A Brief Tutorial on Supervised Learning to Rank

Philipp Hager, Maarten de Rijke

May 15, 2023

University of Amsterdam
p.k.hager@uva.nl, m.derijke@uva.nl

Who are we?

Philipp Hager
PhD student
UvA & Mercury Lab

Maarten de Rijke
Dist. University Professor
UvA

Acknowledgements

Neural Networks for Information Retrieval [32]
SIGIR 2017 Tutorial by Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, Bhaskar Mitra

Unbiased Learning to Rank: Counterfactual and Online Approaches [44]
WWW 2020 Tutorial by Harrie Oosterhuis, Rolf Jagerman, Maarten de Rijke

Lectures on Learning to Rank
Information Retrieval I, UvA by Harrie Oosterhuis, Ilya Markov, Andrew Yates

Motivation

Google

attractions in Amsterdam

TripAdvisor
https://www.tripadvisor.com › Attractions-g188590-A...

THE 15 BEST Things to Do in Amsterdam

Top **Attractions in Amsterdam** · 1. Anne Frank House · 2. Van Gogh Museum · 3. Rijksmuseum · 4. Vondelpark · 5. The Jordaan · 6. Centraal Station · 7. Heineken ...

Attractions: 3,115 Attraction Photos: 309,790
Attraction Reviews: 645,083

iamsterdam.com
https://www.iamsterdam.com › see-and-do › attraction...

Attractions and sights | I amsterdam

Most popular **attractions** · Heineken Experience · ARTIS · Koninklijk Paleis (Royal Palace) · Anne Frank House · **Amsterdam** Canal Cruise - 100 highlights · Johan Cruijff ...

Free things to do in Amsterdam · Rembrandts Amsterdam... · This is holland

PlanetWare
https://www.planetware.com › amsterdam-nl-nh-amst

24 Top-Rated Tourist Attractions in Amsterdam

Signals in Web Search

Textual Signals:

- **Query content**: text
- **Document content**: title, page content

How well does the query text match the document text? [13]

- BM-25
- TF-IDF / vector space models
- Language modeling

Google

attractions near me

TripAdvisor
https://www.tripadvisor.com › Attractions-g188590-A...

THE 15 BEST Things to Do in Amsterdam

Top **Attractions** in Amsterdam · 1. Anne Frank House · 2. Van Gogh Museum · 3. Rijksmuseum · 4. Vondelpark · 5. The Jordaan · 6. Centraal Station · 7. Heineken ...

Attractions: 3,115   Attraction Photos: 309,790
Attraction Reviews: 645,083

Amsterdam Attractions · Things to do near Rijksmuseum · Anne Frank House

iamsterdam.com
https://www.iamsterdam.com › see-and-do › attraction...

Attractions and sights | I amsterdam

Most popular **attractions** · Heineken Experience · ARTIS · Koninklijk Paleis (Royal Palace) · Anne Frank House · Amsterdam Canal Cruise - 100 highlights · Johan Cruijff ...

Free things to do in Amsterdam · Amsterdam's best hidden gems · This is holland

PlanetWare
https://www.planetware.com › amsterdam-nl-nh-amst

24 Top-Rated Tourist Attractions in Amsterdam

Signals in Web Search

Signals beyond text:

- **Query**: type, language
- **Document**: urls, images
- **User context**: location, date, device, search history
- **Metadata**: popularity, recency, page quality, spam, adult content, ...
- **Other stakeholders**: advertisers, auctions, content creators, ...

Search engines use many features:

- **Airbnb** [19]: $>$ 195 features
- **Bing** [50]: $>$ 136 features
- **Istella** [14]: $>$ 220 features
- **Yahoo** [6]: $>$ 700 features

How do we combine all of these signals?

Learning to Rank

Learning to Rank (LTR) is

"...a task to automatically construct a ranking model using training data, such that the model can sort new objects according to their degrees of relevance, preference, or importance." – Liu [37]

Notation

Representation

- For a given query, we want to rank a collection of items: $d \in D$
- Each query-item pair is represented by a feature vector: $\vec{x}_{q,d} \in \mathbb{R}^m$
- Each query-item pair is judged for relevance, typically: $y_{q,d} \in [0, 4]$

A **ranking model** $f : \vec{x} \to \mathbb{R}$ scores each document to optimize the order of items when sorting descendingly by $f(\vec{x}_{q,d}) = s_{q,d}$.

How to measure the quality of a ranking model?

Evaluation

Reciprocal Rank

Reciprocal of the rank of the first relevant item after sorting by our scores s:

$$\mathrm{RR} = \frac{1}{\mathrm{rank}_i}$$

with rank_i being the rank of the first relevant item in our list.

Assumption: Only the position of the first item matters (e.g., in navigational search).

Discounted Cumulative Gain

$$\mathrm{DCG} = \frac{1}{n}\sum_{i=1}^{n} \frac{\mathrm{gain}(y_i)}{\mathrm{discount}(i)} = \frac{1}{n}\sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log(i+1)}$$

Assumptions: Highly relevant items are more useful than somewhat relevant items. Relevant items ranked lower are less useful since the user is less likely to see them.

Pointwise Methods

Pointwise LTR

Predict the relevance of each document from its features

Regression: Relevance as a real-valued score [11, 17]

$$\mathcal{L}_{mse} = \frac{1}{n} \sum_{i=1}^{n} (y_i - s_i)^2 \tag{1}$$

Classification: Relevance as unordered categories [10, 42]

$$\mathcal{L}_{ce} = -\frac{1}{n} \sum_{i=1}^{n} y_i \cdot P(y_i \mid x_i) \tag{2}$$

e.g., with $P(y \mid x) = \text{softmax}(s)$

Ordinal regression: Relevance as ordered categories [12, 54]

Pointwise LTR: pros and cons

Benefits

- Easy to adopt any regression or classification model
- Calibrated output scores

Challenges (solvable)

- Class imbalance due to few relevant documents
- Requires normalization since feature distribution can differ greatly per query

Limitations

- Predicted item scores are independent of each other
- **A lower loss does not necessarily improve ranking metrics**

Pointwise LTR: A lower loss does not imply a better ranking

Relevance labels	Predicted scores
1	0.6
0	0.5
0	0.5
0	0.5
0	0.5

What is the loss?

$$\mathcal{L}_{mse} = \frac{1}{n}\sum_{i=1}^{n}(y_i - s_i)^2$$

Pointwise LTR: A lower loss does not imply a better ranking

Relevance labels	Predicted scores
1	0.6
0	0.5
0	0.5
0	0.5
0	0.5

Loss $\mathcal{L}_{mse} = 1.16$

MRR $= 1$, nDCG $= 1$

Pointwise LTR: A lower loss does not imply a better ranking

Relevance labels	Predicted scores
1	0.2
0	0.2
0	0.2
0	0.2
0	0.1

Loss $\mathcal{L}_{mse} = 0.97$

MRR $= 0.2$, nDCG $= 0.39$

Pairwise Methods

Pairwise LTR

Observation: Ranking requires only relative relevance levels: $s_i > s_j$ if $y_i > y_j$.

Pairwise loss functions generally take the following (unnormalized) form [9]:

$$\mathcal{L}_{pairwise}(s, y) = \sum_{y_i > y_j} \phi(s_i - s_j)$$

with ϕ being the:

- **Hinge** function in RankingSVM [22, 28]: $\phi(z) = \max(0, 1 - z)$
- **Exponential** function in RankBoost [16]: $\phi(z) = e^{-z}$
- **Logistic** function in RankNet [4]: $\phi(z) = \log(1 + e^{-z})$

RankNet

RankNet

Introduced by Burges et al. [4] in 2005 to train neural ranking models.
Popular in industry applications and won the ICML 2015 test of time award.[1]

1. The probability of the event that item d_i should be ranked over d_j is defined by:

$$P(d_i \succ d_j) = P_{ij} = \frac{1}{1 + e^{-\gamma(s_i - s_j)}}$$

$$P(d_i \prec d_j) = P_{ji} = 1 - P_{ij}$$

The desired probabilities when $y_i > y_j$ are $\bar{P}_{ij} = 1$ and $\bar{P}_{ji} = 0$

[1] `https://icml.cc/2015/index.html%3Fp=51.html`

RankNet

2. Compute the cross-entropy loss between P_{ij} and $\bar{P}_{ij}$:

$$\begin{aligned}\mathcal{L}_{RankNet} &= -\bar{P}_{ij}\log P_{ij} - (1-\bar{P}_{ij})\log(1-P_{ij}) \\ &= -\bar{P}_{ij}\log P_{ij} - \bar{P}_{ji}\log P_{ji}\end{aligned}$$

3. Given its symmetry, we only have to compute the loss over pairs where $d_i \succ d_j$:

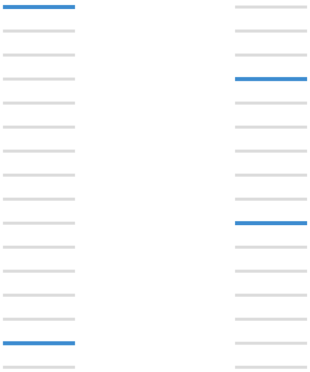
$$\begin{aligned}\mathcal{L}_{RankNet}(s,y) &= \sum_{y_i > y_j} -\bar{P}_{ij}\log P_{ij} \\ &= \sum_{y_i > y_j} -\log\left(\frac{1}{1+e^{-\gamma(s_i - s_j)}}\right) \\ &= \sum_{y_i > y_j} \log\left(1+e^{-\gamma(s_i - s_j)}\right)\end{aligned}$$

Problems with this approach?

Usually implemented using virtual probabilities $\bar{P} \in \{0, 1\}$ and any differences in relevance labels is treated equally. Not very elegant, but works...

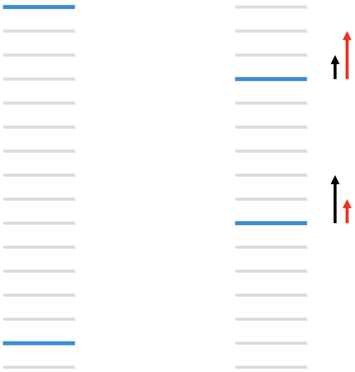
But are all item pairs equally important?

Pairwise LTR: Minimizing pairwise errors



Reducing pairwise errors from 13 (left) to 11 (right),
while top-heavy measures like MRR and nDCG degrade [4, Figure 1].

Pairwise LTR: Minimizing pairwise errors



The **black** arrows denote the RankNet gradients,
while what we'd arguably want are the **red** arrows [4, Figure 1].

Listwise Methods

LambdaRank

Motivation: Can we directly optimize IR metrics such as nDCG, Precision, and MRR?

Reciprocal Rank: Reciprocal of the rank of the first relevant item after sorting by our scores s:

$$\mathrm{RR} = \frac{1}{\mathrm{rank}_i}$$

Discounted Cumulative Gain:

$$\mathrm{DCG} = \frac{1}{n} \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log(i+1)}$$

Non-smooth and discontinuous

- Ranking metrics typically only depend on the rank of an item, not on its score
- Model scores change smoothly, the ranks of documents change abruptly

Listwise LTR

Non-differentiable

Ranking metrics rely on a sorting operation that is non-smooth and discontinuous w.r.t. to model parameters θ:

$$\frac{\partial \mathrm{RR}}{\partial \theta} = ???$$

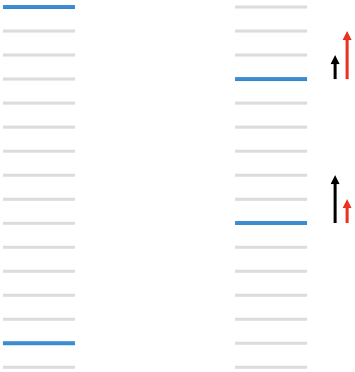
$$\frac{\partial \mathrm{DCG}}{\partial \theta} = ???$$

Thus, ranking metrics are either **flat** (with zero gradient) or **discontinuous**

Holy grail of LTR: Finding methods that (indirectly) optimize listwise IR metrics

LambdaRank

LambdaRank



Observations:

I. To train a model, we don't need the costs just the gradients (of the costs w.r.t model scores)

II. Gradients should be larger for pairs that have a greater impact on our metric

Idea: Scale the RankNet loss/gradients of an item pair based on the change in nDCG when swapping their positions.

1. Let's decompose nDCG into **gains** (relevance-based) and **discounts** (rank-based):

$$\begin{aligned}
\text{NDCG} &= \frac{1}{\text{maxDCG}} \sum_{i=1}^{n} \frac{2^{y_i} - 1}{\log(1+i)} = \sum_{i=1}^{n} \frac{G_i}{D_i} \\
G_i &= \frac{2^{y_i} - 1}{\text{maxDCG}} \\
D_i &= \log(1+i)
\end{aligned}$$

LambdaRank

2. Let $\Delta\text{NDCG}(i,j)$ be the absolute difference in nDCG when swapping d_i and d_j:

$$\Delta\text{NDCG}(i,j) = |G_i - G_j||\frac{1}{D_i} - \frac{1}{D_j}|$$

3. Finally, we weight the loss of each item pair by its difference in nDCG:

$$\mathcal{L}_{LambdaRank}(s,y) = \sum_{y_i > y_j} \Delta\text{NDCG}(i,j) \log\left(1 + e^{-\gamma(s_i - s_j)}\right)$$

The implementation of LambdaRank using multiple additive regression trees (MART) is called **LambdaMART**.

LambdaLoss

Empirical success

LambdaMART has been empirically shown to optimize for nDCG [4]

A late theoretical foundation

It was unclear if the iterative LambdaRank procedure converges
and how the underlying loss relates to nDCG [37, 58]

Wang et al. proved in 2018 that LambdaRank optimizes a lower bound on nDCG
and define it as a special case of their more general **LambdaLoss framework** [58]

ListNet and ListMLE

Listwise LTR

Motivation: Create a probabilistic model for ranking, which is differentiable.

The Plackett-Luce model assumes that the **probability of selecting** an item from a list depends on its value compared to the total item value in the list [38, 47]:

$$P(d_i) = \frac{\phi(s_i)}{\sum_{j=1}^{n} \phi(s_j)}$$

where $\phi(s_i)$ is an increasing and strictly positive function. With $\phi(s_i) = e^{s_i}$, this becomes a **softmax** function.

We can sample a ranking by repeatedly applying the Plackett-Luce model, removing the sampled item from the candidate list.

Example: What is the joint probability of the following ranking $\pi = (d_2, d_1, d_3)$?

$$P(\pi \mid s) = \frac{\phi(s_2)}{\phi(s_1) + \phi(s_2) + \phi(s_3)} \cdot \frac{\phi(s_1)}{\phi(s_1) + \phi(s_3)} \cdot \frac{\phi(s_3)}{\phi(s_3)}$$

Listwise LTR

ListNet [5]
Compute the probability distributions over all possible permutations based on ground-truth labels and predicted scores. Minimize the cross-entropy loss between these two distributions.

Since this is very costly, the authors only compute the top-k permutations (top-1).

ListMLE [61]
Compute the probability of the ideal permutation based on the ground truth. Can get difficult with categorical labels since multiple permutations are possible.

Hybrid methods

Scores of pairwise and listwise methods **are not calibrated**

Problematic for using $s_{d,q}$ in downstream applications (ad ranking, auctions, . . .)

Multi-objective loss [53]

$$\mathcal{L}_{MultiObjective} = \alpha \cdot \mathcal{L}_{ListNet} + (1 - \alpha) \cdot \mathcal{L}_{CrossEntropy}$$

Linear combination of listwise and pointwise loss has conflicting objectives [3]

Combined objective [3]

$$\mathcal{L}_{ListCE} = \frac{1}{\sum_{i=1}^{n} y_i} \cdot \sum_{i=1}^{n} y_i \cdot \frac{\sigma(s_i)}{\sum_{j=1}^{n} \sigma(s_j)}$$

ListCE aligns pointwise and listwise objectives for ad click prediction on YouTube

Summary

Pointwise

- Predict **relevance per item**
- Calibrated scores, but ignores ordering of items

Pairwise

- Predict **relative relevance in item pairs**
- Ignores that not all pairs have the same impact
- Notoriously uncalibrated scores (careful with downstream applications)

Listwise

- Optimize a list of items based on **non-differentiable ranking metrics**
- **Approximations** by heuristics, bounding, or probabilistic ranking methods
- **Check if the assumptions of your ranking metric matches your problem**

Industry Impact

Features used

Features used for **matching queries to products** come in multiple types. A small group of publications lists a large number of ranking features:

- Karmaker Santu et al. [29] list the use of 562 features for product search
- Ludewig and Jannach [39] list 518 features for product search
- Wu et al. [59] list dozens of features for product search (precise number withheld).

Three main types of feature used in these publications:

1. query features,
2. product features, and
3. query-product features.

Features used ii

Papers listed do not provide full details; papers plus discussions with authors have led to the lists below.

Query features Query features are features that are computed using the query only. They include:

- Query length
- Expected product category

Features used iii

Product features Product features are ranking features that are computed using product information.

- Overall product sales
- Total show count, click count, view count, and purchase count of each product
- Total distinct user count of the four types of behavior on each product
- Click-through rate (ctr), view rate and click value rate (cvr) of each product
- Rating
- Number of reviews
- Brand
- Price
- Session-based features: has this product been clicked before in this session?

Features used iv

Query-product features Query-product features are features that concern the relation between query and product. They include:

- Text match, computed using BM25F
- Semantic matching based features
- Whether product belongs to the department predicted for the query.
- Query-product attribute match
- Query-product attribute value match (one feature for each type of attribute (Category, Brand, Price, Color, Size, etc.) available in the product catalog).

LTR from the trenches i

Uses in e-commerce settings

- **Airbnb**: Haldar et al. [20] use a DNN with a LambdaRank loss function; online improvement of bookings over a pointwise GBDT and an ensemble model combining GBDT and factorization machine signals.
- **Alibaba**: Wu et al. [59] use an ensemble method with heavy feature engineering; GBDT as meta-learner with LR.
- **Alibaba**: Pei et al. [46] use a transformer based reranker that is evaluated on a Yahoo! LETOR dataset and on e-commerce data, with online and offline comparisons against LambdaMART and DLCM.

LTR from the trenches ii

- **Allegro.pl**: Pobrotyn et al. [48] use a context-aware, self-attention mechanism for scoring, taking item-level and list-level properties into account.
- **Amazon**: Sorokina and Cantú-Paz [55] use GBDT with pairwise ranking for product search; no comparison against other LTR methods.
- **Etsy**: Wu et al. [60] use a listwise LTR method to optimize both click and purchase probability; they compare against a range of LTR methods, including Lambda-MART.
- **Facebook**: He et al. [21] use GBDT as a feature extractor, then LogReg, for ad click prediction; no comparison with other LTR methods.

LTR from the trenches iii

- **German retailer of products for babies and small children**: Jannach and Ludewig [27] use mixtures of content-based and collaborative filtering based approaches; no comparisons against more traditional LTR methods.
- **Google**: Ai et al. [1] propose GSF (groupwise scoring functions), learned with a DNN, so that the relevance score of a document is determined jointly by multiple documents; comparisons on WEB30K and on a mail dataset.
- **Mercateo**: Anwaar et al. [2] use counterfactual LTR and logged add-to-basket and order signals for product search; comparison of Lambda-MART and a neural method.
- **Microsoft**: Ling et al. [36] use GDBT to boost neural network output on the ad click prediction task; no comparison with other LTR methods.

LTR from the trenches iv

- **Trivago**: Ludewig and Jannach [39] uses extensive feature engineering and a mixture of BPR, doc2vec and GBDT; no systematic comparison against traditional LTR or ablation study.
- **Walmart**: Karmaker Santu et al. [29] compare Lambda-MART and a range of other methods on a product search task; GBDT/GBRT are not considered.
- **Yahoo!**: Yin et al. [62] use GBDT ("LogisticRank") to rank web documents; GBDT beats LambdaMART
- **Yandex**: Gulin et al. [18] use oblivious trees for document ranking; limited comparison with LambdaRank.

Practical Considerations

Neural Networks or Boosted Trees?

Neural networks

+ Easy to integrate non-tabular features: text embeddings, images, ...
+ Integrate advances in deep learning

– Unscaled / non-smooth features
– Higher-order feature interactions in feed-forward networks [56]

Gradient boosted decision trees

+ Strong performance on (unprocessed) tabular data
+ Advances in GBDT in last decade (XGBoost, CatBoost, LightGBM)

– Less trivial for non-tabular data
– Prone to overfitting

NN are starting to catch up with GBDT [51]

Tools and Libraries

GBDT

- LightGBM [31]
- CatBoost [49]
- XGBoost [8]

What are the differences (NeptuneAI post)?

Neural networks

- TensorFlow Ranking [45]: Probably the best industry choice
- PyTorch LTR [24], PTRank [63]
- Rax [26]: Approximate metric optimization with Jax

Try to avoid

- RankLib: Inferior LambdaMART implementation [51]
- Unbiased LambdaMART [23]: Popular library with theoretical deficiencies [43]

A few more things to keep in mind

- **Model complexity** – the more complex the model, the more accurate it might be. A simpler model can be faster and easier to understand, though perhaps less accurate.
- **Rerank depth** – the deeper you rerank, the more you might find additional documents that could be relevant. The deeper you rerank, the higher the latency.
- **Feature complexity** – if you compute very complex features at query time, they might help your model. They will increase latency.
- **Number of features** - a model with many features might lead to higher relevance. Practical LTR systems usually boil the number down to dozens. Choosing the right cut-off threshold on number of features matters.

Conclusion

Summary

In this tutorial, we discussed:

- supervised LTR, the task of learning a model to rank items based on numerical feature representation in order of relevance to a given query.
- pointwise, pairwise, and listwise **approaches to LTR**.
- the **most important algorithms**: RankNet, LambdaRank, ListNet.
- (known) features and algorithms used in **industry applications**.
- **practical considerations** when applying these models.

We conclude by discussing **overall limitations** of supervised LTR and given an **outlook** on work addressing these limitations.

Limitations of supervised LTR

Limitations of expert-annotated datasets:

- **expensive** to obtain [6, 50]
- **unethical** in privacy-sensitive applications [57]
- **impossible** in personalized settings [57]
- **stationary**, not capturing relevance over time [33]
- **misalignment** with actual user intent [52]
- **disagreement** between annotators [30]

Other limitations:

- Relies on typically **handcrafted numerical features** [6, 14, 50]
- Builds on simplified assumptions of user behavior

Outlook

Unbiased learning to rank

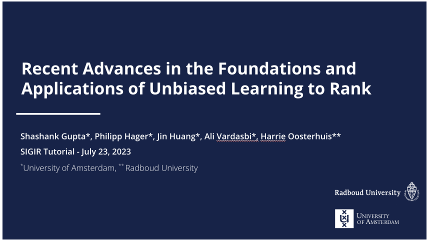
Learn from (biased) user click feedback instead of annotations [44]

Online learning to rank

Learn ranking models while directly interacting with users [44]

Neural IR

Use large language models for ranking and replace handcrafted features [35]



Upcoming SIGIR 2023 tutorial

Outlook

Beyond accuracy goals, including:

- **Diversity** of ranked items [7]
- **Explainability**, a.o., to increase trust in a system [34]
- **Fairness**, e.g., equally relevant items should get equal exposure [41]
- **Multi-sided marketplaces**, optimizing for the utility of multiple stakeholders [40]
- **Resilience**, e.g., to distributional shifts or adversarial actors [25]
- **Composition** of complex SERPs [15]

Questions and Answers

References

[1] Qingyao Ai, Xuanhui Wang, Sebastian Bruch, Nadav Golbandi, Michael Bendersky, and Marc Najork. Learning groupwise multivariate scoring functions using deep neural networks. In *ICTIR 2019: The 2019 ACM SIGIR International Conference on the Theory of Information Retrieval*. ACM, 2019.

[2] Muhammad Umer Anwaar, Dmytro Rybalko, and Martin Kleinsteuber. Mend the learning approach, not the data: Insights for ranking e-commerce products. In *ECML-PKDD*, September 2020.

[3] Aijun Bai, Rolf Jagerman, Zhen Qin, Pratyush Kar, Bing-Rong Lin, Xuanhui Wang, Michael Bendersky, and Marc Najork. Regression compatible listwise objectives for calibrated ranking. *arXiv preprint arXiv:2211.01494*, 2022.

[4] Chris Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Greg Hullender. Learning to rank using gradient descent. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 89–96, 2005. doi: 10.1145/1102351.1102363. URL `https://doi.org/10.1145/1102351.1102363`.

[5] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: From pairwise approach to listwise approach. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 129–136, 2007. doi: 10.1145/1273496.1273513. URL `https://doi.org/10.1145/1273496.1273513`.

[6] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the Learning to Rank Challenge*, volume 14 of *Proceedings of Machine Learning Research (PMLR)*, pages 1–24, 6 2011. URL `https://proceedings.mlr.press/v14/chapelle11a.html`.

[7] Olivier Chapelle, Yi Chang, and Tie-Yan Liu. Future directions in learning to rank. In *Proceedings of the Learning to Rank Challenge*, volume 14 of *Proceedings of Machine Learning Research (PMLR)*, pages 91–100, Jun 2011. URL `https://proceedings.mlr.press/v14/chapelle11b.html`.

[8] Tianqi Chen and Carlos Guestrin. XGBoost: A scalable tree boosting system. In *22nd SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 785–794. ACM, 2016.

[9] Wei Chen, Tie-yan Liu, Yanyan Lan, Zhi-ming Ma, and Hang Li. Ranking measures and loss functions in learning to rank. In *Advances in Neural Information Processing Systems (NIPS)*, volume 22, 2009. URL `https://proceedings.neurips.cc/paper_files/paper/2009/file/2f55707d4193dc27118a0f19a1985716-Paper.pdf`.

[10] William S. Cooper, Fredric C. Gey, and Daniel P. Dabney. Probabilistic retrieval based on staged logistic regression. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 198–210, 1992. doi: 10.1145/133160.133199. URL `https://doi.org/10.1145/133160.133199`.

References iv

[11] David Cossock and Tong Zhang. Subset ranking using regression. In *Proceedings of the Annual Conference on Learning Theory (COLT)*, pages 605–619, 2006. doi: 10.1007/11776420_44. URL https://doi.org/10.1007/11776420_44.

[12] Koby Crammer and Yoram Singer. Pranking with ranking. In *Advances in Neural Information Processing Systems (NIPS)*, volume 14, 2001. URL https://proceedings.neurips.cc/paper_files/paper/2001/file/5531a5834816222280f20d1ef9e95f69-Paper.pdf.

[13] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

[14] Domenico Dato, Sean MacAvaney, Franco Maria Nardini, Raffaele Perego, and Nicola Tonellotto. The istella22 dataset: Bridging traditional and neural learning to rank evaluation. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 3099–3107, 2022. doi: 10.1145/3477495.3531740. URL https://doi.org/10.1145/3477495.3531740.

[15] Romain Deffayet, Thibaut Thonet, Jean-Michel Renders, and Maarten de Rijke. Generative slate recommendation with reinforcement learning. In *WSDM 2023: The Sixteenth International Conference on Web Search and Data Mining*, pages 580–588. ACM, February 2023.

[16] Yoav Freund, Raj Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research (JMLR)*, 4:933–969, 2003. ISSN 1532-4435.

[17] Norbert Fuhr. Optimum polynomial retrieval functions based on the probability ranking principle. *ACM Transactions on Information Systems (TOIS)*, 7(3):183–204, 1989. ISSN 1046-8188. doi: 10.1145/65943.65944. URL https://doi.org/10.1145/65943.65944.

[18] Andrey Gulin, Igor Kuralenok, and Dmitry Pavlov. Winning the transfer learning track of yahoo!'s learning to rank challenge with yetirank. In *Workshop and Conference Proceedings, JMLR*, pages 63–76, 2011.

[19] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, and Thomas Legrand. Applying deep learning to Airbnb search. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1927–1935, 2019. doi: 10.1145/3292500.3330658. URL https://doi.org/10.1145/3292500.3330658.

[20] Malay Haldar, Mustafa Abdool, Prashant Ramanathan, Tao Xu, Shulin Yang, Huizhong Duan, Qing Zhang, Nick Barrow-Williams, Bradley C. Turnbull, Brendan M. Collins, and Thomas Legrand. Applying deep learning to airbnb search. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 1927–1935, 2019. doi: 10.1145/3292500.3330658. URL https://doi.org/10.1145/3292500.3330658.

[21] Xinran He, Junfeng Pan, Ou Jin, Tianbing Xu, Bo Liu, Tao Xu, Yanxin Shi, Antoine Atallah, Ralf Herbrich, Stuart Bowers, and Joaquin Quiñonero Candela. Practical lessons from predicting clicks on ads at facebook. In *ADKDD*, pages 1–9. ACM, 2014.

[22] Ralf Herbrich, Thore Graepel, and Klause Obermayer. Large margin rank boundaries for ordinal regression. In *Advances in Large Margin Classifiers*, chapter 7, pages 115–132. The MIT Press, 1999. URL http://www.herbrich.me/papers/nips98_ordinal.pdf.

[23] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm. In *The World Wide Web Conference (WWW)*, pages 2830–2836, 2019. doi: 10.1145/3308558.3313447. URL https://doi.org/10.1145/3308558.3313447.

[24] Rolf Jagerman and Maarten de Rijke. Accelerated convergence for counterfactual learning to rank. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 469–478, 2020. doi: 10.1145/3397271.3401069. URL https://doi.org/10.1145/3397271.3401069.

[25] Rolf Jagerman, Ilya Markov, and Maarten de Rijke. When people change their mind: Off-policy evaluation in non-stationary recommendation environments. In *WSDM 2019: 12th International Conference on Web Search and Data Mining*, pages 447–455. ACM, February 2019.

[26] Rolf Jagerman, Xuanhui Wang, Honglei Zhuang, Zhen Qin, Michael Bendersky, and Marc Najork. Rax: Composable learning-to-rank using jax. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, 2022.

[27] Dietmar Jannach and Malte Ludewig. Investigating personalized search in e-commerce. In *FLAIRS Conference*, 2017.

[28] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 133–142, 2002. doi: 10.1145/775047.775067. URL https://doi.org/10.1145/775047.775067.

[29] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. On application of learning to rank for e-commerce search. In *SIGIR*, pages 475–484. ACM, 2017.

[30] Gabriella Kazai, Jaap Kamps, Marijn Koolen, and Natasa Milic-Frayling. Crowdsourcing for book search evaluation: Impact of hit design on comparative system ranking. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 205–214, 2011. doi: 10.1145/2009916.2009947. URL https://doi.org/10.1145/2009916.2009947.

[31] Guolin Ke, Qi Meng, Thomas Finley, Taifeng Wang, Wei Chen, Weidong Ma, Qiwei Ye, and Tie-Yan Liu. Lightgbm: A highly efficient gradient boosting decision tree. In *Advances in Neural Information Processing Systems (NIPS)*, volume 30, 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/6449f44a102fde848669bdd9eb6b76fa-Paper.pdf.

[32] Tom Kenter, Alexey Borisov, Christophe Van Gysel, Mostafa Dehghani, Maarten de Rijke, and Bhaskar Mitra. Neural networks for information retrieval. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 1403–1406, 2017. doi: 10.1145/3077136.3082062. URL `https://doi.org/10.1145/3077136.3082062`.

[33] Damien Lefortier, Pavel Serdyukov, and Maarten de Rijke. Online exploration for detecting shifts in fresh intent. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 589–598, 2014. doi: 10.1145/2661829.2661947. URL `https://doi.org/10.1145/2661829.2661947`.

[34] Lei Li, Yongfeng Zhang, and Li Chen. On the relationship between explanation and recommendation: Learning to rank explanations for improved performance. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 14(2), feb 2023. ISSN 2157-6904. doi: 10.1145/3569423. URL `https://doi.org/10.1145/3569423`.

[35] Jimmy Lin, Rodrigo Nogueira, and Andrew Yates. *Pretrained Transformers for Text Ranking: BERT and Beyond*. Synthesis Lectures on Human Language Technologies. Springer, 2021.

[36] Xiaoliang Ling, Weiwei Deng, Chen Gu, Hucheng Zhou, Cui Li, and Feng Sun. Model ensemble for click prediction in bing search ads. In *WWW*, pages 689–698. International World Wide Web Conferences Steering Committee, 2017.

[37] Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009. doi: 10.1561/1500000016. URL `https://doi.org/10.1561/1500000016`.

[38] R Duncan Luce. *Individual choice behavior: A theoretical analysis*. Courier Corporation, 2012.

[39] Malte Ludewig and Dietmar Jannach. Learning to rank hotels for search and recommendation from session-based interaction logs and meta data. In *RecSys Challenge '19*. ACM, 2019.

[40] Rishabh Mehrotra and Benjamin Carterette. Recommendations in a marketplace. In *Proceedings of the ACM Conference on Recommender Systems (RecSys)*, pages 580–581, 2019. ISBN 9781450362436. doi: 10.1145/3298689.3346952. URL https://doi.org/10.1145/3298689.3346952.

[41] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 429–438, 2020. ISBN 9781450380164. doi: 10.1145/3397271.3401100. URL https://doi.org/10.1145/3397271.3401100.

[42] Ramesh Nallapati. Discriminative models for information retrieval. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR)*, pages 64–71, 2004. doi: 10.1145/1008992.1009006. URL https://doi.org/10.1145/1008992.1009006.

[43] Harrie Oosterhuis. Reaching the end of unbiasedness: Uncovering implicit limitations of click-based learning to rank. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval (SIGIR)*, pages 264–274, 2022. doi: 10.1145/3539813.3545137. URL https://doi.org/10.1145/3539813.3545137.

[44] Harrie Oosterhuis, Rolf Jagerman, and Maarten de Rijke. Unbiased learning to rank: Counterfactual and online approaches. In *Proceedings of the Web Conference 2020 (WWW)*, pages 299–300, 2020. doi: 10.1145/3366424.3383107. URL https://doi.org/10.1145/3366424.3383107.

[45] Rama Kumar Pasumarthi, Sebastian Bruch, Xuanhui Wang, Cheng Li, Michael Bendersky, Marc Najork, Jan Pfeifer, Nadav Golbandi, Rohan Anil, and Stephan Wolf. Tf-ranking: Scalable tensorflow library for learning-to-rank. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 2970–2978, 2019. doi: 10.1145/3292500.3330677. URL https://doi.org/10.1145/3292500.3330677.

[46] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, Junfeng Ge, Wenwu Ou, and Dan Pei. Personalized re-ranking for recommendation. In *RecSys 2019*. ACM, September 2019.

[47] Robin. L. Plackett. The analysis of permutations. *Journal of the Royal Statistical Society Series C: Applied Statistics*, 24(2):193–202, 6 1975. ISSN 0035-9254. doi: 10.2307/2346567.

[48] Przemysław Pobrotyn, Tomasz Bartczak, Mikołaj Synowiec, Radosław Białobrzeski, and Jarosław Bojar. Context-aware learning to rank with self-attention. In *eCOM 2020: Proceedings of ACM SIGIR Workshop on eCommerce*. ACM, 2020.

[49] Liudmila Prokhorenkova, Gleb Gusev, Aleksandr Vorobev, Anna Veronika Dorogush, and Andrey Gulin. Catboost: unbiased boosting with categorical features. *Advances in Neural Information Processing Systems (NIPS)*, 31, 2018.

[50] Tao Qin and Tie-Yan Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013. URL `http://arxiv.org/abs/1306.2597`.

[51] Zhen Qin, Le Yan, Honglei Zhuang, Yi Tay, Rama Kumar Pasumarthi, Xuanhui Wang, Michael Bendersky, and Marc Najork. Are neural rankers still outperformed by gradient boosted decision trees? In *International Conference on Learning Representations (ICLR)*, 2021.

[52] Mark Sanderson. Test collection based evaluation of information retrieval systems. *Foundations and Trends in Information Retrieval*, 4(4):247–375, 2010. ISSN 1554-0669. doi: 10.1561/1500000009. URL http://dx.doi.org/10.1561/1500000009.

[53] David Sculley. Combined regression and ranking. In *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining (KDD)*, pages 979–988, 2010. doi: 10.1145/1835804.1835928. URL https://doi.org/10.1145/1835804.1835928.

[54] Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In *Advances in Neural Information Processing Systems (NIPS)*, volume 15, 2002. URL https://proceedings.neurips.cc/paper_files/paper/2002/file/51de85ddd068f0bc787691d356176df9-Paper.pdf.

[55] Daria Sorokina and Erick Cantú-Paz. Amazon search: The joy of ranking products. In *SIGIR*, pages 459–460. ACM, 2016.

[56] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. Deep & cross network for ad click predictions. In *Proceedings of the AdKDD*, 2017. doi: 10.1145/3124749.3124754. URL https://doi.org/10.1145/3124749.3124754.

[57] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the ACM International Conference on Web Search and Data Mining (WSDM)*, pages 610–618, 2018. doi: 10.1145/3159652.3159732. URL https://doi.org/10.1145/3159652.3159732.

[58] Xuanhui Wang, Cheng Li, Nadav Golbandi, Michael Bendersky, and Marc Najork. The LambdaLoss framework for ranking metric optimization. In *Proceedings of the ACM International Conference on Information and Knowledge Management (CIKM)*, pages 1313–1322, 2018. doi: 10.1145/3269206.3271784. URL https://doi.org/10.1145/3269206.3271784.

[59] Chen Wu, Ming Yan, and Luo Si. Ensemble methods for personalized e-commerce search challenge at cikm cup 2016. In *CIKM Cup 2016*, 2016.

[60] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *SIGIR*, pages 365–374. ACM, 2018.

[61] Fen Xia, Tie-Yan Liu, Jue Wang, Wensheng Zhang, and Hang Li. Listwise approach to learning to rank: Theory and algorithm. In *Proceedings of the International Conference on Machine Learning (ICML)*, pages 1192–1199, 2008. doi: 10.1145/1390156.1390306. URL https://doi.org/10.1145/1390156.1390306.

[62] Dawei Yin, Yuening Hu, Jiliang Tang, Tim Daly Jr., Mianwei Zhou, Hua Ouyang, Jianhui Chen, Changsung Kang, Hongbo Deng, Chikashi Nobata, Jean-Marc Langlois, and Yi Chang. Ranking relevance in yahoo search. In *KDD*, pages 323–322. ACM, 2016.

[63] Hai-Tao Yu. PT-Ranking: A benchmarking platform for neural learning-to-rank. *arXiv preprint arXiv:2008.13368*, 2020.