Learning to Rank using Markov Random Fields

Antonino Freno, Tiziano Papini, Michelangelo Diligenti

Learning to Rank using Markov Random Fields

Antonino Freno and Tiziano Papini and Michelangelo Diligenti
Dipartimento di Ingegneria dell'Informazione,
University of Siena, Siena, Italy
Email: {freno, papinit, diligmic}@dii.unisi.it

***Abstract*—Learning to rank from examples is an important task in modern Information Retrieval systems like Web search engines, where the large number of available features makes hard to manually devise high-performing ranking functions. This paper presents a novel approach to learning-to-rank, which can natively integrate any target metric with no modifications. The target metric is optimized via maximum-likelihood estimation of a probability distribution over the ranks, which are assumed to follow a Boltzmann distribution. Unlike other approaches in the literature like BoltzRank, this approach does not rely on maximizing the expected value of the target score as a proxy of the optimization of target metric. This has both theoretical and performance advantages as the expected value can not be computed both accurately and efficiently. Furthermore, our model employs the pseudo-likelihood as an accurate surrogate of the likelihood, so as to avoid to explicitly compute the normalization factor of the Boltzmann distribution, which is intractable in this context. The experimental results show that the approach provides state-of-the-art results on various benchmarks and on a dataset built from the logs of a commercial search engine.**

I. Introduction

Modern Information Retrieval systems like Web search engines have available hundreds of features to represent each document while answering users' queries. These features range from query-independent signals like PageRank to others measuring the match between the query and a document. This high number of ranking signals, which can be strongly correlated, makes hard to manually design ranking functions achieving results that are close to optimality.

In the last few years, learning to rank from examples has emerged as a more flexible approach to design ranking functions. Learning-to-rank approaches have been proved to significantly outperform hand-tuned solutions [1].

Ranking algorithms can be assigned to three classes: pointwise, pairwise and listwise. A pointwise approach [2] takes document/score pairs as training examples to learn a document scoring function. Documents returned for a query are then sorted by score. Pairwise methods like [3] take a set of pairs of documents as input to the training. The training process consists in learning to order the pairs. This is generally preferable to pointwise methods, because it does not impose specific scores to the learning algorithm, leaving it the freedom to select the score range in which to work. Finally, listwise methods [4], [5] get a set of lists of ranked documents as training examples, and the optimization is performed using a loss function over the entire list of documents.

A recent trend in learning to rank approaches is to attempt a direct optimization of the target metrics [6][7], which typically are either *Mean Average Precision* (MAP) for ranks having two relevance scores or *Normalized Discounted Cumulative Gain* (NDCG) when there is an arbitrary number of relevance scores. This class of approaches falls in the listwise category as the target metrics are functions of ranked lists and not of individual pairs. Those approaches are generally considered to outperform pairwise methods as they can direct the learning toward what's most important with regard to the optimization of the target metric. However, direct optimization of the target metrics is typically difficult, because all the commonly employed metrics, such as NDCG and MAP, are not expressed in terms of the scoring functions but in terms of the document ranks (which then depend on the functions). This makes the resulting loss function either constant or not differentiable in any point with respect of the training parameters. Most learning approaches solve this issue by employing a continuous approximation of the target metric [8]. In this context, it is fundamental to approximate the probability of a rank with high precision.

This paper follows an approach similar to BoltzRank [7], which models the distributions of ranks using a Boltzmann distribution and then the target metric was optimized by maximizing its expected value. However, unlike in BoltzRank, the proposed solution directly integrates the target metric into the energy function of the Boltzmann distribution. This yields a more direct optimization problem which has the advantage on not relying on the expected value, which depends on all permutations of a rank and it can not be computed without sub-sampling. Furthermore, BoltzRank was not making any independence assumptions over the rank scores. Therefore, an exact computation of the normalization factor needed to compute the probability distribution over the ranks is intractable whenever there are more than a handful of documents to rank. We explore different independence assumptions and employ the pseudo-likelihood [9] as an accurate surrogate of the likelihood both for learning and Bayesian decision [10][11]. The resulting model has both theoretical and performance advantages over the previous approaches proposed in the literature. The experimental results show the effectiveness of the approach on various benchmarks and on a dataset built from the logs of a commercial search engine.

II. Learning to rank

WITH BOLTZMANN DISTRIBUTIONS

A learning-to-rank problem consists of a set of n queries $\mathcal{Q} = \{q_1, \ldots, q_n\}$, such that for each q_i there is a corresponding set of documents $\mathcal{D}_i = \{d_{i_1}, \ldots, d_{i_{m_i}}\}$, where m_i is the number of documents retrieved for query q_i. A set of permutations $\mathcal{R}_i$ can be constructed from each set $\mathcal{D}_i$. A generic permutation $\boldsymbol{r}_i \in \mathcal{R}_i$ is a set $\boldsymbol{r}_i = \{r_{i_1}, \ldots, r_{i_{m_i}}$, where r_{i_j} indicates the rank of d_{i_j}. We also have a set of associated labels $\mathcal{L}_i = \{l_{i_1}, \ldots, l_{i_{m_i}}\}$, indicating the relevance level of each document d_{i_j} with respect to the corresponding query. Therefore, we may think of the training data as a collection $\mathcal{T} = \{\boldsymbol{e}_1, \ldots, \boldsymbol{e}_n\}$, where $\boldsymbol{e}_i = \{(d_{i_1}, l_{i_1}), \ldots, (d_{i_{m_i}}, l_{i_{m_i}})\}$. What distinguishes relevance judgments from mere label assignments, making the considered task a *ranking* task in the proper sense (rather than a mere classification task), is the fact that the relevance judgments impose a (partial) ordering on each document set $\mathcal{D}_i$. That is, once given the relevance assignments $\mathcal{L}_1, \ldots, \mathcal{L}_n$, it is defined a set of permutations of $\mathcal{D}_i$ in such a way that, if $r_{i_j} < r_{i_k}$, then $l_{i_j} \geq l_{i_k}$ (which means that document d_{i_j} is at least as relevant as document d_{i_k}). In order to keep the notation simple, in the following of the paper we drop the index i whenever a single query is taken into account at a given time.

A. BoltzRank and Rank Distributions over Ordered Lists

BoltzRank [7] is a state-of-the-art approach to learning-to-rank. We start describing this approach and we highlight its main advantages and limitations.

Given a permutation $\boldsymbol{r}$ over a set of documents $\mathcal{D}$ with size m and a function f, with parameters $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_p)$, which assigns a score $f(d, \boldsymbol{\theta})$ to each document d, BoltzRank defines a Boltzmann distribution over document permutations (conditioned on $\boldsymbol{\theta}$) can then be defined as follows:

$$P(\boldsymbol{r} \mid \boldsymbol{\theta}) = \frac{1}{Z(\boldsymbol{\theta})} \exp\big(-E(\boldsymbol{r}|\boldsymbol{\theta})\big) \tag{1}$$

where $E(\boldsymbol{r}|\boldsymbol{\theta})$ denotes the energy of a rank and, if $\mathcal{R}$ is the set of all possible permutations of $\mathcal{D}$, Z denotes the partition function, given by

$$Z(\boldsymbol{\theta}) = \sum_{\boldsymbol{r}' \in \mathcal{R}} \exp\big(-E(\boldsymbol{r}' \mid \boldsymbol{\theta})\big) \tag{2}$$

BoltzRank employs the following form for the energy:

$$E(\boldsymbol{r} \mid \boldsymbol{\theta}) = \frac{2\alpha}{m\,(m-1)} \sum_{r_u > r_v} (r_u - r_v)\,\big(f(d_u, \boldsymbol{\theta}) - f(d_v, \boldsymbol{\theta})\big) \tag{3}$$

where$n$ is the number of documents to rank and $\alpha > 0$ is a parameter. Clearly, the lower the energy $E(\boldsymbol{r} \mid \boldsymbol{\theta})$, the higher the probability determined for $\boldsymbol{r}$ via the function f. For sake of compactness, in the following we omit the $\boldsymbol{\theta}$ from the argument of f. Therefore, we use $f(d)$ instead of $f(d, \boldsymbol{\theta})$.

BoltzRank employs the probability of a total rank defined in equation 1 to compute the expected value $\hat{\Phi}$ of a target metric Φ such as NDCG or MAP. In particular, we indicate as $\Phi(\mathcal{T}, \boldsymbol{\theta})$ the target metric score over the dataset, then the expected value is computed as the average over the expected values obtained for the single queries in the training dataset as

$$\hat{\Phi}(\mathcal{T}, \boldsymbol{\theta}) = \frac{1}{|\mathcal{Q}|} \sum_{i=1}^{|\mathcal{Q}|} \sum_{\boldsymbol{r} \in \mathcal{R}_i} \Phi(\boldsymbol{r}) P(\boldsymbol{r} \mid \boldsymbol{\theta}) \tag{4}$$

where $\Phi(\boldsymbol{r})$ is the score of a rank $\boldsymbol{r}$. Equation 4 can be directly maximized via gradient descent by assigning higher probability values to permutations having a high target score.

There are three main limitations that prevents this method from being applied in its exact formulation and that require a set of approximations. First, in order to compute the probability of a rank as defined in equation 1, it is needed to compute the normalization factor $Z(\boldsymbol{\theta})$ as defined in equation 2. Unfortunately, this can not be computed exactly as the number of possible permutations $\mathcal{R}$ grows factorially with the number of documents. Secondly, the expected value can not be computed exactly as again its computation requires to sum over all possible ranks. BoltzRank faces these issues by approximating these quantities with their Monte Carlo estimates, whose accuracy can not be easily assessed. Finally, please note that BoltzRank is only taking into account total orders. Partial orders are still correctly modeled, since pairs of document with the same score factor out in the computation of the expected value. However, since the expected value is approximated by sub-sampling, it would be preferable that the energy computation could take partial orders directly into account.

B. Boltzmann distributions and rank metrics

In order to overtake the limitations of BoltzRank, the target metric can be directly integrated into the potential function using a technique inspired from LambdaRank [12],

$$E(\boldsymbol{r}_i \mid \boldsymbol{\theta}) = \frac{2\alpha}{m\,(m-1)} \sum_{r_u > r_v} \Delta\Phi(\boldsymbol{r}, u, v)\,\big(f(d_u) - f(d_v)\big)$$

where $\Delta\Phi(\boldsymbol{r}, u, v)$ is the delta of the target metric score when swapping result d_u and d_v in the rank $\boldsymbol{r}$.

A common choice for the metric Φ is the NDCG, defined as:

$$NDCG = \sum_{j=1}^{m} \frac{2^{l_j} - 1}{\log_2(r_j) + 1}\ .$$

For the NDCG, $\Delta\Phi(\boldsymbol{r}, u, v)$ becomes,

$$\Delta NDCG(\boldsymbol{r}, v, u) = \frac{(2^{l_v} - 2^{l_u})}{\log_2(r_v + 1)} - \frac{(2^{l_v} - 2^{l_u})}{\log_2(r_u + 1)} \tag{5}$$

If the evaluation metric is hard-coded into the energy function as described above, a lower energy and, therefore, higher probability is assigned to ranks for which there is a large positive correlation between the function values and the corresponding target metric scores. A key advantage of this energy function is that it allows to natively take into account partial orders which are the norm in most learning-to-rank applications. Indeed, swapping two equivalent results with the same relevance yields that $\Delta\Phi(\boldsymbol{r}, u, v) = 0$, which gives no

contribution to the energy. Please note that this is not the case for the standard BoltzRank energy function, which accounts for rank deltas. Another main advantage is that the target metric can be directly optimized by likelihood maximization over the parameters $\boldsymbol{\theta}$. This does not require to rely on the computation of the expected value of the target metric, which can only be approximated via Monte Carlo estimation.

III. Graphical Models and the Pseudo-Likelihood Function

In Markov random fields, the probability $P(\boldsymbol{x})$ that a vector $\boldsymbol{X}$ of random variables has value $\boldsymbol{x} = (x_1, \ldots, x_d)$ is given by

$$P(x_1, \ldots, x_d) = \frac{1}{Z} \prod_{\mathcal{C} \in \gamma} \varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}) \tag{6}$$

where γ is the set containing all maximal cliques of the graph, and $\boldsymbol{x}_{\mathcal{C}}$ is the state of clique $\mathcal{C}$, as determined by $\boldsymbol{x}$ [13]. Now, the Boltzmann model described in section II amounts to a (fully connected) MRF model where, for the single clique $\mathcal{C}$ of the graph, the potential function $\varphi_{\mathcal{C}}$ is defined as

$$\varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}) = \left(\frac{2\alpha}{m\,(m-1)} \sum_{(r_u, r_v) \in \mathcal{C}^{2^*}} (r_u - r_v) \left(f(d_u) - f(d_v) \right) \right) \tag{7}$$

where, if $\mathcal{C}^2 = \mathcal{C} \times \mathcal{C}$, $\mathcal{C}^{2^*}$ denotes the set $\{(r_u, r_v) : (X_u, X_v) \in \mathcal{C}^2 \ \wedge \ r_u > r_v\}$. The equivalence between the Boltzmann model and a fully connected Markov random field is nothing but a consequence of the Hammersley-Clifford theorem [14]. Clearly, since the graph is fully connected, we have that $\gamma = \{\{X_i \colon 1 \leq i \leq d\}\}$, i.e. there is exactly one maximal clique, containing all of the nodes in the graph.

The main difficulty involved in estimating the quantity referred to in equation 6 lies in computing the value of the partition function, which is typically intractable. One widely used approach to probabilistic modeling in Markov random fields resorts instead to the pseudo-likelihood objective [9], which is a very efficient yet accurate surrogate for likelihood in the strict sense, both for learning and Bayesian decision [15], [10], [11]. Given the random variables $X_1, \ldots, X_d$, the pseudo-likelihood P^* of any state $x_1, \ldots, x_d$ of those variables is measured as follows:

$$P^*(x_1, \ldots, x_d) = \prod_{i=1}^{d} P(x_i \mid x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_d) \tag{8}$$

One convenient property of the pseudo-likelihood measure for application to graphical models (and MRFs in particular) is that, as defined by equation 8, it reduces to the following function:

$$P^*(x_1, \ldots, x_d) = \prod_{i=1}^{n} P\big(x_i \mid mb(X_i)\big) \tag{9}$$

where $mb(X_i)$ denotes the state of the Markov blanket of X_i [13].

Once the structure of the MRF has been specified, the conditional probability of X_i given the state $mb(X_i)$ of its Markov blanket $\mathcal{MB}(X_i)$ for any value x_{i_j} in the domain of the variable X_i takes the following form [16], [17]:

$$P\big(x_{i_j} \mid mb(X_i)\big) = \frac{\prod\limits_{\mathcal{C} \in \gamma_{X_i}} \varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}, x_{i_j})}{\sum\limits_{x_{i_k}} \prod\limits_{\mathcal{C} \in \gamma_{X_i}} \varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}, x_{i_k})} \tag{10}$$

where $\gamma_{X_i} = \{\mathcal{C} \in \gamma \colon X_i \in \mathcal{C}\}$ and $\varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}}, x_{i_k})$ denotes the value returned by $\varphi_{\mathcal{C}}(\boldsymbol{x}_{\mathcal{C}})$ when X_i in $\boldsymbol{x}_{\mathcal{C}}$ is clamped to the value x_{i_k}.

One theoretical virtue of the pseudo-likelihood function is that the argument of its maximum (i.e. the set of parameters that maximize it) with respect to a specified collection of data converges in probability to the true maximum likelihood solution as the number of data points grows to infinity [13].

IV. Pseudo-likelihood for Markov Random Fields on ranks

In order to exploit the pseudo-likelihood function as a replacement for the Boltzmann distribution, it is needed to define the conditional distributions involved in the right-hand side of equation 9. Thus, it is needed to specify what the nodes of the MRF are and which is the Markov blanket of each node.

When modeling a probability distribution of ranks, a natural choice is to associate a variable R_j to each document d_j that needs be ranked for the query under consideration. The value of the variable R_j is the rank of d_j. Assuming that we are given the optimal rank $\boldsymbol{r}^{opt} = \{r_1^{opt}, \ldots, r_n^{opt}\}$ for a query, the value of R_j (the rank of a document with rank j) should depend on the value of the ranks of the documents immediately preceding or following it in the optimal rank. We indicate with W the number of preceding/following documents that are considered when estimating the value of R_j. Therefore, W is a positive integer determining the size of the Markov blanket for each node.

Ideally, $W = 1$ should suffice as the rank r of a document can be establish knowing that it should follow a document with rank $r - 1$ and it should precede a document with rank $r + 1$. However, as the experimental results will confirm, it is always convenient to increase the size of the Markov blanket as this provides more information to the learning algorithm and it helps to generalize in a noisy setting.

The Markov blanket of the node associated with d_j is the set $\mathcal{MB}(R_j)$, such that

$$\mathcal{MB}(R_j) = \{R_k \colon j \neq k \ \wedge \ r_k^{opt} - W \leq r_j^{opt} \leq r_k^{opt} + W\} \ . \tag{11}$$

Since $\mathcal{MB}(R_j)$ is the set of nodes that are connected to R_j, the choice of W determines the structure (i.e. the graph) of the Markov random field. Therefore, we refer to W parameter as the Markov blanket *width*. Two examples of (undirected) graphs resulting from different values of W are depicted in figure IV, for a permutation over a rank of 6 documents.

Given the parameters of the ranking function f to be learned, this procedure results in a query-dependent model. The model computes a probability distribution $P(\boldsymbol{r} \mid \boldsymbol{\theta})$ over an assignment $\boldsymbol{r}$ to the variables $\{R_1, \ldots, R_n\}$, where $n = |\boldsymbol{r}|$

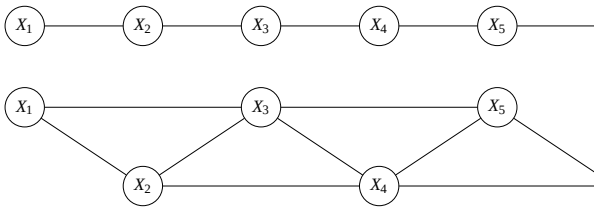

Fig. 1. Two undirected graphs resulting from setting $W = 1$ and W as the width of the respective Markov blankets when modeling a ran documents for a given query. The label X_i in a node of the graph in the R_j variable, which is the rank for document d_j assuming value i optimal rank provided for the query.

is the number of documents to be ranked for the specific q The pseudo-likelihood $P^*(\boldsymbol{r} \mid \boldsymbol{\theta})$ of a ranking $\boldsymbol{r}$ for this r is expressed by equation 10.

The conditional probabilities needed to compute the p likelihood take the following form:

$$P\big(r_i \mid mb(R_i), \boldsymbol{\theta}\big) = \tag{12}$$
$$= \frac{\exp\left(-\frac{2\alpha\left(\sum_{r_j=r_i-W}^{r_i-1} g_{ij}(\boldsymbol{r},\boldsymbol{\theta}) + \sum_{r_j=r_i+1}^{r_i+W} g_{ij}(\boldsymbol{r},\boldsymbol{\theta})\right)}{m\,(m-1)}\right)}{\sum_{r_{i_k}} \exp\left(-\frac{2\alpha\left(\sum_{r_j=r_{i_k}-W}^{r_{i_k}-1} g_{i_k j}(\boldsymbol{r},\boldsymbol{\theta}) + \sum_{r_j=r_{i_k}+1}^{r_{i_k}+W} g_{j i_k}(\boldsymbol{r},\boldsymbol{\theta})\right)}{m\,(m-1)}\right)} .$$

where, using the definition of $\Delta NDCG(\boldsymbol{r}, i, j)$, given in equation 5 $g_{ij}(\boldsymbol{r}, \boldsymbol{\theta}) = \Delta NDCG(\boldsymbol{r}, i, j)\big(f(d_i, \boldsymbol{\theta}) - f(d_j, \boldsymbol{\theta})\big)$. Obviously any other metric Φ can be naively used in place of the NDCG into $g_{ij}(\boldsymbol{r}, \boldsymbol{\theta})$.

It would also be possible to plug the standard energy function of BoltzRank into the pseudo-likelihood computation by instead defining $g_{ij}(\boldsymbol{r}, \boldsymbol{\theta}) = (r_i - r_j)\big(f(d_i, \boldsymbol{\theta}) - f(d_j, \boldsymbol{\theta})\big)$, and then substituting the values into equation 12. This selection would fall back to the BoltzRank potential function (beside the different normalization factor) when removing all independence assumptions by setting $W = n$ (the rank of a document directly depends on the rank of all other documents, yielding a single clique in the graph).

Equation 12 can be plugged into the following equation to compute the pseudo likelihood of a rank:

$$P^*(\boldsymbol{r} \mid \boldsymbol{\theta}) = \prod_{j=1}^{|\boldsymbol{r}|} P\big(r_j \mid mb(R_j), \boldsymbol{\theta}\big) .$$

In order to estimate the model parameters, we maximize the pseudo-likelihood by performing gradient-ascent using an online strategy. In particular, we compute the gradient of the log-pseudo-likelihood function with respect to the parameter vector $\boldsymbol{\theta}$ of f. The partial derivatives of the objective function for each parameter w_k in $\boldsymbol{\theta} = (w_1, \ldots, w_p)$ are given by:

$$\frac{\partial \log P^*(\boldsymbol{r} \mid \boldsymbol{\theta})}{\partial w_k} = \frac{\partial}{\partial w_k} \sum_{j=1}^{|\boldsymbol{r}|} \log P^*\big(r_j \mid mb(R_j), \boldsymbol{\theta}\big) = \tag{13}$$
$$= \sum_{j=1}^{|\boldsymbol{r}|} \frac{1}{P^*\big(r_j \mid mb(R_j), \boldsymbol{\theta}\big)} \cdot \frac{\partial P^*\big(r_j \mid mb(R_j), \boldsymbol{\theta}\big)}{\partial f} \cdot \frac{\partial f}{\partial w_k} .$$

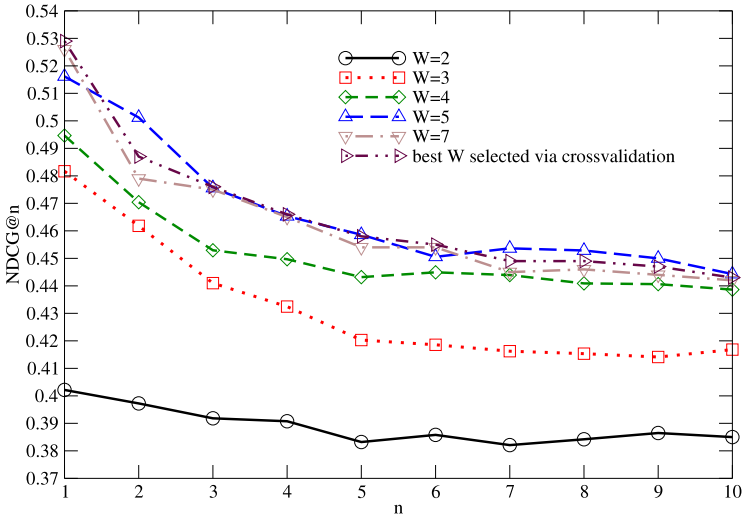

Fig. 2. NDCG@n values for different selection of the W parameter or when selecting the best W for each n via cross-validation on the validation set. The values increase very significantly before stabilizing around W=5, which provides a good trade-off between ranking accuracy and performance.

In our experimental setting, the scoring function f is implemented by a multilayer feed-forward neural network [18], and $\frac{\partial f}{\partial w_k}$ is computed via back-propagation. However, the presented method does not any specific assumption about the underlying implementation of the learned function.

Therefore, f could be implemented using any other machine learning method that can be trained by gradient descent.

V. Experimental Results

In the following of the paper, we refer to the proposed algorithm as *EnergyNDCG*, as we selected the NDCG as the rank goodness metric Φ to integrate into the energy function. For this reason, NDCG@n has been selected as the main metric in this paper for all the selected datasets.

A. Experiments on LETOR datasets

This set of experiments have been carried out on the LETOR 3.0 [1] OHSUMED dataset[1], which contains a set of medical publication abstracts. The dataset contains 106 queries and 16140 (query,document) pairs with associated relevance scores. The relevance scores have three possible levels: where a score equal to 0,1,2 corresponds to not relevant, possibly relevant and definitely relevant result, respectively. Each (query,url) pair is represented using several classical information retrieval (IR) features, such as query term frequency in the documents, inverse document frequency for the query terms, BM25, various language models, and other features proposed in the recent literature, such as HostRank, Feature propagation and Topical PageRank. The dataset contains five precomputed folds, each one composed by a training, validation and test set containing 60%, 20%, 20% of the overall data. All the models have been trained using the training set, the optimal parameters have been selected by cross validation on the validation set and, finally, have been evaluated on the test set. As standard

[1]Freely downloadable at http://research.microsoft.com/en-us/um/beijing/projects/letor/letor3dataset.aspx

Method	EnergyNDCG	AdaRankMAP	AdaRankNDCG
NDCG@1	0.516	0.491	0.533
NDCG@2	0.501	0.460	0.492
NDCG@3	0.476	0.427	0.479
NDCG@4	0.466	0.422	0.469
NDCG@5	0.459	0.416	0.467
NDCG@6	0.451	0.413	0.459
NDCG@7	0.454	0.410	0.459
NDCG@8	0.453	0.412	0.457
NDCG@9	0.450	0.410	0.454
NDCG@10	0.444	0.410	0.450
Method	ListNet	RankSVM	RankBoost
NDCG@1	0.533	0.496	0.463
NDCG@2	0.481	0.433	0.450
NDCG@3	0.473	0.421	0.456
NDCG@4	0.456	0.424	0.454
NDCG@5	0.443	0.416	0.449
NDCG@6	0.440	0.416	0.444
NDCG@7	0.441	0.413	0.441
NDCG@8	0.446	0.407	0.436
NDCG@9	0.446	0.412	0.433
NDCG@10	0.441	0.414	0.430
Method	FRank	SortNet	Regression
NDCG@1	0.530	0.514	0.445
NDCG@2	0.501	0.476	0.453
NDCG@3	0.481	0.473	0.443
NDCG@4	0.469	0.468	0.437
NDCG@5	0.459	0.462	0.428
NDCG@6	0.455	0.461	0.422
NDCG@7	0.453	0.458	0.422
NDCG@8	0.448	0.454	0.419
NDCG@9	0.446	0.446	0.414
NDCG@10	0.443	0.445	0.411

TABLE I
NDCG@N RESULTS ON THE LETOR 3.0 OHSUMED DATASET FOR THE PROPOSED METHOD AND OTHER STATE-OF-THE-ART LEARNING-TO-RANK APPROACHES (AS REPORTED ON THE LETOR DOCUMENTATION).

for the LETOR datasets, the results shown for each model are the averages of the results for the five folds, computed using the default evaluation script, provided together with the dataset. The ranking functions have been implemented by a 2-layer neural network with 10 hidden neurons with tanh activation functions. Cross-validation on the validation set has been performed to select the best performing neural network during the training process.

A key parameter in our model is the width W ruling the amount of dependencies in the graphical model. Figure 2 reports the NDCG@n scores for different values of W. The NDCG@n scores tend to increase significantly at the beginning, as more dependencies are required to model the probability distribution with high accuracy. However, there is little advantage to increase W above 5, as the additional computational requirements are not counterbalanced by a significant increase in the ranking accuracy. Figure 2 also plots the NDCG@n scores obtained when selecting via cross validation for each n the W providing the best results on the validation set. However, this does not significantly improves the results obtained with $W = 5$, which seems to provide a very good trade-off between accuracy and training complexity. For this reason, all the experiments reported in the following of the paper have been obtained by selecting $W = 5$. Table I compares the NDCG@n scores provided by the proposed method against state-of-the-art learning-to-rank methods in the pointwise, pairwise and listwise categories as reported on the LETOR documentation. In particular, EnergyNDCG performs similarly to AdaRankNDCG [19] and Frank [20], which are know to provide state-of-the-art performances. On the other hand, it over performs ListNet [21] and SortNet [22] by a small margin, and significantly improves over AdaRankMap [19], RankSVM [23], RankBoost [24] and Regression (a simple pointwise approach reported as reference on the LETOR documentation).

Method	EnergyNDCG	RankSVM	SortNet
NDCG@1	0.462	0.462	0.450
NDCG@2	0.573	0.563	0.568
NDCG@3	0.616	0.603	0.615
NDCG@4	0.641	0.628	0.639
NDCG@5	0.660	0.647	0.658
NDCG@6	0.675	0.663	0.672
NDCG@7	0.687	0.675	0.685
NDCG@8	0.697	0.686	0.696
NDCG@9	0.705	0.696	0.704
NDCG@10	0.713	0.705	0.712

TABLE II
NDCG@N RESULTS ON THE AOL WEB LOGS DATASET FOR THE PROPOSED METHOD AND OTHER STATE-OF-THE-ART LEARNING-TO-RANK APPROACHES.

B. Experiments on AOL Web logs data

This dataset has been constructed from the logs of the AOL commercial search engine released in 2005. The logs are a sample of the search activity of 658000 anonymized US-based users over a three month period (March-May 2005), which has been estimated to consist of approximately 1.5% of the overall AOL users in the considered period. The dataset contains 4.8 million queries and 1.8 million URLs. Since we are aware of the privacy concerns of this dataset, the logs have been pruned to remove queries that have been issued less than 30 times and by less than four distinct users. This should remove personal queries and documents that could allow associating any anonymous user id to a real person. In this dataset we follow a similar approach to what proposed in [25] by assuming that the relevance of a document for a query is proportional to the number of times the users selected it (click-through-rate), in particular, the click-though-rate ranges in the $[0, 1]$ interval and it has been split into 7 portions of equal size. The first sub-interval is associated to a 0 relevance level (non relevant result), and the relevance level is constantly increased by 0.5 sub-interval by sub-interval up to a maximum relevance level equal to 3 (essential result).

The dataset has been constructed by randomly selecting 10000 queries issued more than 30 times in the dataset. All the documents that have been selected for the query at least once by a user have been downloaded from the Internet. All the documents that were not available anymore at downloading time have been discarded, resulting into 140740 (query, document pairs). 40% of the queries have been selected for inclusion in the training set. The validation and test sets have been created by randomly splitting the remaining queries into two groups containing 20% and 40% of the initial set,

respectively.

The downloaded HTML documents have been parsed and processed together with their associated query. The output of this process is a vectorial representation of each (query, document) pair composed by 140 features, 5 of which depending on the document only and 135 on the document and query. In particular, the entire document and 4 sections of the document are taken into account: title, body, url and anchor. For each portion, 27 features compute the match between the query and specific sub-portions of the document like the BM25, cosine similarity, etc. Most of these features have been implemented as reported for LETOR dataset in [1].

The ranking functions have been implemented by a 2-layer neural network with 40 hidden neurons with tanh activations. Cross-validation on the validation set has been performed to select the best performing neural network during the training process. The W parameter has been set equal to 5 via cross-validation.

Table II reports the NDCG scores obtained on the AOL dataset by the proposed method, RankSVM and SortNet. The proposed method consistently outperforms RankSVM for all NDCG@i. It also generally performs better than SortNet.

VI. Conclusions

This paper presents a novel approach to learning-to-rank based on Markov Random Fields, which integrates the advantages of BoltzRank and LambdaRank. This is realized by natively integrating the target metric into the energy function of a Boltzmann distribution, which is then used to model the distribution over the ranks. This yields a model that is able to deal with both partial and total orders and can easily accommodate and optimize any target metric like NDCG or MAP. A second contribution of this paper is the study of different independence assumptions among the elements of a rank and of how such assumptions can be used to approximate the likelihood of a rank with its pseudo-likelihood surrogate. This overtakes a key limitation of other approaches like BoltzRank, requiring to approximate the normalization factor needed for computing the probability distribution by sub-sampling over the ranks. The experimental results show that this approach provides very good accuracy in spite of its computational lightness.

VII. Acknowledgments

This work has been partially funded by a research award from Google Inc. and by a grant from the MPS foundation. We thank Marco Gori, Marco Maggini, and Edmondo Trentin for the fruitful discussions that helped us to define this model.

References

[1] T. Liu, J. Xu, T. Qin, W. Xiong, and H. Li, "Letor: Benchmark dataset for research on learning to rank for information retrieval," in *Proceedings of SIGIR 2007 workshop on learning to rank for information retrieval*. Citeseer, 2007, pp. 3–10.

[2] K. Crammer and Y. Singer, "Pranking with ranking," in *Advances in Neural Information Processing Systems (NIPS)*. MIT Press, 2001, pp. 641–647.

[3] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender, "Learning to rank using gradient descent," in *Proceedings of the 22nd international conference on Machine learning*. ACM, 2005, pp. 89–96.

[4] Z. Cao, T. Qin, T. Liu, M. Tsai, and H. Li, "Learning to rank: from pairwise approach to listwise approach," in *Proceedings of the 24th international conference on Machine learning*. ACM, 2007, pp. 129–136.

[5] T. Qin, X. Zhang, M. Tsai, D. Wang, T. Liu, and H. Li, "Query-level loss functions for information retrieval," *Information Processing & Management*, vol. 44, no. 2, pp. 838–855, 2008.

[6] Y. Yue, T. Finley, F. Radlinski, and T. Joachims, "A support vector method for optimizing average precision," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 271–278.

[7] M. Volkovs and R. Zemel, "Boltzrank: learning to maximize expected ranking gain," in *Proceedings of the 26th Annual International Conference on Machine Learning*. ACM, 2009, pp. 1089–1096.

[8] T. Qin, T. Liu, and H. Li, "A general approximation framework for direct optimization of information retrieval measures," *Information retrieval*, vol. 13, no. 4, pp. 375–397, 2010.

[9] J. Besag, "Statistical analysis of non-lattice data," *The Statistician*, vol. 24, pp. 179–195, 1975.

[10] M. Richardson and P. Domingos, "Markov logic networks," *Machine Learning*, vol. 62, pp. 107–136, 2006.

[11] A. Freno, E. Trentin, and M. Gori, "Scalable pseudo-likelihood estimation in hybrid random fields," in *Proceedings of the 15th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD 2009)*, 2009, pp. 319–327.

[12] C. Burges, R. Ragno, and Q. Le, "Learning to rank with nonsmooth cost functions," *Advances in neural information processing systems*, vol. 19, p. 193, 2007.

[13] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, Cambridge (MA), 2009.

[14] J. Besag, "Spatial interaction and the statistical analysis of lattice systems," *Journal of the Royal Statistical Society*, vol. 36, pp. 192–236, 1974.

[15] R. Hofmann and V. Tresp, "Nonlinear markov networks for continous variables," in *Advances in Neural Information Processing Systems*, 1997.

[16] R. Kindermann and J. L. Snell, "Markov random fields and their applications," *American Mathematical Society*, 1980.

[17] A. Freno and E. Trentin, *Hybrid Random Fields. A Scalable Approach to Structure and Parameter Learning in Probabilistic Graphical Models*. Springer-Verlag, 2011.

[18] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall, New York, 1999.

[19] J. Xu and H. Li, "Adarank: A boosting al- gorithm for information retrieval. proceedings of the special interest group on information retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 391–398.

[20] M. F. Tsai, T. Y. Liu, T. Qin, H. H. Chen, and W. Y. Ma, "Frank: A ranking method with delity loss. proceedings of the special interest group on in- formation retrieval," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 383–390.

[21] Z. Cao, T. Qin, T. Y. Liu, M. F. Tsai, and H. Li, "Learning to rank: From pairwise approach to listwise approach," in *Proceedings of the International Conference on Machine Learning (ICML)*, 2007, pp. 129–136.

[22] L. Rigutini, T. Papini, M. Maggini, and F. Scarselli, "SortNet: learning to rank by a neural-based sorting algorithm," in *In proceedings of the SIGIR 2008 Workshop on Learning to Rank for Information Retrieval (LR4IR)*, vol. 42, no. 2, 2008, pp. 76–79.

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

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

[25] N. Craswell and M. Szummer, "Random walks on the click graph," in *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval*. ACM, 2007, pp. 239–246.