Learning to Rank using Query-Level Rules

Adriano Veloso, Marcos Gonçalves, Wagner Meira Jr., and Humberto Mossri

Departamento de Ciência da Computação
Universidade Federal de Minas Gerais, Brazil
{adrianov, mgoncalv, meira, hmossri}@dcc.ufmg.br

Abstract. Most existing learning to rank methods neglect query-sensitive information while producing functions to estimate the relevance of documents (i.e., all examples in the training data are treated indistinctly, no matter the query associated with them). This is counter-intuitive, since the relevance of a document depends on the query context (i.e., the same document may have different relevances, depending on the query associated with it). In this paper we show that query-sensitive information is of paramount importance for improving ranking performance. We present novel learning to rank methods. These methods use rules associating document features to relevance levels as building blocks to produce ranking functions. Such rules may have different scopes: global rules (which do not exploit query-sensitive information) and query-level rules. Firstly, we discuss a basic method, RE-GR (Relevance Estimation using Global Rules), which neglects any query-sensitive information, and uses global rules to produce a single ranking function. Then, we propose methods that effectively exploit query-sensitive information in order to improve ranking performance. The RE-SR method (Relevance Estimation using Stable Rules), produces a single ranking function using stable rules, which are rules carrying (almost) the same information no matter the query context. The RE-QR method (Relevance Estimation using Query-level Rules), is much finer-grained. It uses query-level rules to produce multiple query-level functions. The estimates provided by such query-level functions are combined according to the competence of each function (i.e., a measure of how close the estimate provided by a query-level function is to the true relevance of the document). We conducted a systematic empirical evaluation using the LETOR 4.0 benchmark collections. We show that the proposed methods outperform state-of-the-art learning to rank methods in most of the subsets, with gains ranging from 2% to 9%. We further show that RE-SR and RE-QR, which use query-sensitive information while producing ranking functions, achieve superior ranking performance when compared to RE-GR.

Categories and Subject Descriptors: H.3.3 [**Information Search and Retrieval**]: Information Search and Retrieval; I.2.6 [**Artificial Intelligence**]: Learning to Rank

Keywords: Competence, Ranking, Stability

1. INTRODUCTION

The ultimate goal of ranking methods is to achieve the best possible ranking performance for the problem at hand. Recently, a body of empirical evidence has emerged suggesting that methods that learn to rank offer substantial improvements in enough situations to be regarded as a relevant advance for applications that depend on ranking. The conventional approach to this learning task is to assume the availability of examples (i.e., a training data with document features and the corresponding relevance to specific queries), from which learning functions can be learned. When a new query is given, the relevance of documents retrieved for this query are estimated according to the learned function (i.e., this function gives a score to a document indicating its relevance to the query).

There are countless paradigms and strategies for devising learning to rank methods. Such methods usually rely on machine learning techniques, such as neural networks[Burges et al. 2005], genetic programming[de Almeida et al. 2007] and support vector machines [Yue et al. 2007]. The use of association rules has also shown to be valuable for learning ranking functions [Veloso et al. 2008]. The basic idea is to exploit

This research was supported by UOL (www.uol.com.br) through its UOL Bolsa Pesquisa program (grant number 20080131200100), CNPq, Capes, Fapemig (project number 14281), INCTWeb (grant number 573871/2008-6), and InfoWeb (grant number 550874/2007-0).

combinations of document features which are truly associated with relevant/irrelevant documents. In this case, the ranking function is essentially a set of rules $\mathcal{X} \rightarrow r$, where each rule indicates an association between a set of document features $\mathcal{X}$ and a relevance level r. These rules are extracted from the training data and then their predictions are combined in order to estimate the relevance of documents.

In the simplest case, query-sensitive information is neglected while rules are extracted from the training data. Specifically, cross-query and intra-query documents[1] are treated indistinctly while extracting rules. Such rules are called *global rules*, and they naturally lead to a single ranking function. While this function truly reflects the relationship between document features and relevance levels, query-context still provides additional and important information for the sake of relevance estimation. Thus, taking query-sensitive information into account while extracting rules from the training data may be a way of improving ranking performance.

In this paper we look at the problem of learning ranking functions from the perspective of query. We introduce the concept of *query-level rules*, which have the form $q \wedge \mathcal{X} \rightarrow r$, where q represents the query context. Query-level rules capture query-sensitive information by distinguishing cross-query documents from intra-query documents, that is, only intra-query documents are considered while extracting query-level rules. We use query-level rules to find *stable rules*. The predictions performed by stable rules do not change much across different queries. Thus, their predictions are considered very reliable. Further, we also exploit query-level rules to produce multiple (query-level) ranking functions.

We observed that, very often, some particular query-level functions provide extremely accurate relevance estimates for specific documents (i.e., the estimate is very close to the true relevance). On the other hand, the same functions also provide extremely poor estimates for other documents. While this implies that there is no query-level function that can be safely used in isolation to estimate the relevance of all documents, this also indicates that there is an optimal matching between documents and query-level functions. Obviously, knowing this optimal matching would enable the assignment of specific functions to specific documents, and hopefully, ranking performance would be drastically boosted. This would be great, except that the optimal matching is unknown.

Fortunately, we further observed that there is a domain for which a certain query-level function is competent (i.e., a set of documents, which often exhibit certain features, for which the function usually provides accurate relevance estimates). This notion of competence makes possible to approximate the matching between documents and query-level functions. With such approximation, estimates of each query-level function can be combined in a way that maximizes the accuracy of the relevance estimation for each document.

1.1 Contributions

The specific contributions of this paper are summarized as follows:

—We show that query context is valuable information for the sake of improving ranking performance. We introduce stable rules, which are rules that express (almost) the same information, no matter the query. These rules are particularly interesting because they tend to be very reliable. We also introduce query-level rules, which capture query-sensitive information in order to produce query-level functions. Relevance estimates provided by different query-level functions are combined according to the competence of each query-level function, resulting in a hybrid ranking function.

—We look at the problem of learning ranking functions from the perspective of query. We propose learning to rank methods which are based on stable rules (RE-SR), and query-level rules (RE-QR).

—A deep evaluation of these methods, using the LETOR 4.0 benchmark, revealed that producing document-specific ranking functions is, most of the times, beneficial. We show that the RE-QR method, which combines query-level functions according to their competence, outperforms all baselines in most of the subsets used, with gains in terms of MAP ranging from 2% to 9%.

[1] Cross-query documents are those associated with different queries. Intra-query documents are those associated with the same query. In Table I on Section 3, d_1 and d_2 are intra-query documents, while d_1 and d_4 are cross-query documents.

1.2 Organization

Related work is discussed in the next section. In Section 3 we discuss the RE-GR basic method, which produces a single ranking function using global rules. In Section 4 we propose the RE-SR method (which produces a single ranking function using stable rules), and the RE-QR method (which produces multiple query-level functions using query-level rules). In Section 5 we demonstrate the effectiveness of the proposed methods through a systematic set of experiments. Finally, in Section 6 we conclude the paper.

2. RELATED WORK

Many prior efforts have been devoted to exploit machine learning techniques in order to improve ranking performance. Particularly noteworthy contributions include [Matveeva et al. 2006; Gao et al. 2005; Yue et al. 2007; Trotman 2005; Joachims 2002; Liu et al. 2007; Qin et al. 2007; Xu and Li 2007; Burges et al. 2005; Cao et al. 2007; Tsai et al. 2007; Cao et al. 2006; Xia et al. 2008; Cohen et al. 2008; Xu et al. 2008; Geng et al. 2008; Qin et al. 2008; Veloso et al. 2008; Qin et al. 2008].

According to Cao et al. [Cao et al. 2007], current learning to rank methods fall into three categories: (i) point-wise, (ii) pair-wise and (iii) list-wise approaches. In the point-wise approach [Nallapati 2004; Veloso et al. 2008], each training example is composed of a document and its corresponding relevance relative to a query. The learning process tries to map document features into relevance estimates. In the pair-wise approach [Burges et al. 2005; Freund et al. 2003; Joachims 2002; Tsai et al. 2007], each example is composed of pairs of documents and the preference relation among them. In this case, the goal is to classify each pair into correctly or incorrectly ranked categories. Finally, in the list-wise approach [Cao et al. 2007; Xu and Li 2007], a list of documents are used as examples. A function is learned, and then used to sort documents.

Most of the existing learning to rank methods are designed under the conventional assumption that there is a single distribution governing the relationship between inputs (i.e., documents) and outputs (i.e., relevance). In practice, however, this is not a realistic assumption, mainly because the relevance of documents retrieved for different queries (i.e., cross-query documents) are interpreted in different ways (i.e., a document which is relevant for a given query, may be not relevant for another query). Attempts to exploit query-sensitive information include [Veloso et al. 2008], where the authors proposed to use query terms in order to produce improved ranking functions. While query terms may carry valuable information, different queries may share the same terms, and thus query terms are not appropriate to define query-context (i.e., similar documents retrieved for queries that share terms, are still cross-query documents, and thus they may have very different relevances). The necessity to employ different ranking functions depending on the query was also pointed in [Geng et al. 2008]. Further analysis about the use of query-sensitive information was presented in [Cohen et al. 2008].

In this paper we are also interested in learning to rank methods that exploit query-sensitive information while estimating the relevance of documents. The proposed methods differ significantly from existing ones. They are based on stable rules and query-level rules. Predictions performed by stable rules are very reliable, and are shown to improve ranking performance. Predictions performed by query-level rules are combined in a way that maximizes the accuracy of the estimates for each document. The proposed methods are intuitive (easily understood using a set of illustrative examples), but also extremely effective, as will be shown in the experiments.

3. LEARNING TO RANK USING GLOBAL RULES

In our context, the task of learning to rank is defined as follows. We have as input the *training data* (referred to as $\mathcal{D}$), which consists of a set of records of the form $< q, d, r^d >$, where q is a query, d is a document (represented as a list of m attribute-values or features $\{f_1, f_2, \ldots, f_m\}$), and r^d is the *relevance* of d to q. Attributes include BM25, Page Rank, and many other document properties. The relevance of a document draws its values from a discrete set of possibilities (e.g., r_0, r_1, ..., r_k). The training data is used to build functions relating features of the documents to their corresponding relevance. The *test set* (referred to as $\mathcal{T}$)

consists of records $< q, d, ? >$ for which only the query q and the document d are known, while the relevance of d to q is unknown. Ranking functions learned from $\mathcal{D}$ are used to estimate the relevance of such documents to the corresponding queries.

Ranking functions exploit the relationship between document features and relevance levels. This relationship can be represented by association rules. We denote as $\mathcal{R}$ a rule-set composed of rules of the form $\{f_j \wedge \ldots \wedge f_l \xrightarrow{\theta} r_i\}$. These rules can contain any mixture of the available features in the antecedent and a relevance level in the consequent. The strength of the association between antecedent and consequent is measured by a statistic, θ, which is known as *confidence* [Agrawal et al. 1993] and is simply the conditional probability of the consequent given the antecedent.

3.1 Demand-Driven Rule Extraction

The search space for rules is huge, and thus, computational cost restrictions must be imposed during rule extraction. Typically, a minimum support threshold (σ_{min}) is employed in order to select frequent rules (i.e., rules occurring at least σ_{min} times in $\mathcal{D}$) from which the ranking function is produced. This strategy, although simple, has some problems. If σ_{min} is set too low, a large number of rules will be extracted from $\mathcal{D}$, and often most of these rules are useless for estimating the relevance of documents in $\mathcal{T}$ (a rule $\{\mathcal{X} \rightarrow r_i\}$ is only useful to estimate the relevance of document $d \in \mathcal{T}$ if the set of features $\mathcal{X} \subseteq d$, otherwise the rule is meaningless to d). On the other hand, if σ_{min} is set too high, some important rules will not be included in $\mathcal{R}$, causing problems if some documents in $\mathcal{T}$ contain rare features (i.e., features occurring less than σ_{min} times in $\mathcal{D}$). Usually, there is no optimal value for σ_{min}, that is, there is no single value that ensures that only useful rules are included in $\mathcal{R}$, while at the same time important rules are not missed. The method to be proposed next deals with this problem by extracting rules on a demand-driven basis.

Demand-driven rule extraction is delayed until a set of documents is retrieved for a given query in $\mathcal{T}$. Then, each individual document d in $\mathcal{T}$ is used as a filter to remove irrelevant features and examples from $\mathcal{D}$. This process produces a projected training data, $\mathcal{D}_d$, which is obtained after removing all attribute-values not present in d. Then, a specific rule-set, $\mathcal{R}_d$ extracted from $\mathcal{D}_d$, is produced for each document d in $\mathcal{T}$.

LEMMA 3.1. *All rules extracted from $\mathcal{D}_d$ (i.e., $\mathcal{R}_d$) are useful to estimate r^d.*

Proof:. Since all examples in $\mathcal{D}_d$ contain only attribute-values that are present in d, the existence of a rule $\{\mathcal{X} \rightarrow r_i\} \in \mathcal{R}_d$, such that $\mathcal{X} \nsubseteq d$, is impossible. ■

THEOREM 3.2. *The number of rules extracted from $\mathcal{D}_d$ increases polynomially with the number of distinct attribute-values in $\mathcal{D}$, no matter the value of σ_{min}.*

Proof:. Let n be the number of distinct attribute-values in $\mathcal{D}$. Obviously, the number of all rules is exponential in n (i.e., $O(2^n)$ rules). However, since an arbitrary document $d \in \mathcal{T}$ contains at most l attribute-values (with $l \ll n$), then any rule matching d (i.e., an useful rule) can have at most l attribute-values in its antecedent. That is, for any rule $\{\mathcal{X} \rightarrow r_i\}$, such that $\mathcal{X} \subseteq d$, $|\mathcal{X}| \leq l$. Consequently, for $\sigma_{min} \approx 0$, the number of possible rules matching d is $k \times (l + \binom{l}{2} + \ldots + \binom{l}{l}) = O(2^l) \ll O(n^l)$, where k is the number of distinct relevances. Thus, the number of useful rules increases polynomialy in n. Since, according to Lemma 1, only useful rules are extracted from $\mathcal{D}_d$, then the number of rules extracted for all documents in $\mathcal{T}$ is $O(|\mathcal{T}| \times n^l)$. ■

3.2 Relevance Estimation using Global Rules

In order to estimate the relevance of a document d, it is necessary to combine all rules in $\mathcal{R}_d$. Our strategy is to interpret $\mathcal{R}_d$ as a poll, in which each rule $\{\mathcal{X} \xrightarrow{\theta} r_i\} \in \mathcal{R}_d$ is a vote given by a set of features $\mathcal{X}$ for relevance level r_i. Votes have different weights, depending on the strength of the association they represent (i.e., θ). The weighted votes for relevance level r_i are summed and then averaged (by the total number of rules

in $\mathcal{R}_d$ that predict relevance level r_i), forming the score associated with relevance r_i for document d, as shown in Equation 1 (where $\theta(\mathcal{X} \rightarrow r_i)$ is the value θ assumes for rule $\{\mathcal{X} \rightarrow r_i\}$):

$$s(d, r_i) = \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{\mid \mathcal{R}_d \mid}, \text{ where } \mathcal{X} \subseteq d \tag{1}$$

Therefore, for a document d, the score associated with relevance r_i is given by the average θ values of the rules in $\mathcal{R}_d$ predicting r_i. The likelihood of d having a relevance level r_i is obtained by normalizing the scores, as expressed by $\hat{p}(r_i|d)$, shown in Equation 2:

$$\hat{p}(r_i|d) = \frac{s(d, r_i)}{\sum_{j=0}^{k} s(d, r_j)} \tag{2}$$

Finally, the relevance of document d is estimated by a linear combination of the likelihoods associated with each relevance level, as expressed by the ranking function $rank(d)$, which is shown in Equation 3:

$$rank(d) = \sum_{i=0}^{k} \big(r_i \times \hat{p}(r_i|d)\big) \tag{3}$$

The value of $rank(d)$ is an estimate of the true relevance of document d (i.e., r^d) using $\hat{p}(r_i|d)$. This estimate ranges from r_0 to r_k, where r_0 is the lowest relevance and r_k is the highest one. Thus, both $rank(d)$ and r^d assume values in the same range. Relevance estimates are used to produce ranked lists of documents. This is the strategy adopted by RE-GR.

Example. Table I shows an example where $\mathcal{D}$ contains three queries. For each query, three documents are retrieved, and each document is represented by three attributes − PageRank, BM25 and tf (in our experiments we represented a document using many more attributes, but for simplicity we restricted this example to only three attributes). Document features were obtained by discretizing these attributes (for this example, the boundaries of the intervals are merely illustrative). Suppose we want to estimate the relevance of d_{10}. In this case, the original training data is projected according to d_{10}. resulting in $\mathcal{D}_{d_{10}}$, which is shown in Table II.

The following 4 rules are extracted from $\mathcal{D}_{d_{10}}$:

(1) tf=[0.28-0.45] $\rightarrow$ r=0 ($\theta = 1.00$)
(2) BM25=[0.36-0.55] $\rightarrow$ r=0 ($\theta = 0.50$)
(3) BM25=[0.36-0.55] $\rightarrow$ r=1 ($\theta = 0.50$)
(4) BM25=[0.36-0.55] $\wedge$ tf=[0.28-0.45] $\rightarrow$ r=0 ($\theta = 1.00$)

The predictions of these rules are combined according to Equations 1 and 2, in order to produce $\hat{p}(r_i|d_{10})$. Finally, according to Equation 3, $rank(d_{10})$=0.37. Following the same process, we obtain $rank(d_{11})$=0.54 and $rank(d_{12})$=0.24.

4. LEARNING TO RANK USING STABLE AND QUERY-LEVEL RULES

Cross-query documents are usually interpreted in different ways, according to the query associated with them. A document which is considered relevant for a certain query, may have similar documents that are not considered relevant for other queries. In this section we take query-context into account while extracting rules, in

	Query	Retrieved Documents				r^d
		id	PageRank	BM25	*tf*	
$\mathcal{D}$	q_1	d_1	[0.85-0.92]	[0.36-0.55]	[0.23-0.27]	1
		d_2	[0.74-0.84]	[0.36-0.55]	[0.23-0.27]	1
		d_3	[0.74-0.84]	[0.56-0.70]	[0.46-0.61]	0
	q_2	d_4	[0.93-1.00]	[0.36-0.55]	[0.46-0.61]	0
		d_5	[0.85-0.92]	[0.56-0.70]	[0.62-0.76]	1
		d_6	[0.74-0.84]	[0.36-0.55]	[0.28-0.45]	0
	q_3	d_7	[0.74-0.84]	[0.22-0.35]	[0.12-0.22]	0
		d_8	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	0
		d_9	[0.85-0.92]	[0.71-0.80]	[0.46-0.61]	1
$\mathcal{T}$	q_4	d_{10}	[0.51-0.64]	[0.36-0.55]	[0.28-0.45]	0
		d_{11}	[0.85-0.92]	[0.00-0.21]	[0.46-0.61]	1
		d_{12}	[0.74-0.84]	[0.56-0.70]	[0.46-0.61]	0

Table I. Training Data and Test Set.

	id	PageRank	BM25	*tf*	r^d
$\mathcal{D}_{d_{10}}$	d_1	–	[0.36-0.55]	–	1
	d_2	–	[0.36-0.55]	–	1
	d_4	–	[0.36-0.55]	–	0
	d_6	–	[0.36-0.55]	[0.28-0.45]	0

Table II. Training Data projected according to d_{10}.

order to improve ranking performance. The extracted rules (which are called query-level rules) have the form $\{q \wedge \mathcal{X} \rightarrow r_i\}$, where q represents the query context. In the following sections we will discuss how to exploit query-level rules to find stable rules, and how we use query-levels rules to estimate the relevance of documents. For the discussion that follows we need to define query-context.

Definition 4.1. Query-Context: The context of a query q is the set of all documents retrieved for this query.

4.1 Relevance Estimation using Stable Rules

A rule $\{\mathcal{X} \rightarrow r_i\}$ is said to be stable if the association between $\mathcal{X}$ and r_i does not change much across different query contexts.

Definition 4.2. Rule Stability: A rule $\{\mathcal{X} \rightarrow r_i\}$ is stable, if:

$$\forall q_j, |\theta(\mathcal{X} \rightarrow r_i) - \theta(q_j \wedge \mathcal{X} \rightarrow r_i)| \leq \phi_{min}$$

The lower ϕ_{min} is, the more stable is the rule. Stable rules are particularly important because their predictions tend to be very reliable. We denote as $\mathcal{R}^{\phi}$ the rule-set composed of stable rules. In order to estimate the relevance of document d, ϕ-stable rules are combined according to Equation 4. Then, Equations 2 and 3 are used to estimate r^d. This is the strategy adopted by RE-SR.

$$s(d, r_i) = \frac{\sum \theta(\mathcal{X} \rightarrow r_i)}{\mid \mathcal{R}_d^{\phi} \mid}, \text{ where } \mathcal{X} \subseteq d \tag{4}$$

Example. Suppose we want to estimate the relevance of d_{11}. If ϕ_{min}=0.05, then $\mathcal{R}^{\phi}_{d_{11}}$ is composed of the following rules:

(1) PageRank=[0.85-0.92]→ r=1 (θ=1.00, ϕ=0.00)
(2) PageRank=[0.85-0.92]∧tf=[0.46-0.61]→ r=1 (θ=1.00, ϕ=0.00)

The predictions of these rules are combined in order to produce the function $\hat{p}(r_i|d_{11})$, according to Equation 4. Finally, according to Equation 3, $rank(d_{11})$=1.00. Following the same process, we obtain $rank(d_{10})$=0.00. No stable rules can be extracted from $\mathcal{D}_{12}$, and, in this case, global rules are used to produce the ranking function.

4.2 Relevance Estimation using Query-Level Rules

A single ranking function, $\hat{p}(r_i|d)$, is not likely to reflect the true relationship between documents and their relevances. This is because the relevance of documents is not draw from a single distribution, but rather, from several different distributions, depending on the context of each query. In this section we directly use query-level rules to produce multiple query-level functions. Such functions take into account query-sensitive information, as shown in Equations 5, 6 and 7:

$$s(q, d, r_i) = \frac{\sum \theta(q \wedge \mathcal{X} \rightarrow r_i)}{\mid \mathcal{R}_d \mid}, \text{ where } \mathcal{X} \subseteq d \tag{5}$$

$$\hat{p}(r_i|d, q) = \frac{s(q, d, r_i)}{\sum_{j=0}^{k} s(q, d, r_j)} \tag{6}$$

$$rank(q, d) = \sum_{i=0}^{k} (r_i \times \hat{p}(r_i|d, q)) \tag{7}$$

Example. Suppose we want to estimate the relevance of d_{12}. The projected training data for d_{12} (i.e., $\mathcal{D}_{d_{12}}$) is shown in Table III. The following 15 query-level rules are extracted from $\mathcal{D}_{d_{12}}$:

(1) q_1 ∧ BM25=[0.56-0.70] → r=0 (θ=1.00)
(2) q_1 ∧ tf=[0.46-0.61] → r=0 (θ=1.00)
(3) q_1 ∧ PageRank=[0.74-0.84] → r=1 (θ=0.50)
(4) q_1 ∧ PageRank=[0.74-0.84] → r=0 (θ=0.50)
(5) q_1 ∧ PageRank=[0.74-0.84] ∧ BM25=[0.56-0.70] → r=0 (θ=1.00)
(6) q_1 ∧ BM25=[0.56-0.70] ∧ tf=[0.46-0.61] → r=0 (θ=1.00)
(7) q_1 ∧ PageRank=[0.74-0.84] ∧ tf=[0.46-0.61] → r=0 (θ=1.00)
(8) q_2 ∧ BM25=[0.56-0.70] → r=1 (θ=1.00)
(9) q_2 ∧ PageRank=[0.74-0.84] → r=0 (θ=1.00)
(10) q_2 ∧ tf=[0.46-0.61] → r=0 (θ=1.00)
(11) q_3 ∧ PageRank=[0.74-0.84] → r=0 (θ=1.00)
(12) q_3 ∧ BM25=[0.56-0.70] → r=0 (θ=1.00)
(13) q_3 ∧ BM25=[0.56-0.70] tf=[0.46-0.61] → r=0 (θ=1.00)
(14) q_3 ∧ tf=[0.46-0.61] → r=0 (θ=0.50)

	Query	id	PageRank	BM25	*tf*	r^d
$\mathcal{D}_{d_{12}}$	q_1	d_2	[0.74-0.84]	–	–	1
		d_3	[0.74-0.84]	[0.56-0.70]	[0.46-0.61]	0
	q_2	d_4	–	–	[0.46-0.61]	0
		d_5	–	[0.56-0.70]	–	1
		d_6	[0.74-0.84]	–	–	0
	q_3	d_7	[0.74-0.84]	–	–	0
		d_8	–	[0.56-0.70]	[0.46-0.61]	0
		d_9	–	–	[0.46-0.61]	1

Table III. Training Data projected according to d_{12}.

(15) $q_3 \wedge tf$=[0.46-0.61] $\rightarrow$ r=1 (θ=0.50)

Different query-level functions may provide different relevance estimates for the same document. For instance, $rank(q_1, d_{12})$=0.35, $rank(q_2, d_{12})$=0.50, and $rank(q_3, d_{12})$=0.36. This suggests that different query-level functions are only able to accurately estimate the relevances of certain documents. The optimal matching between functions and documents is valuable information. In the following we present an approach to estimate such matching. We start by defining the ranking competence of a function. Then, we discuss how to separate documents that are competently ranked by a function from documents that are not.

Definition 4.3. Ranking Competence: The ranking competence of a function, which is denoted as $\Delta(q, d)$, is defined as:

$$\Delta(q, d) = |rank(q, d) - r^d| \tag{8}$$

The competence of a function with respect to a document d, is essentially the discrepancy between the estimated relevance of d (i.e., $rank(q, d)$) and the true relevance of d (i.e., r^d). A query-level function $rank(q_a, d)$ is more competent than function $rank(q_b, d)$ if $\Delta(q_a, d) < \Delta(q_b, d)$.

The competence of a query-level function is novel information which may be used to enhance the original training data, $\mathcal{D}$. Specifically, for each document $d \in \mathcal{D}$, it is informed from which query-context it is produced the most competent function for this document. This information is obtained by estimating the relevance of each document in $\mathcal{D}$. This process results in an enhanced training data, denoted as $\mathcal{D}^*$. Initially, $\mathcal{D}^*$ is empty. At each iteration, document $d \in \mathcal{D}$ along with the context of the most competent query-level function with regard to d are inserted into $\mathcal{D}^*$. The process continues until all documents in $\mathcal{D}$ are inserted into $\mathcal{D}^*$, as shown in Table IV.

	Retrieved Documents				Query
	id	PageRank	BM25	*tf*	Context
$\mathcal{D}^*$	d_1	[0.85-0.92]	[0.36-0.55]	[0.23-0.27]	q_3
	d_2	[0.74-0.84]	[0.36-0.55]	[0.23-0.27]	q_1
	d_3	[0.74-0.84]	[0.56-0.70]	[0.46-0.61]	q_3
	d_4	[0.93-1.00]	[0.36-0.55]	[0.46-0.61]	q_2
	d_5	[0.85-0.92]	[0.56-0.70]	[0.62-0.76]	q_2
	d_6	[0.74-0.84]	[0.36-0.55]	[0.28-0.45]	q_3
	d_7	[0.74-0.84]	[0.22-0.35]	[0.12-0.22]	q_2
	d_8	[0.65-0.73]	[0.56-0.70]	[0.46-0.61]	q_1
	d_9	[0.85-0.92]	[0.71-0.80]	[0.46-0.61]	q_3

Table IV. Enhanced Training Data. The last column denotes the most competent query-level function.

Matching Documents and Functions. The enhanced training data, $\mathcal{D}^*$, can be exploited to approximate the matching between documents and query-level functions. Specifically, instead of directly extracting rules of the form $\{\mathcal{X} \rightarrow r_i\}$, we first extract rules of the form $\{\mathcal{X} \rightarrow q_i\}$ (i.e., the antecedent is a set of document features and the consequent is a query context). These rules are used to approximate the matching between documents and functions, according to Equations 9 and 10 (where n is the number of queries in $\mathcal{D}$). The higher $\hat{p}(q_i|d)$ is, the higher is the likelihood of $\Delta(q_i, d)$ being low (i.e., it is likely that the function produced using documents associated with query q_i will competently estimate r^d).

$$s(d, q_i) = \frac{\sum \theta(\mathcal{X} \rightarrow q_i)}{\mid \mathcal{R}_d \mid}, \text{ where } \mathcal{X} \subseteq d \tag{9}$$

$$\hat{p}(q_i|d) = \frac{s(d, q_i)}{\sum_{j=0}^{n} s(d, q_j)} \tag{10}$$

Hybridization based on Competence. A hybrid function is a combination of two or more query-level functions. Such a combination involves finding the appropriate parameters, so that the estimate provided by the resulting function minimizes $|rank(d) - r^d|$. The matching between documents and query-level functions (i.e. Eq. 10) can be used as parameter, as shown in Equation 11.

$$rank(d) = \sum_{i=0}^{k} \left(r_i \times \sum_{j=0}^{n} \left(\hat{p}(r_i|d, q_j) \times \hat{p}(q_j|d) \right) \right) \tag{11}$$

The basic idea is to weigh the estimates provided by different query-level functions according to the likelihood of competence of each of these functions. Intuitively, if a query-level function is likely to provide accurate estimates to a document, then such estimates will be heavily weighted. This is the strategy adopted by RE-QR.

Example. Suppose we want to estimate the relevance of document d_{12}. The first step, in this case, is to extract rules of form $\{\mathcal{X} \rightarrow q_i\}$ from $\mathcal{D}^*_{d_{12}}$. According to Equations 9 and 10, $\hat{p}(q_1, d_{12})$= 0.26, $\hat{p}(q_2, d_{12})$=0.21, and $\hat{p}(q_3, d_{12})$=0.53. Then, query-level rules of the form $\{q \wedge \mathcal{X} \rightarrow r_i\}$ are extracted from $\mathcal{D}^*_{d_{12}}$. Finally, according to Equation 11, $rank(d_{12})$=0.27. Following the same process, we obtain $rank(d_{10})$=0.00, and $rank(d_{11})$=0.48.

5. EXPERIMENTAL EVALUATION

In this section we empirically analyze the proposed learning to rank methods, RE-GR, RE-SR, and RE-QR. We first present the collections employed in the evaluation, and then we discuss the effectiveness of the methods in these collections.

5.1 The LETOR Benchmark

LETOR [Liu et al. 2007] is a benchmark for research on learning to rank, released by Microsoft Research Asia[2]. It makes available seven subsets (OHSUMED, TD2003, TD2004, HP2003, HP2004, NP2003 and NP2004). Each subset contains a set of queries, document features, and the corresponding relevance judgments. Features cover a wide range of properties, such as term frequency, BM25, PageRank, HITS etc. In order to conduct five-fold cross validation, each subset is arranged in five folds, including training, validation and test data. Ranking performance is evaluated using NDCG@$\times$ (normalized discounted cumulative gain), P@$\times$ (precision), and

[2]LETOR Web page: http://research.microsoft.com/users/LETOR/

MAP (mean average precision) measures. A detailed explanation of these measures can be found in [Liu et al. 2007]. For RE-GR and RE-SR, pre-processing involved only the discretization of attribute-values in $\mathcal{D}$ [Fayyad and Irani 1993]. For RE-QR, pre-processing also involved the creation of $\mathcal{D}^*$.

5.2 Baselines

Our evaluation is based on a comparison against state-of-the-art learning to rank methods such as R-SVM [Yue et al. 2007], FRank [Tsai et al. 2007], R-Boost [Freund et al. 2003], SVMMAP[Joachims 2002], AdaRank[Xu and Li 2007], and ListNet[Cao et al. 2007]. The ranking performance for these methods are also available at the LETOR Web page.

5.3 Results

All experiments were performed on a Linux PC with an Intel Core 2 Duo 1.63GHz and 2GBytes RAM. Validation set was used to select appropriate parameters. For RE-GR, RE-SR and RE-QR, we set σ_{min}=10^{-10}. For RE-SR we set ϕ_{min}=0.10. Parameters for the baselines can be found in the LETOR Web page.

How accurate are the proposed methods? How effective are the proposed methods when compared to other learning to rank methods?

Tables V, VI, VII, and VIII show MAP numbers for all subsets. The result for each trial is obtained by averaging partial results obtained from each query in the trial. The final result is obtained by averaging the five trials. We conducted two sets of significance tests (t-test) on each subset. The first set of significance tests was carried on the average of the results for each query. The second set of significance tests was carried on the average of the five trials.

In five, out of seven subsets, RE-QR was the best overall performer, demonstrating the effectiveness of exploiting query-sensitive information. RE-GR and RE-SR showed to be effective in most of the subsets, being (together with RE-QR) the best performers in the NP2003 subset. In most of the subsets, all proposed methods achieved superior ranking performance when compared to the best baseline. The only exceptions occurred in HP2003 and HP2004 subsets, where AdaRank was the best performer. Still, the gains in performance provided by RE-QR range from 7% (relative to FRank in NP2003) to 48% (relative to FRank in TD2003). RE-SR outperformed RE-GR in the OHSUMED, TD2003, HP2003 and HP2004 subsets, but RE-GR showed to be superior in the remaining subsets. This is because TD2004, NP2003 and NP2004 subsets contains only few stable rules, hurting the performance of RE-SR.

	OHSUMED				
Trial	RE-GR	RE-SR	RE-QR	R-SVM	AdaRank
1	0.352	**0.366**	**0.369**	0.304	0.344
2	**0.463**	**0.469**	**0.465**	0.447	0.446
3	0.460	0.460	**0.469**	**0.465**	**0.469**
4	0.521	**0.535**	**0.540**	0.499	0.514
5	0.482	0.475	**0.490**	0.453	0.471
Avg	0.456	0.460	**0.465**	0.433	0.449

Table V. MAP numbers for OHSUMED subset. Worst and best baselines are also shown. Best results, including statistical ties, are shown in bold.

We also evaluated the proposed methods in terms of precision and NDCG. Figure 1 shows precision numbers obtained from the execution of the proposed methods. Due to lack of space, only the best baseline is shown for comparison. RE-QR and RE-SR improved the precision at the first positions (they are always the best performer at P@1). Precision in the subsequent positions are similar to the precision achieved by RE-GR.

	TD2003					TD2004				
Trial	RE-GR	RE-SR	RE-QR	FRank	ListNet	RE-GR	RE-SR	RE-QR	SVMMAP	R-Boost
1	0.169	0.178	0.171	0.113	**0.192**	0.213	0.221	0.219	0.185	**0.247**
2	0.293	0.304	**0.327**	0.297	**0.325**	0.276	0.256	**0.279**	0.192	**0.281**
3	0.365	0.381	**0.403**	0.155	0.381	**0.285**	0.277	**0.283**	0.201	0.241
4	**0.394**	**0.394**	0.382	0.211	0.275	0.267	0.249	**0.275**	0.211	0.238
5	0.219	0.201	0.216	**0.238**	0.202	0.276	0.273	0.283	0.235	**0.299**
Avg	0.288	0.292	**0.300**	0.203	0.275	0.263	0.255	**0.268**	0.205	0.261

Table VI. MAP numbers for TD2003 and TD2004 subsets. Worst and best baselines are also shown.

	HP2003					HP2004				
Trial	RE-GR	RE-SR	RE-QR	FRank	AdaRank	RE-GR	RE-SR	RE-QR	R-Boost	AdaRank
1	**0.717**	**0.722**	0.709	0.674	**0.715**	0.666	**0.678**	**0.671**	0.621	**0.674**
2	0.808	0.839	0.834	0.804	**0.855**	0.756	**0.770**	**0.763**	0.618	0.678
3	0.737	0.762	0.744	0.737	**0.801**	0.806	0.812	0.818	0.637	**0.848**
4	0.762	0.762	**0.774**	0.684	0.752	0.635	**0.645**	0.641	0.611	**0.648**
5	0.755	0.749	**0.769**	0.648	0.732	0.627	0.624	0.639	0.638	**0.762**
Avg	0.756	0.767	0.766	0.709	**0.771**	0.696	0.706	0.706	0.625	**0.722**

Table VII. MAP numbers for HP2003 and HP2004 subsets. Worst and best baselines are also shown.

	NP2003					NP2004				
Trial	RE-GR	RE-SR	RE-QR	FRank	R-Boost	RE-GR	RE-SR	RE-QR	R-Boost	ListNet
1	**0.695**	**0.702**	**0.701**	0.591	0.685	**0.592**	**0.594**	**0.585**	0.550	0.550
2	**0.676**	**0.674**	**0.679**	0.645	0.666	0.648	**0.652**	**0.659**	0.559	**0.659**
3	0.670	0.661	0.682	0.673	**0.711**	**0.870**	**0.877**	**0.873**	0.609	0.739
4	0.751	0.738	0.746	**0.769**	0.733	0.611	0.602	0.649	0.531	**0.728**
5	**0.748**	**0.762**	**0.756**	0.642	0.743	0.650	0.633	0.657	0.570	**0.684**
Avg	**0.708**	**0.707**	**0.712**	0.664	**0.707**	0.675	0.672	**0.685**	0.564	0.672

Table VIII. MAP numbers for NP2003 and NP2004 subsets. Worst and best baselines are also shown.

NDCG numbers are shown in Figure 3. Again, RE-SR and RE-QR showed some improvements at the first positions, and a performance which is similar to the one achieved by RE-GR in the subsequent positions. RE-QR outperformed the best baselines in five (out of seven) subsets. AdaRank showed to be the best performer in HP2003 and HP2004 subsets.

How is competence distributed among different query-level functions?

Figure 2 shows the domain of competence of each query-level function using the OHSUMED subset. Lighter colored regions indicate documents in the x-axis for which relevances were competently estimated by the corresponding query-level function in the y-axis (i.e., $\Delta(q, d)$ is low). Darker colored regions, on the other hand, indicate documents for which relevances were not competently estimated by the corresponding query-level function (i.e., $\Delta(q, d)$ is high). We divided the documents in three graphs, according to their relevance. As expected, some query-level functions are competent in estimating the relevance of relevant documents, while others are competent in estimating the relevance of irrelevant documents. Some functions are also able to competently estimate the relevance of both relevant and irrelevant documents. RE-QR is likely to avoid poor estimates, since it takes into account the competence of each query-level function. Thus, RE-QR take advantage from selecting appropriate regions of each query-level function.

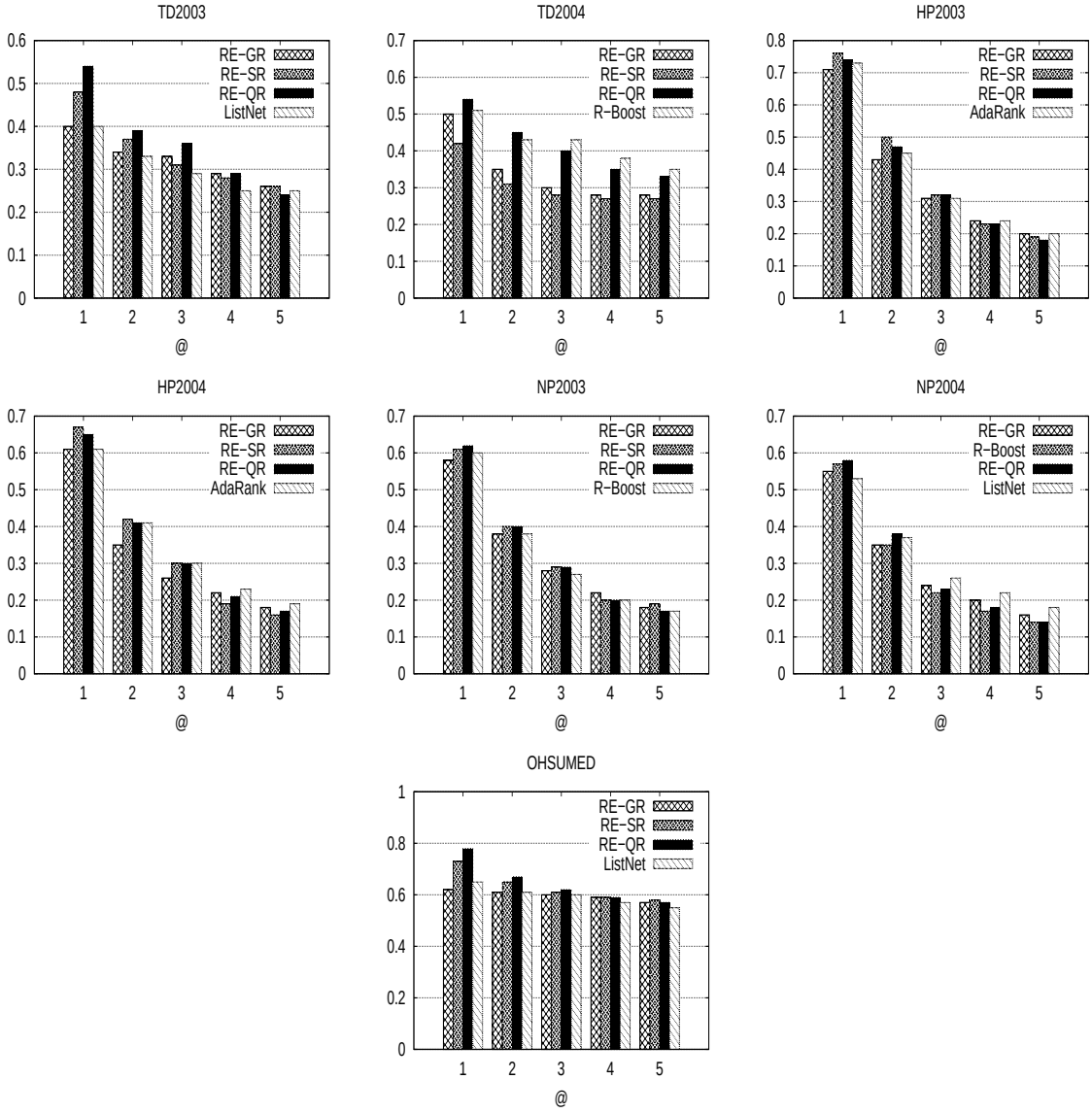

Fig. 1. Precision numbers. Only the best baseline is shown.

How fast are the proposed methods?

The computational efficiency of the proposed methods was evaluated through the average execution time per query, that is, the average processing time spent in extracting rules from $\mathcal{D}$ (or $\mathcal{D}^*$) and estimating the relevance of all documents retrieved for a query. Table IX shows the execution times for each subset. RE-GR is usually the fastest method, since it only extracts global rules from $\mathcal{D}$. Processing query-sensitive information incurs some overhead. Specifically, RE-SR has to perform the additional process of selecting stable rules from the set of all global rules. RE-QR has to perform the additional process of extracting rules of the form $\mathcal{X} \rightarrow q_i$ in order to approximate the competence of query-level functions. These overheads make RE-SR and RE-QR slower than RE-GR. However, the magnitude of that increase in execution time is almost imperceptible for the final user. We also compared the execution times of the proposed methods against R-SVM and ListNet, and we found that the proposed methods are also competitive in terms of computational efficiency.

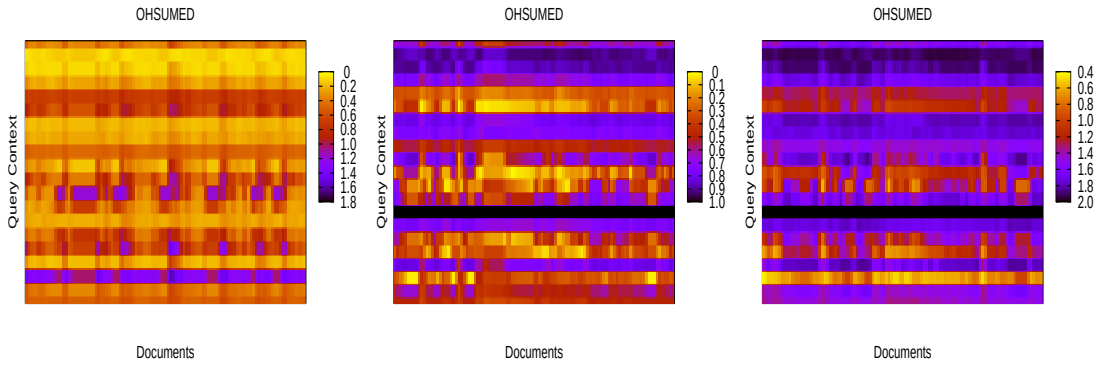

Fig. 2. Left – Documents with relevance 0. Middle – Documents with relevance 1. Right – Documents with relevance 2.

Subset	Method				
	RE-GR	RE-SR	RE-QR	R-SVM	ListNet
OHSUMED	0.10	0.12	0.14	0.13	0.19
TD2003	1.15	1.22	1.27	1.12	1.34
TD2004	1.31	1.39	1.47	1.33	1.75
HP2003	0.93	1.00	1.03	0.99	1.12
HP2004	1.27	1.35	1.41	1.37	1.51
NP2003	0.88	0.92	0.97	0.95	1.10
NP2004	1.13	1.21	1.26	1.21	1.33

Table IX. Execution time (per query) in seconds.

6. CONCLUSIONS

In this paper we propose novel learning to rank methods using association rules for the sake of relevance estimation. The first method (RE-SR) is based on the concept of stable rules, which are rules able to perform very trustworthy predictions. The other proposed method (RE-QR) is much finer-grained. It takes into account the query context extracting query-level rules in order to produce a hybrid ranking function by the combination of multiple query-level functions. In fact, each query-level function has a particular domain of competence, being able to provide highly accurate relevance estimates for certain documents.

Experimental results, obtained using the LETOR 3.0 benchmark, indicate that our methods outperform all state-of-the-art learning to rank methods in most of the subsets, with gains in terms of MAP ranging from 7% to 48%. Results obtained by the execution of RE-QR lead us to conclude that improved ranking performance is obtained by exploiting domains of competence in order to produce hybrid functions. Thus, as future work, we intend to move forward by investigating how to provide hybrid ranking functions using multiple learning to rank approaches according to their domains of competence. Further, we also intend to investigate the reasons for variations in ranking performance depending on the characteristics of the collections.

REFERENCES

AGRAWAL, R., IMIELINSKI, T., AND SWAMI:, A. N. Mining association rules between sets of items in large databases. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*. Washington, USA, pp. 207–216, 1993.

BURGES, C. J. C., SHAKED, T., RENSHAW, E., LAZIER, A., DEEDS, M., HAMILTON, N., AND HULLENDER, G. N. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning*. Bonn, Germany, pp. 89–96, 2005.

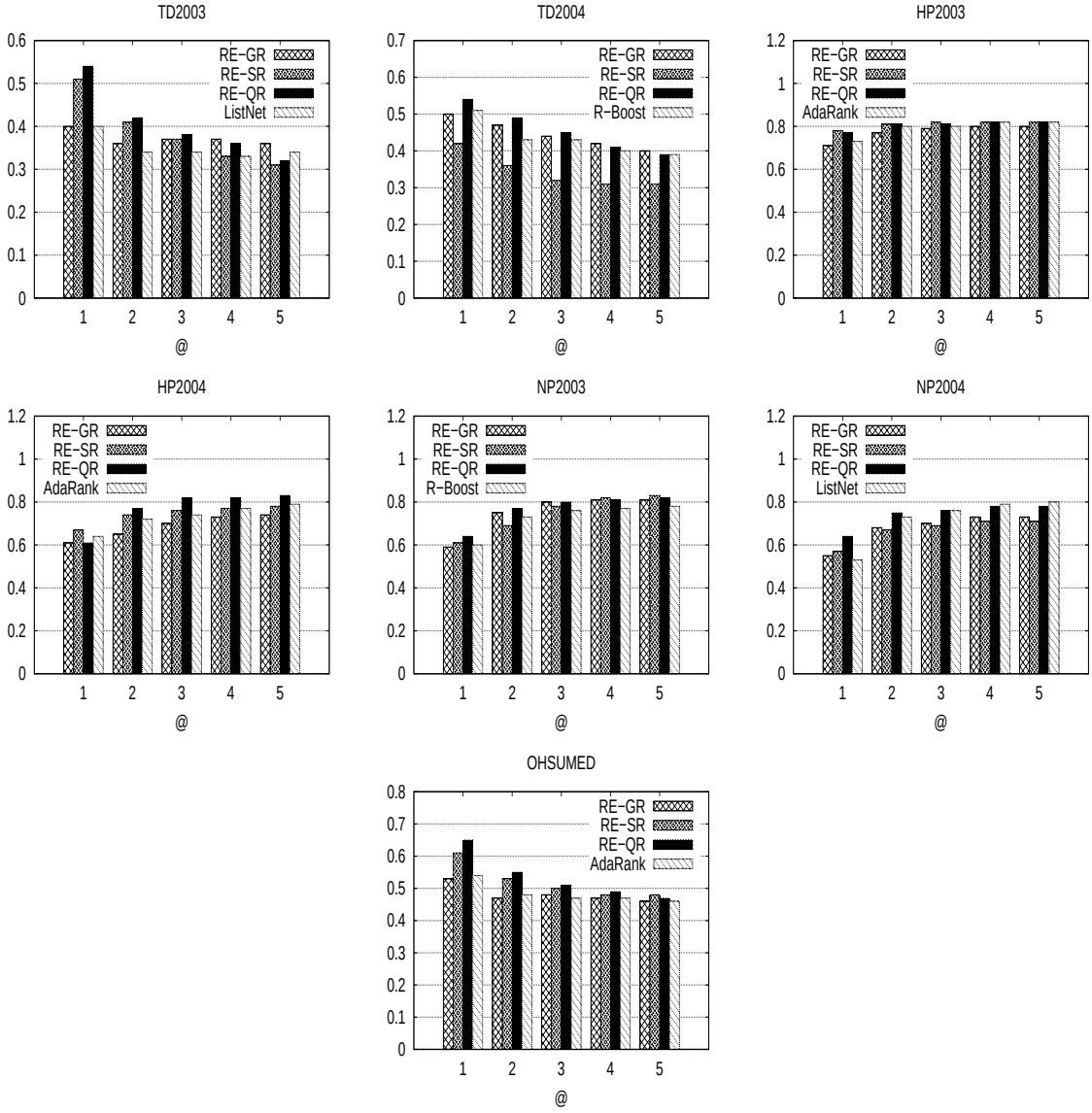

Fig. 3. NDCG Numbers. Only the best baseline is shown.

CAO, Y., XU, J., LIU, T.-Y., LI, H., HUANG, Y., AND HON, H.-W. Adapting ranking SVM to document retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Seattle, USA, pp. 186–193, 2006.

CAO, Z., QIN, T., LIU, T.-Y., TSAI, M.-F., AND LI, H. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning*. Corvalis, USA, pp. 129–136, 2007.

COHEN, W. W., MCCALLUM, A., AND ROWEIS, S. T. Query-level stability and generalization in learning to rank. In *Proceedings of the International Conference on Machine Learning*. Helsinki, Finland, pp. 512–519, 2008.

DE ALMEIDA, H. M., GONÇALVES, M. A., CRISTO, M., AND CALADO, P. A combined component approach for finding collection-adapted ranking functions based on genetic programming. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, pp. 399–406, 2007.

FAYYAD, U. M. AND IRANI, K. B. Multi interval discretization of continuous-valued attributes for classification learning. In *Proceedings of the International Joint Conference on Artificial Intelligence*. Chambéry, France, pp. 1022–1027, 1993.

FREUND, Y., IYER, R., SCHAPIRE, R., AND SINGER, Y. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research* vol. 4, pp. 933–969, 2003.

GAO, J., QI, H., XIA, X., AND NIE, J. Linear discriminant model for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Salvador, Brazil, pp. 290–297, 2005.

GENG, X., LIU, T.-Y., QIN, T., ARNOLD, A., LI, H., AND SHUM, H.-Y. Query dependent ranking using k-nearest neighbor. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Singapore, pp. 115–122, 2008.

JOACHIMS, T. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. Edmonton, Canada, pp. 133–142, 2002.

LIU, Y., XU, J., QIN, T., XIONG, W., AND LI, H. LETOR: Benchmark dataset for research on learning to rank for information retrieval. In *Learning to Rank at SIGIR 2007*, 2007.

LIU, Y.-T., LIU, T.-Y., QIN, T., MA, Z., AND LI, H. Supervised rank aggregation. In *Proceedings of the International Conference on World Wide Web*. Banff, Canada, pp. 481–489, 2007.

MATVEEVA, I., BURGES, C., BURKARD, T., LAUCIUS, A., AND WONG, L. High accuracy retrieval with multiple nested ranker. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Seattle, USA, pp. 437–444, 2006.

NALLAPATI, R. Discriminative models for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Sheffield, UK, pp. 64–71, 2004.

QIN, T., LIU, T., ZHANG, X., WANG, D., XIONG, W., AND LI, H. Learning to rank relational objects and its application to web search. In *Proceedings of the International Conference on World Wide Web*. Beijing, China, pp. 407–416, 2008.

QIN, T., ZHANG, X., WANG, D., LIU, T., LAI, W., AND LI, H. Ranking with multiple hyperplanes. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, pp. 279–286, 2007.

QIN, T., ZHANG, X.-D., TSAI, M.-F., WANG, D.-S., LIU, T.-Y., AND LI, H. Query-level loss functions for information retrieval. *Information Processing and Management* 44 (2): 838–855, 2008.

TROTMAN, A. Learning to rank. *Information Retrieval* 8 (3): 359–381, 2005.

TSAI, M.-F., LIU, T.-Y., QIN, T., CHEN, H.-H., AND MA, W.-Y. FRank: a ranking method with fidelity loss. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, pp. 383–390, 2007.

VELOSO, A., DE ALMEIDA, H. M., GONÇALVES, M. A., AND JR., W. M. Learning to rank at query-time using association rules. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Singapore, pp. 267–274, 2008.

XIA, F., LIU, T.-Y., WANG, J., ZHANG, W., AND LI, H. Listwise approach to learning to rank: theory and algorithm. In *Proceedings of the International Conference on Machine Learning*. Helsinki, Finland, pp. 1192–1199, 2008.

XU, J. AND LI, H. Adarank: a boosting algorithm for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, pp. 391–398, 2007.

XU, J., LIU, T., LU, M., LI, H., AND MA, W. Directly optimizing evaluation measures in learning to rank. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Singapore, pp. 107–114, 2008.

YUE, Y., FINLEY, T., RADLINSKI, F., AND JOACHIMS, T. A support vector method for optimizing average precision. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. Amsterdam, The Netherlands, pp. 271–278, 2007.