A Real-World Framework for Translator as Expert Retrieval

Navid Rekabsaz[1] and Mihai Lupu[2]

[1] Faculty of Informatics, Vienna University of Technology
navid.rekabsaz@student.tuwien.ac.at
[2] Information and Software Engineering Group Vienna University of Technology
A-1040 Vienna, Austria lupu@ifs.tuwien.ac.at

Abstract This article describes a method and tool to identify expert translators in an on-demand translation service. We start from existing efforts on expert retrieval and factor in additional parameters based on the real-world scenario of the task. The system first identifies topical expertise using an aggregation function over relevance scores of previously translated documents by each translator, and then a learning to rank method to factor in non-topical relevance factors that are part of the decision-making process of the user, such as price and duration of translation. We test the system on a manually created test collection and show that the method is able to effectively support the user in selecting the best translator.

1 Introduction

We look at the technology of Information Retrieval from the perspective of a real-world user scenario involving the selection of human translators based on a combination of expertise and practical factors. It has become more and more common place to consider search technology in a series of applications previously served only by database technology, if at all by a computer system [1]. In such cases, new data, new users, and new scenarios need to be observed, existing methods have to be adapted to the task at hand, and new evaluation procedures have to be devised.

This paper addresses the problem of searching translators as experts. We offer a novel translator-expert retrieval platform and evaluate different expert retrieval methods based on a multilingual dataset. In contrast to common expert retrieval systems, we also include non-topical factors involved in the search for a translator (such as price and delivery time). The proposed method has two distinct components: A proficiency estimation phase, in which different aggregation algorithms related to documents of translators are studied based on the proofreaders' assessments as gold standard; and a Learning-to-Rank phase in which different features are tested under different Learning-to-Rank methods based on a manually created ground truth, which we make available together with the

gold standard and the document similarities, under GPL[3]. The contributions of the report are three-fold:

1. the application and adaptation of state-of-the-art IR methods to a new use-case
2. extensive evaluation in a realistic scenario, including non-topical relevance criteria as part of the evaluation
3. creation of a publicly available test collection for both of the steps involved in the retrieval framework

The remainder of the paper is organised as follows: In Section 2, the use-case is presented and the Translator-Expert Retrieval framework is described in detail. Then, Section 3 explains the methods used in the study. Section 4 shows the result of applied methods on the framework. We discuss these results and conclude the study in Section 5.

2 Translator recommendation

2.1 Use-case

The user model for this application is that of an online user in possession of a document in a language other than a desired one. The need for a different language comes either from an internal need to know the contents of the document, or from an external requirement to provide the document in a high-quality translation in the desired language. However, the document is not simply an official document (e.g. a birth certificate) since the platform does not provide legal translation services. Therefore, we can assume that the document to be translated has a particular narrative and a certain topic.

The task of this user is to identify a translator who balances translation quality with non-functional requirements such as cost and delivery time.

The system is therefore charged to estimate the proficiency of the translator on the topic of the document at hand by considering previously translated documents, and to learn a preference model that a typical user will have in combining this proficiency estimation with the other aspects involved in the decision making process (monetary, temporal and social). A reasonable hypothesis, which we will verify in what follows, is that a high-proficiency, low-cost, fast-delivery, professionally known translator will be preferred.

2.2 The Platform

Essential components of the platform as well as the workflow of searching for the translators are depicted in Figure 1. The platform consists of four main components: *Ranking*, *Proficiency Estimator*, *Scheduler* and *Profiles*.

The *Profiles* component stores translator profile information i.e. source and target languages, offered price and translation duration per word, as well as the number of translations the translator has performed for the same client.

[3] `https://github.com/neds/expert-retrieval-translators`

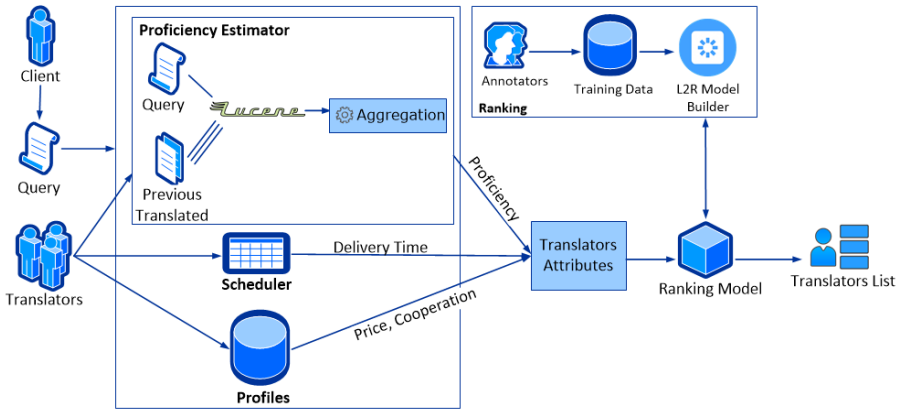


Figure 1: Translator-Expert Search Workflow

The *Scheduler* system calculates the delivery time based on the timetable of each translator. The scheduler builds an efficient data structure to calculate the delivery time in a reasonable response time. The details of the process are out of scope of the paper. In this report we focus on the Proficiency Estimator and the Ranking elements.

The *Proficiency Estimator* sub-system stores the previously-translated documents of each translator and indexes them using Lucene. The similarity between query and indexed documents is used as a basis for the estimation of translator's proficiency for the task at hand. The proficiency score is obtained by aggregating the documents' similarity scores. Different aggregation functions are analyzed in Section 4. Finally, the *Ranking* sub-system uses all the data generated in the previous steps to create the ranking model. It uses Learning to Rank techniques to return the most relevant candidate translators. The training data is provided by a group of annotators familiar with the business of the company, using an evaluation system created specifically for generating this ground truth. The evaluation system presents three translators and the annotators rank them based on the values of each of their attributes (i.e. proficiency, delivery time, price, cooperation). In order to prevent bias in evaluation, the translators are suggested randomly and without name and picture. The applied learning to rank methods and their results are described in Section4.

Separately from the mentioned workflow, after finishing the translation, another expert (a proofreader) revises the translation. The proofreader is selected by the client and guarantees the quality of the final translation. As well as revising, the proofreader assesses the quality of translator's task from different points of view (grammar, style, accuracy, content and language). The assessment value can be from 1 (very bad) to 5 (perfect). In Section 4 we use these assessments to evaluate and compare aggregation algorithms.

3 Methods and Related Work

With the development of information retrieval (IR) techniques, many research efforts go beyond traditional document retrieval and address high-level IR such as entity retrieval and expertise retrieval [2]. The goal of expertise retrieval is to link humans to expertise areas, and vice versa. In other words, the task of expertise retrieval is to identify a set of persons with relevant expertise for the given query [3,4]. The launch of the Expert Finding task at TREC has generated a lot of interest in expertise retrieval, following by rapid progress being made in terms of modeling, algorithms, and evaluation aspects [5,4].

Cao and colleagues [5] propose two principal approaches in the expertise retrieval area based on probabilistic language modeling techniques. They were formalized as so-called *candidate models* and *document models*. The candidate-based approach, also referred to as profile-based method, builds a textual representation of candidate experts and then ranks them based on the query. The document models first find documents relevant to the topic and then locate the experts associated with these documents [3].

In either of the two models, aggregation functions have a significant effect on the performance of expert retrieval systems. Aggregate tasks are those where documents' similarities are not the final outcome, but instead an intermediary component. In expert search, a ranking of candidate persons with relevant expertise to a query is generated after aggregation of their related documents [6].

Ranking techniques are an essential part of each IR framework. In recent years, Learning to Rank (L2R) has been studied extensively especially for document retrieval. It refers to machine learning techniques for training the model in a ranking task [3]. In essence, expert search is a ranking problem and thus the existing L2R techniques can be naturally applied to it [7].

For the task at hand, we found that the two methods have to be used in two different steps. Aggregation can be used to bring together in one value elements essentially of the same nature. In this case - query similarity scores of different documents. The test collection at hand relies for its topical similarity exclusively on term frequencies, as there are no hyperlinks, metadata, or other sources of information in the documents. As a second step, in order to bring in attributes orthogonal to topical similarity, learning to rank methods are an obvious choice. In the following, we describe related work related to these two aspects, and in doing so prepare the ground for our experiments, which we present in the next section.

3.1 Aggregation Functions

The aggregation function has a significant impact on the performance of Expert Retrieval system [8]. As a usual scenario in expert retrieval systems, first each document related to an expert is scored and ranked regarding to query. Then, the top N document scores associated with a candidate expert are aggregated in order to rank the experts.

MacDonald and Ounis [9] consider expert search as a voting problem, where documents vote for the candidates with relevant expertise. Eleven data fusion methods as well as three statistically different document weighting models were tested in their experiments. In practice, the approach considers both the number of documents and expert features regarding to the ranking score of the documents. The results show that while some of adapted voting techniques most likely outperform others, the proposed approach is effective when using the appropriate one.

Cummins and colleagues [8] study the effect of different features on the aggregation function. They show that the number of documents is an important factor, in that the performance of different queries are optimal for different values of N. Comparing query-based features using statistical measures, they infer that the document features (such as TF, IDF) may not, in general, be able to predict the optimal number of documents to aggregate for each query. In contrast, individual Expert Features have been shown to be more informative such that relevant experts are associated with a higher ranked document than non-relevant experts. More interestingly, relevant experts are associated with less documents on average.

Focusing on these features Cummins et al. [8] introduce a new aggregation method. It uses genetic programming to learn a formula for the weights of document associations within the candidate profiles. The formula, denoted as $GP2$, is as follows:

$$GP2 = \frac{\sqrt{\sqrt{2/no\_docs_{x_i}}/(\sqrt{(10/R)+R})}}{\sqrt{sq(10/R)+R+sq(10/R)+\sqrt{R*2}}}$$

where R is the rank of the document in the initial ranking and $no\_docs_{x_i}$ is the total number of documents associated with expert x_i.

3.2 Learning To Rank

Learning to rank refers to machine learning techniques for training a model in a ranking task. Due to importance of ranking problems, learning to rank has been drawing broad attention in the machine learning community recently.

In the learning to rank approach, the ranking problem is transformed to classification, regression and ordinal classification, and existing methods and techniques for solving machine learning problems are applied. As Hang [7] points out, the relation between learning to rank and ordinal classification is that, in ranking, one cares more about accurate ordering of objects, while in ordinal classification, one cares more about accurate ordered-categorization of objects.

The first step in accumulating data required for learning to rank, is relevance judgments, normally done by human annotators. Lie [10] presents the three main strategies in learning to rank:

- *Relevance degree*: In this method, the annotator specifies whether an object is relevant or not to the query. It can be either in binary judgment or by specifying the degree of relevance (e.g., Perfect, Excellent, Good, Fair, or Bad).
- *Pairwise preference*: The annotator compares a pair of objects in order to specify which one is more relevant with regards to a query.
- *Total order*: The annotator specifies the total order of all objects with respect to a query by rating each object.

Among the three mentioned kinds of judgments, the first one is the most popularly used judgment since is the easiest to obtain, while the third one is more accurate but laborious for human annotators. In our case, we have used the total order method because our ranked lists consisted of only 3 translators.

The learning to rank techniques are categorized in three main groups: *Pointwise*, *Pairwise* and *Listwise*.

In the pointwise approach, the ranking problem is transformed to classification, regression or ordinal classification. Therefore, the group structure of ranking is ignored in this approach [7]. Here, linear or polynomial regression are widely used methods.

The pairwise approach transforms the ranking problem into pairwise classification or regression. In fact, it cares about the relative order between two documents. Similar to the pointwise approach, the pairwise method also ignores the group structure of ranking [7]. Here is a brief explanation of some pairwise algorithms:

- *RankNet* [11]: Widely applied by commercial search engines, it uses gradient descent method and neural network to model the underlying ranking function.
- *RankBoost* [12]: It adopts AdaBoost algorithm for the classification over the object pairs.
- *LambdaRank* [13]: It considers the evaluation measures to set its pair weight. In particular, the evaluation measures (which are position based) are directly used to define the gradient with respect to each document pair in the training process.
- *LambdaMART* [14]: It combines the strengths of boosted tree classification and LambdaRank.

The listwise approach takes the entire set of documents associated with a query in the training data as the input and predicts their ground truth labels [10]. In contradiction to two previous approaches, it maintains the group structure of ranking. In addition, ranking evaluation measures can be more directly incorporated into the loss functions in learning [7]. In the following, two common listwise algorithms are briefly discussed:

- *AdaRank* [15]: It applies the evaluation measures on the framework of Boosting and focuses on effectively optimization.
- *ListNet* [16]: It uses different probability distributions in order to define the loss function.

Lie [10] compares the algorithms by applying on different data-sets. It concludes that listwise techniques are in general the most effective among the others. However, the choice of the learning evaluation measure and the rank cutoff may have a noticeable impact on the effectiveness of the learned model [17].

3.3 Evaluation

A critical point in all information retrieval systems is the evaluation of results. The evaluation on the performance of a ranking model is carried out by comparison between the ranking lists output of the model and the ranking lists given as the ground truth. Some common IR evaluation methods like Mean average precision (MAP), [Normalized] Discounted Cumulative Gain ([N]DCG), Mean Reciprocal Rank (MRR) are also widely user in leaning to rank evaluation. Among the mentioned metrics, DCG/NDCG is the only one used for graded relevance.

Recently, Chapelle and Zhang [18] have proposed Expected Reciprocal Rank (ERR) which claims to model user's satisfaction with search results better than the DCG metric. Their work addresses the underlying independence assumption of DCG that a document in a given position has always the same gain and discount independently of the documents shown above it. It asserts that based on research on modeling user click behavior [19,20], the likelihood a user examines the document at rank i is dependent on how satisfied the user was with previously observed documents in the ranked list. In other words, it assumes that a user is more likely to stop browsing if they have already seen one or more highly relevant documents. Introducing the ERR formula, Chapelle and Zhang claim that results reflect real user browsing behavior better and quantifies user satisfaction more accurately than DCG.

4 Experimental Results

In this section, we applied the different approaches presented in the previous section within our platform. By comparing the methods, we aim to discover the most appropriate one considering to the project's characteristics and data.

4.1 Aggregation Functions

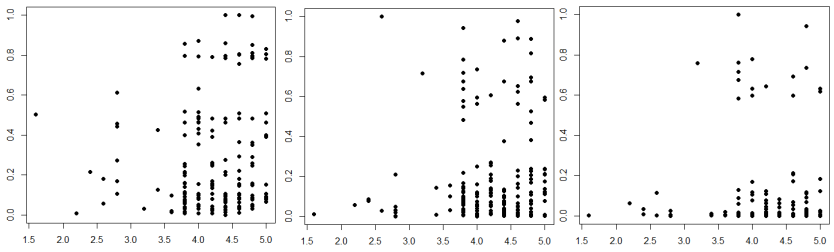
Translator's proficiency on the topic of the query is one of the attributes used for creating the ranking model. In order to obtain the value, we aggregate the similarity values of the previously translated documents with the query document. Selecting the most appropriate aggregation function is the objective of the current section.

In order to evaluate aggregation functions results, we use the assessment of the proof-readers after every translation. As mentioned before, the hypothesis is that a translator is likely to do a better job if she is familiar with the vocabulary of the query. In other words, when the query document is more similar to the

translator's pre-translated documents, we expect that this existing vocabulary expertise of the translator will result in a higher quality translation and therefore a higher score from the proof-reader. We can then use the proof-readers assessments as a gold standard and observe which of the three aggregation methods correlates more with the scores given by the proof-readers.

We repeat the experiment done by Cummins and colleagues [8] for the particular use-case and data at hand, comparing three aggregation algorithms: $GP2$, $Top5$ and $Top1$. $GP2$ is the state of the art, while $Top1$ and $Top5$ are two common forms of $TopN$ aggregation algorithm which refers to algorithm that summarizes the N top documents. Furthermore, $Top1$ can be interpreted as the maximum similarity score, and follows the intuition that it is sufficient for a translator to have had only one highly similar translation job before in order to do a good job on the new task.

As input data we have 181 translation orders collected from the live system. For each of them, we know the translator who performed the translation (including all her previously translated documents), the text of the document to be translated, and the score given by the proof-reader. Because this is historical data, evaluating the aggregation method cannot be done by giving the same translation task to different translators. However, by calculating the value of each mentioned algorithm on all finished translations, we achieve three lists of translations each annotated with an additional aggregation value field. Then, in order to compare the algorithms, we test the correlation of each list with the list of translations annotated with proof-readers' assessments. Again, the assumption being that if we rank translation jobs by aggregation scores, the top elements would have high proof-reader scores, while the bottom elements lower scores. The distribution of data between the values of each aggregation function and the proof-reader's assessments is shown in Figure 2.



(a) GP2 (b) TOP5 (c) TOP1

Figure 2: Data distribution of aggregation values against proof-readers' assessments. The x-axis shows the assessments of proof-readers, while the y-axis represents the values of the corresponding aggregation function

In order to calculate the correlation value, we applied Spearman Rank Order and Kendall Rank Correlation as two common methods. Table 1 shows the results of Spearman correlation coefficient (r_s) and Kendall's tau coefficient

(τ) using the 181 records of purchased orders. In addition, it represents the Significance Test of both methods. For Spearman's test, p-values are computed using algorithm AS 89 [21].

The outcome shows an approximately weak correlation between aggregation functions and assessments of proof-readers. In all the algorithms, the coefficient value of Spearman is slightly higher than Kendall's. Regarding to p-Value of significance test, a meaningful relation between $GP2$ and assessments can be considered. $Top1$ which has the worst values in the table shows a meaningless and near random correlation though.

Comparing the algorithms, $GP2$ outperforms the others in both correlation tests. In comparison to $Top1$, $Top5$ has slightly better performance. The results are also nearly the same when comparing based on language-pairs.

Table 1: Correlation test between algorithms and proof-readers' assessments as well as P-Value of significance of correlation test

		Top1	Top5	GP2
r_s	Correlation Test	0.052	0.089	0.145
	p-Value	0.4866	0.2295	0.05038
τ	Correlation Test	0.034	0.059	0.102
	p-Value	0.5157	0.2562	0.05263

4.2 Learning To Rank

In order to accumulate data required for ranking model, we conduct a survey with eight human annotators. The questions of survey represent three translators each with four criteria (price, delivery time, proficiency and number of cooperation times). As mentioned before, in order to prevent bias in results, the name and the picture of the translators are hidden from the annotators. The annotators rate the questions from one to three based on *Total Order* strategy. The accumulated data consists of 400 annotated list and overall 1200 records.

We use the RankLib library[4] to apply a large set of Learning to Rank methods. The library provides the implementation of some Learning to Rank algorithms as well as evaluation measures in Java. By splitting the data in train, validation and test datasets, we use 5-Fold Cross-Validation on each run. One pointwise, three pairwise and two listwise methods are tested.

Since evaluation of the results should be applied on the entire result list (with 3 items), we have to use relevance grading evaluation measures like DCG, NDCG or ERR. In the current study, NDGC and ERR both with rank position at 3 are used.

Because of the short final predicted list (3 items), every measure returns a very high score. In order to understand the real benefit of our framework compared with just presenting the un-ranked set of three translators, we evaluate two random approaches. In the first approach, we run all the algorithms on data

[4] http://sourceforge.net/p/lemur/wiki/RankLib

with random generated labels. In order to increase the accuracy of results, the process of generating randomized label is repeated 5 times. The second approach is developing a simple ranker which randomly predicts the rank of each record. In fact, the first tests whether there is anything to learn in the data and the second examines if the learning algorithms learn from the data.

The result is shown in Table 2. Figure 3 depicts the corresponding data in diagrams. As it is shown, random values define a base line for comparison the goodness of methods. Among the applied methods, Linear Regression and LambdaMART tend to have better results. In particular, Linear Regression shows a narrower confidence interval and hence more stable. Furthermore, tracing NDCG and ERR diagrams shows a considerable similarity in behavior of both evaluation methods regarding to the data.

In addition to comparing the methods, features comparison can be an interesting point. Table 3 shows the coefficients of features, calculated in Linear Regression model. The coefficient value is a measure for understanding the importance of each feature in comparison to the others.

As it is shown, price and delivery time seem to be the most effective features while the number of cooperation times has the lowest importance. Surprisingly, proficiency plays a small role in the ranking of the translators. It can be because of the proposed business plan of the platform to the clients which guarantees an acceptable quality of translation. In addition, we expect that by applying the methods on much more amount of data, the proficiency feature gains more importance.

Table 2: Results of applying Learning to Rank methods based on NDCG and ERR evaluation measures

Method	NDCG@3		ERR@3	
	Result	Random	Result	Random
Linear Regression	**0.935**	0.833	**0.451**	0.375
RankNet	0.876	0.834	0.394	0.378
RankBoost	0.909	0.831	0.432	0.374
LambdaMART	0.93	0.832	0.447	0.373
ListNet	0.915	0.831	0.439	0.375
AdaRank	0.857	0.83	0.399	0.373
Random Ranker	0.832	0.832	0.375	0.378

Table 3: Coefficient value of features in Linear Regression model

Feature	Value
Price	2.002
Duration	0.057
Proficiency	-0.048
Number of Cooperation Times	-0.313

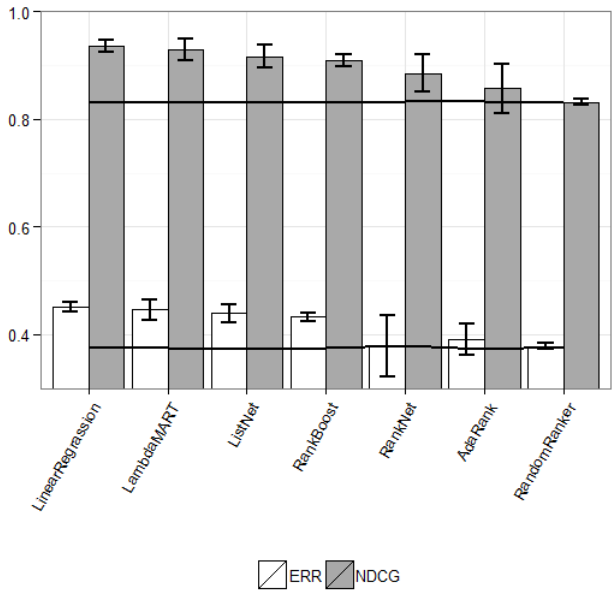


Figure 3: Learning to Rank Results

5 Conclusion and Future Work

We propose a comprehensive solution for a translator-expert retrieval system. As well as system architecture, we thoroughly study the obstacles and pitfalls in implementing such a system. Multilingual IR tools are adapted to solve two essential steps in a practical system: Estimating the translators proficiency on a particular topic (document aggregation), ranking translators based on real-world factors (learning to rank).

To address the first issue we have compared three commonly used aggregation methods. The aggregation methods estimate the proficiency of each translator based on the similarity values of the previous translated documents with the query document. Using the assessment of the proof-readers on the final translation as the gold standard, we compare three aggregation methods. We found that the GP2 method shows better performance in comparison to the others with reasonable good results. Future work in addressing the estimation of translators' proficiency will allow us to increase the correlation between the aggregation and proof-readers' scores.

The second issue tackles the problem of experts ranking. By applying different learning to rank algorithms, we obtain a ranking model based on linear regression with a very high performance. The model's performance is tested with both NDCG and ERR evaluation measures. Feature analysis of data shows that real users consider price and delivery time much more important than the other features. This is relatively disappointing, but in retrospective not surprising for a real-world system.

References

1. Grefenstette, G., Wilber, L.: Search Based Applications. Morgan & Claypool Publishers (2011)
2. Balog, K., Bogers, T., Azzopardi, L., de Rijke, M., van den Bosch, A.: Broad expertise retrieval in sparse data environments. In: Proc. of SIGIR. (2007)
3. Balog, Krisztian, Fang, Y., de Rijke, M., Serdyukov, P., Si., L.: Expertise retrieval. Foundations and Trends in Information Retrieval **6** (2012)
4. Deng, Hongbo, King, I., Lyu., M.R.: Enhanced models for expertise retrieval using community-aware strategies. Systems Man and Cybernetics Part B: Cybernetics IEEE Transactions (2012)
5. Cao, Y., Liu, J., Bao, S., Li, H.: Research on expert search at enterprise track of trec 2005. In: Proc. of TREC. (2005)
6. Macdonald, Craig, Ounis, I.: Learning models for ranking aggregates. Advances in Information Retrieval. Springer Berlin Heidelberg (2011)
7. Hang, L.I.: A short introduction to learning to rank. Transactions on Information and Systems (2011)
8. Cummins, R., Lalmas, M., O'Riordan, C.: Learning aggregation functions for expert search. In: Proc. of ECAI. (2010)
9. Macdonald, C., Ounis, I.: Voting for candidates: adapting data fusion techniques for an expert search task. In: Proc. of CIKM. (2006)
10. Lie, T.Y.: Learning to rank for information retrieval. Foundations and Trends in Information Retrieval. Springer (2011)
11. Burges, C., Shaked, T., Renshaw, E., Lazier, A., Deeds, M., Hamilton, N., Hullender, G.: Learning to rank using gradient descent. In: Proc. of ICML, New York, NY, USA (2005)
12. Freund, Y., Iyer, R., Schapire, R.E., Singer, Y.: An efficient boosting algorithm for combining preferences (1998)
13. Burges, C., C.J., Ragno, Robert, Le, V., Q.: Learning to rank with nonsmooth cost functions, MIT Press (2006)
14. Wu, Q., Burges, C.J.C., Svore, K.M., Gao, J.: Adapting boosting for information retrieval measures, Springer Science (2009)
15. Xu, J., Li, H.: Adarank: A boosting algorithm for information retrieval. In: Procs of SIGIR. (2007)
16. Cao, Z., Qin, T., Liu, T.Y., Tsai, M.F., Li, H.: Learning to rank: From pairwise approach to listwise approach. In: Procs of ICML. (2007)
17. Macdonald, C., Santos, R.L., Ounis, I.: The whens and hows of learning to rank for web search. Inf. Retr. **16** (2013)
18. Chapelle, O., Zhang, Y.: Expected reciprocal rank for graded relevance. In: Proc. of CIKM. (2009)
19. Craswell, N., Zoeter, O., Taylor, M., Ramsey, B.: An experimental comparison of click position-bias models. In: Proc. of WSDM. (2008)
20. Büttcher, S., Clarke, C.L.A., Yeung, P.C.K., et al.: Reliable information retrieval evaluation with incomplete and biased judgements (2007)
21. Best, D.J., Roberts, D.E.: Algorithm as 89: The upper tail probabilities of spearman's rho. Journal of the Royal Statistical Society. Series C (Applied Statistics) **24** (1975)