this regard, the theoretical analysis on the listwise approach can have a more direct value to 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 multi-variate functions h that operate on a set of documents and predict their permutation. For practical reasons, the hypothesis h is usually implemented with a scoring function f, e.g., $h(\mathbf{x}) = \text{sort} \circ f(\mathbf{x})$. That is, first a scoring function f is used to give a score to each document, and then these documents are sorted in descending order of the scores to produce the desired permutation.

There are two types of *loss functions*, widely used in the listwise approach. For the first type, the loss function is explicitly related to the evaluation measures (which we call the measure-specific loss function), while for the second type, the loss function is not (which we call the non-measure-specific loss function). Note that sometimes it is not very easy to determine whether a loss function is listwise, since some building blocks of a listwise loss may also seem to be pointwise or pairwise. In this book, we mainly distinguish a listwise loss from a pointwise or pairwise loss according to the following criteria:

- A listwise loss function is defined with respect to all the training documents associated with a query.
- A listwise loss function cannot be fully decomposed to simple summation over individual documents or document pairs.
- A listwise loss function emphasizes the concept of a ranked list, and the positions of the documents in the final ranking result are visible to the loss function.

Just because of these properties of the loss function, the listwise approach is in more accordance with the ranking task in information retrieval than the pointwise and pairwise approaches.

Example algorithms that belong to the listwise approach include [10–12, 38, 59, 72, 80, 82, 84, 85, 88, 89, 96]. We will introduce some of them in Chap. 4.

To sum up, we list the key components for each approach to learning to rank in Table 1.2. In this table, for simplicity, we assume the use of a scoring function f in the hypothesis of all the approaches, although it is not always necessary to be the case.

It should be noted that although the scoring function is a kind of "pointwise" function, it is not to say that all the approaches are in nature pointwise approaches. The categorization of the aforementioned three approaches is mainly based on the four pillars of machine learning. That is, different approaches regard the same training data as in different input and output spaces, and use different loss functions and hypotheses. Accordingly, they will have difference theoretical properties. We will make more discussions on this in Part VI, with the introduction of a new theory called the *statistical learning theory for ranking*.

Table 1.2 Summary of approaches to learning to rank

Category	Pointwise		
	Regression	Classification	Ordinal regression
Input space	Single document x_j		
Output space	Real value y_j	Non-ordered category y_j	Ordered category y_j
Hypothesis space	$f(x_j)$	Classifier on $f(x_j)$	$f(x_j)$ + thresholding
Loss function	$L(f; x_j, y_j)$		

Category	Pairwise	Listwise	
	–	Non-measure-specific	Measure-specific
Input space	Document pair (x_u, x_v)	Set of documents $\mathbf{x} = \{x_j\}_{j=1}^m$	
Output space	Preference $y_{u,v}$	Ranked list π_y	
Hypothesis space	$2 \cdot I_{\{f(x_u) > f(x_v)\}} - 1$	$sort \circ f(\mathbf{x})$	
Loss function	$L(f; x_u, x_v, y_{u,v})$	$L(f; \mathbf{x}, \pi_y)$	

1.4 Book Overview

In the rest of this book, we will first give a comprehensive review on the major approaches to learning to rank in Chaps. 2, 3 and 4. For each approach, we will present the basic framework, give example algorithms, and discuss its advantages and disadvantages. It is noted that different loss functions are used in different approaches, while the same evaluation measures are used for testing their performances. Then a natural question is concerning the relationship between these loss functions and the evaluation measures. The investigation on this can help us explain the empirical results of these approaches. We will introduce such investigations in Chap. 5.

Considering that there have been some recent advances in learning to rank that cannot be simply categorized into the three major approaches, we use a separate part to introduce them. These include relational ranking, query-dependent ranking, transfer ranking, and semi-supervised ranking. Details can be found in Chaps. 6, 7, 8 and 9.

Next we introduce the benchmark datasets for the research on learning to rank in Chaps. 10 and 12, and discuss the empirical performances of typical learning-to-rank algorithms on these datasets in Chap. 11.

Then in order to fit the needs of practitioners of learning to rank, we introduce some practical issues regarding learning to rank in Chap. 13, e.g., how to mine relevance judgment from search logs, and how to select documents and features for effective training. In addition, we will also show several examples of applying learning-to-rank technologies to solve real information retrieval problems in Chap. 14.

"Nothing is more practical than theory" [77]. After introducing the algorithms and their applications, we will turn to the theoretical part of learning to rank. In particular, we will discuss the theoretical guarantee of achieving good ranking performance on unseen test data by minimizing the loss function on the training data. This is related to the generalization ability and statistical consistency of ranking methods. We will make discussions on these topics in Chaps. 15, 16, 17 and 18.

In Chaps. 19 and 20, we will summarize the book and present some future research topics.

As for the writing of the book, 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. We have listed some basics of machine learning, probability theory, algebra, and optimization in Chaps. 21 and 22. We also provide some related materials and encourage readers to refer to them in order to obtain a more comprehensive overview of the background knowledge for this book.

Throughout the book, we will use the notation rules as listed in Table 1.3. Here we would like to add one more note. Since in practice the hypothesis h is usually defined with scoring function f, 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 f, we will use $f(w, x)$ instead of $f(x)$ in the discussion, although they actually mean the same thing. We sometimes also refer to w as the ranking model directly if there is no confusion.

1.5 Exercises

1.1 How can one estimate the size of the Web?
1.2 Investigate the relationship between the formula of BM25 and the log odds of relevance.
1.3 List different smooth functions used in LMIR, and compare them.
1.4 Use the view of the Markov process to explain the PageRank algorithm.
1.5 Enumerate all the applications of ranking that you know, in addition to document retrieval.
1.6 List the differences between generative learning and discriminative learning.
1.7 Discuss the connections between different evaluation measures for information retrieval.
1.8 Given text classification as the task, and given linear regression as the algorithms, illustrate the four components of machine learning in this case.
1.9 Discuss the major differences between ranking and classification (regression).
1.10 List the major differences between the three approaches to learning to rank.

Table 1.3 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	$TF(q, d)$
Inverse document frequency of query q	$IDF(q)$
Length of document d	$LEN(d)$
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 associate with query q	π_y
Ground-truth permutation set for documents associate with query q	Ω_y
Original document index of the jth element in permutation π	$\pi^{-1}(j)$
Rank position of document j in permutation π	$\pi(j)$
Number of classes	K
Index of class, or top positions	k
VC dimension of a function class	V
Indicator function	$I_{\{\cdot\}}$
Gain function	$G(\cdot)$
Position discount function	$\eta(\cdot)$

References

1. Amento, B., Terveen, L., Hill, W.: Does authority mean quality? Predicting expert quality ratings of web documents. In: Proceedings of the 23th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000), pp. 296–303 (2000)
2. Baeza-Yates, R., Ribeiro-Neto, B.: Modern Information Retrieval. Addison-Wesley, Reading (1999)

3. Banerjee, S., Chakrabarti, S., Ramakrishnan, G.: Learning to rank for quantity consensus queries. In: Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2009), pp. 243–250 (2009)
4. Bartell, B., Cottrell, G.W., Belew, R.: Learning to retrieve information. In: Proceedings of Swedish Conference on Connectionism (SCC 1995) (1995)
5. Bianchini, M., Gori, M., Scarselli, F.: Inside pagerank. ACM Transactions on Internet Technologies **5**(1), 92–128 (2005)
6. Boldi, P., Santini, M., Vigna, S.: Pagerank as a function of the damping factor. In: Proceedings of the 14th International Conference on World Wide Web (WWW 2005), pp. 557–566. ACM, New York (2005)
7. Burges, C.J., Ragno, R., Le, Q.V.: Learning to rank with nonsmooth cost functions. In: Advances in Neural Information Processing Systems 19 (NIPS 2006), pp. 395–402 (2007)
8. Burges, C.J., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 89–96 (2005)
9. Cao, Y., Xu, J., Liu, T.Y., Li, H., Huang, Y., Hon, H.W.: Adapting ranking SVM to document retrieval. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), pp. 186–193 (2006)
10. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: from pairwise approach to listwise approach. In: Proceedings of the 24th International Conference on Machine Learning (ICML 2007), pp. 129–136 (2007)
11. Chakrabarti, S., Khanna, R., Sawant, U., Bhattacharyya, C.: Structured learning for non-smooth ranking losses. In: Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2008), pp. 88–96 (2008)
12. Chapelle, O., Wu, M.: Gradient descent optimization of smoothed information retrieval metrics. Information Retrieval Journal. Special Issue on Learning to Rank 13(3), doi:10.1007/s10791-009-9110-3 (2010).
13. Chu, W., Ghahramani, Z.: Gaussian processes for ordinal regression. Journal of Machine Learning Research **6**, 1019–1041 (2005)
14. Chu, W., Ghahramani, Z.: Preference learning with Gaussian processes. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 137–144 (2005)
15. Chu, W., Keerthi, S.S.: New approaches to support vector ordinal regression. In: Proceedings of the 22nd International Conference on Machine Learning (ICML 2005), pp. 145–152 (2005)
16. Ciaramita, M., Murdock, V., Plachouras, V.: Online learning from click data for sponsored search. In: Proceeding of the 17th International Conference on World Wide Web (WWW 2008), pp. 227–236 (2008)
17. Clarke, C.L., Kolla, M., Cormack, G.V., Vechtomova, O., Ashkan, A., Buttcher, S., MacKinnon, I.: Novelty and diversity in information retrieval evaluation. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 659–666 (2008)
18. Cohen, W.W., Schapire, R.E., Singer, Y.: Learning to order things. In: Advances in Neural Information Processing Systems 10 (NIPS 1997), vol. 10, pp. 243–270 (1998)
19. Cooper, W.S., Gey, F.C., Dabney, D.P.: Probabilistic retrieval based on staged logistic regression. In: Proceedings of the 15th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1992), pp. 198–210 (1992)
20. Cossock, D., Zhang, T.: Subset ranking using regression. In: Proceedings of the 19th Annual Conference on Learning Theory (COLT 2006), pp. 605–619 (2006)
21. Crammer, K., Singer, Y.: Pranking with ranking. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 641–647 (2002)
22. Craswell, N., Hawking, D., Wilkinson, R., Wu, M.: Overview of the trec 2003 web track. In: Proceedings of the 12th Text Retrieval Conference (TREC 2003), pp. 78–92 (2003)
23. Deerwester, S., Dumais, S.T., Furnas, G.W., Landauer, T.K., Harshman, R.: Indexing by latent semantic analysis. Journal of the American Society for Information Science **41**, 391–407 (1990)

24. Drucker, H., Shahrary, B., Gibbon, D.C.: Support vector machines: relevance feedback and information retrieval. Information Processing and Management **38**(3), 305–323 (2002)
25. Freund, Y., Iyer, R., Schapire, R., Singer, Y.: An efficient boosting algorithm for combining preferences. Journal of Machine Learning Research **4**, 933–969 (2003)
26. Fuhr, N.: Optimum polynomial retrieval functions based on the probability ranking principle. ACM Transactions on Information Systems **7**(3), 183–204 (1989)
27. Gao, J., Qi, H., Xia, X., Nie, J.: Linear discriminant model for information retrieval. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), pp. 290–297 (2005)
28. Gey, F.C.: Inferring probability of relevance using the method of logistic regression. In: Proceedings of the 17th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1994), pp. 222–231 (1994)
29. Gyongyi, Z., Garcia-Molina, H., Pedersen, J.: Combating web spam with trustrank. In: Proceedings of the 30th International Conference on Very Large Data Bases (VLDB 2004), pp. 576–587 (2004). VLDB Endowment
30. Harrington, E.F.: Online ranking/collaborative filtering using the perceptron algorithm. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), vol. 20(1), pp. 250–257 (2003)
31. Harrington, E.F.: Online ranking/collaborative filtering using the perceptron algorithm. In: Proceedings of the 20th International Conference on Machine Learning (ICML 2003), pp. 250–257 (2003)
32. Haveliwala, T.: Efficient computation of pageRank. Tech. rep. 1999-31, Stanford University (1999)
33. Haveliwala, T., Kamvar, S.: The second eigenvalue of the Google matrix. Tech. rep., Stanford University (2003)
34. Haveliwala, T., Kamvar, S., Jeh, G.: An analytical comparison of approaches to personalizing pagerank. Tech. rep., Stanford University (2003)
35. Haveliwala, T.H.: Topic-sensitive pagerank. In: Proceedings of the 11th International Conference on World Wide Web (WWW 2002), Honolulu, Hawaii, pp. 517–526 (2002)
36. He, B., Ounis, I.: A study of parameter tuning for term frequency normalization. In: Proceedings of the 12th International Conference on Information and Knowledge Management (CIKM 2003), pp. 10–16 (2003)
37. Herbrich, R., Obermayer, K., Graepel, T.: Large margin rank boundaries for ordinal regression. In: Advances in Large Margin Classifiers, pp. 115–132 (2000)
38. Huang, J., Frey, B.: Structured ranking learning using cumulative distribution networks. In: Advances in Neural Information Processing Systems 21 (NIPS 2008) (2009)
39. Järvelin, K., Kekäläinen, J.: IR evaluation methods for retrieving highly relevant documents. In: Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2000), pp. 41–48 (2000)
40. Järvelin, K., Kekäläinen, J.: Cumulated gain-based evaluation of IR techniques. ACM Transactions on Information Systems **20**(4), 422–446 (2002)
41. Joachims, T.: Optimizing search engines using clickthrough data. In: Proceedings of the 8th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2002), pp. 133–142 (2002)
42. Joachims, T.: Evaluating retrieval performance using clickthrough data. In: Text Mining, pp. 79–96 (2003)
43. Kendall, M.: Rank Correlation Methods. Oxford University Press, London (1990)
44. Kramer, S., Widmer, G., Pfahringer, B., Groeve, M.D.: Prediction of ordinal classes using regression trees. Funfamenta Informaticae **34**, 1–15 (2000)
45. Lafferty, J., Zhai, C.: Document language models, query models and risk minimization for information retrieval. In: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2001), pp. 111–119 (2001)
46. Langville, A.N., Meyer, C.D.: Deeper inside pagerank. Internet Mathematics **1**(3), 335–400 (2004)

47. Li, P., Burges, C., Wu, Q.: McRank: Learning to rank using multiple classification and gradient boosting. In: Advances in Neural Information Processing Systems 20 (NIPS 2007), pp. 845–852 (2008)
48. Liu, T.Y., Xu, J., Qin, T., Xiong, W.Y., Li, H.: LETOR: Benchmark dataset for research on learning to rank for information retrieval. In: SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007) (2007)
49. Mao, J.: Machine learning in online advertising. In: Proceedings of the 11th International Conference on Enterprise Information Systems (ICEIS 2009), p. 21 (2009)
50. Maron, M.E., Kuhns, J.L.: On relevance, probabilistic indexing and information retrieval. Journal of the ACM **7**(3), 216–244 (1960)
51. McSherry, F.: A uniform approach to accelerated pagerank computation. In: Proceedings of the 14th International Conference on World Wide Web (WWW 2005), pp. 575–582. ACM, New York (2005)
52. Metzler, D.A., Croft, W.B.: A Markov random field model for term dependencies. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), pp. 472–479 (2005)
53. Metzler, D.A., Kanungo, T.: Machine learned sentence selection strategies for query-biased summarization. In: SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR 2008) (2008)
54. Mitchell, T.: Machine Learning. McGraw-Hill, New York (1997)
55. Nallapati, R.: Discriminative models for information retrieval. In: Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2004), pp. 64–71 (2004)
56. Pavlu, V.: Large scale ir evaluation. PhD thesis, Northeastern University, College of Computer and Information Science (2008)
57. Ponte, J.M., Croft, W.B.: A language modeling approach to information retrieval. In: Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 1998), pp. 275–281 (1998)
58. Qin, T., Liu, T.Y., Lai, W., Zhang, X.D., Wang, D.S., Li, H.: Ranking with multiple hyperplanes. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 279–286 (2007)
59. Qin, T., Liu, T.Y., Li, H.: A general approximation framework for direct optimization of information retrieval measures. Information Retrieval **13**(4), 375–397 (2009)
60. Qin, T., Liu, T.Y., Zhang, X.D., Chen, Z., Ma, W.Y.: A study of relevance propagation for web search. In: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2005), pp. 408–415 (2005)
61. Qin, T., Liu, T.Y., Zhang, X.D., Wang, D., Li, H.: Learning to rank relational objects and its application to web search. In: Proceedings of the 17th International Conference on World Wide Web (WWW 2008), pp. 407–416 (2008)
62. Qin, T., Zhang, X.D., Tsai, M.F., Wang, D.S., Liu, T.Y., Li, H.: Query-level loss functions for information retrieval. Information Processing and Management **44**(2), 838–855 (2008)
63. Radlinski, F., Joachims, T.: Query chain: learning to rank from implicit feedback. In: Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD 2005), pp. 239–248 (2005)
64. Richardson, M., Domingos, P.: The intelligent surfer: probabilistic combination of link and content information in pagerank. In: Advances in Neural Information Processing Systems 14 (NIPS 2001), pp. 1441–1448. MIT Press, Cambridge (2002)
65. Robertson, S.E.: Overview of the okapi projects. Journal of Documentation **53**(1), 3–7 (1997)
66. Rochhio, J.J.: Relevance feedback in information retrieval. In: The SMART Retrieval System—Experiments in Automatic Document Processing, pp. 313–323 (1971)
67. Shakery, A., Zhai, C.: A probabilistic relevance propagation model for hypertext retrieval. In: Proceedings of the 15th International Conference on Information and Knowledge Management (CIKM 2006), pp. 550–558 (2006)
68. Shashua, A., Levin, A.: Ranking with large margin principles: two approaches. In: Advances in Neural Information Processing Systems 15 (NIPS 2002), pp. 937–944 (2003)

69. Shen, L., Joshi, A.K.: Ranking and reranking with perceptron. Journal of Machine Learning **60**(1–3), 73–96 (2005)
70. Singhal, A.: Modern information retrieval: a brief overview. IEEE Data Engineering Bulletin **24**(4), 35–43 (2001)
71. Surdeanu, M., Ciaramita, M., Zaragoza, H.: Learning to rank answers on large online qa collections. In: Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies (ACL-HLT 2008), pp. 719–727 (2008)
72. Talyor, M., Guiver, J., et al.: Softrank: optimising non-smooth rank metrics. In: Proceedings of the 1st International Conference on Web Search and Web Data Mining (WSDM 2008), pp. 77–86 (2008)
73. Tao, T., Zhai, C.: Regularized estimation of mixture models for robust pseudo-relevance feedback. In: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2006), pp. 162–169 (2006)
74. Tao, T., Zhai, C.: An exploration of proximity measures in information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 295–302 (2007)
75. Taylor, M., Zaragoza, H., Craswell, N., Robertson, S., Burges, C.J.: Optimisation methods for ranking functions with multiple parameters. In: Proceedings of the 15th International Conference on Information and Knowledge Management (CIKM 2006), pp. 585–593 (2006)
76. Tsai, M.F., Liu, T.Y., Qin, T., Chen, H.H., Ma, W.Y.: Frank: a ranking method with fidelity loss. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 383–390 (2007)
77. Vapnik, V.N.: Statistical Learning Theory. Wiley-Interscience, New York (1998)
78. Veloso, A., Almeida, H.M., Goçalves, M., Meira, W. Jr.: Learning to rank at query-time using association rules. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 267–274 (2008)
79. Verberne, S., Halteren, H.V., Theijssen, D., Raaijmakers, S., Boves, L.: Learning to rank qa data. In: SIGIR 2009 Workshop on Learning to Rank for Information Retrieval (LR4IR 2009) (2009)
80. Volkovs, M.N., Zemel, R.S.: Boltzrank: learning to maximize expected ranking gain. In: Proceedings of the 26th International Conference on Machine Learning (ICML 2009), pp. 1089–1096 (2009)
81. Voorhees, E.M.: The philosophy of information retrieval evaluation. In: Lecture Notes in Computer Science (CLEF 2001), pp. 355–370 (2001)
82. Xia, F., Liu, T.Y., Wang, J., Zhang, W., Li, H.: Listwise approach to learning to rank—theorem and algorithm. In: Proceedings of the 25th International Conference on Machine Learning (ICML 2008), pp. 1192–1199 (2008)
83. Xu, J., Cao, Y., Li, H., Zhao, M.: Ranking definitions with supervised learning methods. In: Proceedings of the 14th International Conference on World Wide Web (WWW 2005), pp. 811–819. ACM Press, New York (2005)
84. Xu, J., Li, H.: Adarank: a boosting algorithm for information retrieval. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 391–398 (2007)
85. Xu, J., Liu, T.Y., Lu, M., Li, H., Ma, W.Y.: Directly optimizing IR evaluation measures in learning to rank. In: Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2008), pp. 107–114 (2008)
86. Yang, Y.H., Hsu, W.H.: Video search reranking via online ordinal reranking. In: Proceedings of IEEE 2008 International Conference on Multimedia and Expo (ICME 2008), pp. 285–288 (2008)
87. Yang, Y.H., Wu, P.T., Lee, C.W., Lin, K.H., Hsu, W.H., Chen, H.H.: Contextseer: context search and recommendation at query time for shared consumer photos. In: Proceedings of the 16th International Conference on Multimedia (MM 2008), pp. 199–208 (2008)
88. Yeh, J.Y., Lin, J.Y., et al.: Learning to rank for information retrieval using genetic programming. In: SIGIR 2007 Workshop on Learning to Rank for Information Retrieval (LR4IR 2007) (2007)

89. Yue, Y., Finley, T., Radlinski, F., Joachims, T.: A support vector method for optimizing average precision. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 271–278 (2007)
90. Zhai, C.: Statistical language models for information retrieval: a critical review. Foundations and Trends in Information Retrieval **2**(3), 137–215 (2008)
91. Zhai, C., Cohen, W.W., Lafferty, J.: Beyond independent relevance: methods and evaluation metrics for subtopic retrieval. In: Proceedings of the 26th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2003), pp. 10–17 (2003)
92. Zhai, C., Lafferty, J.: Model-based feedback in the language modeling approach to information retrieval. In: Proceedings of the 10th International Conference on Information and Knowledge Management (CIKM 2001), pp. 403–410 (2001)
93. Zhai, C., Lafferty, J.: A risk minimization framework for information retrieval. Information Processing and Management **42**(1), 31–55 (2006)
94. Zheng, Z., Chen, K., Sun, G., Zha, H.: A regression framework for learning ranking functions using relative relevance judgments. In: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR 2007), pp. 287–294 (2007)
95. Zheng, Z., Zha, H., Sun, G.: Query-level learning to rank using isotonic regression. In: SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR 2008) (2008)
96. Zoeter, O., Taylor, M., Snelson, E., Guiver, J., Craswell, N., Szummer, M.: A decision theoretic framework for ranking using implicit feedback. In: SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR 2008) (2008)

http://www.springer.com/978-3-642-14266-6

Learning to Rank for Information Retrieval
Liu, T.-Y.
2011, XVII, 285 p., Hardcover
ISBN: 978-3-642-14266-6