Using the Cross-Entropy Method to Re-Rank Search Results

Haggai Roitman, Shay Hummel
IBM Research - Haifa
Haifa 31905, Israel
{haggai,shayh}@il.ibm.com

Oren Kurland
Faculty of Industrial Engineering and Management, Technion
Haifa 32000, Israel
kurland@ie.technion.ac.il

ABSTRACT

We present a novel unsupervised approach to re-ranking an initially retrieved list. The approach is based on the Cross Entropy method applied to permutations of the list, and relies on performance prediction. Using pseudo predictors we establish a lower bound on the prediction quality that is required so as to have our approach significantly outperform the original retrieval. Our experiments serve as a proof of concept demonstrating the considerable potential of the proposed approach. A case in point, only a tiny fraction of the huge space of permutations needs to be explored to attain significant improvements over the original retrieval.

Categories and Subject Descriptors: H.3.3 [Information Search and Retrieval] Retrieval Models

General Terms: Algorithms, Experimentation

Keywords: Re-ranking, Optimization, Performance Prediction

1. INTRODUCTION

We present a novel unsupervised approach to the challenge of re-ranking a document list that was retrieved in response to a query so as to improve retrieval effectiveness. The approach utilizes a Monte-Carlo-based optimization method, named the Cross Entropy (CE) method [13], which is applied to permutations of the list. The approach relies on a retrieval performance predictor that can be applied to any ranking of the list.

Given the reliance on performance prediction, we present a novel pseudo predictor that enables to fully control the prediction quality. We use the pseudo predictor in our approach to set a lower bound on the prediction quality that is needed so as to have our approach significantly outperform the initial ranking. Further empirical evaluation provides a proof of concept for our approach. Specifically, via the exploration of a tiny fraction of the huge space of all possible permutations, our approach finds highly effective permutations. The retrieval effectiveness of these permutations is substantially, and statistically significantly better, than that of the original ranking of the list.

SIGIR'14, July 6–11, 2014, Gold Coast, Queensland, Australia.

http://dx.doi.org/10.1145/2600428.2609454.

2. RELATED WORK

The Cross Entropy (CE) method [13] that is used in our approach is a Monte Carlo framework for rare event estimation and combinatorial optimization. The CE method has been previously applied in many domains such as machine learning, simulation, networks, etc. [13]. To the best of our knowledge, our work is the first to use the CE method in the information retrieval domain.

Our approach relies on predicting the retrieval performance of permutations of a document list. Applying performance prediction to select one of two retrieved lists was explored in some work [2, 6, 3, 11]. However, the conclusions regarding the resultant effectiveness of using the proposed predictors were inconclusive. In contrast, we do not present a concrete predictor. Rather, we devise a pseudo predictor that enables to control prediction quality, and accordingly set a lower bound on the prediction quality required so as to have our approach outperform the initial ranking.

Using a simulation study, a lower bound on the prediction quality required for effective selective query expansion was set [8]. While this work focused on performance prediction over queries, our approach relies on prediction over rankings for the same query. Furthermore, our approach is not committed to any ranking paradigm. In addition, rather than use a simulation, we propose a novel pseudo predictor that allows to control prediction quality.

Finally, we note that some list-wise learning to rank approaches [10] are also based on finding effective permutations of the same list, although not with the Cross Entropy method that we employ. Permutations are explored during the training phase and a ranker is induced. In contrast, our approach employs optimization over permutations as an unsupervised re-ranking mechanism.

3. FRAMEWORK

3.1 Problem Def nition

Let $\mathcal{D}_q^k$ denote the list of the k documents in a corpus $\mathcal{D}$ that are the most highly ranked by *some* initial search performed in response to query q. Let $\Pi_{\mathcal{D}_q^k}$ denote the set of all $k!$ possible permutations (rankings) of the documents in $\mathcal{D}_q^k$. Let $\pi \in \Pi_{\mathcal{D}_q^k}$ denote a single permutation of $\mathcal{D}_q^k$ and let $\pi(d)$ further denote the position (rank) of document d ($\in \mathcal{D}_q^k$) in π. Let $\mathcal{Q}(\pi)$ denote the retrieval performance (effectiveness) of the permutation π ($\in \Pi_{\mathcal{D}_q^k}$).

The goal is to find a permutation π ($\in \Pi_{\mathcal{D}_q^k}$) such that $\mathcal{Q}(\pi)$ is maximized. Unfortunately, finding an optimal per-

mutation is NP-Complete [1]. In addition, with no prior relevance judgement on the documents in $\mathcal{D}_q^k$, the true performance $\mathcal{Q}(\pi)$ for any given permutation $\pi \in \Pi_{\mathcal{D}_q^k}$ is *unknown*. Hence, the performance of any given permutation can only be *predicted*, and the task becomes even more challenging.

Let $\widehat{\mathcal{Q}}(\pi)$ denote the predicted performance of the permutation π ($\in \Pi_{\mathcal{D}_q^k}$). Potential predictors may utilize any available pre-retrieval features [9] (e.g., induced from the query q and the corpus $\mathcal{D}$), post-retrieval features (e.g., induced from the result list $\mathcal{D}_q^k$ or the permutation π) or their combination [4].

3.2 An Optimization Approach

We next propose an optimization approach that effectively finds "promising" permutations which have the best predicted performance according to a given predictor $\widehat{\mathcal{Q}}(\pi)$.

Since the resultant retrieval effectiveness of the optimization procedure depends on the prediction quality of the predictor employed, we empirically derive a lower bound for the prediction quality of "effective" predictors. That is, if a predictor with a prediction quality higher than the lower bound is used in our approach then the approach is guaranteed to find — as determined based on the benchmarks we have used — as a solution a permutation π^* whose retrieval performance is better than that of the initial ranking.

3.2.1 Optimization using the Cross Entropy method

We propose a Monte-Carlo optimization approach to our permutation-based re-ranking task. The approach, which we term *Cross Entropy Re-ranking Optimization* (CERO), uses the Cross Entropy (CE) method [13]. Within the CE method, optimal solutions to hard problems (such as that we try to solve) are modeled as rare events whose occurrence probability is effectively estimated [13]. Given such estimates, optimal solutions can then be efficiently generated. Under a certain condition, the CE method is expected to converge to the optimal solution [12].

We now describe our algorithm and provide its pseudo code in Algorithm 1. The algorithm gets as an input the initial ranked list $\pi_{\mathcal{D}_q^k}$, a performance predictor $\widehat{\mathcal{Q}}(\pi)$ and several tuning parameters (mentioned below) that control the learning rate and convergence of the algorithm. The algorithm iteratively explores permutations in $\Pi_{\mathcal{D}_q^k}$ using random sampling. To this end, the algorithm induces a probability space of "promising" permutations over $\Pi_{\mathcal{D}_q^k}$ using the feedback it gets about the relative performance of permutations that were explored in previous iterations. It is easy to show that for a given k, a unique bijection exists between the permutation set $\Pi_{\mathcal{D}_q^k}$ and the set of all $k!$ possible Hamiltonian paths in a complete graph with k nodes. Therefore, random permutations can be efficiently drawn by sampling Hamiltonian paths using a simple constrained random walk method [13]. Let $P^t_{(i,j)}$ denote the the probability for a single step transition between node i and node j in the graph, which corresponds to the event $\pi(d_j) = \pi(d_i) + 1$, i.e., document d_i is ranked in π one position before document d_j. With no prior knowledge on the permutations probability space, the algorithm is initialized with the uniform probability (having the maximum entropy) and $\pi_{\mathcal{D}_q^k}$ is considered as the current best performing permutation. The algorithm's

Algorithm 1 Cross Entropy Re-ranking Optimization

input: $\pi_{\mathcal{D}_q^k}, \widehat{\mathcal{Q}}(\pi), N, \alpha, \lambda$
initialize:
$P^0_{(i,j)} = \begin{cases} \frac{1}{k-1}, & i \neq j \\ 0, & i = j \end{cases}$
$\pi^* = \pi_{\mathcal{D}_q^k}$
$t = 1$
loop
  Randomly draw N permutations $\pi_l \in \Pi_{\mathcal{D}_q^k}$ using P^{t-1}
  if $\widehat{\mathcal{Q}}(\pi^*) < \max_{l=1,\ldots,N} \widehat{\mathcal{Q}}(\pi_l)$ **then**
    $\pi^* = \arg\max_{l=1,\ldots,N} \widehat{\mathcal{Q}}(\pi_l)$
  end if
  Sort permutations π_l according to $\widehat{\mathcal{Q}}(\pi_l)$
  Let γ_t be the sample $(1-\alpha)$-quantile of the performances: $\widehat{\mathcal{Q}}(\pi_l); l = 1, .., N$
  for $i = 1, \ldots, k; j = 1, \ldots, k$ **do**
    $P^t_{(i,j)} = \frac{\sum_{l=1}^{N} I\{\pi_l(d_j)=\pi_l(d_i)+1\} I\{\widehat{\mathcal{Q}}(\pi_l)\geq\gamma_t\}}{\sum_{l=1}^{N} I\{\widehat{\mathcal{Q}}(\pi_l)\geq\gamma_t\}}$
    $P^t_{(i,j)} = \lambda P^{t-1}_{(i,j)} + (1-\lambda) P^t_{(i,j)}$
  end for
  if γ_t converged **then**
    stop and **return** π^*
  else
    $t = t + 1$
  end if
end loop

goal is to converge (via cross entropy minimization) to the unknown probability space of optimal permutations, from which optimal solutions can be generated [13].

On each iteration t, N random permutations are sampled based on the last induced promising permutations probability space P^{t-1}. Next, the predicted performance of each sampled permutation is calculated and the performance of the current best performing permutation π^* is updated accordingly. "New" promising permutations are then explored by first sorting the sampled permutations according to their predicted performance and then updating the transition probabilities based on the top-$\lceil \alpha N \rceil$ performing samples, whose minimum performance is γ_t. Given γ_t, each transition probability $P^t_{(i,j)}$ is induced according to the relative number of permutations out of the top-$\lceil \alpha N \rceil$ permutations (which have predicted performance equal to or higher than γ_t) that also ranked document d_i one position above document d_j. A fixed smoothing scheme, controlled by parameter λ, further allows to trade between the exploration (given by $P^t_{(i,j)}$) and the exploitation (given by $P^{t-1}_{(i,j)}$) of the algorithm.

The algorithm continuous until some convergence criteria is met. In this work we follow the convergence criteria suggested in [13] and stop the algorithm if the sample $(1-\alpha)$-performance quantile γ_t does not change within several consecutive iterations.

3.2.2 A criterion for effective prediction

The CERO algorithm is generic and can employ any performance predictor. Naturally, however, the prediction quality of the predictor has significant impact on the retrieval effectiveness of the ranking produced by the algorithm.

Thus, we turn to devise a method for determining the lower bound of prediction quality that will result in the CERO algorithm outperforming the initial ranking of $\mathcal{D}_q^k$. The bound is independent of a prediction approach.

Herein, we measure retrieval performance using average precision (AP@k); i.e., $\mathcal{Q}(\pi)$ in our case is the AP of the per-

corpus	# of documents	queries	disks
GOV2	25,205,179	701-850	GOV2
WT10G	1,692,096	451-550	WT10g
TREC8	528,155	401-450	4&5-{CR}

Table 1: TREC data used for experiments.

mutation π. Following standard practice in work on query-performance prediction [4], prediction quality is measured by the Pearson correlation between the true AP of permutations ($\mathcal{Q}(\pi)$) and their predicted performance ($\widehat{\mathcal{Q}}(\pi)$).

To derive a lower bound on prediction quality, we next present an approach for generating pseudo AP predictors, whose prediction quality can be controlled. Following previous observations [5], we assume that true AP values follow a normal distribution[1].

We first normalize the AP of the permutation π ($\in \Pi_{\mathcal{D}_q^k}$):

$$\mathcal{Q}_{norm}(\pi) = \frac{\mathcal{Q}(\pi) - E_{\Pi_{\mathcal{D}_q^k}}(AP)}{\sqrt{Var_{\Pi_{\mathcal{D}_q^k}}(AP)}}; \quad (1)$$

$E_{\Pi_{\mathcal{D}_q^k}}(AP)$ and $Var_{\Pi_{\mathcal{D}_q^k}}(AP)$ are the expectation and the variance of the true AP values of the permutations in $\Pi_{\mathcal{D}_q^k}$, respectively. The two statistics can be estimated using maximum likelihood estimation for normal distribution, by sampling a large enough random (uniform) sample of permutations in $\Pi_{\mathcal{D}_q^k}$ (e.g., $N = 1000$). Since AP follows a normal distribution, we get that as $k \to \infty$, for any permutation $\pi \in \Pi_{\mathcal{D}_q^k}$: $\mathcal{Q}_{norm}(\pi) \sim N(0,1)$ [5].

Proposition 1 defines a ρ-correlated pseudo AP predictor; that is, a predictor with a ρ prediction quality (i.e., Pearson correlation with true AP). The proof is quite straightforward and is ommitted due to space considerations.

PROPOSITION 1. *Given a query q, initial result list $\mathcal{D}_q^k$, and permutation $\pi \in \Pi_{\mathcal{D}_q^k}$, a ρ-correlated pseudo AP predictor, denoted $\widehat{\mathcal{Q}}_\rho(\pi)$, is obtained as follows:*

$$\widehat{\mathcal{Q}}_\rho(\pi) = \mathcal{Q}_{norm}(\pi)\rho + \sqrt{1-\rho^2}X \quad (2)$$

where $\mathcal{Q}_{norm}(\pi)$ is the normalized true AP value according to Eq. 1 and $X \sim N(0,1)$.

4. EVALUATION

4.1 Setup

The TREC corpora and queries used for experiments are specified in Table 1. Titles of TREC topics served for queries. The Apache Lucene[2] search library (version 4.3) was used for the experiments. Documents and queries were processed using Lucene's default analysis (i.e., tokenization, stemming, stopwords, etc). For each query, 100 documents were retrieved using Lucene's implementation, employed with default free-parameter values, of each of the following retrieval methods: vector space model (TF-IDF), query-likelihood (QL with Dirichlet smoothing) and Okapi BM25. Thus, we obtained three initial lists, $\mathcal{D}_q^k$, composed of $k = 100$ documents, for each query. Mean average precision (MAP@k) serves as the evaluation measure. Statistically significant differences of performance are measured using the paired t-test with a 95% confidence level.

Each initial list was re-ranked using the CERO algorithm employed with pseudo AP predictors of varying prediction quality levels. To control prediction quality, the pseudo predictors were generated according to Eq. 2; the prediction quality was varied from $\rho = 0.05$ (worst predictor) to $\rho = 1.0$ (best predictor). Following previous recommendations [13], the algorithm's learning parameters were set as follows: $N = 1000$, $\alpha = 0.01$ and $\lambda = 0.7$.

Eff ciency considerations. To implement CERO, we used a parallelized version of the Cross Entropy method [7]. On average, CERO converged in 21.32 iterations (with a 15.6 standard deviation). CERO explores at each iteration a maximum of $N = 1000$ permutations. Thus, all together, CERO considered *only* about $21k - 36k$ permutations out of the 100! possible permutations of the initial list.

4.2 Results

CERO's effectiveness. We first study the potential of our permutation-based optimization approach; specifically, in finding highly effective permutations in the huge space (100!) of permutations. To this end, we neturilize the effect of prediction quality by applying CERO with a "predictor" which reports the true AP of the considered permutations (i.e., $\rho = 1.0$). The resultant MAP performance is presented in Table 2. As reference comparisons we use the initial ranking and an optimal re-ranking where all relevant documents from the initial list are positioned at the highest ranks.

We can see in Table 2 that, overall, CERO results in very good approximations. Specifically, CERO's MAP is at least as 91% as good as that of the optimal MAP. Recall from above that CERO explores only a tiny fraction of all permutations of the documents in the result list. It is worth noting that an even better approximation may be obtained by finer tuning of the algorithm (e.g., following [12]). We leave this exploration for future work, and view the results presented in Table 2 as a solid proof of concept for the optimization approach we have employed.

The effect of prediction quality. In Table 3 we present the effect of the prediction quality of the pseudo AP predictors used in CERO on its MAP retrieval performance. Evidently, and as should be expected, the higher the prediction quality (ρ), the better the performance. Furthermore, as from $\rho = 0.3$ CERO improves over the initial ranking for all the retrieval methods and across all corpora; for $\rho \geq 0.35$ the improvements are always statistically significant. With higher prediction quality, the improvements over the initial ranking become very substantial.

5. CONCLUSIONS AND FUTURE WORK

We presented a novel approach to re-ranking an initially retrieved list. The approach is based on applying the Cross Entropy optimization method [13] upon permutations of the list. Query-performance predictors are used to evaluate the performance of permutations. Empirical evaluation pro-

[1]The assumption was further verified in our experiments using the χ^2 goodness-of-fit test. Details are omitted due to space considerations.

[2]http://lucene.apache.org

	GOV2			WT10G			TREC8		
	BM25	QL	TF-IDF	BM25	QL	TF-IDF	BM25	QL	TF-IDF
Initial	.151	.172	.137	.156	.164	.158	.198	.206	.190
Optimal	.274	.296	.253	.352	.391	.349	.400	.407	.388
CERO ($\rho = 1.0$)	.251 (92%)	.275 (93%)	.232 (91%)	.320 (91%)	.367 (94%)	.322 (92%)	.367 (92%)	.373 (92%)	.356 (92%)

Table 2: The MAP of the initial retrieval, optimal re-ranking and the CERO algorithm when employed with the true AP as the "predictor". The percentages are with respect to Optimal.

	GOV2			WT10G			TREC8		
	BM25	QL	TF-IDF	BM25	QL	TF-IDF	BM25	QL	TF-IDF
Initial	.151	.172	.137	.156	.164	.158	.198	.206	.190
$\rho = 1.0$	.251 (66%)✓	.275 (60%)✓	.232 (69%)✓	.320 (105%)✓	.367 (124%)✓	.322 (104%)✓	.367 (85%)✓	.373 (81%)✓	.356 (87%)✓
$\rho = .95$	.249 (65%)✓	.272 (58%)✓	.231 (69%)✓	.320 (105%)✓	.362 (121%)✓	.321 (104%)✓	.364 (83%)✓	.371 (80%)✓	.356 (87%)✓
$\rho = .90$	.245 (62%)✓	.267 (56%)✓	.225 (64%)✓	.320 (104%)✓	.361 (120%)✓	.317 (101%)✓	.361 (82%)✓	.370 (80%)✓	.354 (86%)✓
$\rho = .85$	.242 (60%)✓	.262 (53%)✓	.225 (64%)✓	.319 (104%)✓	.354 (115%)✓	.317 (101%)✓	.358 (81%)✓	.364 (77%)✓	.348 (83%)✓
$\rho = .80$	.237 (57%)✓	.255 (49%)✓	.218 (59%)✓	.311 (99%)✓	.355 (116%)✓	.313 (99%)✓	.354 (79%)✓	.362 (76%)✓	.342 (80%)✓
$\rho = .75$	.232 (54%)✓	.248 (44%)✓	.212 (55%)✓	.309 (97%)✓	.342 (108%)✓	.309 (96%)✓	.348 (76%)✓	.358 (74%)✓	.339 (78%)✓
$\rho = .70$	.222 (47%)✓	.245 (42%)✓	.208 (52%)✓	.301 (93%)✓	.345 (110%)✓	.303 (93%)✓	.342 (72%)✓	.345 (68%)✓	.330 (73%)✓
$\rho = .65$	.222 (47%)✓	.231 (35%)✓	.203 (48%)✓	.301 (92%)✓	.333 (103%)✓	.296 (88%)✓	.332 (67%)✓	.344 (67%)✓	.322 (70%)✓
$\rho = .60$	.215 (43%)✓	.229 (33%)✓	.198 (45%)✓	.292 (87%)✓	.328 (100%)✓	.293 (86%)✓	.325 (64%)✓	.332 (62%)✓	.317 (67%)✓
$\rho = .55$	.207 (37%)✓	.219 (28%)✓	.194 (42%)✓	.290 (85%)✓	.312 (90%)✓	.285 (81%)✓	.311 (57%)✓	.321 (56%)✓	.304 (60%)✓
$\rho = .50$	.199 (32%)✓	.214 (24%)✓	.181 (32%)✓	.280 (79%)✓	.303 (85%)✓	.284 (81%)✓	.303 (53%)✓	.310 (51%)✓	.294 (54%)✓
$\rho = .45$	.192 (27%)✓	.202 (18%)✓	.173 (27%)✓	.260 (66%)✓	.272 (66%)✓	.260 (65%)✓	.285 (44%)✓	.293 (42%)✓	.278 (46%)✓
$\rho = .40$	.182 (20%)✓	.194 (13%)✓	.172 (25%)✓	.257 (64%)✓	.258 (57%)✓	.249 (58%)✓	.267 (35%)✓	.268 (30%)✓	.259 (36%)✓
$\rho = .35$	.170 (12%)✓	.180 (5%)✓	.157 (15%)✓	.211 (35%)✓	.246 (50%)✓	.225 (43%)✓	.244 (23%)✓	.248 (21%)✓	.237 (24%)✓
$\rho = .30$	.161 (6%)✓	.173 (1%)	.144 (5%)	.199 (27%)✓	.222 (35%)✓	.213 (35%)✓	.221 (11%)✓	.227 (10%)✓	.212 (12%)✓
$\rho = .25$	.154 (2%)	*.165* (−4%)	*.135* (−2%)	.172 (10%)✓	.177 (8%)✓	.172 (9%)✓	.201 (1%)	*.194* (−6%)	*.179* (−6%)
$\rho = .20$	*.143* (−5%)	*.150* (−13%)	*.125* (−8%)	*.151* (−3%)	*.139* (−15%)	*.130* (−18%)	*.167* (−16%)	*.159* (−22%)	*.165* (−13%)
$\rho = .15$	*.134* (−11%)	*.144* (−16%)	*.120* (−12%)	*.134* (−14%)	*.124* (−24%)	*.112* (−29%)	*.145* (−27%)	*.148* (−28%)	*.139* (−27%)
$\rho = .10$	*.120* (−20%)	*.141* (−18%)	*.117* (−14%)	*.110* (−30%)	*.119* (−28%)	*.105* (−33%)	*.123* (−38%)	*.133* (−35%)	*.120* (−37%)
$\rho = .05$	*.119* (−21%)	*.137* (−20%)	*.110* (−20%)	*.79* (−50%)	*.93* (−43%)	*.103* (−35%)	*.115* (−42%)	*.122* (−41%)	*.109* (−43%)

Table 3: The MAP of the initial retrieval and of CERO when using different ρ-correlated pseudo AP predictors. ✓ marks a statistically significant improvement over the initial ranking. Numbers in italics are lower than those for the initial ranking. The reported percentages are with respect to the initial ranking.

vided a proof of concept for our approach. That is, the optimization procedure finds highly effective permutations by exploring only a tiny fraction of the space of all possible permutations. In addition, we devised novel pseudo predictors that allow to carefully control prediction quality and to infer the minimal prediction quality required for our approach to (significantly) outperform the original ranking.

Our main plan for future work is devising query-performance predictors that yield a prediction quality that is higher than that we established as a lower bound for effective application of our approach. We note that almost all previously proposed query-performance predictors [4] are not suited for this task as they operate over different queries rather than over different lists retrieved for the same query.

Acknowledgment

We would like to thank David Carmel for discussions on an earlier version of this work. Oren Kurland's work is supported in part by the Israel Science Foundation under grant no. 433/12 and by a Google faculty research award.

6. REFERENCES

[1] N. Alon. Ranking tournaments. *SIAM Journal on Discrete Mathematics*, 20(1):137–142, 2006.

[2] G. Amati, C. Carpineto, and G. Romano. Query difficulty, robustness, and selective application of query expansion. In *Proc. of ECIR*, pages 127–137, 2004.

[3] N. Balasubramanian and J. Allan. Learning to select rankers. In *Proc. of SIGIR*, pages 855–856, 2010.

[4] D. Carmel and E. Yom-Tov. *Estimating the Query Difficulty for Information Retrieval*. Synthesis Lectures on Information Concepts, Retrieval, and Services. Morgan & Claypool Publishers, 2010.

[5] B. Carterette, J. Allan, and R. Sitaraman. Minimal test collections for retrieval evaluation. In *Proc. of SIGIR*, pages 268–275, 2006.

[6] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. A language modeling framework for selective query expansion. Technical Report IR-338, Center for Intelligent Information Retrieval, University of Massachusetts, 2004.

[7] G. E. Evans, J. M. Keith, and D. P. Kroese. Parallel cross-entropy optimization. In *Proc. of WSC*, pages 2196–2202, 2007.

[8] C. Hauff and L. Azzopardi. When is query performance prediction effective? In *Proc. of SIGIR*, pages 829–830, 2009.

[9] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *Proc. of CIKM*, pages 1419–1420, 2008.

[10] T.-Y. Liu. *Learning to Rank for Information Retrieval*. Springer, 2011.

[11] X. Liu and W. B. Croft. Experiments on retrieval of optimal clusters. Technical Report IR-478, Center for Intelligent Information Retrieval (CIIR), University of Massachusetts, 2006.

[12] L. Margolin. On the convergence of the cross-entropy method. *Annals of Operations Research*, 134(1):201–214, 2005.

[13] R. Y. Rubinstein and D. P. Kroese. *The cross-entropy method: a unified approach to combinatorial optimization, Monte-Carlo simulation and machine learning*. Springer, 2004.