ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ

INFORMATION TECHNOLOGY

DOI: 10.20998/2079-0023.2023.01.12
UDC 519.2

K. S. YAMKOVYI, National Technical University "Kharkiv Polytechnic Institute", Assistant Professor of Computer Mathematics and Data Analysis Department, Kharkiv, Ukraine, e-mail: klym.yamkovyi@cs.khpi.edu.ua; ORCID: https://orcid.org/0000-0001-9512-4150

ADAPTATION OF LAMBDAMART MODEL TO SEMI-SUPERVISED LEARNING

The problem of information searching is very common in the age of the internet and Big Data. Usually, there are huge collections of documents and only multiple percent of them are relevant. In this setup brute-force methods are useless. Search engines help to solve this problem optimally. Most engines are based on learning to rank methods, i.e. first of all algorithm produce scores for documents based on they feature and after that sorts them according to the score in an appropriate order. There are a lot of algorithms in this area, but one of the most fastest and a robust algorithm for ranking is LambdaMART. This algorithm is based on boosting and developed only for supervised learning, where each document in the collection has a rank estimated by an expert. But usually, in this area, collections contain tons of documents and their annotation requires a lot of resources like time, money, experts, etc. In this case, semi-supervised learning is a powerful approach. Semi-supervised learning is an approach to machine learning that combines a small amount of labeled data with a large amount of unlabeled data during training. Unlabeled data, when used in combination with a small quantity of labeled data, can produce significant improvement in learning accuracy. This paper is dedicated to the adaptation of LambdaMART to semi-supervised learning. The author proposes to add different weights for labeled and unlabeled data during the training procedure to achieve higher robustness and accuracy. The proposed algorithm was implemented using Python programming language and LightGBM framework that already has supervised the implementation of LambdaMART. For testing purposes, multiple datasets were used. One synthetic 2D dataset for a visual explanation of results and two real-world datasets MSLR-WEB10K by Microsoft and Yahoo LTRC.

Keywords: learning to rank, information retrieval, semi-supervised learning, pairwise ranking, LambdaMART, pseudo labeling, NDCG.

Introduction. A modern person searches for useful information or some stuff every day. Usually, we sort a list of items to find the best option. But today in the Big Data age almost all information stored on the internet and information retrieval systems like search engines can help us to handle this. Search engines based on learning to rank (LR) methods. LR is the application of machine learning that constructs ranking models for information retrieval systems. The ranking model proposes to rank, i.e. producing a permutation of items in new, unseen lists in a similar way to rankings in the training data.

One of the naive and simple to understand approaches in LR is pairwise comparison. In this case, we take two objects and try to classify which of them is more preferable than another. For example, bubble sort works with the same rules. The main advantages of this approach are simplicity and the ability to describe an algorithm's behavior. The pairwise approach is popular in the state-of-the-art ranking methods, i.e. LambdaMART [1]. This method was developed more than ten years ago, but still one of the most efficient and faster.

As we say above, today we are dealing with a huge dataset. And not always there is an opportunity to label all data, because of cost or complexity. So when we can't mark the whole dataset we can use some compromise in the form of semi-supervised learning: train model on labeled and unlabeled data at the same time. In our research, we've developed a semi-supervised version of the LambdaMART algorithm.

This paper is organized as follows: Section 2 describes the problem statement. Section 3 provides a short review of related literature in the area of the research. Section 4 describes the proposed semi-supervised learning to rank approach. After that, we'll show the accuracy of the proposed algorithm on multiple datasets in Section 5. Finally, Section 6 contains conclusions.

Problem statement. Suppose that $D = \{d_1, ..., d_N\}$ – set of documents, where N – denotes number of documents, $Q = \{q_1, ..., q_M\}$ – set of queries, where M – number of queries and $N > M$. $R = \{r_1, ..., r_K\}$ – ordinal set of ranks, where K denotes the number of ranks. Usually in Information Retrieval (IR) $K \in [2,5]$ e.g. R = {irrelevant, relevant} in binary case and if $K = 5$ then R = {bad, fair, good, excellent, perfect} [2]. There exists a total order between the ranks $r_L \succ r_{L-1} \succ ... \succ r_1$, where "$\succ$" denotes a preference relationship.

Each query q_i related with a subset of retrieved documents $D_i = \{d_{i1},...,d_{in(q_i)}\}$, where $n(q_i)$ – size of D_i and list of labels $Y_i = \{y_{i1},...,y_{im(q_i)}\}$, where $y_{ij} \in Y$ and denotes rank of document d_{ij}, $n(q_i) > m(q_i)$ – some documents haven't labels as a part of the query. Feature vector $\boldsymbol{x} \in X$ represent the "query – document" pair $\boldsymbol{x}_{ij} = \Psi(q_i, d_{ij})$ [3]. Usually, only query – document features are available, queries and documents feature not disclosed because it's a commercial secret of search engines. Thus training data can be formally represent as subset of labeled query – document pairs $L = \{(\boldsymbol{x}_{ij}, y_{ij})\}$, $|L| = l$ and subset of unlabeled pairs, $U = \{\boldsymbol{x}_{ij}\}$, $|U| = u$ the set U will be used in an unsupervised manner.

In IR information about labels can take several forms such as absolute labels, preference relations, complete orderings, or partial ordering. We'll use pairwise preference relations of the form document "i is preferred to j", and denoted by $i \succ j$. This information can be obtained by comparison of documents' labels.

Our goal is to estimate a function $f(\boldsymbol{x}, \boldsymbol{w})$, where $\boldsymbol{w}$ – the function's parameters, which compute the score of a document related to a query: $o_i = f(\boldsymbol{x}_i, \boldsymbol{w})$. For any pair of items $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ we'll compute the score o_i and o_j. The two scores are mapped to a learned probability that x_i should be ranked higher that $\boldsymbol{x}_j$ using a probabilistic model $P(\boldsymbol{x}_i \succ \boldsymbol{x}_j)$. The one of the widely used in IR is Bradley – Terry model:

$$P(i \succ j) = \frac{o_i}{o_i + o_j}. \quad (1)$$

Let $O = \{i_1 \succ j_1, i_2 \succ j_2, ...\}$ be a set of observed pairwise preferences. We can estimate parameters $\boldsymbol{w}$ by maximizing likelihood.

$$C = \sum_{(i,j)\in L} I_{i\succ j} \log P(i \succ j) + \\ + \sum_{(i,j)\in U} I_{i\succ j} \log P(i \succ j) \to \max_{\boldsymbol{w}}, \quad (2)$$

where $I_{i\succ j}$ – the indicator that denotes the observed pairwise preference, $I_{i\succ j} = 1$ when $i \succ j$, 0 otherwise [4];

α – regularization coefficient.

Related works. During this research, we relied on the different groups of papers. First of all is an overview of ranking algorithms, especially pairwise methods. Christopher J.C. Burges showed in his work evolution from RankNet – and neural networks-based algorithm to LambdaMART boosting-based approach for ranking. And nowadays LambdaMART is one of the state-of-the-art algorithms in LR. According to him the key observation of LambdaRank is thus that to train a model, we don't need the costs themselves: we only need the gradients (of the costs with respect to the model scores). The arrows (λ's) mentioned on fig. 1 are exactly those gradients. Note that this does not mean that the gradients are not gradients of a cost. The model was designed to learn Normalized Discounted Cumulative Gain (NDCG). LambdaMART combines MART (multiple additive regression trees) and LambdaRank [5].

Fig. 1. An example of RankNet gradients (black) and LambdaRank gradients (red)

In [6] the authors covered clustering. They compare unsupervised and semi-supervised clustering, describing a typology of methods. The main difference of semi-supervised clustering is the availability of some external knowledge. But they also consider that the available knowledge is too far from being representative of a target classification of the items, so that supervised learning is not possible, even in a transductive form.

So if we have labels of good quality but still not enough for supervised learning the good choice is transductive learning. One of the first approaches of transductive learning is T-SVM [7] by Vapnik. Also, the transductive approach was applied in LR by Rahangdale A. U., and Raut S. in [8]. The authors propose rank preserving clustering with PLoacalSearch and get pseudo labels for unlabeled data. They present semi-supervised learning that adopts a clustering-based transductive method and combines it with a non-measure-specific listwise approach to learn the LTR model.

And another group of methods is proposed cost function for boosting algorithms with regularization on unlabeled data and. [9] presents a boosting-based algorithm for learning a bipartite ranking function (BRF) with partially labeled data. The proposed approach is a semi-supervised inductive ranking algorithm which, as opposed to transductive algorithms, can infer an ordering on new examples that were not used for its training. And in [10] the authors made the assumption that unlabeled data gives information about the data distribution $P(x)$ and the

structure of the unlabeled data tells us about the ranking function. The authors show it on non-convex clusters. So unlabeled data defines the extent and shape of clusters and labeled data determines the class/function value of each cluster.**Semi-supervised ranking function.** The proposed semi-supervised approach consists of two steps: pseudo labeling and learning of ranking function. The main focus of the paper is a second step.

During the pseudo labeling step we used the k-NN algorithm for label assignment. This algorithm is very common choice in semi-supervised learning. The k-NN algorithm uses the compactness hypothesis: nearest documents have the same labels. For each example in the labeled training set L, we assigned the same label to its nearest neighbors from U [11]. The k-NN algorithm with a small number of neighbors (<3) showed the best experimental result. It could be explained by complicated structure of cluster, overlapping between them and noise in the data.

As we described above the pairwise ranking approach based on the Bradley – Terry model, which could be expressed as sigmoid (fig. 2):

$$P(i \succ j) = \frac{o_i}{o_i + o_j} = \frac{e^{o_i - o_j}}{1 + e^{o_i - o_j}} = 1 - P(j \succ i). \qquad (3)$$

So sigmoid usually optimized using cross-entropy loss:

$$\begin{aligned} C_{ij} &= -\overline{P_{ij}} \log P_{ij} - (1 - \overline{P_{ij}}) \log(1 - P_{ij}) = \\ &= -\overline{P_{ij}} [\log P_{ij} - \log(1 - P_{ij})] - \log(1 - P_{ij}) = \\ &= -\overline{P_{ij}} (o_i - o_j) + \log(1 + e^{o_{ij}}), \end{aligned} \qquad (4)$$

where $\overline{P_{ij}}, P_{ij}$ – true and predicted probabilities respectively.

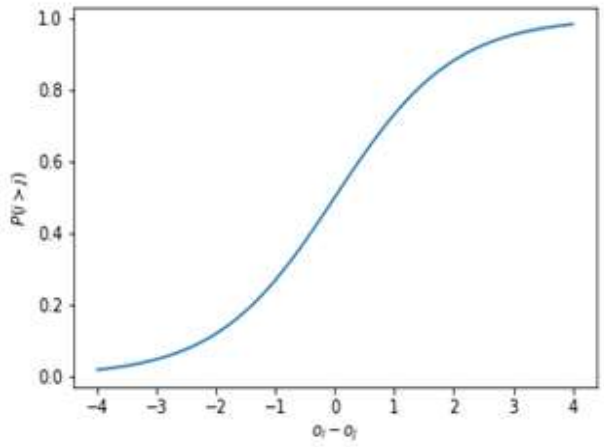


Fig. 2. Bradley – Terry model

Derivative of the loss function:

$$\frac{\delta C_{ij}}{\delta \boldsymbol{w}} = \frac{\delta C_{ij}}{\delta o_{ij}} \frac{\delta o_{ij}}{\delta \boldsymbol{w}} = \frac{\delta C_{ij}}{\delta o_{ij}} [\frac{\delta o_i}{\delta \boldsymbol{w}} - \frac{\delta o_j}{\delta \boldsymbol{w}}]. \qquad (5)$$

Let's describe the derivative:

$$\frac{\delta C_{ij}}{\delta o_{ij}} = \frac{\delta(-\overline{P_{ij}} o_{ij})}{\delta o_{ij}} + \frac{\delta \log(1 + e^{o_{ij}})}{\delta o_{ij}} = -\overline{P_{ij}} + P_{ij}. \qquad (6)$$

Training process using mini-batch gradient descent, but the batch is a query. And the query consists of labeled and unlabeled documents, the LambdaMART learning looks like this:

$$\boldsymbol{w} \to \boldsymbol{w} - \eta \sum_L \frac{\delta C_L}{\delta \boldsymbol{w}}, \qquad (7)$$

where η – learning rate.

Formula (7) should be modified for semi-supervised model:

$$\boldsymbol{w} \to \boldsymbol{w} - \eta [\sum_L \frac{\delta C_L}{\delta \boldsymbol{w}} + \alpha \sum_U \frac{\delta C_U}{\delta \boldsymbol{w}}]. \qquad (8)$$

The equation above introduced a semi-supervised version of the LambdaMART model.

For clarity, we'll use an example: a query consisting of 5 documents: 1, 2, and 3 with labels and 4, 5 without labels. In this case, describe the equation (8):

$$\begin{aligned} &\sum_L \frac{\delta C_L}{\delta \boldsymbol{w}} + \alpha \sum_U \frac{\delta C_U}{\delta \boldsymbol{w}} = \\ &= \frac{\delta C_{12}}{\delta o_{12}} (\frac{\delta o_1}{\delta \boldsymbol{w}} - \frac{\delta o_2}{\delta \boldsymbol{w}}) + \frac{\delta C_{13}}{\delta o_{13}} (\frac{\delta o_1}{\delta \boldsymbol{w}} - \frac{\delta o_3}{\delta \boldsymbol{w}}) + \\ &+ \frac{\delta C_{23}}{\delta o_{23}} (\frac{\delta o_2}{\delta \boldsymbol{w}} - \frac{\delta o_3}{\delta \boldsymbol{w}}) + \alpha \frac{\delta C_{45}}{\delta o_{45}} (\frac{\delta o_4}{\delta \boldsymbol{w}} - \frac{\delta o_5}{\delta \boldsymbol{w}}) + \\ &+ \alpha \frac{\delta C_{14}}{\delta o_{14}} (\frac{\delta o_1}{\delta \boldsymbol{w}} - \frac{\delta o_4}{\delta \boldsymbol{w}}) + \alpha \frac{\delta C_{15}}{\delta o_{15}} (\frac{\delta o_1}{\delta \boldsymbol{w}} - \frac{\delta o_5}{\delta \boldsymbol{w}}) + \\ &+ \alpha \frac{\delta C_{24}}{\delta o_{24}} (\frac{\delta o_2}{\delta \boldsymbol{w}} - \frac{\delta o_2}{\delta \boldsymbol{w}}) + \alpha \frac{\delta C_{25}}{\delta o_{25}} (\frac{\delta o_2}{\delta \boldsymbol{w}} - \frac{\delta o_5}{\delta \boldsymbol{w}}) + \\ &+ \alpha \frac{\delta C_{34}}{\delta o_{34}} (\frac{\delta o_3}{\delta \boldsymbol{w}} - \frac{\delta o_2}{\delta \boldsymbol{w}}) + \alpha \frac{\delta C_{35}}{\delta o_{35}} (\frac{\delta o_3}{\delta \boldsymbol{w}} - \frac{\delta o_5}{\delta \boldsymbol{w}}). \end{aligned} \qquad (9)$$

For optimization, we'll introduce factorization, because $\frac{\delta C_{ij}}{\delta o_{ij}}$ computed faster than $\frac{\delta o_i}{\delta \boldsymbol{w}}$:

$$\begin{aligned} &\frac{\delta o_1}{\delta \boldsymbol{w}} [\frac{\delta C_{12}}{\delta o_{12}} + \frac{\delta C_{13}}{\delta o_{13}} + \alpha \frac{\delta C_{14}}{\delta o_{14}} + \alpha \frac{\delta C_{15}}{\delta o_{15}}] + \\ &+ \frac{\delta o_2}{\delta \boldsymbol{w}} [\frac{\delta C_{23}}{\delta o_{23}} + \alpha \frac{\delta C_{24}}{\delta o_{24}} + \alpha \frac{\delta C_{25}}{\delta o_{25}} - \frac{\delta C_{12}}{\delta o_{12}}] + \\ &+ \frac{\delta o_3}{\delta \boldsymbol{w}} [\alpha \frac{\delta C_{34}}{\delta o_{34}} + \alpha \frac{\delta C_{35}}{\delta o_{35}} - \frac{\delta C_{13}}{\delta o_{13}} - \frac{\delta C_{23}}{\delta o_{23}}] + \\ &+ \frac{\delta o_4}{\delta \boldsymbol{w}} [\alpha \frac{\delta C_{45}}{\delta o_{45}} - \alpha \frac{\delta C_{14}}{\delta o_{14}} - \alpha \frac{\delta C_{24}}{\delta o_{24}} - \alpha \frac{\delta C_{34}}{\delta o_{34}}] + \\ &+ \frac{\delta o_5}{\delta \boldsymbol{w}} [-\alpha \frac{\delta C_{45}}{\delta o_{45}} - \alpha \frac{\delta C_{15}}{\delta o_{15}} - \alpha \frac{\delta C_{25}}{\delta o_{25}} - \alpha \frac{\delta C_{35}}{\delta o_{35}}]. \end{aligned} \qquad (10)$$

Let's introduce lambda:

$$\lambda_{ij} = \frac{\delta C_{ij}}{\delta o_{ij}},$$
$$\lambda_i = \sum_{i:\{i,j\}\in I_L} \lambda_{ij} - \sum_{j:\{i,j\}\in I_L} \lambda_{ij} + \quad (11)$$
$$+\alpha[\sum_{i:\{i,j\}\in I_U} \lambda_{ij} - \sum_{j:\{i,j\}\in I_U} \lambda_{ij} +],$$

where I – set of pairs of indices $\{i, j\}$.

So in general:

$$\sum_{i,j\in I} \frac{\delta C_{ij}}{\delta o_{ij}} = \sum_i \frac{\delta o_i}{\delta \boldsymbol{w}} \lambda_i. \quad (12)$$

Only need to add an NDCG component to the gradient. The idea is quite straightforward: if the change in NDCG by swapping i and j is large, we expect the gradient to be large as well to make this happen in the hope of maximizing NDCG. And indeed, the paper showed that this works empirically:

$$\frac{\delta C_{\text{LambdaRank},ij}}{\delta \boldsymbol{w}} = \frac{\delta C_{ij}}{\delta \boldsymbol{w}} |\Delta NDCG_{ij}|, \quad (13)$$

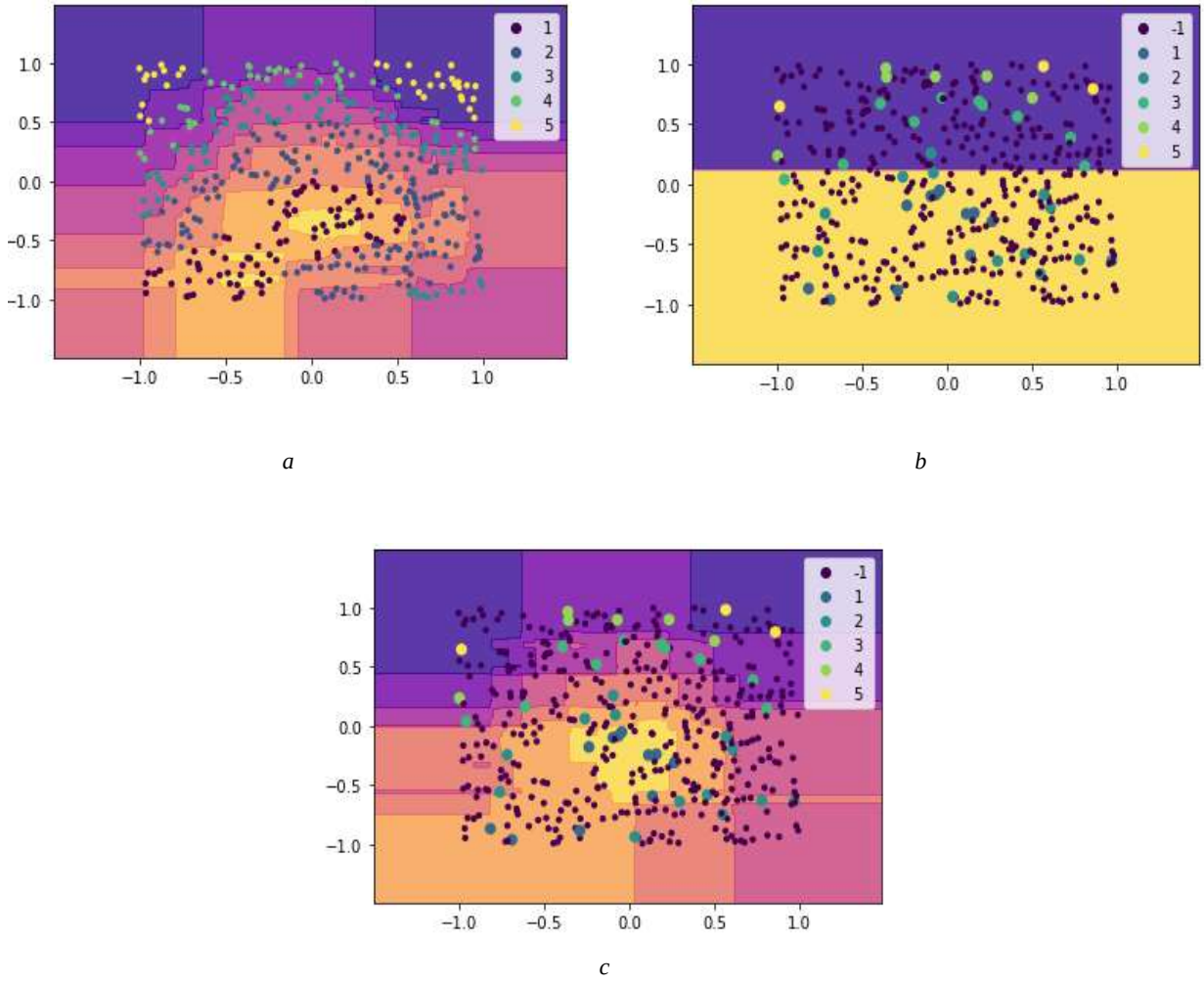
where $|\Delta NDCG_{ij}|$ – change in NDCG score if document i swapped with j.

$$|\Delta NDCG_{ij}| =$$
$$= \frac{1}{iDCG} |(\frac{2^{o_i}-1}{\log(1+r_i)} + (\frac{2^{o_j}-1}{\log(1+r_j)}) + ...) -$$
$$- (\frac{2^{o_j}-1}{\log(1+r_j)} + (\frac{2^{o_i}-1}{\log(1+r_i)}) + ...)| = \quad (14)$$
$$= |\frac{1}{iDCG}(2^{o_i} - 2^{o_l})(\frac{1}{\log(1+r_i)} - \frac{1}{\log(1+r_j)})|,$$

where s_i – true label of document i;

r_i – rank position of document i returned by the model [12].

Experiments. For the experiment was used LightGBM – an open-source gradient boosting framework that uses tree-based learning algorithm. It is based on decision tree algorithms and used for ranking, classification, and other machine learning tasks. The development focus is on performance and scalability. The



a *b*

c

Fig. 3. Ranking methods comparison: *a* – supervised on full dataset, *b* – supervised using only available part of dataset, *c* – semi-supervised method. Here: −1 – no label, 1–5 – ranks

framework has an implementation of a pairwise ranking approach and allows the use of custom learning objections for ranking.

First of all, we've generated a dataset with 2D feature space and multiple queries. These allow us to visualize results and understand the models. The fig. 3 compares supervised and semi-supervised methods.

Fig. 3, *c* shows that a semi-supervised approach with a small number of labels can build decision function almost the same as the supervised with all labels. For pseudo labeling was used the k-NN method with the number of neighbors equal to 3. With synthetic data, the semi-supervised clustering method works also well, but for concordance with real-world, datasets were chosen exactly the k-NN approach.

A dataset was used Microsoft Learning to Rank Datasets (MSLR-WEB10K) and C14 – Yahoo! Learning to Rank Challenge (Yahoo LTRC). Both datasets consist of feature vectors extracted from query – URL pairs along with relevance judgment labels from 0 to 4.

As we can see in tab. 1 the datasets are imbalanced: the higher the rank, the fewer documents are found in the dataset.

So to make a semi-supervised dataset we removed labels of 90 % documents per query. Documents for label removal were chosen by uniform distribution and this aggravated the disbalance problem. Because in some queries there were only a couple of relevant labels and after labeling these queries lost their relevant document with labels at all. Let's say in this way we generate the worst case.

As a metric NDCG was chosen. The metric works with multiple relevance levels and always in range [0, 1]. For better measuring, we take NDCG on different levels 1, 3, and 10.

First, we measured baseline on full (upper) and only semi-supervised (lower) editions for both datasets (tab. 1). As we can see in the table below the additional 90 % of labeled data can increase metric only by 0.05.

Table 1 – Dataset description

	MSLR-WEB10K	Yahoo LTRC
Number of train docs	723412	473134
Number of train queries	6000	19944
Number of docs per query mean	120.57	23.72
Number of docs per query std	70.15	9.00
Number of docs per query mode	110.00	9.00
Ranks distribution	0 – 0.52 1 – 0.32 2 – 0.13 3 – 0.02 4 – 0.01	0 – 0.25 1 – 0.36 2 – 0.29 3 – 0.08 4 – 0.02

The tab. 2 and tab. 3 show that the proposed semi-supervised method can improve accuracy. Also, we can see that improvement grows with relevance level (an improvement on level 1 higher than on level 10).

For pseudo labeling was used the k-NN method with the number of neighbors equal to 1. This is caused by the method's sensitivity to labeling quality. So datasets complex inner structure has a negative influence on pseudo labeling quality with a high number of neighbors or semi-supervised clustering approaches. And as a result, it's bringing the model accuracy decreasing.

Table 2 – MSLR-WEB10K results

	NDCG@1	NDCG@3	NDCG@10
Lower baseline	0.5169	0.5089	0.5163
Upper baseline	0.5621	0.5440	0.5416
Proposed method	0.5290	0.5118	0.5184

Table 3 – Yahoo LTRC results

	NDCG@1	NDCG@3	NDCG@10
Lower baseline	0.7187	0.7206	0.7738
Upper baseline	0.7583	0.7545	0.7965
Proposed method	0.7310	0.7286	0.7778

Conclusion. The given research described a semi-supervised adaptation of the LambdaMART algorithm. The proposed semi-supervised method is based on the regularization and pseudo labeling approach and improves the ranking accuracy on datasets with a small number of labeled documents. It especially improves accuracy on a high relevance level. But the proposed method is very sensitive to the quality of pseudo labeling. Also, the task is complicated by unbalancing the target datasets. Because of that, we should choose labeling algorithms very carefully.

References

1. Burges C. J. C., Svore K. M., Wu Q., Gao J. *Ranking, boosting and model adaptation*. URL: https://www.microsoft.com/en-us/research/publication/ranking-boosting-and-model-adaptation/ (accessed 07.04.2023).
2. Chang Y., Chapelle O. Yahoo! Learning to Rank Challenge Overview. *JMLR: Workshop and Conference Proceedings 14*. 2011. P. 1–24.
3. Xu H., Li H. AdaRank: A Boosting Algorithm for Information Retrieval. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2007. P. 391–398.
4. Yilmaz E., Szummer M. Semi-supervised Learning to Rank withPreference Regularization. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. 2011. P. 269–278.
5. Burges C. J. C. *From RankNet to LambdaMART to LambdaMART: An Overview*. URL: https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/ (accessed 07.04.2023).
6. Grira N., Crucianu M., Boujemaa N. *Unsupervised and Semi-supervised Clustering: a Brief Survey*. URL: http://cedric.cnam.fr/~crucianm/src/BriefSurveyClustering.pdf (accessed 07.04.2023)
7. Vapnik V. N. *Statistical Learning Theory*. New York: Wiley, 1998. 768 p.
8. Rahangdale A. U., Raut, S. Clustering Based Transductive Semi-supervised Learning for Learning-to-Rank. *International Journal of Pattern Recognition and Artificial Intelligence*. 2019. Vol. 33, no. 12. P. 1951007:1–1951007:27. DOI: 10.1142/s0218001419510078.
9. Amini M., Truong T., Goutte C. A Boosting Algorithm for Learning Bipartite Ranking Functions with Partially Labeled Data. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008*. 2008. P. 99–106.

10. Szummer M., Yilmaz E. Semi-supervised Learning to Rank with Preference Regularization. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. 2011. P. 269–278.
11. Weston J., Leslie C., Ie E., Zhou D., Elisseeff A., Noble W. S. Semi-supervised protein classification using cluster kernels. *Bioinformatics*. 2005. Vol. 21, no. 15. P. 3241–3247.
12. Valizadegan H., Jin R., Zhang R., and Mao J. Learning to Rank by Optimizing NDCG Measure. *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*. 2009. P. 1883–1891.

References (transliterated)

1. Burges C. J. C., Svore K. M., Wu Q., Gao J. *Ranking, boosting and model adaptation*. Available at: https://www.microsoft.com/en-us/research/publication/ranking-boosting-and-model-adaptation/ (accessed 07.04.2023).
2. Chang Y., Chapelle O. Yahoo! Learning to Rank Challenge Overview. *JMLR: Workshop and Conference Proceedings 14*. 2011, pp. 1–24.
3. Xu H., Li H. AdaRank: A Boosting Algorithm for Information Retrieval. *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2007, pp. 391–398.
4. Yilmaz E., Szummer M. Semi-supervised Learning to Rank withPreference Regularization. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. 2011, pp. 269–278.
5. Burges C. J. C. *From RankNet to LambdaMART to LambdaMART: An Overview*. Available at: https://www.microsoft.com/en-us/research/publication/from-ranknet-to-lambdarank-to-lambdamart-an-overview/ (accessed 07.04.2023).
6. Grira N., Crucianu M., Boujemaa N. *Unsupervised and Semi-supervised Clustering: a Brief Survey*. Available at: http://cedric.cnam.fr/~crucianm/src/BriefSurveyClustering.pdf (accessed 07.04.2023)
7. Vapnik V. N. *Statistical Learning Theory*. New York, Wiley, 1998. 768 p.
8. Rahangdale A. U., Raut, S. Clustering Based Transductive Semi-supervised Learning for Learning-to-Rank. *International Journal of Pattern Recognition and Artificial Intelligence*. 2019, vol. 33, no. 12, pp. 1951007:1–1951007:27. DOI: 10.1142/s0218001419510078.
9. Amini M., Truong T., Goutte C. A Boosting Algorithm for Learning Bipartite Ranking Functions with Partially Labeled Data. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008*. 2008, pp. 99–106.
10. Szummer M., Yilmaz E. Semi-supervised Learning to Rank with Preference Regularization. *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*. 2011, pp. 269–278.
11. Weston J., Leslie C., Ie E., Zhou D., Elisseeff A., Noble W. S. Semi-supervised protein classification using cluster kernels. *Bioinformatics*. 2005, vol. 21, no. 15, pp. 3241–3247.
12. Valizadegan H., Jin R., Zhang R., and Mao J. Learning to Rank by Optimizing NDCG Measure. *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009*. 2009, pp. 1883–1891.

Received 11.04.2023

УДК 519.2

К. С. ЯМКОВИЙ, Національний технічний університет «Харківський політехнічний інститут», асистент кафедри комп'ютерної математики і аналізу даних, м. Харків, Україна, e-mail: klym.yamkovyi@cs.khpi.edu.ua; ORCID: https://orcid.org/0000-0001-9512-4150

АДАПТАЦІЯ МОДЕЛІ LAMBDAMART ДО НАПІВКОНТРОЛЬОВАНОГО НАВЧАННЯ

Проблема пошуку інформації дуже поширена в епоху Інтернету та великих даних. Зазвичай існують величезні колекції документів, і лише кілька відсотків з них є актуальними. У цьому налаштуванні методи перебору неефективні. Пошукові системи допомагають вирішити цю проблему оптимальним способом. Більшість пошукових двигунів засновані на методах навчання ранжируванню. тобто спочатку алгоритм видає оцінки документам на основі їх ознак, а потім сортує їх відповідно до цих оцінок у відповідному порядку. Існує багато алгоритмів у цій галузі, але одним із найшвидших і надійних алгоритмів ранжирування є LambdaMART. Цей алгоритм заснований на бустінгу та розроблений лише для навчання з вчителем, де кожен документ у колекції має ранг, оцінений експертом. Але зазвичай у цій сфері колекції містять масу документів, і їх анотація вимагає багато ресурсів, як-от часу, грошей, експертів тощо. У цьому випадку напівавтоматичне навчання є потужним підходом. Напівавтоматичне навчання – це підхід у машинному навчанні, який поєднує невелику кількість позначених даних із великою кількістю не позначених даних під час навчання. Дані без міток у поєднанні з невеликою кількістю мічених даних можуть значно підвищити точність навчання. Ця стаття присвячена адаптації LambdaMART до напівавтоматичного навчання. Автор пропонує додавати різні ваги для розмічених і нерозмічених документів під час процедури навчання, щоб досягти більшу надійність і точність. Запропонований алгоритм було реалізовано з використанням мови програмування Python та фреймворку lightGBM, який уже має реалізацію LambdaMART для навчання з вчителем. Для цілей тестування було використано кілька наборів даних. Один синтетичний 2D-набір даних для візуального пояснення результатів і два реальних набори даних MSLR-WEB10K від Microsoft і Yahoo LTRC.

Ключові слова: навчання ранжуванню, пошук інформації, напівавтоматичне навчання, попарне ранжування, LambdaMART, псевдомаркування, NDCG.

Повне ім'я автора / Author's full name

Ямковий Клим Сергійович, Yamkovyi Klym Serhiyovych