

# A Stochastic Learning-To-Rank Algorithm and its Application to Contextual Advertising

Maryam Karimzadehgan<sup>\*</sup>  
 Department of Computer Science  
 University of Illinois at Urbana-Champaign  
 Urbana, IL 61801  
 mkarimz2@illinois.edu

Wei Li, Ruofei Zhang, Jianchang Mao  
 Advertising Sciences, Yahoo! Labs  
 4401 Great America Parkway  
 Santa Clara, CA 95054  
 {wei,rzhang,jmao}@yahoo-inc.com

## ABSTRACT

This paper is concerned with the problem of learning a model to rank objects (Web pages, ads and etc.). We propose a framework where the ranking model is both optimized and evaluated using the same information retrieval measures such as Normalized Discounted Cumulative Gain (NDCG) and Mean Average Precision (MAP). The main difficulty in direct optimization of NDCG and MAP is that these measures depend on the rank of objects and are not differentiable. Most learning-to-rank methods that attempt to optimize NDCG or MAP approximate such measures so that they can be differentiable. In this paper, we propose a simple yet effective stochastic optimization algorithm to *directly* minimize any loss function, which can be defined on NDCG or MAP for the learning-to-rank problem. The algorithm employs Simulated Annealing along with Simplex method for its parameter search and finds the global optimal parameters. Experiment results using NDCG-Annealing algorithm, an instance of the proposed algorithm, on LETOR benchmark data sets show that the proposed algorithm is both effective and stable when compared to the baselines provided in LETOR 3.0. In addition, we applied the algorithm for ranking ads in contextual advertising. Our method has shown to significantly improve relevance in offline evaluation and business metrics in online tests in a *real large-scale advertising serving system*. To scale our computations, we parallelize the algorithm in a MapReduce framework running on Hadoop.

## Categories and Subject Descriptors

H.3.3 [Information Search and Retrieval]: Retrieval models

## General Terms

Algorithms, Experimentation

## Keywords

Learning to Rank, Simulated Annealing, Simplex Algorithm, NDCG-Annealing, Contextual Advertising, NDCG, IR Measures.

<sup>\*</sup>Work conducted during an internship at Yahoo! Labs.

Copyright is held by the International World Wide Web Conference Committee (IW3C2). Distribution of these papers is limited to classroom use, and personal use by others.

WWW 2011, March 28–April 1, 2011, Hyderabad, India.  
 ACM 978-1-4503-0632-4/11/03.

## 1. INTRODUCTION

Ranking is the central part of many applications including document retrieval, recommender systems, advertising and so on. Many models for ranking functions have been proposed previously, including vector space model [43], probabilistic model [41] and language model [35]. Web page ranking previously has been done on a manually tuned ranking function, e.g., BM25 [42]. Manual parameter tuning is usually very difficult especially when there are many parameters. In recent years, supervised learning methods (learning-to-rank methods) have been devoted to automatically learn an effective ranking function from training data. Learning-to-rank methods (e.g., [6, 8, 9, 15, 17, 47]) have been proposed to optimize a ranking function that incorporates a variety of features and avoids tuning a large number of parameters empirically. In document retrieval, the ranking results are usually measured in terms of MAP (Mean Average precision) [2] and NDCG (Normalized Discounted Cumulative Gain) [21] which are non-differentiable.

Learning to rank when applied to document retrieval is as follows: Training data consists of a set of queries, a set of documents for each query and a label for each document which shows the degree of relevancy of the document to its corresponding query. Each query-document pair is represented by a feature vector. A ranking function is then created using the training data and the learned model can predict the ranking list in the training data well. In retrieval ranking (i.e., the test stage), given a query, the ranking function assigns a score to each document and then documents are ranked in descending order according to the assigned scores.

Several methods of learning to rank have been proposed in the machine learning community. Most existing methods for document retrieval (e.g., [6, 7, 17, 19, 22, 31]) are designed to optimize loss functions loosely related to the IR performance measures, not loss functions directly based on the measures. Recently, there have been some attempts to directly optimize the performance measures in terms of the IR evaluation measures. Some of these methods minimize upper bounds of the loss function defined on the IR measures [47, 50]. The others either approximate the IR measures with functions that are easy-to-handle [12, 44] or use especially designed methods for optimizing the non-smooth IR measures [1, 49].

In this paper, we propose a stochastic algorithm to learn a ranking function that minimizes the loss function defined **directly** on the IR evaluation measures including NDCG and MAP. The algorithm is based on Simulated Anneal-

ing [23] framework which uses a modification of downhill Simplex method [32] for finding the next candidate to move, to minimize the loss function. The method of Simulated Annealing is a global optimization technique for large-scale problems especially for the problems where the global minimum is hidden among many local minima. Each step of the algorithm replaces the current solution by a nearby solution which is chosen with a probability that depends on the difference between their corresponding function values and on a global parameter  $T$  (i.e., temperature) which is gradually decreased. The loss function in our method is application specific; we do not have certain restrictions on the properties of the loss function, such as continuity and differentiability. As an instance of the method, we define the loss function on NDCG measure for ranking in this paper, but other metrics can also be applied directly. We call the proposed method *NDCG-Annealing algorithm*. The NDCG-Annealing algorithm is general to be applied to both *linear* and *non-linear* ranking functions at test time.

We compare the NDCG-Annealing algorithm (with linear ranking function described in section 3) with baselines provided in the LETOR 3.0 datasets. The results on seven datasets in LETOR 3.0 show that the NDCG-Annealing algorithm can outperform the baselines and it is more stable.

We also applied the algorithm to rank ads (with non-linear ranking function described in section 7) in contextual advertising in both online and offline scenarios. In contextual advertising, ads are shown to a user on a Web page; the user clicks on an ad, visits the advertiser site and then converts. Ads are ranked to optimize different objectives, such as relevance, Click-through-Rate (CTR) [10] or revenue. In this paper, our ranking problem considers how to rank ads for a given page to optimize relevance and CTR, which consequently also increase revenue. Matching Web pages with ads is done using the standard Cosine similarity metric. The ranking results are first evaluated by editors based on relevancy between (page, ad) pairs. Our approach is shown to significantly improve the relevance over the baseline. Further evaluation on a real large-scale advertising serving system shows that the algorithm improves CTR and Revenue Per 1000 Impressions (RPM) in online bucket tests. Also, to scale our computations (building an inverted index, extracting features and training the model), we parallelize our algorithm in a MapReduce framework running on Hadoop [16].

The rest of the paper is organized as follows: We discuss related work in Section 2. The problem formulation is given in Section 3. We then present our NDCG-Annealing algorithm in Section 4. We discuss experiment design and experiment results on LETOR 3.0 dataset in Sections 5 and 6, respectively. Section 7 presents the application of the proposed algorithm in contextual advertising. Finally, we conclude the paper in Section 8.

## 2. RELATED WORK

“Ranking” is a key problem of many applications especially to create a model that can sort documents for a given query. Previous models in Information Retrieval such as BM25 [42] and language model [35] *only* have a few parameters to tune. As the ranking models become more sophisticated by considering more relevant features for documents and queries, how to manually tune the parameters becomes a challenge.

Recently, learning-to-rank methods have been proposed to automatically learn a ranking function using labeled train-

ing data. Three main approaches have been proposed:

**1) Pointwise approaches** learn a ranking function to minimize a loss function that is defined on individual document relevance judgment and ranking score which could be based on regression [12] or classification [25, 31]. One of the problems with these approaches is that the training model might be biased towards queries with more document pairs [8].

**2) Pairwise approaches** learn a ranking function to minimize a loss function that is defined on pair-wise preferences. The ranking problem is then transformed into the binary classification problem. Examples of such a model are RankSVM [19, 22], RankBoost [17], RankNet [6] and FRank [15]. The main problem with these methods is that the objective function is formalized as minimizing errors in classification rather than minimizing errors in ranking of documents. To overcome such a problem, the final approach, i.e., Listwise approaches consider document lists instead of document pairs as instances in learning.

**3) Listwise approaches** can be classified into two categories: the first one directly optimizes the IR measures. Optimizing IR measures directly is difficult [7] since they depend on the rank and are not differentiable. To avoid the computational difficulty, in [7], authors perform gradient descent on a smoothed version of the objective function, however, the objective function is calculated implicitly through its gradients. In [46], optimizing the expectation of IR measures is proposed, however the Monte Carlo sampling is used to address the intractable task of computing the expectation which might not be a good approximation for queries with a large number of documents. SVM-MAP [50] also relaxes the MAP metric by incorporating it into the constraints of SVM. Since SVM-MAP optimizes MAP, it can only work with binary relevance and is not suitable for more than two levels of relevance judgments. AdaRank [47] uses boosting to optimize NDCG, for that they deploy the NDCG value of each query in the current iteration as the weight for that query in constructing the weak ranker. The convergence of AdaRank is conditional and not guaranteed. NDCG-Boost [45] is a recent algorithm that optimizes the expectation of NDCG over all the possible permutations of documents. Some other approaches for directly optimizing IR measures use Genetic Programming [1, 49] or approximate the IR measures with the functions that are easy-to-handle [44, 12]. Our method is similar to these methods as we directly optimize the IR evaluation measure (i.e., NDCG) by using the Simulated Annealing which uses a modification of downhill Simplex method for the next candidate move to find the global minimum.

The second category of listwise algorithms defines loss function as an indirect way to optimize the IR evaluation metrics. For example, RankCosine [36] uses the cosine similarity between the ranking list and the ground truth as a query level loss function. ListNet [9] uses the KL-divergence as a loss function by defining a probability distribution. There is a problem with these approaches that optimizing the listwise loss function does not necessarily result in the optimization of IR metrics.

For a complete survey about all learning-to-rank methods, a reader is referred to [26].

Also, our work is related to other applications that use Simulated Annealing including clustering [28, 48], classification [4] and [33] which uses the Simulated Annealing algorithm to rank Web objects. The work in [33] calculates the

object popularity scores of domain-dependent Web objects based on their Web popularity and the object relationship graph. They automatically assign a popularity propagation factor for each type of object relationship. For the parameter estimation, they adopt Simulated Annealing algorithm. Also, authors in [34] have applied Simulated Annealing algorithm to select a framework that selectively applies an appropriate ranking function for each query. The work in [20] adopts the algorithm for object detection in computer vision by proposing a collaborative learning algorithm enhanced by a Simulated Annealing step in combination with AdaBoost algorithm. Our work is similar to all previous work in a sense that we also use Simulated Annealing algorithm, however we combine it with the downhill Simplex algorithm and systematically examine the algorithm in comparison with other state-of-the-art algorithms, and apply it in a different application, i.e., ranking ads in contextual advertising.

Since we consider ranking in advertising as a direct application of our method, we review the related work in contextual advertising. Some previous work has focused on developing methods to match ads to pages or users' search queries (e.g. [5, 39]). In these studies, the problem of matching ads with pages is transformed into a similarity search in a vector space. Each page or ad is represented as a vector of features, which include words, along with higher-level semantic classes [5]. Our method is similar to these methods in a sense that our ranking function is based on vector space model, however differs in that we learn weights for the importance of different parts of the page and ad by using NDCG-Annealing algorithm. Indeed, the work in [10] learns weights for each word in a page, however their method introduces a lot of parameters which makes it harder to scale up for real ad serving system. Some other studies have examined the use of semantic classes to match pages and ads for a common set of semantic classes [5]. In addition, [37] introduces a page-ad probability model in which semantic relationships between page terms and ad terms are modeled with hidden classes. Another work [30] applies translation techniques to improve the ad-page matching. A recent study [3] close to our work considers a ranking method for conversions. However, we propose a learning method to maximize Click-through-Rate (CTR) for impressions. Also, the work in [24] applies Genetic Programming to learn ranking functions that select the most appropriate ads.

Another line of work uses click data to produce a CTR estimate for an ad, independent of the page or query in the Sponsored Search scenario [38, 40]. These studies make a simplifying assumption that the ads are selected by matching the bid phrase to a phrase from the page or query in Sponsored Search and therefore to select the most clickable ads, one only needs to estimate the CTR on the ads with the matching bid phrase.

### 3. PROBLEM FORMULATION

We assume there is a collection of  $n$  queries for training, i.e.,  $\mathcal{Q} = \{q^1, \dots, q^n\}$  and for each query  $q^k$ , we have a collection of  $m_k$  documents, i.e.,  $D^k = \{d_i^k, i = 1, \dots, m_k\}$  where relevance of  $d_i^k$  is given by a vector  $r^k = (r_1^k, \dots, r_{m_k}^k)$ . Thus, the training set can be represented as  $S = \{(q^k, D^k, r^k)\}_{k=1}^n$ .

Linear feature-based ranking functions are of the form:

$$\mathcal{F}_\lambda(d, q) = \sum_j \lambda_j f_j(q, d) \quad (1)$$

where  $q$  is a query,  $d$  is a document,  $f_j(q, d)$  is a feature function and  $\lambda_j$  is the weight assigned to feature  $j$ . The ranking function  $\mathcal{F}_\lambda(d, q)$  is linear with respect to the model parameters,  $\lambda_j$ 's. This model is simple in that it can combine many different features in a straightforward manner. There are different learning methods to find the parameters that optimize retrieval effectiveness for linear ranking functions. For our case, we use *NDCG-Annealing algorithm* to learn the model parameters. Note that this algorithm is a general approach that can also be applied to non-linear ranking functions with respect to the model parameters (as we see in section 7). Then, for test queries, our linear ranking function  $\mathcal{F}_\lambda(d, q)$  takes a document-query pair  $(d, q)$  and outputs a real number as a score. Documents are then ranked according to this score.

In the next section, we describe our objective function.

### 3.1 Objective Function

The Discounted Cumulative Gain (DCG) score [21] is a commonly used measure for multi-level relevance judgments. It has a logarithmic position discount, i.e., the benefit of seeing a relevant document at position  $i$  is  $\frac{1}{\log_2(i+1)}$ .

The *NDCG*( $\mathcal{Q}, \mathcal{F}_\lambda$ ) for ranking function  $\mathcal{F}_\lambda(q, d)$  is defined as:

$$NDCG(\mathcal{Q}, \mathcal{F}_\lambda) = \frac{1}{n} \sum_{k=1}^n \frac{1}{Z_k} \sum_{i=1}^{m_k} \frac{2^{r_i^k} - 1}{\log(1 + j_i^k)} \quad (2)$$

where  $j_i^k$  is the rank of document  $d_i^k$  within  $D^k$  for query  $q^k$ ,  $Z_k$  is the normalization factor [21] and  $n$  is the total number of queries.

NDCG measure is usually truncated at a particular truncation (rank) level  $J$  (e.g., the first 10 retrieved documents), denoted by *NDCG@J* by ignoring the documents after rank  $J$  to emphasize the importance of the first retrieved documents.

We define the loss function as:

$$\mathcal{L}(\mathcal{Q}, \mathcal{F}_\lambda) = 1 - NDCG@10(\mathcal{Q}, \mathcal{F}_\lambda) \quad (3)$$

where 1 indicates the ideal NDCG value and *NDCG@10*( $\mathcal{Q}, \mathcal{F}_\lambda$ ) is the NDCG value for all queries given the learned parameters  $\lambda_j$ 's. The learning process minimizes the aforementioned loss function by iteratively finding the best  $\lambda_j$ 's that minimize the loss function. We consider *NDCG@10* (without loss of generality) to emphasize the importance of the first ten documents but any other value  $J$  can also be used.

In the next section, we present NDCG-Annealing algorithm for learning the model parameters, i.e.,  $\lambda_j$ 's.

## 4. PARAMETER ESTIMATION

### 4.1 NDCG-Annealing Algorithm

In this section we describe the details of integrating Simulated Annealing and downhill Simplex method in the optimization framework to minimize the loss function associated directly to NDCG measure.

The idea of Simulated Annealing comes from Metropolis et al. [29] in which the authors described the cooling of

material in a heat bath. The idea is that at high temperatures, molecules of a liquid move freely and if the liquid is cooled slowly, the mobility is lost and the atoms can then line up. The essence of the whole process is slow cooling. Later, Kirkpatrick et al. [23] applied the idea of Metropolis algorithm to optimization problems by searching for feasible solutions and converging to an optimal solution.

For estimating the model parameters, we use Simulated Annealing algorithm [14], while the parameter search at each temperature applies downhill Simplex method [32]. Simulated Annealing algorithm is applicable to optimizing  $N$ -dimensional spaces (e.g.,  $N$  feature spaces) by finding the minimum of some function, i.e.,  $\mathcal{L}(\mathcal{Q}, \mathcal{F}_\lambda)$  where  $\lambda$  is a  $N$ -dimensional vector, and  $\mathcal{L}$  is the objective function. There are some components associated with the algorithm:

1. There is a generator of random changes which is a procedure that takes a random step from  $\lambda$  to  $\lambda + \Delta\lambda$ .
2. An objective function  $\mathcal{L}(\mathcal{Q}, \mathcal{F}_\lambda)$  (analogy of energy) whose minimization is the goal of the procedure.
3. There is a control parameter called Temperature (T), with an annealing schedule by which tells how it is lowered from high to low values.

The most important of these components is  $\Delta\lambda$ . There are many different schemes for choosing  $\Delta\lambda$ . One efficient way of doing Simulated Annealing minimization on continuous control spaces is to use a modification of downhill Simplex method. This method only requires function evaluations, not derivatives. A *simplex* is a geometrical figure which in  $N$  dimensions, consists of  $N + 1$  points. In  $N$ -dimensional minimization, the *downhill Simplex* algorithm starts with a guess, i.e.,  $(N+1)$  points, which define an initial simplex. The algorithm then makes its own way downhill through an  $N$ -dimensional topology until it finds a minimum. For doing that, the downhill Simplex method takes a set of steps. Most steps just move the point of the simplex where the objective value is largest (highest point) to a lower point with the smaller objective value. These steps are called *reflections* which conserve the volume of the simplex. When it reaches a valley floor, the algorithm *contracts* itself in the transverse direction and finally it contracts itself in all directions (*multiple contractions*) which results in pulling itself in around the lowest point (lowest objective value) [32]. Figure 1 shows appropriate sequences of such steps.

Downhill Simplex method approximates the size of the region that can be reached at temperature T, and it samples new points. If the temperature T is reduced *slowly* enough, the downhill Simplex method shrinks into the region containing the lowest minimum value. Also, the starting temperature  $T_0$  must be hot enough to allow a move to any neighborhood state. On the other hand, if the starting temperature is set too high, the search can move to any neighborhood which results in a random search. So, finding the correct starting temperature is a key issue in this algorithm. Also, the way in which we decrement the temperature is critical to the success of the algorithm. It is stated that we should allow enough iterations at each temperature [14]. For the choice of cooling down of annealing scheduling, we tried the following:

$$T = T_0 \times \left(1 - \frac{k}{K}\right)^\alpha \quad (4)$$

Where K is the budget of total moves, and reduce T after k moves to the above value and k is the cumulative number of moves thus far,  $\alpha$  is a constant. Larger values of  $\alpha$  spend more iterations at lower temperature.  $T_0$  is the initial temperature value. We set  $K = 1000$  and we tune  $\alpha$  and  $T_0$  based on validation datasets.

The algorithm also needs a probability distribution to select a move that minimizes the objective function. The following probability distribution indicates that even at low temperature, there is a chance of a system to be in a high energy state which indicates that there is a chance for the system to get out of a local minimum for finding a global one. As a result, the system goes uphill and downhill. Therefore, for selecting the next move  $\lambda'$ , if the move is better than the current position, i.e.,  $\mathcal{L}(\mathcal{Q}, \mathcal{F}_{\lambda'}) < \mathcal{L}(\mathcal{Q}, \mathcal{F}_\lambda)$  (true downhill step), the Simulated Annealing will select  $\lambda'$  as a new move, otherwise if the move is worse (uphill), it will be accepted based on the following probability:

$$Prob \approx \exp\left(\frac{-[\mathcal{L}(\mathcal{Q}, \mathcal{F}_{\lambda'}) - \mathcal{L}(\mathcal{Q}, \mathcal{F}_\lambda)]}{kT}\right) \quad (5)$$

In the next section, we describe the evaluation of this algorithm.

## 5. EXPERIMENT DESIGN

For our experiments, we use version 3.0 of LETOR package provided by Microsoft Asia [27]. The LETOR package includes several benchmark datasets and baseline results for research on learning to rank. The datasets provided in the LETOR package are: OHSUMED, Top Distillation 2003 (TD2003), Top Distillation 2004 (TD2004), Homepage Finding 2003 (HP2003), Homepage Finding 2004 (HP2004), Named Page Finding 2003 (NP2003) and Named Page Finding 2004 (NP2004). There are 106 queries in the OHSUMED dataset. The relevancy judgments provided in OHSUMED are scored 0, 1 or 2 and there are 45 features for each query-document pair. For HP2003, HP2004, NP2003, NP2004, TD2003 and TD2004, there are 150, 75, 150, 75, 50 and 75 queries, respectively. For these datasets, there are 64 features extracted for each query-document pair and a binary relevance judgment for each pair is provided. Table 1 shows the statistics of the datasets included in the LETOR 3.0 benchmark.

In LETOR 3.0 package, each dataset is partitioned into five for five-fold cross validation and each fold includes training, testing and validation sets. The results of the state-of-the-art algorithms are provided in the LETOR 3.0. Since these baselines are representatives from each category of learning-to-rank algorithms (i.e., pointwise, pairwise and listwise), we use them to compare with our proposed algorithm. The followings are the baselines provided in the LETOR 3.0 package:

Regression: This is a simple linear regression which is a basic pointwise approach.

RankSVM: RankSVM is a pairwise approach using Support Vector Machine [18].

FRank: FRank is a pairwise approach with a novel loss function called Fidelity loss function [15].

ListNet: ListNet is a listwise learning-to-rank algorithm [9] which uses cross-entropy loss for its loss function.

AdaRank-NDCG: This is a listwise boosting algorithm that incorporates NDCG in computing the samples [47].

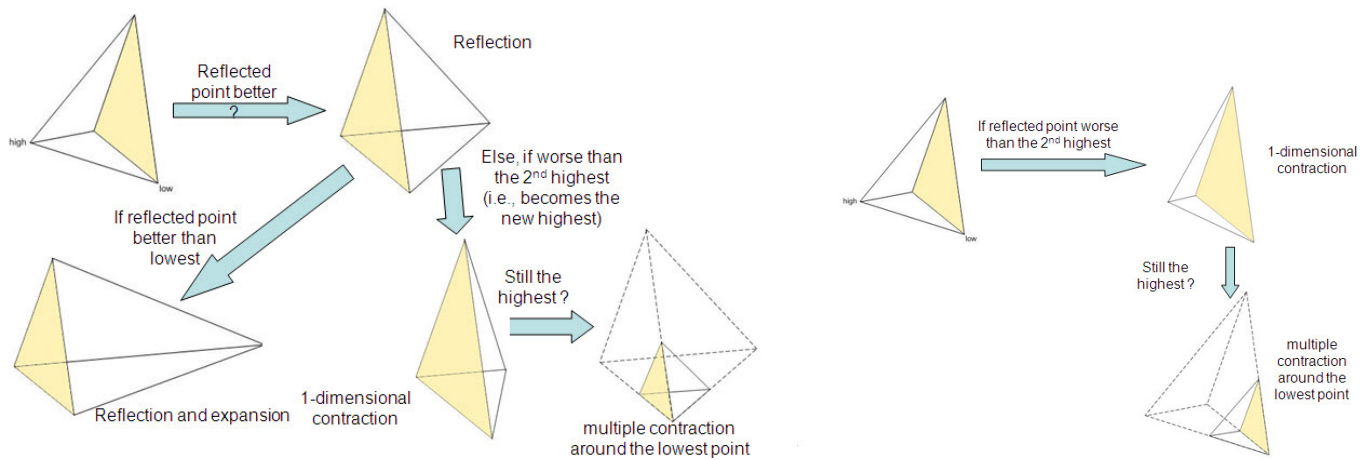


Figure 1: Sequence of Steps for Downhill Simplex Method

Table 1: Statistics of the datasets included in the LETOR 3.0 benchmark.

	Queries	Rel. Level	Features	Avg. Doc Per Query
<b>OHSUMED</b>	106	3	45	152
<b>HP 2003</b>	150	2	64	984
<b>HP 2004</b>	75	2	64	992
<b>NP 2003</b>	150	2	64	991
<b>NP 2004</b>	75	2	64	984
<b>TD 2003</b>	50	2	64	981
<b>TD 2004</b>	75	2	64	989

SVM-MAP: This is a listwise that uses support vector machine with MAP measure [50].

We also compare our algorithm with a new algorithm called NDCG-Boost [45]<sup>1</sup> which optimizes the expectation of NDCG over all possible permutations of documents.

We use the validation sets for each fold to find the best values for hyper-parameters (i.e.,  $\alpha$  and  $T_0$ ) and fix  $K = 1000$ . Once we find the best values of the hyper-parameters based on the validation sets, we fix them on training data and use training data to find the best parameters for features (weights for features), i.e.,  $\lambda_j$ 's. And finally, we use the optimal learned weight parameters on the training data to test the performance on the test data. We use the linear scoring function to rank documents for each query on test data, e.g., Equation 1. We then get the average across all folds for each data set.

We adopted the common IR evaluation measure, i.e., Normalized Discounted Cumulative Gain (NDCG) [21]. NDCG is designed to measure ranking accuracy when there are more than two levels of relevance judgments.

<sup>1</sup>The performance measured in our experiment for NDCG-Boost is different to the performance reported in the original paper [45] for some datasets. The reason is that the evaluations use different versions of MATLAB which have different implementations for decision stump, the base classifier utilized in the algorithm.

## 6. EXPERIMENT RESULTS

### 6.1 Comparison of NDCG-Annealing Algorithm with Baselines in LETOR 3.0

We compare our proposed NDCG-Annealing algorithm with those baselines provided in LETOR 3.0. The results for all the datasets are shown in Figures 2, 3, 4 in terms of NDCG measure. The results show that our algorithm outperforms all the baselines in most of the datasets. The interesting observation from these results is that the performance of the baseline algorithms varies from one dataset to another; however, the performance of NDCG-Annealing algorithm is the best in most datasets which indicates the consistency of the algorithm for ranking purposes. For example, FRank which performs well on OHSUMED yields a poor performance on TD2003, HP2003 and HP2004. Similarly, AdaRank-NDCG achieves a good performance on OHSUMED data set; however it does not perform well on other data sets such as TD2003, HP2003 and NP2003.

As also indicated in [11], since the OHSUMED dataset is the only dataset with more than 2 levels of relevance, the difference between various learning algorithms based on the NDCG measure can be more distinguishable. We also see from Figure 4 that our NDCG-Annealing algorithm outperforms all the other baseline algorithms on this dataset.

As a result, the NDCG-Annealing algorithm is more stable and pronounced compared to the baselines in LETOR 3.0 dataset.

### 6.2 Parameter Sensitivity Study on LETOR 3.0

As discussed before, the starting temperature of the Simulated Annealing algorithm must be hot enough. However, if the starting temperature is set too high, the search can move to any neighbor which leads to a random search. As a result, finding a correct starting temperature is important. Also, the way in which we decrement the temperature is critical to the success of the algorithm. Finally, how many iterations we make at each temperature is less important and can be set to a constant number [14].

As a result, we tuned the two important hyper-parameters,  $\alpha$  and  $T_0$  on validation datasets to achieve the optimal performance by the algorithm. Learning the optimal values for

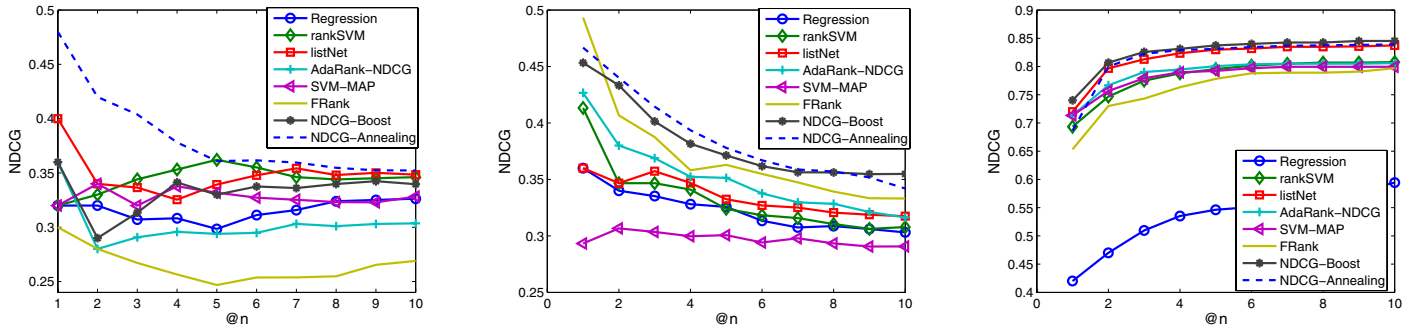


Figure 2: Comparison of NDCG-Annealing algorithm with other baselines in terms of NDCG measure, Left: TD2003, middle: TD2004 and right: HP2003.

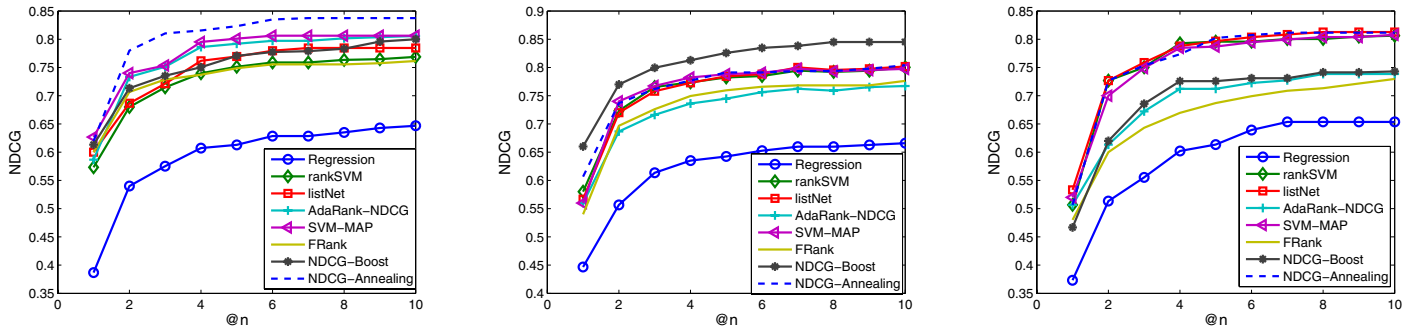


Figure 3: Comparison of NDCG-Annealing algorithm with other baselines in terms of NDCG measure, Left: HP2004, middle: NP2003 and right: NP2004.

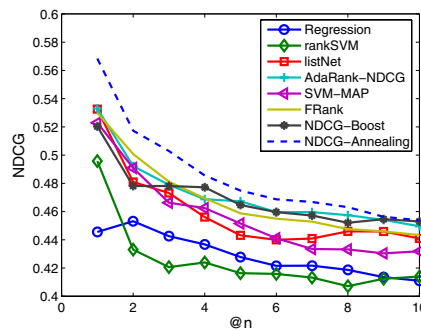


Figure 4: Comparison of NDCG-Annealing algorithm with other baselines in terms of NDCG measure, OHSUMED dataset.

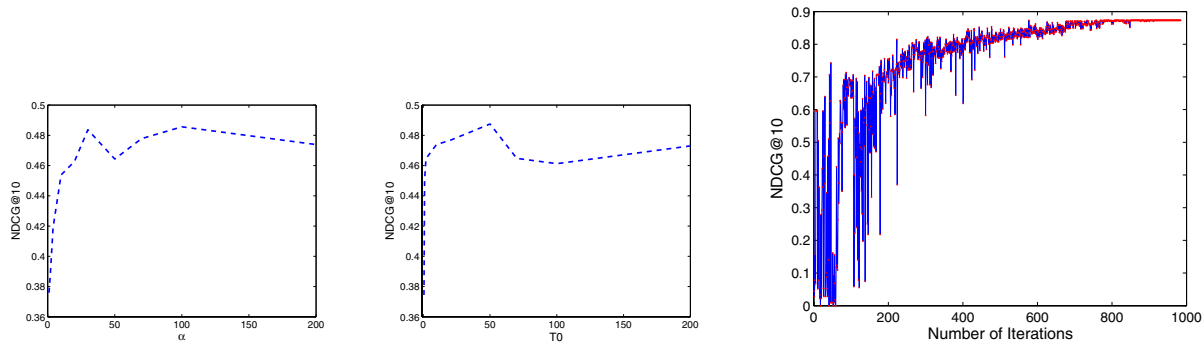


Figure 5: Sensitivity of  $\alpha$  (left) and  $T_0$  (middle) to NDCG measure on TD2003 dataset in LETOR 3.0. Sensitivity of the number of iterations to NDCG (right).

Table 2: CTR and RPM gains by running the NDCG-Annealing algorithm online.

	CTR gain over baseline	RPM gain over baseline
Yahoo! Answers	+86.19%	+38.58%
Sports Game	+10.88%	+9.39%
Sports News	+59.88%	+49.09%

the hyper-parameters  $\alpha$  and  $T_0$  is another way and we leave it for future work.

We show the sensitivity of NDCG@10 measure to each of these parameters. We only show the sensitivity analyses on TD2003 data set; similar trends can be seen on other datasets as well. Figure 5 (left) shows the sensitivity of NDCG@10 to  $\alpha$  parameter. For each fold, we set  $T_0$  to its optimal value and vary  $\alpha$  and get the average over all folds. Figure 5 (middle) shows the sensitivity of NDCG@10 to  $T_0$  parameter. For each fold, we set  $\alpha$  to its optimal value and vary  $T_0$  and get the average over all folds. The results of these sensitivity analyses indicate that the performance is not very sensitive to the hyper-parameters when they are large enough.

Figure 5 (right) shows the change in NDCG@10 as the NDCG-Annealing algorithm iterates. It is shown that in the first few iterations, the NDCG@10 varies a lot as the algorithm tries to find the best point, however for the larger number of iterations, the NDCG-Annealing algorithm stabilizes which shows that the algorithm needs to iterate enough to find the optimal parameters.

## 7. RESULTS ON APPLICATION TO ADVERTISING

In this section, we describe an application of the NDCG-Annealing algorithm in an online ad serving system. We present the results of evaluating this algorithm using Yahoo! editorial dataset (offline experiments) and online experiments by selecting and ranking the relevant ads on web pages and measuring the performance based on business evaluation metrics.

### 7.1 Motivation and Background

Contextual advertising has received much attention nowadays as one of the major revenue generation mechanisms for publishers. In contextual advertising, some ads are shown to a user when he/she visits a Web page. Such an event

is called *impression*. Most major Internet companies provide some contextual advertising products with a pay-per-click (PPC) business model where the advertiser is charged a fee every time a user clicks on an ad. Click-through-Rate (CTR), defined as the ratio between the number of clicks and the number of impressions, is thus an important metric to evaluate an ad ranking system. Furthermore, it is often believed that ads which are relevant to the page are more likely to receive clicks and have less impact on user experience (users may not come back to visit the Web page if we put too many irrelevant ads on the pages). Therefore, our ranking function seeks to optimize page-ad relevance as an indirect way to improve CTR as well as user experience.

In the next section, we describe the ad ranking function.

### 7.2 Ad Ranking Function

This problem of ranking ads ( $a$ ) in a page ( $p$ ) can be viewed as a special setup of information retrieval, where pages are queries and ads are documents. Our ranking function adopts the Cosine similarity metric in vector space model to calculate page-ad relevancy score:

$$F(a, p) = \frac{\sum_i tf.idf_{pi} * tf.idf_{ai}}{\|tf.idf_p\|_2 * \|tf.idf_a\|_2}, \quad (6)$$

where  $i$  corresponds to each token in the vocabulary,  $tf.idf_{pi}$  is the tf-idf score for token  $i$  in page  $p$  and  $tf.idf_{ai}$  is the tf-idf score for token  $i$  in ad  $a$ .

Under this framework, the tf-idf score for a particular token does not depend on its location in the pages or ads. However, a typical Web page consists of multiple sections and it is generally believed that the title, headings, emphasized and strong text on a page, carry more important information than plain text in the body. For example, on a Yahoo! Sports page, the body sometimes contains a brief summary of recent sports news. Ads that match this part are not necessarily relevant to this particular page. On the other hand, ads that match the page title are more likely to

be relevant. One way to differentiate among various sections is to assign different weights to them. For doing that, we use a DOM-based parser to analyze Web pages and decompose the content on the page based on the html tags (e.g.,  $\langle title \rangle$ ,  $\langle h1 \rangle$ ,  $\langle em \rangle$ ,  $\langle body \rangle$ ). Similarly an ad can be divided into multiple sections including title, description, landing page URL and bid terms<sup>2</sup>. Now the tf-idf score for token  $i$  in page  $p$  is calculated as:

$$tf.idf'_{pi} = \sum_j \lambda_j \times tf.idf_{p_j i} \quad (7)$$

where  $\lambda_j$  is the weight for section  $j$  and  $tf.idf_{p_j i}$  is the ordinary tf-idf score for token  $i$  in section  $j$  of page  $p$ . Similarly, we define the new tf-idf score for each token in each ad. Replacing the  $tf.idf$  values in equation (5) with the  $tf.idf'$  values, we obtain a new function to calculate the relevancy score between a (page, ad) pair.

It is shown in the experiments that applying weights to the sections improves retrieval relevance, but how to set appropriate weights of different sections is a challenge. Because the ranking function is predefined<sup>3</sup> and nonlinear, many existing learning-to-rank algorithms are simply not applicable. Therefore in the early version of our ranking system, these weights were tuned by experts through several rounds of trial-and-error<sup>4</sup> procedures. In this paper, we propose to apply the NDCG-Annealing algorithm to automatically learn the optimal weights that minimize the loss function.

To scale our computations, we adopt a parallelization approach based on the MapReduce [13] framework. MapReduce is a programming model for processing large-scale data which is highly scalable. The run-time system automatically takes care of the data, scheduling job across machines and managing inter-machine communication. In our case, every time the NDCG-Annealing algorithm needs to evaluate a parameter set, we start a new MapReduce job. In the Map step, the similarity score for each (page, ad) pair is calculated in parallel. Then in the Reduce step, we rank ads based on these scores and calculate the NDCG. By doing so, we are able to speed up the training process for large amounts of data.

In the next sections, we describe evaluations in both offline and online scenarios.

### 7.3 Offline evaluation

In this section, we report the results based on the Yahoo! editorial dataset. This dataset is created by editors assigning judgments to the (page, ad) pairs. The Yahoo! editorial dataset is composed of a set of Web pages, each associated with the contextual ads displayed on the page, and the corresponding editorial relevance judgment on each (page, ad) pair. The editorial relevance judgment uses 3-

<sup>2</sup>In sponsored search, advertisers specify which keywords they want to bid for. When a user searches for such keywords, ads with these bid terms are qualified to be shown to the user. However, in contextual advertising, there are no such search keywords. But the bid terms somewhat describe what the ad is about and should also be considered as an ad feature.

<sup>3</sup>Because of system restrictions, we are unable to change the ranking function format but only allowed to change the section weights.

<sup>4</sup>We use several sets of values based on heuristics for the parameters and run online tests to pick the best one.

grade scores with 0 (relevant), 1 (somewhat relevant), and 2 (irrelevant). The dataset has 7,259 unique Web pages. This editorial dataset was collected between years 2008 and 2009. It is a combination of 7 rounds of editorial judgments. In each round, the editorial team randomly crawls 1000-2000 Web pages. Some of them are not judgeable (with no or little content). After removing those pages, we obtain a total number of 7,259 pages. Each page has 3-15 associated ads which amounts to a total of 83,505 (page, ad) pairs.

A total of 14 sections are extracted from each (page, ad) pair. The sections extracted are shown in table 3. EM, H1, Strong and Meta keywords sections are the words contained in html tags such as  $\langle em \rangle$ ,  $\langle h1 \rangle$ ,  $\langle strong \rangle$  and meta tags of Web pages, respectively. Page URL Segments are extracted from page URLs, including domain name, host name and etc.

We randomly partitioned this dataset into five folds and created five training sets, each using four folds for training and leaving one fold out for evaluation. We report the NDCG averaged over five folds. We compare the NDCG-Annealing algorithm with the heuristic approach. Since we only show three ads on each result page, we calculate NDCG@3. NDCG@3 of the learned weights for sections with NDCG-Annealing algorithm is improved 11.02% compared with that of the baseline for section weights in the offline evaluation using Yahoo! editorial data.

**Table 3: Sections extracted for offline and online evaluations.**

Page Title
Page URL Segments
Page Description
Page Keywords
Body
Strong
EM
H1
Meta Keywords
Page Anchor Text
Ad Title
Ad Short Description
Ad Display URL
Ad Bid Terms

### 7.4 Online Evaluation

We applied the NDCG-Annealing algorithm on a real advertising serving system to select and rank ads to show on Web pages for real users. To measure the performance, we set up two buckets in our online ad serving system (each Web page initiates an ad call to request a set of ads from the ad server). One bucket is a control bucket, which uses the expert-tuned weights in the ranking system of the ad server. The second bucket uses the weights learned (in offline evaluation) by NDCG-Annealing algorithm. All other settings are the same. The two buckets receive random ad calls sent over equally so the other factors which may affect the experiment results are statistically the same. The goal of the online evaluation is to measure the impact of the NDCG-Annealing learned weights to the business metrics to check if the NDCG gains observed offline can be transferred to the gains online (this can be regarded as transferring the knowledge, i.e., learned weights). Although NDCG-Annealing algorithm is to optimize relevance, we have observed that relevance is highly correlated with user response, CTR, and thus the revenue generated from clicks. It is difficult to directly

measure the relevance improvement in online testing, so instead we measure the business metrics to be a proxy of the relevance, and those business metrics are also more tangible and financially important than NDCG. We use the following two metrics, Click-through-Rate (CTR) and Revenue per 1000 Impressions (RPM) for the online traffic evaluation:

$$CTR = \frac{\#clicks}{\#impressions}$$

$$RPM = \frac{Revenue * 1000}{\#impressions}$$

Revenue is defined as follows:

Consider we have  $N$  impressions and  $C$  clicks and each click  $c_i$  in  $C$  has a winning bid of  $b_i$ . Then the Revenue for these  $N$  impressions is:  $\sum_i b_i$ .

We do not directly train the model for online traffic, but use the learned weights from offline training. We use the same sections as in table 3. In both two buckets, we return three ads per page on Yahoo! Answers page, Sports game and Sports news. An example of ads shown to the users on Yahoo! Answers page is shown in Figure 6 (yellow part). The ad server with the two buckets were put online to receive real ad call traffic for two weeks. The page impressions, clicks received on the ads and the revenue generated are logged, based on which we compare the online performance of the two buckets. The results for CTR and RPM are shown in table 2. The percentage is the relative gain. As we can see, the learned weights significantly improve the evaluation metrics over the baseline.

### 7.5 Findings

- The followings are the importance of different sections. Note that the numbers in parentheses show the learned weights.
- Ad title has high positive impact (3.03).
  - Ad bid terms (2.61) have lower impact than titles which is about 86% of ad title. This maybe because that the ad structure has high perceived impact than the invisible inserted bid term itself to the users.
  - Ad description impact is relatively low (0.76).
  - Page title has less positive impact (1.92) than that of page description (2.91).
  - HTML tags do not have significantly higher impact than plain text in page body. The tag strong (0.01) even has a lower weight than body (0.93). H1 (1.47), EM (1.09), and page keywords (1.21) have almost similar positive impacts.
  - Page anchor has quite high positive impact (3.88).
  - Page part of URL (1.83) and query part of URL (1.16) both have quite high positive impacts.

## 8. CONCLUSIONS AND FUTURE WORK

Listwise approach is a new approach to learning to rank. In this paper, we proposed a stochastic learning-to-rank algorithm that uses Simulated Annealing algorithm along with Simplex method for parameter search to minimize the loss function that is directly defined on any IR evaluation measures including NDCG and MAP. The loss function in our method is application specific and restriction-free. As an instance of the method, we optimize NDCG in this paper, but other metrics in ranking can also be applied directly. Our experiments on LETOR 3.0 benchmark dataset show that the

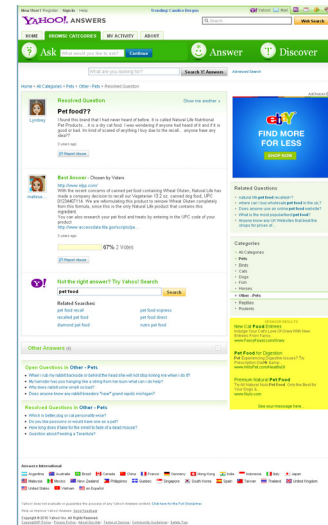


Figure 6: Ads (in yellow) shown on Yahoo! Answers page.

NDCG-Annealing algorithm outperforms the state-of-the-art algorithms both in terms of performance and stability. We also showed an application of the proposed method in an online ad serving system for matching and ranking ads for a given Web page which indicates the applicability of the proposed algorithm in real online applications as we improved the CTR and RPM significantly in the online bucket tests.

There is still remaining work in this direction; it would be interesting to see the performance of the algorithm for ranking real web documents. In addition, adding more advanced functionalities to handle complex constraints of the parameters to learn, and standardize the library to be easily applied to other applications is another line for future work. Also, it would be interesting to propose some other ranking functions other than vector space model for ad ranking. Although, different methods for matching and ranking ads are proposed in the literature, there is no good understanding of which method performs better for a real ad system. We plan to develop a large-scale benchmark ad dataset, so that comparing different methods would be easier. Finally, the optimization (training) of the NDCG-Annealing algorithm takes time and one future work is to expedite the algorithm by better extracting and exploiting the parallelism in the algorithm to leverage MapReduce programming model more effectively.

## 9. ACKNOWLEDGMENTS

We thank the anonymous reviewers for their useful comments. The authors would like to acknowledge Fernando Diaz, Olivier Chapelle, Wei Chu and Jangwon Seo for useful discussions.

## 10. REFERENCES

- [1] H. M. D. Almeida, M. A. Concalves, M. Cristo, and P. Calado. A combined component approach for finding collection-adapted ranking functions based on genetic programming. *ACM SIGIR*, pages 399–406, 2007.
- [2] R. Baeza-Yates and B. Ribeiro-Neto. Modern information retrieval. *Addison Wesley*.
- [3] A. Bagherjeiran, A. Hatch, and A. Ratnaparkhi. Ranking for the conversion funnel. *ACM SIGIR*, pages 146–153, 2010.
- [4] S. Bandyopadhyay, S. K. Pal, and C. A. Murthy. Simulated annealing based pattern classification. *Information Sciences*, pages (109):165–184, 1998.
- [5] A. Z. Broder, M. Fontoura, V. Josifovski, and L. Riedel. A semantic approach to contextual advertising. *SIGIR*, pages 559–566, 2007.
- [6] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. *ICML*, pages 89–96, 2005.
- [7] C. J. Burges, R. Ragno, and Q. V. Le. Learning to rank with nonsmooth cost functions. *NIPS*, pages 193–200, 2006.
- [8] Y. Cao, J. Xu, T.-Y. Liu, H. Li, Y. Huang, and H.-W. Hon. Adapting ranking svm to document retrieval. *ACM SIGIR*, pages 186–193, 2006.
- [9] Z. Cao, T. Qin, T.-Y. Liu, M.-F. Tsai, , and H. Li. Learning to rank: From pairwise approach to listwise approach. *ICML*, pages 129–136, 2007.
- [10] D. Chakrabarti, D. Agrawal, and V. Josifovski. Contextual advertising by combining relevance with click feedback. *WWW*, pages 416–426, 2008.
- [11] O. Chapelle and M. Wu. Gradient descent optimization of smoothed information retrieval metrics. *IR Journal*, pages 13(3):201–215, 2010.
- [12] D. Cossock and T. Zhang. Subset ranking using regression. *CoLT*, pages 605–619, 2006.
- [13] J. Dean and S. Ghemawat. Mapreduce: simplified data processing on large clusters. *Symposium on Operating System Design and Implementation*, pages 137–150, 2004.
- [14] K. A. Dowsland. Simulated annealing. *Modern Heuristic Techniques for Combinatorial Problems*, 1993.
- [15] M. feng Tsai, T.-Y. Liu, T. Qin, H.-H. Chen, and W.-Y. Ma. Frank: A ranking method with fidelity loss. *ACM SIGIR*, pages 383–390, 2007.
- [16] A. Foundation. Apache hadoop project. <http://hadoop.apache.org/>.
- [17] Y. Freund, R. Iyer, R. E. Schapire, and Y. Singer. An efficient boosting algorithm for combining preferences. *ICML*, pages 170–178, 1998.
- [18] R. Herbrich, T. Graepel, and K. Obermayer. Support vector learning for ordinal regression. *Artificial Neural Networks*, pages 97–102, 1999.
- [19] R. Herbrich, T. Graepel, and K. Obermayer. Large margin rank boundaries for ordinal regression. *Advances in Large Margin Classifiers*, pages 115–132, 2000.
- [20] C. Huang and R. Nevatia. High performance object detection by collaborative learning of joint ranking of granules features. *CVPR*, pages 41–48, 2010.
- [21] K. Jarvelin and J. Kekalainen. Ir evaluation methods for retrieving highly relevant documents. *ACM SIGIR*, pages 41–48, 2000.
- [22] T. Joachims. Optimizing search engines using clickthrough data. *ACM KDD*, pages 133–142, 2002.
- [23] S. Kirkpatrick, C. D. Gelatt, and M. P. Vecchi. Optimization by simulated annealing. *Science*, pages 671–680, 1983.
- [24] A. Lacerda, M. Cristo, M. Goncalves, W. Fan, N. Ziviani, and B. Ribeiro-Neto. Learning to advertise. *ACM SIGIR*, pages 549–556, 2006.
- [25] P. Li, C. J. Burges, and Q. Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. *NIPS*, 2007.
- [26] T. Y. Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, pages 3(3):225–331, 2009.
- [27] T.-Y. Liu, J. Xu, T. Qin, W. Xiong, and H. Li. Letor: Benchmark dataset for research on learning to rank for information retrieval. *SIGIR-LTR*, 2007.
- [28] A. V. Lukashin and R. Fuchs. Analysis of temporal gene expression profiles: Clustering by simulated annealing and determining the optimal number of clusters. *Bioinformatics*, pages 405–414, 2000.
- [29] N. Metropolis, A. W. Rosenbluth, M. N. Rosenbluth, A. H. Teller, and E. Teller. Equation of state calculation by fast computing machines. *J. Of Chem. Phys.*, pages 1087–1091, 1953.
- [30] V. Murdock, M. Ciaramita, and V. Plachouras. A noisy-channel approach to contextual advertising. *ADKDD*, pages 21–27, 2007.
- [31] R. Nallapati. Discriminative models for information retrieval. *ACM SIGIR*, pages 64–71, 2004.
- [32] J. A. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal*, pages (7):308–313, 1965.
- [33] Z. Nie, Y. Zhang, J.-R. Wen, and W.-Y. Ma. Object-level ranking: Bringing order to web objects. *WWW*, pages 567–574, 2005.
- [34] J. Peng, C. Macdonald, and I. Ounis. Learning to select a ranking function. *ECIR*, pages 111–126, 2010.
- [35] J. Ponte and W. Croft. A language modeling approach to information retrieval. *ACM SIGIR*, pages 275–281, 1998.
- [36] T. Qin, T.-Y. Liu, M. feng Tsai, X.-D. Zhang, and H. Li. Learning to search web pages with query-level loss functions. *Technical Report*, 2006.
- [37] A. Ratnaparkhi. A hidden class page-ad probability model for contextual advertising. *WWW Workshop*, 2008.
- [38] M. Regelson and D. Fian. Predicting click-through rate using keyword clusters. *ACM EC*, 2006.
- [39] B. Ribeiro-Neto, M. Cristo, P. Golgher, and E. S. D. Moura. Impedance coupling in content-targeted advertising. *ACM SIGIR*, pages 496–503, 2005.
- [40] M. Richardson, E. Dominowska, and R. Rango. Predicting clicks: Estimating the click-through rate for new ads. *WWW*, pages 521–530, 2007.
- [41] S. Robertson and K. S. Jones. Relevance weighting of search terms. *Journal of the American Society for Information Science*, pages 27:129–146, 1976.
- [42] S. Robertson and S. Walker. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. *ACM SIGIR*, pages 232–241, 1994.
- [43] G. Salton and C. Buckley. Term-weighting approaches in automatic text retrieval. *Information Processing and Management*, pages 24(5):513–523, 1988.
- [44] M. Taylor, J. Guiver, S. Robertson, and T. Minka. Softrank: optimizing non-smooth rank metrics. *ACM WSDM*, pages 77–86, 2008.
- [45] H. Valizadegan, R. Jin, R. Zhang, and J. Mao. Learning to rank by optimizing ndcg measure. *NIPS*, 2009.
- [46] M. N. Volkovs and R. S. Zemel. Boltzrank: learning to maximize expected ranking gain. *ICML*, pages 1089–1096, 2009.
- [47] J. Xu and H. Li. Adarank: a boosting algorithm for information retrieval. *ACM SIGIR*, pages 391–398, 2007.
- [48] W. Yang, L. Rueda, and A. Ngom. A simulated annealing approach to find the optimal parameters for fuzzy clustering microarray data. *IEEE International Conference on The Chilean Computer Science Society*, 2005.
- [49] J.-Y. Yeh, J.-Y. Lin, H.-R. Ke, and W.-P. Yang. Learning to rank for information retrieval using genetic programming. *SIGIR-LTR Workshop*, 2007.
- [50] Y. Yue, T. Finley, F. Radlinski, and T. Joachims. A support vector method for optimizing average precision. *ACM SIGIR*, pages 271–278, 2007.