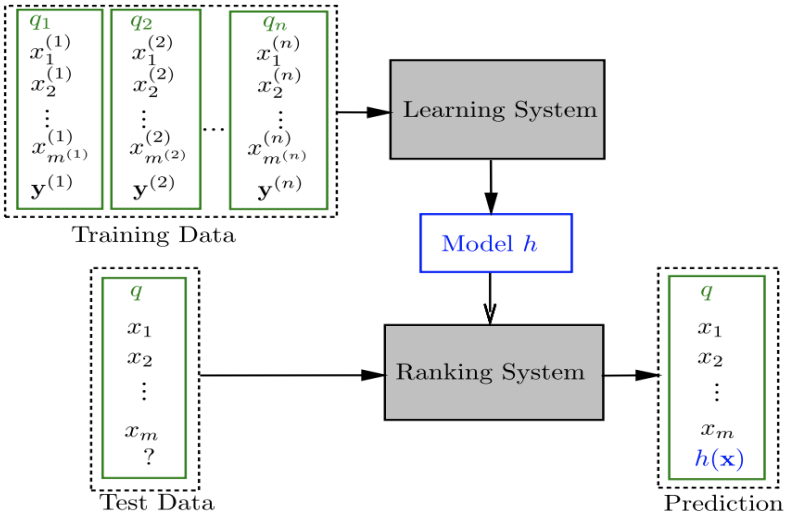
Boyu Pang (bpang4)
Fall 2021

CS 410 Technology Review - Unbiased LambdaMART

Background

Learning-to-rank (LTR) refers to applying machine learning methods to rank a set of documents by relevance.[1] It is in contrast to more traditional ranking methods such as BM25, which is a probabilistic retrieval function that outputs a score based on term match frequency, document frequency, and document length. In LTR, a model is trained using training data and is evaluated on test data using a loss function. The framework for LTR can be summarized in the diagram below:[2]



According to Liu 2009, methods that satisfy the following two criteria are considered LTR methods:

1. It is *feature-based*, meaning that for each query, each associated document d can be represented as a feature vector, where the features are usually pre-specified.
2. The training process is *discriminative*, meaning it is an automatic learning process based on training data, with a well-defined input space, output space, hypothesis space, and loss function.[3]

There are three types of approaches in LTR: pointwise, pairwise, and listwise. They differ in how the loss function considers documents. In the pointwise approach, each document is considered independently of the others, which is to say that the loss function considers one document at a

[1] Liu, Tie-Yan. "Learning to rank for information retrieval." (2011), 225
[2] Ibid., 239.
[3] Ibid., 258.

time and each document's score does not depend on any features of any other documents. In the pairwise approach, pairs of documents are considered during scoring, and the ranking function tries to minimize the number of pairwise inversions in the final ranking (compared to the ground truth). Lastly, the listwise approach considers the entire document set as a list during ranking, so the ranking function attempts to find the optimal ranking of the entire list of documents.[4]

Introduction

In the paper *Unbiased LambdaMART: An Unbiased Pairwise Learning-to-Rank Algorithm*, Hu et al. (henceforth "the authors") propose a method to train a pairwise ranking model which accounts for positional bias in user's clicks (i.e. it is in some sense an "unbiased " algorithm).[5] Positional bias refers to the fact that highly-ranked documents tend to be clicked more frequently by users simply due to their position, regardless of whether they are truly relevant or not. There are also other types of biases such as presentation bias, which is the tendency for users to click more on results that look good (i.e. are presented well). The paper by Hu et al. considers only positional bias. The authors then test their unbiased model on a benchmark dataset as well as in a real-world application through a commercial search engine. This report will give a high-level overview of the unbiased LambdaMART algorithm proposed by the authors and future steps in this area of research.

Framework

The unbiased ranking algorithm proposed by the authors is based on the LambdaMART algorithm published by Burges et al. at Microsoft research in 2010.[6] A brief background of that algorithm is provided here to form a basis for the formulation of the unbiased version put forth by Hu et al.

LambdaMART is essentially the gradient-boosted tree version of an earlier algorithm called LambdaRank (MART stands for Multiple Additive Regression Trees), which itself is an improvement of an even earlier method called RankNet. The key idea behind LambdaRank was the discovery that scaling the gradients (of the cost function with respect to the predicted scores given by the model) by the change in NDCG given by swapping the rank positions of two documents gave good results.[7] LambdaMART is an algorithm that uses gradient boosted decision trees (GBDT) with LambdaRank as the cost function. Since the cost function considers pairs of scores, LambdaMART is a pairwise algorithm.

---

[4] Dandekar, Nikhil. "Pointwise vs. Pairwise vs. Listwise Learning to Rank." *Medium*, Medium, 22 Mar. 2019, https://medium.com/@nikhilbd/pointwise-vs-pairwise-vs-listwise-learning-to-rank-80a8fe8fadfd.
[5] Hu, Ziniu, et al. "Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm." *The World Wide Web Conference*. (2019), 1
[6] Burges, Christopher JC. "From ranknet to lambdarank to lambdamart: An overview." *Learning* 11.23-581 (2010)
[7] Ibid.

Hu et al. then take the LambdaMART algorithm and propose a way to account for position bias during the model training. They define the positional bias as follows:

$$t_i^+ = \frac{P(c_i^+|x_i)}{P(r_i^+|x_i)} = \frac{P(c_i^+, r_i^+|x_i)}{P(r_i^+|x_i)} = P(c_i^+|r_i^+, x_i)$$

where the first numerator is the probability that the i-th document is clicked, and the denominator is the probability that the i-th document is relevant to the query. Thus, we can think of positional bias as a ratio, and the click probability as proportional to the relevance probability. It is also equivalent to say that the position bias ratio is equal to the probability a document is clicked given that it was deemed to be relevant.

$$P(c_i^+|x_i) = t_i^+ P(r_i^+|x_i)$$
$$P(c_j^-|x_j) = t_j^- P(r_i^-|x_j)$$

The first relation can be rewritten as a product, shown above.[8] Here the minus (-) subscripts indicate unclicked and irrelevant documents. In order to find a pairwise objective function that takes the clicked and unclicked biases into account, Hu et al. propose the following relationship:[9]

$$\hat{f}_{unbiased} = \arg\min_f \sum_q \sum_{(d_i, d_j) \in I_q} \frac{L(f(x_i), c_i^+, f(x_j), c_j^-)}{t_i^+ \cdot t_j^-}$$

Here, L represents a pairwise loss function, and the t terms in the denominator are the click and unclick bias ratios. A detailed explanation of how t_i and t_j can be calculated is found in the paper by Hu et al.

Learning Algorithm

To train an unbiased model using the positional bias formulations mentioned above, the key step is to apply the click and unclick positional bias ratios (the two t terms in the previous equation) when calculating the lambda gradient in the LambdaMART algorithm. In the original LambdaMART algorithm, at each step there is a lambda gradient defined on a pair of documents:

[8] Hu, Ziniu, et al. "Unbiased LambdaMART: An unbiased pairwise learning-to-rank algorithm." *The World Wide Web Conference*. (2019), 4
[9] Ibid.

$$\lambda_{ij} = \frac{-\sigma}{1 + e^{\sigma(f(x_i) - f(x_j))}} |\Delta Z_{ij}|$$

where sigma is a parameter set to a constant = 2, delta of Z_ij represents the difference in NDCG scores of documents i and j, and f(x_i)/f(x_j) are the scores of documents i and j given by LambdaMART respectively. Thus, to train an unbiased ranker using LambdaMART, the lambda ij above can be replaced with[10]:

$$\tilde{\lambda}_{ij} = \frac{\lambda_{ij}}{(t_i^+)^* \cdot (t_j^-)^*}$$

For reference, the full algorithm proposed by Hu et al. can be summarized with the pseudocode below:[11]

Algorithm 1 Unbiased LambdaMART

Require: click dataset $\mathcal{D} = \{(q, D_q, C_q)\}$; hyper-parameters p, M;
Ensure: unbiased ranker f; position biases (ratios) t^+ and t^-;

```
Initialize all position biases (ratios) as 1;
for m = 1 to M do
    for each query q and each document d_i in D_q do
        Calculate λ̃_i with (t^+)* and (t^-)* using (35) and (36);
    end for
    Re-train ranker f with λ̃ using LambdaMART algorithm
    Re-estimate position biases (ratios) t^+ and t^- with ranker
    f* using (30) and (31)
end for
return f, t^+, and t^-;
```

Thus, we can see that the unbiased algorithm proposed by the authors is essentially a simple extension of the LambdaMART algorithm which can account for positional biases during the gradient boosting step.

Results and Conclusion

The unbiased LambdaMART algorithm was evaluated on 1) a benchmark dataset and 2) A/B testing through a commercial search engine.

For 1), the dataset used was the Yahoo! Learning-to-rank challenge dataset, which consists of almost 30k queries and 710k documents. Hu et al. synthetically generated click data by simulating users' browsing behavior. The unbiased LambdaMART algorithm significantly

[10] Ibid., 6.
[11] Ibid.

outperformed the baseline methods (including …) in terms of NDCG at 1, 3, 5, and 10[12]. (For a full summary of the experiment results, see the paper by Hu et al.).

For 2), the authors ran an A/B test applying the Unbiased LambdaMART algorithm to the search engine of Jinri Toutiao, which is a news recommendation app popularly used in China. The treatment group used Unbiased LambdaMART while the control group retained the existing ranker; the authors then compared the two treatments by looking at the first click ratios, defined as the percentages of sessions having first clicks at the top 1, 3, 5 positions among all sessions. The treatment group proved to be a statistically significant improvement over the control for all three positions. When the two rankers' results were then evaluated by human assessors, the treatment group (Unbiased LambdaMART ranker) scored better as well.[13]

In conclusion, Hu et al. have proposed a new pairwise approach by extending the original LambdaMART algorithm to account for position biases in user clicks. The authors have shown that this new approach is a promising improvement, demonstrated over its performance on a benchmark dataset as well as the real world in a commercial application. In the future, this pairwise debiasing method can be applied to more general pairwise LTR algorithms. Furthermore, while the approach discussed in Hu et al.'s paper accounts for position bias, there are other types of bias in the real world such as presentation bias, and future work needs to be done in exploring ways to remove these types of biases in learned ranking algorithms.

[12] Ibid., 7.
[13] Ibid., 10.