A Robust Ranking Methodology based on Diverse Calibration of AdaBoost

Róbert Busa-Fekete[1,2] Balázs Kégl[1,3]
Tamás Éltető[3] György Szarvas[2,4]

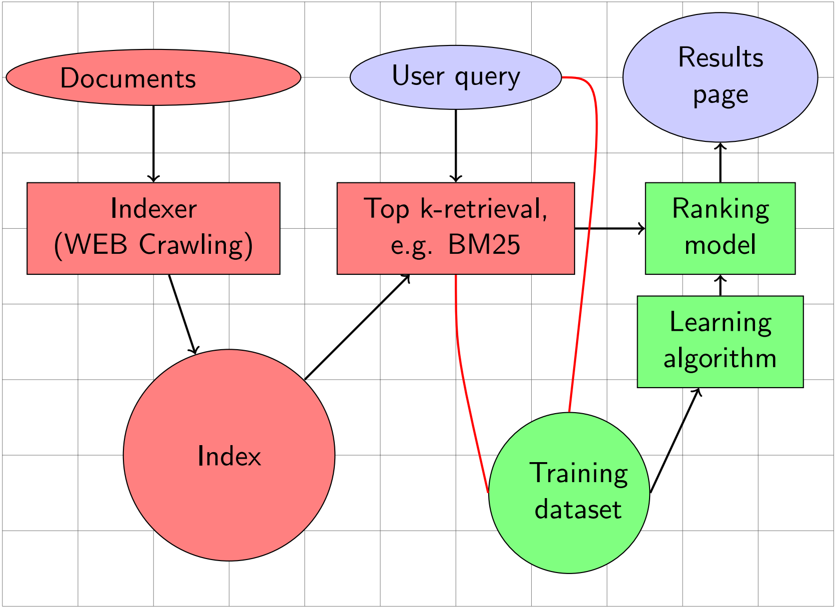
[1]Linear Accelerator Laboratory (LAL), University of Paris-Sud, CNRS

[2]Research Group on Artificial Intelligence of the Hungarian Academy of Sciences and University of Szeged (RGAI)

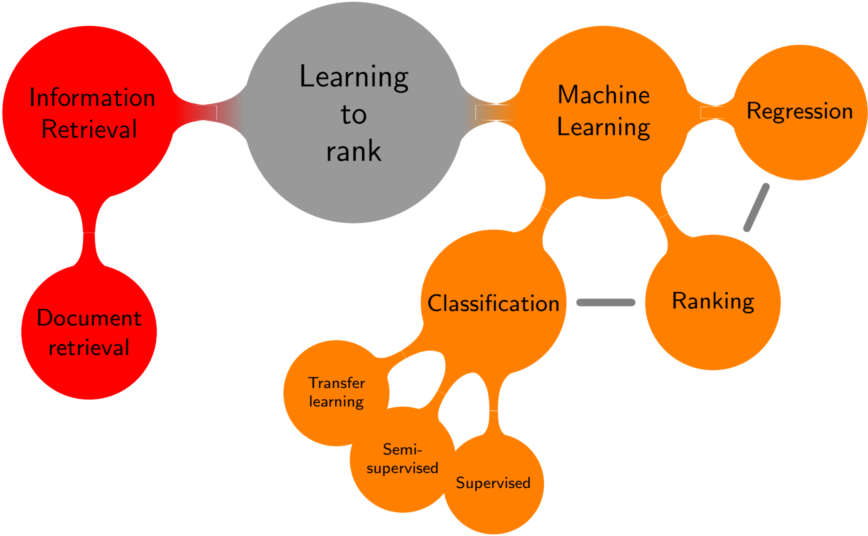
[3]Computer Science Laboratory (LRI), University of Paris-Sud, CNRS and INRIA-Saclay

[4]Ubiquitous Knowledge Processing (UKP) Lab, Computer Science Department Technische Universität Darmstadt

September 6, 2011


Documents
User query
Results page
Indexer (WEB Crawling)
Top k-retrieval, e.g. BM25
Ranking model
Learning algorithm
Index
Training dataset

Learning-to-rank



Definition of the Learning-To-Rank(LTR) task

- $\mathbf{D} = \{\mathcal{D}^{(1)}, \ldots, \mathcal{D}^{(M)}\}$ are the query objects
- a query object consists of a set of $n^{(k)}$ pairs:

$$\mathcal{D}^{(k)} = \left\{ \left(\mathbf{x}_1^{(k)}, \ell_1^{(k)}\right), \ldots \left(\mathbf{x}_{n^{(k)}}^{(k)}, \ell_{n^{(k)}}^{(k)}\right) \right\}.$$

- $\mathbf{x}_i^{(k)} \in \mathbb{R}^B$ represents the kth query and the ith document received as a potential hit for the query
- $\ell_i^{(k)}$ represents the label index of the query-document pair $\mathbf{x}_i^{(k)}$. They are typically integers between 1 and K
- They define only partial ordering for a query $\mathcal{D}^{(k)}$ (since typically $n^{(k)} > K$)
- **GOAL** of the ranker is to output a permutation $\mathbf{j}^{(k)} = (j_1, \ldots, j_{n^{(k)}})$ over the integers $(1, \ldots, n^{(k)})$ for each query object $\mathcal{D}^{(k)}$

Normalized Discounted Cumulative Gain (NDCG)

- Relevance grades expresses the relevance of the ith document to the kth query on a numerical scale
- A popular choice for the numerical relevance grades is $z_\ell = 2^{\ell-1} - 1$ for all $\ell = 1, \ldots, K$
- Discounted Cumulative Gain (DCG) for $\mathbf{j}^{(k)}$ and $\mathcal{D}^{(k)}$ is

$$\widehat{\mathrm{DCG}}\left(\mathbf{j}^{(k)}, \mathcal{D}^{(k)}\right) = \sum_{i=1}^{n^{(k)}} c_i z_{j_i}^{(k)},$$

  where c_i is the *discount factor* in the form of $c_i = \frac{1}{\log(1+i)}$

- Example: $\mathcal{D}^{(k)} =$ 3 3 1 1 0 $\Rightarrow \mathbf{j}^{(k)} \Rightarrow$ 0, 1, 3, 1, 3 $\Rightarrow$ $\widehat{\mathrm{DCG}}\left(\mathbf{j}^{(k)}, \mathcal{D}^{(k)}\right) =$
  0 $*1.44+$ 1 $*0.91+$ 3 $*0.72+$ 1 $*0.62+$ 3 $*0.55 \approx 5.37$
  - To normalize DCG between 0 and 1, one can divide it with the DCG score of the best permutation.
  - Truncated toplist like ROC_{10}
  - Averaging over all queries

Basic approaches

- **Pointwise:** the relevance grades are learned directly using either a classification or a regression method
  - Only slightly different from conventional machine learning methods
  - McRank (classification based), PRank (regression based)
- **Pairwise:** the pairwise preferences of documents with respect to a query are learned typically by a classification method
  - RankBoost, RankSVM
- **Listwise:** the whole partial/total order are learned
  - Most computationally intensive
  - For example, optimizing a smooth and differentiable upper bound of the evaluation measure (such as NCDG) using a conventional machine learning technique
  - AdaRank, SVM-MAP

Bayes optimal permutation

- $\ell_i^{(k)}$ is considered as a random variable

$$p^*(\ell|\mathbf{x}_i^{(k)}) = P(\ell_i^{(k)} = \ell|\mathbf{x}_i^{(k)})$$

- Bayes scoring function

$$v^*(\mathbf{x}_i^{(k)}) = \mathbb{E}\left\{z|\mathbf{x}_i^{(k)}\right\} = \sum_{\ell=1}^{K} z_\ell p^*(\ell|\mathbf{x}_i^{(k)})$$

- The *expected* DCG for any permutation $\mathbf{j}^{(k)}$ is

$$\mathrm{DCG}(\mathbf{j}^{(k)}, \mathcal{D}^{(k)}) = \sum_{i=1}^{n^{(k)}} c_i \mathbb{E}\left\{z|\mathbf{x}_{j_i}^{(k)}\right\} = \sum_{i=1}^{n^{(k)}} c_i v^*(\mathbf{x}_{j_i}^{(k)}).$$

- Bayes optimal permutation

$$\mathbf{j}^{(k)*} = \arg\max_{\mathbf{j}^{(k)}} \mathrm{DCG}(\mathbf{j}^{(k)}, \mathcal{D}^{(k)}).$$

A good property of Bayes optimal permutation

- Cossock&Zhang (2008) showed[1] that a Bayes optimal permutation $\mathbf{j}^{(k)*}$ has the property that if $c_i > c_{i'}$, then for the Bayes-scoring function we have $v^*\left(\mathbf{x}_{j_i^{(k)*}}\right) > v^*\left(\mathbf{x}_{j_{i'}^{(k)*}}\right)$
- Consequences:
  1. $\mathbf{j}^{(k)*}$ can be easily obtained from the Bayes scoring function $\Rightarrow v\left(\mathbf{x}^{(k)}_{j_1^{(k)*}}\right) \geq \ldots \geq v\left(\mathbf{x}^{(k)}_{j^*_{n(k)}}\right)$.
  2. This result justifies those *pointwise* approaches where either v^* is estimated in a *regression* setup or $p^*\left(\ell|\mathbf{x}_j^{(k)}\right) \approx p\left(\ell|\mathbf{x}_j^{(k)}\right)$ in a *discrete density estimation* setup

[1] Cossock, D., Zhang, T.: Statistical analysis of Bayes optimal subset ranking. IEEE Transactions on Information Theory 54(11), 5140-5154 (2008)

Upper bound for the excess of DCG

- Assume that there is given $p^*(\ell|\mathbf{x}_i) \approx p(\ell|\mathbf{x}_i)$.
- This estimate generates a permutation over $\mathcal{D}$
  1. scoring function: $v(\mathbf{x}_i) = \sum_{\ell=1}^{K} z_\ell p(\ell|\mathbf{x}_i)$
  2. permutation: $v(\mathbf{x}_{j_1^v}) \geq \ldots \geq v(\mathbf{x}_{j_n^v})$
- Let $p, q \in [1, \infty]$ and $1/p + 1/q = 1$. Then

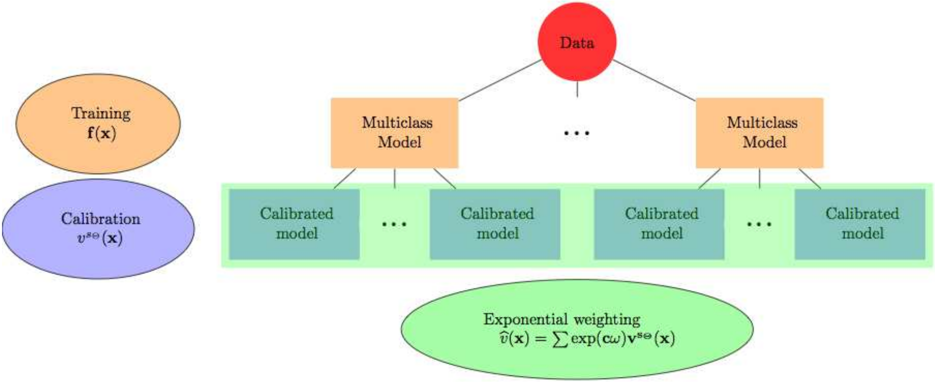
$$\mathrm{DCG}(\mathbf{j}^*, \mathcal{D}) - \mathrm{DCG}(\mathbf{j}^v, \mathcal{D}) \leq$$

$$\underbrace{\max_{\mathbf{j},\mathbf{j}'} \left(\sum_{i=1}^{n} \sum_{\ell=1}^{K} \left| (c_{j_i} - c_{j'_i}) z_\ell \right|^p \right)^{\frac{1}{p}}}_{\text{constant}} \underbrace{\left(\sum_{i=1}^{n} \sum_{\ell=1}^{K} \left| p(\ell|\mathbf{x}_i) - p^*(\ell|\mathbf{x}_i) \right|^q \right)^{\frac{1}{q}}}_{\text{quality of approximation}}$$

Our approach

- GOAL: $p^*\left(\ell|\mathbf{x}_j^{(k)}\right) \approx p\left(\ell|\mathbf{x}_j^{(k)}\right)$
- REAL GOAL: We will estimate $p^*\left(\ell|\mathbf{x}_i^{(k)}\right)$ in many ways – hoping that we can obtain many diverse estimation – and we will mix them using a proper weighting scheme!

Our approach



1. Training cost-sensitive multi-class AdaBoost.MH

Training AdaBoost.MH

- Feature vectors: $\mathbf{X} = \left(\mathbf{x}_1^1, \ldots, \mathbf{x}_{n(1)}^1, \ldots, \mathbf{x}_1^M, \ldots, \mathbf{x}_{n(M)}^M\right)$
- Labels: $\mathbf{Y} = \left(\mathbf{y}_1^1, \ldots, \mathbf{y}_{n(1)}^1, \ldots, \mathbf{y}_1^M, \ldots, \mathbf{y}_{n(M)}^M\right)$

$$y_{i,\ell}^{(k)} = \begin{cases} +1 & \text{if } \ell_i^{(k)} = \ell, \\ -1 & \text{otherwise.} \end{cases}$$

- The training instances were upweighted exponentially proportionally to their relevance
- Base learners: decision trees and decision products[2]
- Hyperparameters of base learners were not validated
- All models were used in an "ensemble of ensembles" scheme
- Only the number of iterations were validated
- Open source C++ package: http://www.multiboost.org

[2] Kégl and Busa-Fekete: Boosting products of base classifiers, ICML'09

2. Calibration

Class-probability-based calibration (CPC)

- The output of AdaBoost.MH is

$$\mathbf{f}\left(\mathbf{x}_i^{(k)}\right) = \left(f_1\left(\mathbf{x}_i^{(k)}\right), \ldots, f_K\left(\mathbf{x}_i^{(k)}\right)\right).$$

- The class probability can be calibrated using *sigmoidal* function

$$s_\theta(f) = s_{a,b}(f) = \frac{1}{1 + \exp\left(-a(f-b)\right)}$$

to obtain

$$p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right) = \frac{s_\theta\left(f_\ell\left(\mathbf{x}_i^{(k)}\right)\right)}{\sum_{\ell'=1}^{K} s_\theta\left(f_{\ell'}\left(\mathbf{x}_i^{(k)}\right)\right)}.$$

- scoring function: $v(\mathbf{x}_i^{(k)}) = \sum_{\ell=1}^{K} z_\ell p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right)$ *expected rel. grade*
- permutation: $v\left(\mathbf{x}_{j_1^v}^{(k)}\right) \geq \ldots \geq v\left(\mathbf{x}_{j_{n(k)}^v}^{(k)}\right)$

Class-probability-based calibration: obtaining Θ

- Θ can be tuned by minimizing a so-called *target calibration function* (TCF) $L^{\mathcal{A}}(\theta, \mathbf{f}) \Rightarrow \theta^{\mathcal{A},\mathbf{f}} = \arg\min_{\theta} L^{\mathcal{A}}(\theta, \mathbf{f})$
  1. Log-sigmoid TCF

$$L^{\mathrm{LS}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} - \log p^{s_\theta}\left(\ell_i^{(k)} | \mathbf{x}_i^{(k)}\right)$$

  2. Entropy weighted log-sigmoid TCF
  3. Expected loss TCF

$$L^{\mathrm{EL}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} \sum_{\ell=1}^{K} \mathcal{L}\left(\ell, \ell_i^{(k)}\right) p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right)$$

  4. Expected label loss TCF, similar to the Expected loss TCF, but the loss are calculated for expected label $\overline{\ell}_i^{(k)} = \sum_{\ell=1}^{K} \ell p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right)$
  5. The surrogate function of SmoothGrad can be also used[3]

[3] Chapelle&Wu: Gradient descent optimization of smoothed information retrieval metrics. Inform. Retr. 13(3), 216235 (2010)

Regression-based Calibration (RBC)

- Let us recall that the output of AdaBoost.MH is

$$\mathbf{f}\left(\mathbf{x}_i^{(k)}\right) = \left(f_1\left(\mathbf{x}_i^{(k)}\right), \dots, f_K\left(\mathbf{x}_i^{(k)}\right)\right).$$

- We need a scalar scoring function:

$$\widehat{v}\left(\mathbf{x}_i^{(k)}\right) = g\left(\mathbf{f}(\mathbf{x}_i^{(k)})\right)$$

- Standard multi-class solution:

$$g(\mathbf{f}) = \arg\max_k f_k$$

- We regress the relevance grades $z_i^{(k)}$ vs. $\mathbf{f}(\mathbf{x}_i^{(k)})$

$$g = \arg\min_{g' \in \mathcal{G}} \sum_{k,i} \left(g'\left(\mathbf{f}(\mathbf{x}_i^{(k)})\right) - z_i^{(k)}\right)^2$$

- $\mathcal{G}$: linear, Gaussian process, neural network, polynomial

3. Ensemble of ensembles:

putting the calibrated models into a huge ensemble classifier

Ensemble of ensembles: choosing $\pi(\mathcal{A}, \mathbf{f})$

1. $\pi(\mathcal{A}, \mathbf{f}) = \exp(c\omega^{\mathcal{A},\mathbf{f}})$, where $\omega^{\mathcal{A},\mathbf{f}}$ is the NDCG_{10} score of the ranking obtained by using $v^{\mathcal{A},\mathbf{f}}(\mathbf{x})$
2. c is hyperparameter
3. Ultimate scoring function:

$$v^{\mathrm{ENSEMBLE}}(\mathbf{x}) = \sum_{\mathcal{A},\mathbf{f}} \exp\left(c\omega^{\mathcal{A},\mathbf{f}}\right) v^{\mathcal{A},\mathbf{f}}(\mathbf{x}).$$

4. This gives a slight listwise touch to our approach
5. Advantages:
    - computationally efficient
    - theoretically well-founded: Exponentially Weighted Average Forecaster[4]

[4] Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, New York, NY, USA (2006)

LETOR datasets

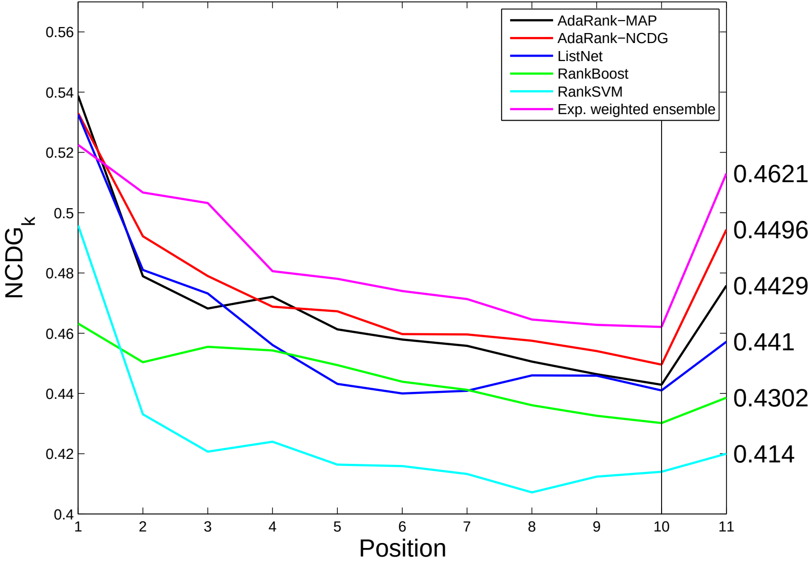
- Most widely used datasets
- LETOR 3.0 consists of 7 datasets
- Only OHSUMED has three relevance grades
- LETOR 4.0 consists of 2 datasets (MQ2007 and MQ2008)
- 46 features in both datasets, but webpages and query terms are also available
- Baseline performances are provided using 5-fold cross validation

	Number of docs	Number of queries	Docs. per query
LETOR 3.0 Ohsumed	16140	106	≈ 152
LETOR 4.0 MQ2007	69623	1692	≈ 41
LETOR 4.0 MQ2008	15211	784	≈ 19

Experiments

1. AdaRank-MAP, AdaRank-NDCG, ListNet, RankBoost, RankSVM
2. In our setup we validated the number of iterations of AdaBoost.MH based on the NCDG_{10} performance of the ultimate scoring function $v^{\mathrm{ENSEMBLE}}(\mathbf{x})$
3. The calibration was carried out on validation set.
4. We used the official evaluation tools

Results



(a) LETOR 3.0/Ohsumed

NDCG values for various ranking algorithms.

Method	Letor 3.0 Ohsumed	Letor 4.0 MQ2007	Letor 4.0 MQ2008
Eval. metric	$NDCG_{10}$	Avg. NDCG	Avg. NDCG
AdaRank-MAP	0.4429	0.4891	0.4915
AdaRank-NDCG	0.4496	0.4914	0.4950
ListNet	0.4410	0.4988	0.4914
RankBoost	0.4302	**0.5003**	0.4850
RankSVM	0.4140	0.4966	0.4832
Exp. w. ensemble	0.4561	0.4974	0.5006
Exp. w. ensemble(CPC)	**0.4621**	0.4975	0.4998
Exp. w. ensemble(RBC)	0.4493	0.4976	**0.5004**
AdaBoost+D. Tree	0.4164	0.4868	0.4843
AdaBoost+D. Product	0.4162	0.4785	0.4768

Conclusions and further work

1. CPC achieved significant improvement only on OHSUMED
2. all relevance levels are well represented in OHSUMED
3. We plan to investigate the robustness of our method to label noise
4. Accelerate the testing phase by using Markov Decision Process (on-going work)

Thanks for Your Attention!

- Our boosting package is available at
- `http://www.multiboost.org/`
- Hope to see you at our poster!

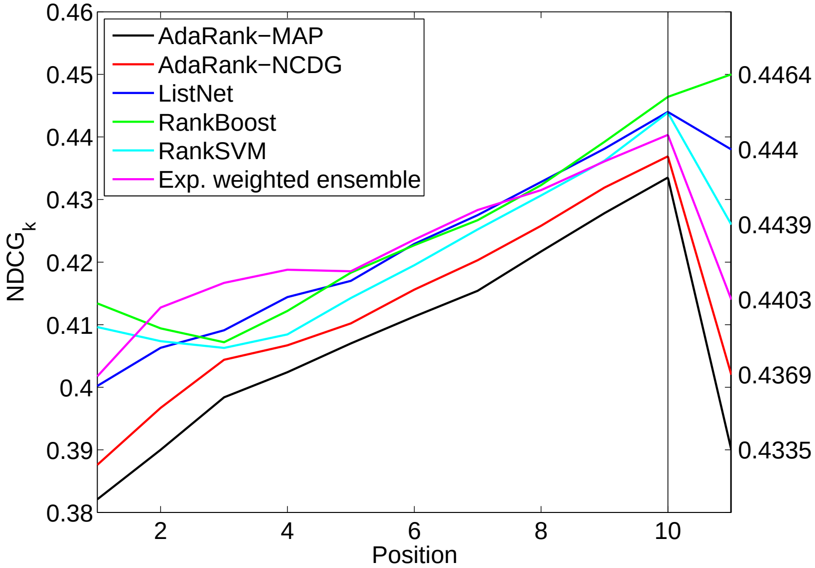
Class-probability-based calibration: obtaining Θ

- Θ can be tuned by minimizing a so-called target calibration function (TCP) $L^{\mathcal{A}}(\theta, \mathbf{f}) \Rightarrow \theta^{\mathcal{A},\mathbf{f}} = \arg\min_\theta L^{\mathcal{A}}(\theta, \mathbf{f})$
  1. $L^{\mathrm{LS}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} -\log p^{s_\theta}\left(\ell_i^{(k)} | \mathbf{x}_i^{(k)}\right)$
  2. $L_C^{\mathrm{EWLS}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} -\log p^{s_\theta}\left(\ell_i^{(k)} | \mathbf{x}_i^{(k)}\right) \times H_M\left(p^{s_\theta}\left(\ell_1 | \mathbf{x}_i^{(k)}\right), \ldots, p^{s_\theta}\left(\ell_K | \mathbf{x}_i^{(k)}\right)\right)^C$, where $H_M(p_1, \ldots, p_K) = -\sum_{\ell=1}^{K} p_\ell \log p_\ell$
  3. $L^{\mathrm{EL}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} \sum_{\ell=1}^{K} \mathcal{L}\left(\ell, \ell_i^{(k)}\right) p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right)$
  4. $L^{\mathrm{ELL}}(\theta) = \sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} \mathcal{L}\left(\bar{\ell}_i^{(k)}, \ell_i^{(k)}\right)$, where the expected label is defined as $\bar{\ell}_i^{(k)} = \sum_{\ell=1}^{K} \ell p^{s_\theta}\left(\ell | \mathbf{x}_i^{(k)}\right)$
  5. $L_\sigma^{\mathrm{SNDCG}}(\theta) = -\sum_{k=1}^{M} \sum_{i=1}^{n^{(k)}} \sum_{i'=1}^{n^{(k)}} z_i^{(k)} c_{i'} h_{\theta,\sigma}\left(\mathbf{x}_i^{(k)}, \mathbf{x}_{j_{i'}}^{(k)}\right)$, where[5]

$$h_{\theta,\sigma}\left(\mathbf{x}_i^{(k)}, \mathbf{x}_{i'}^{(k)}\right) = \frac{\exp\left(-\frac{1}{\sigma}\left(v^{s_\theta}\left(\mathbf{x}_i^{(k)}\right) - v^{s_\theta}\left(\mathbf{x}_{i'}^{(k)}\right)\right)^2\right)}{\sum_{i''=1}^{n^{(k)}} \exp\left(-\frac{1}{\sigma}\left(v^{s_\theta}\left(\mathbf{x}_i^{(k)}\right) - v^{s_\theta}\left(\mathbf{x}_{i''}^{(k)}\right)\right)^2\right)}$$

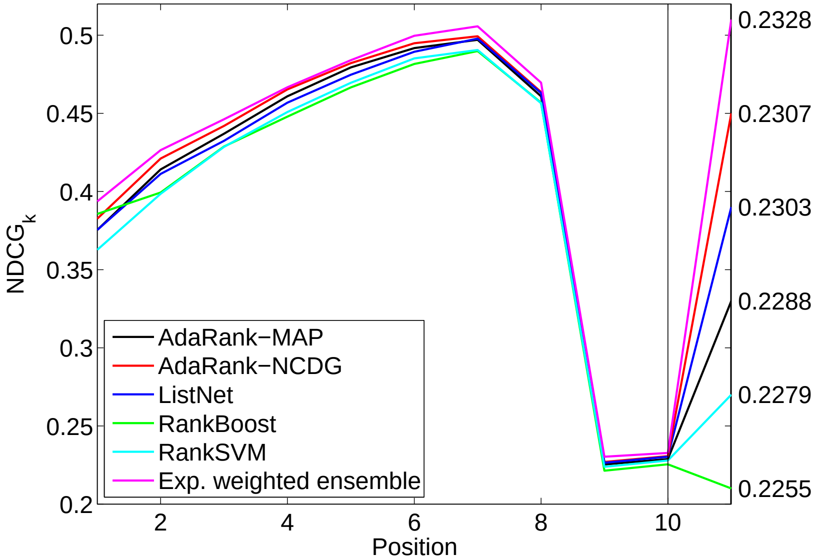
[5] Chapelle&Wu: Gradient descent optimization of smoothed information retrieval metrics. Inform. Retr. 13(3), 216235 (2010)

Results



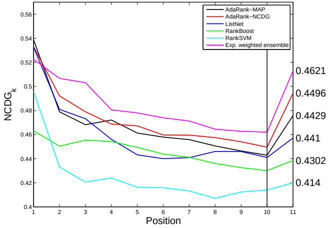
(b) LETOR 4.0/MQ2007

Results

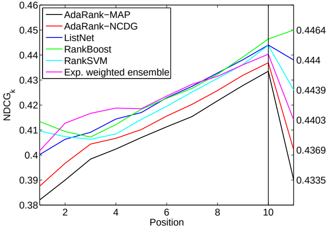


(c) LETOR 4.0/MQ2008

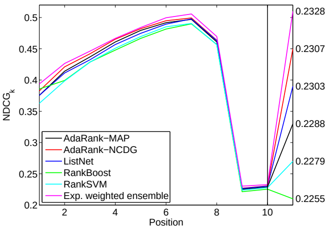
Results



(d) LETOR 3.0/Ohsumed



(e) LETOR 4.0/MQ2007



(f) LETOR 4.0/MQ2008