Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original software publication

RankEval: Evaluation and investigation of ranking models

Claudio Lucchese [a,1], Cristina Ioana Muntean [b,*,1], Franco Maria Nardini [b,1], Raffaele Perego [b,1], Salvatore Trani [b,1]

[a] *Ca' Foscari University of Venice, Italy*
[b] *ISTI-CNR, Pisa, Italy*

ARTICLE INFO

Article history:
Received 30 April 2020
Received in revised form 15 October 2020
Accepted 15 October 2020

Keywords:
Learning to Rank
Evaluation
Analysis

ABSTRACT

RankEval is a Python open-source tool for the *analysis and evaluation* of ranking models based on ensembles of decision trees. Learning-to-Rank (LtR) approaches that generate tree-ensembles are considered the most effective solution for difficult ranking tasks and several impactful LtR libraries have been developed aimed at improving ranking quality and training efficiency. However, these libraries are not very helpful in terms of hyper-parameters tuning and in-depth analysis of the learned models, and even the implementation of most popular Information Retrieval (IR) metrics differ among them, thus making difficult to compare different models. RankEval overcomes these limitations by providing a unified environment where to perform an easy, comprehensive inspection and assessment of ranking models trained using different machine learning libraries. The tool focuses on ensuring efficiency, flexibility and extensibility and is fully interoperable with most popular LtR libraries.

Code metadata

Current code version	Latest stable release, v0.8.2
Permanent link to repository	https://github.com/ElsevierSoftwareX/SOFTX_2020_188
Code Ocean compute capsule	none
Legal code license	Mozilla Public License 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Python 2, Python 3, Cython
Compilation requirements, operating environments, and dependencies	Compilation requirements: C compiler required (GCC, Clang) to build from source code. Operating Environments: Windows, MacOS, Linux. Python requirements: numpy, scipy, matplotlib, cython. Python dependencies: six, pandas, xarray, seaborn, (graphviz).
Link to documentation	http://rankeval.isti.cnr.it/docs/
Support email for questions	rankeval@isti.cnr.it

1. Motivation and significance

The widespread adoption of complex machine-learned ranking models has generated a demand for novel tools aimed at inspecting and analyzing them so to enforce ranking trust and to understand how they can be fine-tuned and improved.

In this paper we present RankEval, an open-source tool for the *analysis and assessment* of ranking models based on additive *ensembles of regression trees*. These ensembles are generated by efficient Learning to Rank (LtR) algorithms that leverage large amounts of training data to iteratively learn simple decision trees incrementally optimizing the given loss function. A training dataset consists in a collection of query-document pairs where each example is annotated with a relevance label. The goal

[*] Corresponding author.
E-mail addresses: claudio.lucchese@unive.it (C. Lucchese), cristina.muntean@isti.cnr.it (C.I. Muntean), francomaria.nardini@isti.cnr.it (F.M. Nardini), raffaele.perego@isti.cnr.it (R. Perego), salvatore.trani@isti.cnr.it (S. Trani).

[1] All authors have contributed equally to this work.

https://doi.org/10.1016/j.softx.2020.100614
2352-7110/

of the LtR algorithm is learning a document scoring function that approximates the ideal ranking induced by relevance labels observed in the training samples.

Nowadays, tree-based models are considered the most competitive LtR solution for "difficult" ranking tasks [1], including the ranking of web search results or the ranking of items/posts in e-Commerce platforms and online social networks [2]. In such large-scale production systems, complex ensembles with thousands of trees are commonly deployed to achieve high quality results that meets user requirements. In these contexts, the models based on Gradient Boosted Regression Trees (GBRTs) [3] were adopted by the majority of the winning solutions of both the Kaggle and KDD Cup 2015 competitions. Among the most popular open-source implementations of GBRTs we cite LightGBM [4], XGBoost [5] and RankLib.[2] We also contributed to the open-source implementations with our QuickRank [6] framework. The need for RankEval stems from the fact that all these libraries offer very limited support for fine-tuning the models and evaluating their performance under different aspects. Moreover, engineers suffer from the lack of a common open-source framework for comprehensively evaluating LtR models trained with different tools. The evaluation of a ranking solution in isolation is not trivial. For example, the implementations available for most common Information Retrieval (IR) evaluation metrics differ between each other: for what the Normalized Discounted Cumulative Gain (NDCG) metric [7] is concerned, a query with no relevant results is assigned a score equal to −1 in LightGBM, equal to 0 in QuickRank and RankLib, and equal to 0.5 in the Yahoo Learning to Rank Challenge.

By providing a single comprehensive framework for tuning, evaluating and comparing GBRT LtR models, possibly learned with different gradient boosting libraries, RankEval aims to fill the above gap. While the open-source libraries mentioned above provide efficient and effective implementations of the state-of-the-art LtR algorithms, RankEval offers an easy-to-use framework that is interoperable across the most popular libraries, and to engineers the possibility to deeply understand the models and tune their effectiveness. RankEval is implemented as a lightweight Python library and can be used within a Jupyter notebook[3] to take advantage of its recently introduced interactive visualization capabilities. Finally, RankEval can exploit a high performance infrastructure to speedup the most demanding analytics functionalities.

1.1. Major improvements in the current release

A demo paper presenting the preliminary implementation of RankEval can be found in [8]. Since first published in 2017, the RankEval framework has been constantly maintained and updated with several releases of the code. Improvements introduced over time and available in the current release include:

- **PyPI availability.** Starting from 0.61 release version, RankEval is available from the Python Package Index (PyPI). You can easily install the tool by running the command:

  ```
  pip install rankeval
  ```

  Additionally, the code is packaged into several binary wheels according to the most adopted platforms (UNIX and MacOS, python versions 2.7, 3.4, 3.5, 3.6 and 3.7). These binary packages are provided through PyPi thus allowing for a fast and simple installation without the need to compile low-level code from scratch (some functionalities are written using cython for efficiency reason).
- **Interactive Notebooks.** Starting from 0.61 release version, several Jupyter notebooks are delivered with the source code as to provide to the users an interactive introduction to the main functionalities implemented in RankEval, and a user-friendly how-to guide on the usage of the library and on an in-depth understanding of the code structure.
- **Support for CatBoost and Jforests.** Starting from 0.7 release version, RankEval provides two new model proxies extending its interoperability capabilities to two additional LtR libraries.
- **Python 3 compatibility.** of RankEval. Starting from the 0.7 release version, RankEval fully supports Python 3 functionalities and code conventions.
- **New analysis and visualization functionalities.** Starting from the 0.7 release version, several new functionalities have been introduced for analyzing and visualizing the shape of the trees in the ensemble, with the possibility to personalize the plots in several ways. This feature permits a better understanding of the output predicted by each tree, as well as to have a clear picture of the trees' shape (impacting, for example, the scoring time [2]). See Section 2.2 for additional information on tree-wise and topological analysis functionalities.
- **Continuous Integration.** Modern software development often uses Agile principles which focus on the customer's requirement for continuous delivery of new functionality. In light of this vision, starting from the 0.81 release version, we introduced Continuous integration (CI) using Travis,[4] adding automated tests and controls to the RankEval pipeline in order to save programming and debugging time and increase the reliability to the software. Each commit/release/PR is now tested before being merged into the repository. Even releasing the library on PyPi has been automatized: each tagged commit on the master will now activate the deployment actions that will test again the code, package and release it for the various supported platforms.
- **Bug fixes.** Since the first release of the library, many issues have been solved, with the help of the community that contributed by signaling them by email or using the issue panel on GitHub. Similarly, many of the features listed above (and others as well, e.g., the K-Fold splitting of a Learning-to-Rank dataset on a per-query basis) were requested by the users and implemented in the library as to provide support to the user base.

The GitHub repository of RankEval is currently watched by 13 developers and has 65 stars, and the community engages with the software by adding issues and pull requests.

2. Software description

2.1. Software architecture

RankEval is implemented in Python, and its architecture is depicted in Fig. 1. Green boxes highlight major software components, blue boxes are Python modules, while the yellow box provides purely static functionalities. The red boxes highlight the functionalities implemented in Cython for efficiency reasons. The software is logically split in two layers: (i) the *core components* which include the base classes (Dataset, Model, Metric) and some utility methods (e.g., the static Proxy classes for loading models from different libraries, or the Cython implementation of the scoring process); (ii) the *high level components*, providing the implementation of several analysis tools with integrated visualization capabilities that can be easily exploited within Jupyter notebooks.

[2] https://github.com/codelibs/ranklib

[3] http://jupyter.org/

[4] https://travis-ci.org

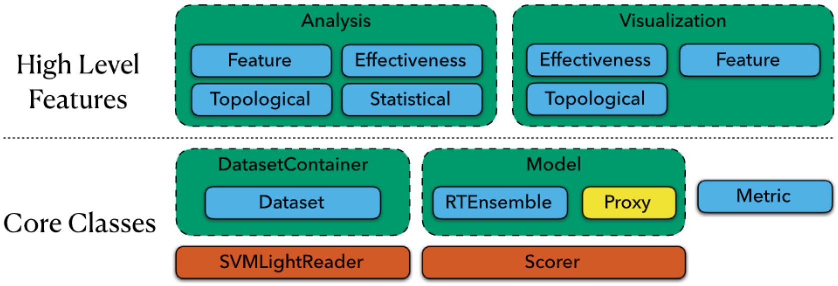

Fig. 1. `RankEval` software architecture. (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

2.2. Software functionalities

`RankEval` functionalities can be grouped into five categories: *effectiveness analysis*, *feature analysis*, *structural analysis*, *topological analysis*, *interoperability among GBRT libraries*. `RankEval` supports the analysis of multiple ranking models and multiple datasets simultaneously, so as to permit a clear and direct comparative evaluation of different models.

Effectiveness Analysis This group of functionalities aims at evaluating models performance in terms of accuracy.

- **Model performance** implements several ranking metrics for model evaluation. The metrics include: Precision, Precision-Max, Mean average precision (MAP), Ranked biased Precision (RBP), Recall, Expected Reciprocal Rank (ERR), Mean Reciprocal Rank (MRR), Mean squared error (MSE), Root mean squared error (RMSE), Discounted cumulative gain (DCG), Normalized discounted cumulative gain (NDCG), Kendall's Tau, Spearman's Rho and Pfound. This large offer of different evaluation metrics permits the user to choose the right metrics for the specific retrieval task by avoiding common IR evaluation mistakes (see for example Fuhr's[5] and Sakai's[6] opinion papers).
- **Tree-wise performance** allows to evaluate different effectiveness metrics incrementally by varying the number of trees in the model, i.e., top-k trees. This analysis allows to investigate efficiency/effectiveness trade-offs [9] by scoring a large model on different datasets by varying the number of trees considered. In Fig. 2 (left) we show the incremental per-tree break-down for NDCG@10 on the three dataset splits: Train, Validation and Test. It shows that increasing the number of trees on the Train set improves performance in terms of NDCG, while on the other two splits the performance tends to flatten after approximately 400 trees. This indicates that the model starts to overfit from 400 trees on. In Fig. 2 (right) the analysis shows the average per-tree contribution to the final accuracy score, regardless of the tree ordering in the forest. This allows, for example, to rank trees by importance. The figure shows that the trees in the beginning of the forest (0–200) contribute more and should come first in the model compared to the others. This analysis can give an indication of which trees can be pruned due to their limited contribution if the developer is looking for a smaller, more efficient model.
- **Query-wise performance** computes the metric scores *cumulative distribution* over a given query set.
- **Document graded-relevance performance** similarly as above, it provides the cumulative performance distribution with a breakdown over the available relevance labels.
- **Query class performance** allows to investigate the quality performance breakdown over different query classes (e.g., navigational, informational, transactional, etc.).
- **Rank confusion matrix** provides a rank-oriented confusion matrix, where for any given relevance label l_i, the number of documents with a predicted score smaller than documents with label l_j is given.

Feature analysis `RankEval` makes available several feature analysis tools that allow the investigation of: (i) how features are used by a trained model and (ii) their impact on the final accuracy of the model. While `LtR` libraries allow to perform a similar analysis only in terms of a single metric, i.e., root mean squared error (RMSE), `RankEval` allows for an improved analysis across all the ranking metrics supported.

- **Feature importance** provides an evaluation of the contribution of each feature to the global error reduction (i) on the whole ensemble or (ii) incrementally, by considering each estimator of the forest in isolation.
- **Feature usage statistics** provides further details on the use of features, i.e., advanced statistics on where and how features appear in the forest, e.g., the number of nodes of the forest using a given feature, their average depth in the model, the distribution of the thresholds used by the model, etc.

Statistical analysis provides the following novel functionalities:

- **Statistical significance** allows the evaluation of when and how a model provides statistically significant performance w.r.t a given competitor. Several state-of-the-art significance tests are implemented, including the "randomization test" [10].
- **Bias vs. variance decomposition** investigates the strengths and weaknesses of a given `LtR` algorithm with the help of the bias vs. variance analysis.

Topological analysis provides the topological properties of a `LtR` model. The analysis provided by `RankEval` covers min/max/avg depth of trees, and the investigation of the shape of the trees. In Fig. 3 (left) the XGBoost tends to build ensemble models that are on average more balanced (i.e., with a similar number of nodes on the left and right sides) compared to the QuickRank framework in Fig. 3 (right), which on average tends to build very skewed models with long chain of nodes on the left branch.

Interoperability `RankEval` aims at providing a unified and user-friendly framework for the analysis of `LtR` solutions. `RankEval` is able to analyze models stored in different formats used by the five most popular libraries for learning ensembles of trees, namely `XGBoost`, `LightGBM`, `scikit-learn`, `CatBoost`, `Jforests`, `QuickRank`, and `RankLib`. `RankEval` also provides format conversion tools to easily move from one format to another one.

[5] http://sigir.org/wp-content/uploads/2018/01/p032.pdf
[6] http://www.sigir.org/wp-content/uploads/2020/06/p14.pdf

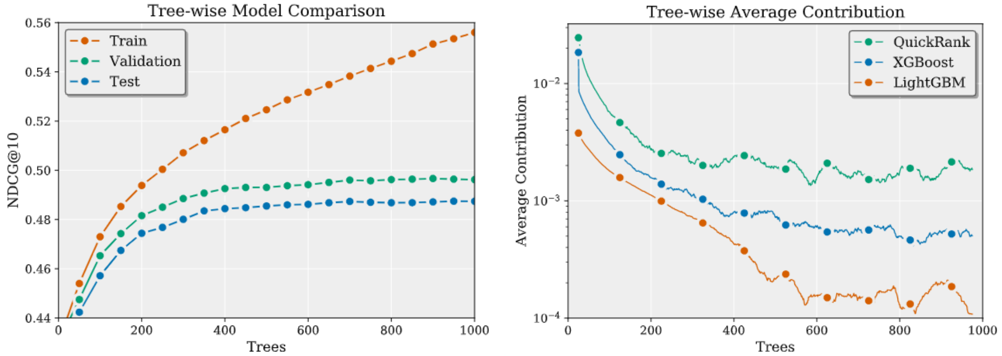

Fig. 2. Tree-Wise model comparison (left) and average contribution of each tree to the final score (right).

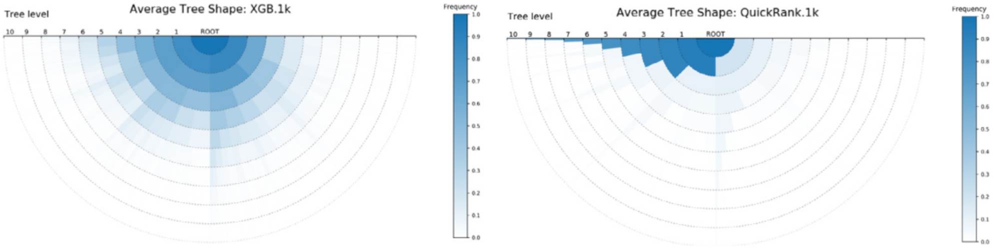

Fig. 3. The topological analysis tool investigates the shape of tree-based ensembles in two different models: XGB.1k (left) and QuickRank.1k (right).

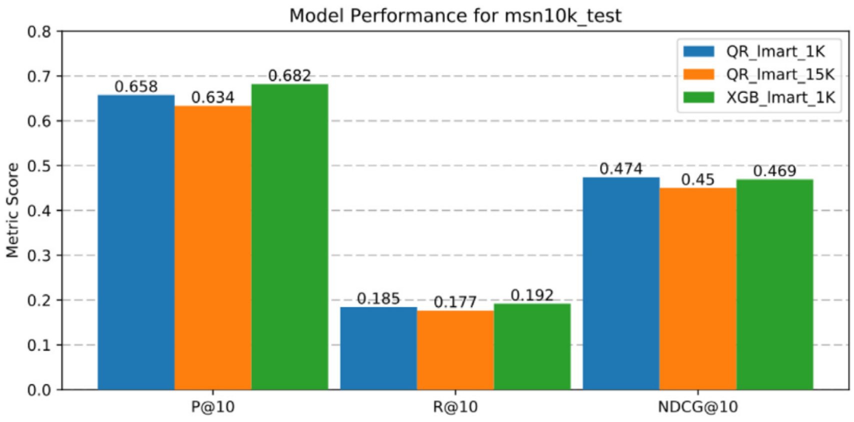

Fig. 4. Model performance evaluation generated with the code snippet above..

3. Illustrative examples

RankEval is flexible as it can be used in several different ways. For example, it can be used as a Python module and, with the help of the shell, python scripts invoking the desired functionalities can be run. The other way to use it is by importing it in a Jupyter notebook. Interested readers can test all the functionalities presented in Section 2.2. RankEval offers a comprehensive way of visualizing the results as report tables (with the support of Pandas dataframes) and plots (with the support of Matplotlib). The results can be stored in files or visualized inline, when running RankEval in a Python Jupyter notebook.

Fig. 4 reports a code snippet showing how easy, user-friendly and customizable is the evaluation process offered by RankEval. The example illustrates a simple code snippet for measuring model performance between three pre-trained models, two trained with QuickRank (QR_lmart_1k, QR_lmart_15k) and one with XGBoost (XGB_lmar_1k). The effectiveness of the models is assessed based on three different metrics, i.e. *P*@10, *R*@10 and *NDCG*@10. Several other examples of code snippets and use of functionalities can be found in the Notebook examples of the RankEval GitHub.[7]

```
from rankeval.model import RTEnsemble
from rankeval.metrics import Precision, Recall, NDCG
from rankeval.analysis.effectiveness import model_performance
from rankeval.visualization.effectiveness import plot_model_performance

msn_qr_lm_1Ktrees = RTEnsemble("msn_qr_lmart_1Ktrees.xml", name="QR_lmart_1K",
format="QuickRank")
msn_qr_lm_15Ktrees = RTEnsemble("msn_qr_lmart_15Ktrees.xml", name="QR_lmart_15K",
format="QuickRank")
msn_xgb_lm_1Ktrees = RTEnsemble("msn_xgb_lmart_1Ktrees", name="XGB_lmart_1K",
format="XGBoost")

msn_model_perf = model_performance(datasets=[msn_test],
                        models=[msn_qr_lm_1Ktrees, msn_qr_lm_15Ktrees, msn_xgb_lm_1Ktrees],
                        metrics=[Precision(cutoff=10), Recall(cutoff=10), NDCG(cutoff=10)])

fig = plot_model_performance(msn_model_perf, compare="models", show_values=True)
```

4. Impact

RankEval brings a significant added value to both the scientific community and IR engineers and practitioners. RankEval

[7] https://github.com/hpclab/rankeval/tree/master/notebooks

can positively impact both communities since it allows a fair comparison and analysis among state-of-the-art LtR methods. It unifies the evaluation of models trained with several different tools (e.g. XGBoost, LightGBM, scikit-learn, QuickRank, RankLib) in a single framework offering objective and straightforward metrics. What before was done in a very minimal and non-unified way by the LtR libraries themselves, RankEval succeeds at putting together in a comprehensive framework of model analysis and evaluation, with tools for integration and interoperability. Indeed, RankEval was successfully used in different LtR tutorials presented at major conferences [11–13] in which the audience could test out and compare their models, trained with tools of their choice. Such a "high value, low cost" tool can speed up the process of model tuning by supporting also feature analysis and engineering. Lastly, RankEval is designed with the goal of speeding up the process of investigating strengths and weaknesses of given LtR models, and to support users in achieving an optimal model fine-tuning.

5. Conclusions

RankEval is a unique tool focusing on the evaluation of ensemble-based LtR models. It gathers and implements, in a unifying framework, all the evaluation methods and metrics commonly used for measuring the effectiveness of ranking models, by also adding new functionalities and features to their objective evaluation, interpretation and in-depth analysis.

The tool is designed to give a comprehensive view on model performance but also to improve reporting. It provides a topological view on the model structure, an analysis on feature importance, statistical significance tests and several performance indicators on LtR datasets, making it easier for users to understand where the model fails to achieve maximum accuracy and highlighting problematic issues.

RankEval is an open source project available on GitHub under the Mozilla Public License 2.0. We acknowledge all the contributors who participated in the project and added/improved the RankEval functionalities.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

Work partially supported by the BIGDATAGRAPES project funded by the EU Horizon 2020 research and innovation programme under grant agreement No. 780751, and by the OK-INSAID project funded by the Italian Ministry of Education and Research (MIUR) under grant agreement No. ARS01_00917.

References

[1] Gulin A, Kuralenok I, Pavlov D. Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank. In: Yahoo! learning to rank challenge. 2011, p. 63–76. http://dx.doi.org/10.5555/3045754.3045761.

[2] Dato D, Lucchese C, Nardini FM, Orlando S, Perego R, Tonellotto N, et al. Fast ranking with additive ensembles of oblivious and non-oblivious regression trees. ACM Trans Inf Syst 2016;35(2):15:1–31. http://dx.doi.org/10.1145/2987380.

[3] Friedman JH. Greedy function approximation: A gradient boosting machine. Ann Statist 2000;29:1189–232, URL www.jstor.org/stable/2699986.

[4] Ke G, Meng Q, Finley T, Wang T, Chen W, Ma W, et al. LightGBM: A highly efficient gradient boosting decision tree. In: Guyon I, Luxburg UV, Bengio S, Wallach H, Fergus R, Vishwanathan S, Garnett R, editors. Advances in neural information processing systems, vol. 30. Curran Associates, Inc.; 2017, p. 3146–54. http://dx.doi.org/10.5555/3294996.3295074.

[5] Chen T, Guestrin C. XGBoost: A scalable tree boosting system. In: Proceedings of the 22Nd ACM SIGKDD international conference on knowledge discovery and data mining. New York, NY, USA: ACM; 2016, p. 785–94. http://dx.doi.org/10.1145/2939672.2939785.

[6] Capannini G, Dato D, Lucchese C, Mori M, Nardini FM, Orlando S, et al. Quickrank: a C++ suite of learning to rank algorithms. In: 6th Italian information retrieval workshop. 2015, URL http://ceur-ws.org/Vol-1404/paper_9.pdf.

[7] Järvelin K, Kekäläinen J. Cumulated gain-based evaluation of IR techniques. ACM Trans Inf Syst 2002;20(4):422–46. http://dx.doi.org/10.1145/582415.582418.

[8] Lucchese C, Muntean CI, Nardini FM, Perego R, Trani S. Rankeval: An evaluation and analysis framework for learning-to-rank solutions. In: Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval. New York, NY, USA: Association for Computing Machinery; 2017, p. 1281–4. http://dx.doi.org/10.1145/3077136.3084140.

[9] Capannini G, Lucchese C, Nardini FM, Orlando S, Perego R, Tonellotto N. Quality versus efficiency in document scoring with learning-to-rank models. Inf Process Manag 2016;52(6):1161–77. http://dx.doi.org/10.1016/j.ipm.2016.05.004.

[10] Smucker MD, Allan J, Carterette B. A comparison of statistical significance tests for information retrieval evaluation. In: Proceedings of the sixteenth ACM conference on conference on information and knowledge management. New York, NY, USA: ACM; 2007, p. 623–32. http://dx.doi.org/10.1145/1321440.1321528.

[11] Lucchese C, Nardini FM. Efficiency/effectiveness trade-offs in learning to rank. In: Machine learning and knowledge discovery in databases - european conference. 2018, http://dx.doi.org/10.1145/3121050.312110.

[12] Lucchese C, Nardini FM, Pasumarthi RK, Bruch S, Bendersky M, Wang X, et al. Learning to rank in theory and practice: from gradient boosting to neural networks and unbiased learning. New York, NY, USA: Association for Computing Machinery; 2019, p. 1419–20. http://dx.doi.org/10.1145/3331184.3334824.

[13] Lucchese C, Nardini FM. Efficiency/effectiveness trade-offs in learning to rank. In: Proceedings of the ACM SIGIR international conference on theory of information retrieval. New York, NY, USA: Association for Computing Machinery; 2017, p. 329–30. http://dx.doi.org/10.1145/3121050.3121109.