Counterfactual Learning from Logs for Improved Ranking of E-Commerce Products

Muhammad Umer Anwaar, Dmytro Rybalko, and Martin Kleinsteuber

Abstract—Improved search quality enhances users' satisfaction, which directly impacts sales growth of an E-Commerce (E-Com) platform. Learning to Rank (LTR) algorithms require relevance judgments (labels) on products for learning. In commercial scenarios, getting such judgments poses an immense challenge in application of LTR algorithms. In the literature, it is proposed to employ user feedback signals such as clicks, orders etc to generate relevance judgments. It is done by aggregating the logged data and calculating click rate etc of products for the queries in the logs. In this paper, we advocate counterfactual risk minimization (CRM) approach which circumvents the need of such data pre-processing and is better suited for learning from logged data, i.e. contextual bandit feedback. Due to unavailability of public E-Com LTR dataset, we provide *Commercial dataset* from our E-Com platform. It contains more than 10 million click log entries and 1 million order logs from a catalogue of about 3.5 million products and 3000 queries. To the best of our knowledge, this is the first work which examines effectiveness of CRM approach in learning ranking model from real-world logged data. Our empirical evaluation shows that CRM approach not only learns effectively from logged contextual-bandit feedback but also that our CRM based method outperforms full-information (e.g. cross-entropy) loss on various deep neural network models as well as traditional models like LambdaMART. These findings show that by adopting CRM learning approach, E-Com platforms can get better product search quality compared to full-information approach, without artificially mending the data.

Keywords—Learning to Rank, E-Commerce Search, Implicit Feedback, Counterfactual Risk Minimization, Dataset, Mining Data Logs

I. Introduction

E-COM industry is growing fast, with a projected global retail sales worldwide of 4.8 trillion USD in 2021[1]. Virtually every E-Com platform is leveraging machine learning (ML) techniques for increasing their users' satisfaction and optimizing business value for the company. Optimal ranking of the products plays a vital role in achieving these goals. The successful application of traditional ML algorithms such as LambdaMART [1] and AdaRank [2], requires hand-crafted features. This greatly reduces the applicability of such algorithms in commercial settings. Deep learning (DL) is nowadays a well established framework for automatically learning relevant features from raw data and has also inspired research in LTR [3], [4], [5]. But DL needs large-scale data for training a model. Fortunately, such data is readily available in almost every E-Com platform in the form of search, clicks and orders logs. It will be referred to as *log data* in the remainder of the paper.

As log data is abundantly and cheaply available, it is quite lucrative to devise learning algorithms which can learn effective ranking models from it. In contrast to online learning methods [6], [7], [8], learning from log data does not require intrusive interactions in the live system. This is highly desirable in practice because it avoids badly affecting users' experience on the E-Com platform. But there is one significant challenge in learning from log data; i.e., how to generate relevance judgments on the products for training supervised LTR algorithms. Relevance judgments for benchmark datasets in LTR problems are performed either by human experts or crowd sourcing. The studies conducted by [9] and [10] have shown conclusively that crowd-sourcing is not a reliable technique for getting relevance judgments on products of E-Com platform. This has created a gap in application of DL research for improving E-Com search quality.

In this paper, we aim to bridge this gap and improve product search with practical constraints of a commercial setting[2]. In the literature, Kramaker et.al. [10] have made an attempt to overcome this issue. They proposed to aggregate the log data and for a given query, product pair, calculate the click (order) rate. This rate is then used to get relevance judgments. We discuss it in detail in subsubsection III-C5.

But their method of modelling the relevance judgments, ignores the fact that logged data is in the form of so-called *contextual-bandit* feedback. This means that we have access to *only* those feedback signals which were generated in response to the actions taken by the system. For instance, we do not know how the user would have responded to the search results if another set of candidate products was shown. That is why traditional supervised learning approach, where information about all possible actions is required, is not well-suited for learning from log data. We refer to the traditional approach as full-information (Full-Info) approach and their loss (such as cross-entropy and hinge loss) as Full-Info loss. Furthermore, calculating relevance judgments for Full-Info approach requires unnecessary data pre-processing. One can devise more efficient ways of utilizing the information contained in log files. However, it requires reformulation of the learning problem such that LTR algorithm is adapted to learn directly from the logged data.

Due to these reasons, we advocate employing counterfactual

Muhammad Umer Anwaar is with the Department of Electrical and Computer Engineering, Technical University of Munich (corresponding author, email: umer.anwaar@tum.de).

Dmytro Rybalko is with IBM, Russia (email: dmitriy.rybalko@ibm.com). This work was done during his research stay at Mercateo Group, Munich.

Martin Kleinsteuber is Chief Information Officer and Lead Data Scientist for the Mercateo Group, Munich (email: martin.kleinsteuber@mercateo.com).

[1] https://www.statista.com/statistics/379046/worldwide-retail-e-commerce-sales/

[2] This work is part of ongoing research in solving practical issues faced by our E-Com platform.

risk minimization (CRM) approach [11] to learn directly from logged contextual-bandit feedback. CRM loss requires the knowledge of logging policy currently running on the system, the action taken by the logging policy and users' feedback on these actions. All this information is contained in entries of log files. CRM approach is better suited for learning from such logged feedback, as in contrast to Full-Info approach, it does not require full-information about all actions and their rewards (e.g. clicks, orders etc). Moreover, it also circumvents the need to generate relevance judgments.

In section III, we present in detail a novel LTR *Commercial dataset* constructed from the logs of our E-Com platform. In section IV, we explain how to apply CRM approach to LTR problem. In section V, we conduct several experiments that establish that CRM loss is successful at learning directly from log data. Furthermore, we show that our method outperforms deep neural network (DNN) models and LambdaMART trained with Full-Info loss.

II. Related work

In contrast to web search, there has been little work on learning effective ranking models for E-Com search. Kramaker et.al. [10] have done a systematic study of applying popular LTR algorithms on E-Com search. They studied the query-attribute sparsity problem, the effects of popularity based features and the reliability of relevance judgments via crowd-sourcing. They compared traditional ranking algorithms and reported that LambdaMART showed best performance. They did not include DNN models in their work.

One major issue which has inhibited research in LTR for E-Com is the unavailability of public E-Com LTR dataset. There are some open-source E-Com datasets available for different tasks like for recommender systems: retail rocket dataset and KASANDR [12] but no dataset for LTR. Some researchers [10], [13], [14] mentioned that they have conducted experiments on proprietary E-Com dataset, but they did not publish the dataset due to data confidentiality reasons. We think that such a dataset is critical in advancing the research of designing effective and robust LTR algorithms for E-Com search.

The biggest hurdle in constructing such dataset is how to get relevance judgments on millions of product. We discuss this in detail in subsubsection III-C4. Our approach is inspired from counterfactual risk minimization (CRM) principle proposed by Swaminathan et. al. [11]. Recently, [15] have shown that CRM loss is able to achieve predictive accuracy comparable to cross-entropy loss on object recognition task in computer vision. They replace cross-entropy loss with self-normalized inverse propensity estimator (SNIPS) [16] that enables learning from propensity-logged bandit feedback. It is worth noting that [15] conducted all their experiments on simulated contextual-bandit feedback from CIFAR10 dataset. This is, according to authors' best knowledge, the first work which applies SNIPS estimator on actual logged bandit feedback and verify its effectiveness in solving a real-world problem.

III. E-Com Dataset for LTR

In this section, we present the dataset constructed from Business-to-Business (B2B) E-Com platform. This dataset contains information of queries from real users, actions taken by the policy running on the system, the probability of these actions and feedback of users in response to such actions. *Commercial dataset* is publicly available to facilitate research on ranking products of E-Com platform. The dataset can be accessed at: https://github.com/ecom-research/CRM-LTR.

A. Why a new dataset?

1) Why not use existing datasets from related IR domains?: The task of ranking products for E-Com platforms differs from related tasks in IR like ranking documents or webpages, term extraction, recommender systems etc. E-Com platforms have to tackle diverse needs and preferences of their users while maximizing their profit. This translates to certain unique problems in practice, which are not that severe or even relevant in other domains. For instance, relevance judgments from human judges is not feasible. On the other hand, we have multiple implicit feedback signals generated from the users' interactions with the E-Com website, e.g. clicks, add-to-basket, orders, reviews etc. These signals are usually logged in every E-Com system and are abundantly available. These signals can serve as a proxy for the relevance of search results from users' perspective, users' satisfaction with search results and business value for the E-Com company.

2) Unavailablity of open source E-Com datasets: To the best of our knowledge, there is no E-Com dataset publicly available for LTR problem. We think this is because such datasets may contain confidential or proprietary information, and E-Com platforms are unwilling to share such information publicly. For instance, [10], [13], [14] worked on LTR for E-Com search, and constructed a dataset from *Walmart and Amazon* but they did not publish it. Moreover, some E-Com datasets that are publicly available are not suitable for LTR tasks, since they were constructed to deal with problems like clustering or recommendation systems [17], [12]. Furthermore, they have very few features (e.g. 8 features in [18]) and few products to be considered viable for LTR task.

3) Logged dataset: Since CRM loss does not require any relevance judgment for training the LTR model, but instead requires contextual bandit logged data. Due to absence of such open-source log dataset, we also publish the logged data. As our dataset is constructed from real-world log data, we hope it will prove instrumental in advancing the research on learning directly from logs.

B. Scope of the Dataset

For a given query, E-Com search can be divided into two broad steps. The first step retrieves potentially relevant products from the product catalogs and the second step ranks these retrieved products in such a fashion which optimizes users' satisfaction and business value for the E-Com platform [19]. Our proprietary algorithm performs the first part effectively. Thus, for this dataset, we are chiefly concerned with overcoming the challenges faced in optimizing the second part of the E-Com search.

C. Dataset Construction

We explain briefly the steps performed in creating this dataset.

1) Data sources: We collected data from only those sources which are available in almost every E-Com platform. Namely: (i) title of the products, (ii) click data from user logs, (iii) search logs (containing information about the products displayed on the search result page in response to the query) and (iv) number of orders for the products sold during the time period of interest.

2) Sampling the Products: We have more than 23 Million products in our product set, which is typical for E-Com platforms. It would be highly inefficient to consider all of them for a given query and learn a ranker. For instance, the task of extracting features for all of the products for every query is neither desirable nor viable. In IR tasks, it is common to employ some sort of sampling of documents (products), in order to remove highly irrelevant documents with respect to the sampling model. For example, consider LETOR [20]: For the "Gov" corpus, they use BM25 model to rank all documents for each query and then choose the top one thousand documents for that query for the construction of dataset (feature extraction etc). For the *Commercial dataset*, we sample the products with our proprietary algorithm, which removes *irrelevant* products. Its implementation details can not be shared publicly. But it utilizes knowledge graph based techniques for predicting the most probable product category (or categories) and the relevant product type(s) in that category for the given query. After this product sampling step, we have a considerably smaller set of *potentially relevant products* for a given query.

3) Selecting the queries: Most queries included in this dataset are those which were really challenging for our current ranking algorithm. Queries were subsampled from the search logs, keeping in mind the following considerations: (i) ensure statistical significance of the learning outcomes (especially for dev and test sets) and (ii) include queries with variable *specificity*. We say a query has low *specificity*, if the query entered by the user has a broad range of products associated with it. This can be ensured simply by looking at the number of *potentially relevant products* returned by our proprietary algorithm. For instance, for a very specific query like *pink brush for painting* (20) products are returned, while for broad and common queries like *iphone* (354) or *beamer* (1,282) products are returned.

4) How to get relevance judgments on E-Com products for supervised dataset?: Our CRM based approach does not require relevance judgments for learning but such judgments are required for full-information supervised loss and traditional learning algorithms like LambdaMART. These relevance judgments will also be required for doing a fair comparison of CRM approach with other approaches. But how do we get them for millions of products? Relevance judgments for benchmark datasets [20], [1] are performed by human judges; who can be domain experts or crowd-source workers. Unfortunately it is too expensive to ask experts to judge products in an E-Com setting.

The problems associated with generating relevance judgments through crowd-sourcing have been analyzed extensively by [9]. They used Amazon Mechanical Turk to validate their hypothesis that users of E-Com platforms have a very complex utility function and their criteria of relevance may depend on product's value for money, brand, warranty etc. Thus, it is quite difficult for crowd-source workers to capture all these aspects of relevance. This was also confirmed by [10]; they found crowd-sourcing an unreliable method for relevance judgments.

An alternative approach is to generate relevance judgments from multiple feedback signals present in user interaction logs and historical sales data of products. Major benefits of this approach are: (i) such relevance judgments are closer to the notion of relevance in E-Com Search and quantify relevance as a proxy of user satisfaction and business value, (ii) it costs less time and money, as compared to other two approaches, and (iii) large-scale E-Com datasets can be constructed, as it only requires data generated by users' interaction with the E-Com website. Such data is abundantly available in almost every E-Com platform. Potential drawbacks are that (i) the quality of these relevance judgments may not be as good as human expert judgments, and (ii) it is computationally expensive to do this data pre-processing for large scale data for an extended period of time.

Based on the analysis of available choices for supervised setting, generating relevance judgments from log files and historical sales data in such manner is justified. [10] also advocated this approach.

5) Calculating Relevance Judgments (Labels): We follow the approach advocated by Kramaker et.al. [10] for calculating the relevance judgments. For a given query q, we now show the steps taken to calculate relevance judgments on the set of products, $\mathcal{P}_q$, associated with it:

1) We calculate visibility of a product, i.e. the number of times a product p was shown to the user. We filter out the products with a visibility less than 100.
2) We then convert click (order) signal to *relevance rate* (RR) by dividing the clicks (orders) for product p, with its visibility.
3) Next, we normalize RR with the maximum value of RR for that query q to get normalized relevance rate (NRR).

We publish NRR for all query, product pairs. This allows researchers the flexibility in computing different type of relevance judgments. The relevance judgment, $l(p,q)$, used in the experiments are calculated by the following formula:

$$l(p,q) = ceil[4 \cdot NRR(p,q)] \tag{1}$$

6) Dataset format: Our clients (sellers) have proprietary rights on the product title and description available on our platform. So, raw text can not be published in *Commercial dataset*. Therefore, we train GloVe model [21] on the corpus comprising of queries and product titles to learn word embeddings [3]. We publish these 100-dimensional word embeddings.

For each context (query and product pair), we also provide some proprietary features that can be used as additional

[3]One can also employ more advanced embedding techniques like ELMo, BERT to get these embeddings. Since this was not the focus of our study, thus we resorted to a simpler model to learn embeddings.

TABLE I: Statistics of the *Commercial dataset*

Total # of Queries	3060
# of Queries in Train set	1836
# of Queries in Dev set	612
# of Queries in Test set	612
Total # of Products	3507965
# of Products in Train	2032393
# of Products in Dev	745874
# of Products in Test	729698
Avg. # of Products per Query [Train]	1106.9
Avg. # of Products per Query [Dev]	1218.7
Avg. # of Products per Query [Test]	1192.3
Entries in Click Logs	10156042
Entries in Order Logs	1146555

(dense) features in many neural network approaches. These features contain information about price, delivery time, profit margin etc of the product. Specific details can not be shared as these dense features are also part of current ranking algorithm running on our platform.

Although there are plenty of feedback signals logged by E-Com platforms, but we considered only two most common feedback signals in our dataset. One is click on the results shown in response to query and second is the order signal on the same list of results. The data for clicks [orders] is inherently in the log format and contains one entry for each click [order] by the user. For each query, we retain all products that were clicked (ordered) and negatively sample remaining products from the logs, i.e. products which were shown to the user but not clicked (ordered). A separate supervised train set is also published along with click and order labels. The queries and products are same for both supervised train set and logged training set. Test and development sets are published *only* with supervisory labels. The split of queries among train, dev and test sets is done randomly with 60%, 20% and 20% of total number of queries.

D. Glimpse of the Commercial dataset

The *Commercial dataset* contains more than 10 million and 1 million click and order log entries respectively. Table I shows some statistics about the dataset. Compared to existing LTR datasets, the size of this dataset is much larger and better reflects a practical use case. For instance, in LETOR 4.0 [20], there are about 1700 queries in MQ2007 dataset with labeled documents and about 800 queries in MQ2008 dataset with labeled documents. The E-Com dataset (*not publicly available*) used by Kramaker et.al. [10], has 2800 queries and average 94.2 products per query.

IV. Problem Formulation

In traditional supervised setting, for instance in the point-wise LTR case, a query and product pair defines training instance and we have access to true relevance judgment of this product for the given query. The algorithm has to learn to predict correct relevance label of a product for a given query. In CRM setting, we say that a query and product pair defines the context and the action taken by the system is that whether this product is to be displayed among top-k positions or not. A reward (loss) is accumulated based on feedback signals from the user. Examples are click, order or revenue generated by this action. Additional information can be incorporated into context such as users' model for personalized ranking, product description etc. But for simplicity, we considered a query and product title as context.

It is to be noted that top-k positions are extremely important for real-world LTR systems. And also that k is usually not same for all queries in E-Com platforms. It varies depending on the query specificity. For instance, if a query is too broad like 'copy paper', then it signals navigational or informational intent of the user. Thus, k has big value for such queries as compared to more specific queries like 'green A4 copy paper'.

More formally, let context $c \in \mathcal{C}$ be word embeddings of search query and product title. Let $a \in \mathcal{A}$ denote the action taken by the system and $\pi_0(a|c)$ denote the logging policy running on the E-Com platform, which determines the probability of an action a given the context c. For a particular action a_i and context c_i, this probability is also referred as propensity, p_i. i.e., $p_i = \pi_0(a_i|c_i)$.

For a particular a_i and c_i , we denote the loss by δ_i. The logged data is a collection of n tuples: $D = [(c_1, a_1, p_1, \delta_1), ..., (c_n, a_n, p_n, \delta_n)]$.

The goal of counterfactual risk minimization is to learn an unbiased stochastic policy π_w from logged data, which can be interpreted as a conditional distribution $\pi_w(A|c)$ over actions $a \in \mathcal{A}$. We model this conditional distribution using a neural network with softmax output layer:

$$\pi_w(a|c) = \frac{exp(f_w(c,a))}{\sum_{a' \in \mathcal{A}} exp(f_w(c,a'))}, \tag{2}$$

where $f_w(c,a)$ is a deep neural network, which takes word embeddings of two texts as input and outputs a relevance score for these texts.

Swaminathan et.al. [16], [15] proposed SNIPS as an efficient estimator to counterfactual risk. For details, we refer to [15]. We use SNIPS in our experiments.

$$\hat{R}_{SNIPS}(\pi_w) = \frac{\frac{1}{n}\sum_{i=1}^{n} \delta_i \frac{\pi_w(a_i|c_i)}{\pi_0(a_i|c_i)}}{\frac{1}{n}\sum_{i=1}^{n} \frac{\pi_w(a_i|c_i)}{\pi_0(a_i|c_i)}}. \tag{3}$$

V. Experiments and Evaluation

A. Experimental Setup

In our experiments, we treat the loss δ and actions a as binary variables. The action a is equal to 1, if logging policy (π_0) *decides* to show a product in the top-k results for a given query and it is 0 otherwise. If $a = 1$ and the user clicked (ordered), we set loss $\delta = 0$. This implies that decision of π_0 is correct and product is relevant to the user. Conversely, if it was not clicked (ordered); then this is logged with $\delta = 1$. On the other hand, if $a = 0$ is selected by π_0 and the user browsed beyond top-k results and clicked (ordered), we set $\delta = 1$. This means that decision of π_0 is incorrect and product is actually relevant to the user, thus it must have been shown in top-k results. Conversely, if $a = 0$ and the user did not clicked (ordered), we set $\delta = 0$.

We selected a simple yet powerful CNN model proposed by Severyn et.al [22] for empirical evaluation of our CRM

TABLE II: Performance comparison for target label: clicks.

Ranker (Loss)	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
Logging policy	0.4704	0.6123	0.4686	0.4613	0.2052	0.2537
LambdaMART	0.3715	0.4114	0.2678	0.2799	0.0497	0.0734
CNN (Cross-Entropy)	0.5074	0.8036	0.6552	0.6261	0.3362	0.3835
CNN (CRM) - ours	**0.5993**	**0.8391**	**0.7346**	**0.7093**	**0.4332**	**0.4964**

TABLE III: Performance comparison for target label: orders.

Ranker (Loss)	MAP	MRR	P@5	P@10	NDCG@5	NDCG@10
Logging policy	0.2057	0.3225	0.1693	0.1717	0.0945	0.1250
LambdaMART	0.1339	0.1632	0.0647	0.074	0.022	0.0358
CNN (Cross-Entropy)	0.2728	0.4601	0.2869	0.2562	0.1720	0.1973
CNN (CRM) - ours	**0.3181**	**0.4609**	**0.2841**	**0.2791**	**0.1854**	**0.2266**

approach. It utilizes convolutional filters for ranking pairs of short texts. It requires word embeddings of query and product as input. The model can also take dense features as additional input. With CNN ranking model, we compare the performance of CRM loss with cross-entropy loss. For both losses, we use Adam optimizer [23] and select the model which yields best performance on dev set. All network parameters are kept same in both cases for fair comparison. Further implementation details can be found in the code available at: https://github.com/ecom-research/CRM-LTR. For measuring the performance, we considered popular ranking metrics like Mean Average Precision (MAP), Mean Reciprocal Rank (MRR), Precision at top-k positions (P@k) and Normalized Discounted Cumulative Gain at top-k positions (NDCG@k) [24]. All these metrics are computed using the standard *trec_eval* [4] tool provided by TREC conference.

We also compare the performance of CRM approach with LambdaMART [25] model. For training LambdaMART, we use open source $RankLib$ toolkit, with default parameters [5]. LambdaMART uses full-information supervised loss that directly optimizes information retrieval metrics (like NDCG). It has been recently shown that LambdaMART outperforms traditional LTR algorithms on E-Com dataset [10]. We use dense features engineered by domain experts as input for LambdaMART model. The ranker [not based on DNN] currently running on our E-Com platform also employs the same dense features. We also evaluate the performance of the logging policy to serve as a baseline for our experiments. Predictions of this policy are used by our CRM method for actions a_i and propensities $\pi_0(a_i|c_i)$.

B. Click Logged Data

The most abundant, albeit noisy, available indicator for relevance are clicks on products for a given query. Table II shows the comparison of performance of different ranking models on test set with graded click labels (see Equation 1). First, we note that both CRM and supervised losses are able to perform significantly better than the logging policy. Specifically, on MAP, the performance improvement against logging policy is approximately 7.8% for cross-entropy loss and 27.4% for CRM loss.

Second we observe that LambdaMART model is performing quite worse on all metrics. It is performing significantly worse than even logging policy. Particularly, performance degradation is 245% on NDCG@10 and 26.6% on MAP. This huge difference in metrics suggest that LambdaMART has failed to learn graded relevance. Since MAP treats all relevance other than zero as one, whereas NDCG metric is sensitive to graded relevance.

These results also suggest that deep learning approach can significantly outperform LambdaMART, given enough training data. This is consistent with findings of [3]. It can be argued that since LambdaMART has access to limited number of hand-crafted features, so its performance can improve by adding more features. But feature engineering requires domain expertise and is quite time-consuming process. Moreover, our logging policy, which uses the same features performs quite better than LambdaMART. Third we note that our CRM based approach performs significantly better than supervised loss on all metrics. Concretely, on NDCG@10, the performance gain of our method against cross-entropy loss is 29.4%.

These results show that CRM loss is very effective in learning click-based relevance from logs and significantly outperforms the baseline and models trained in supervised fashion on aggregated data.

C. Orders Logged Data

The order logs are extremely valuable source of relevance for products for a given query. Orders, though more sparse than clicks, are less noisy and correlate strongly with users' satisfaction with E-Com search results. They also serve as proxy to business value for the E-Com platform.

Table III summarizes results for cross-entropy and CRM models evaluated on the *Commercial dataset* with *orders* as relevance source. Similar to click logs, our approach significantly outperforms all other rankers on order logs. Both CRM and CE loss outperform the logging policy. Our CRM loss has performance gain of 39.7% w.r.t MAP and 14.9% w.r.t NDCG@10 against CE loss. Whereas the gain in performance over logging policy is 54.6% on MAP and 81.3% on NDCG@10.

The results of this experiment also confirm the ability of CRM approach to effectively learn from logged data.

D. Learning Progress with increasing number of bandit feedback

It is quite insightful to see how the performance of our CRM model is changing with number of training samples. In order to

[4]Available at https://github.com/usnistgov/trec_eval
[5] https://sourceforge.net/p/lemur/wiki/RankLib/

TABLE IV: Comparison of Deep Learning Models for target label: orders.

Ranker	Loss	MAP	NDCG@5	NDCG@10
Logging policy	-	0.2057	0.0945	0.1250
CNN [22]	Full-Info	0.2728	0.1720	0.1973
CNN [22]	**CRM**	**0.3181**	**0.1854**	**0.2266**
ARCII [26]	Full-Info	0.2891	0.2044	0.2238
ARCII [26]	**CRM**	**0.3208**	**0.2319**	**0.2472**
DRMMTKS [27]	Full-Info	0.3112	0.2183	0.2309
DRMMTKS [27]	**CRM**	**0.3361**	**0.2436**	**0.2617**
ConvKNRM [28]	Full-Info	0.2818	0.1159	0.1494
ConvKNRM [28]	**CRM**	**0.2942**	**0.1344**	**0.1604**
DUET [4]	Full-Info	0.3488	0.2501	0.2866
DUET [4]	**CRM**	**0.3562**	**0.2679**	**0.3055**

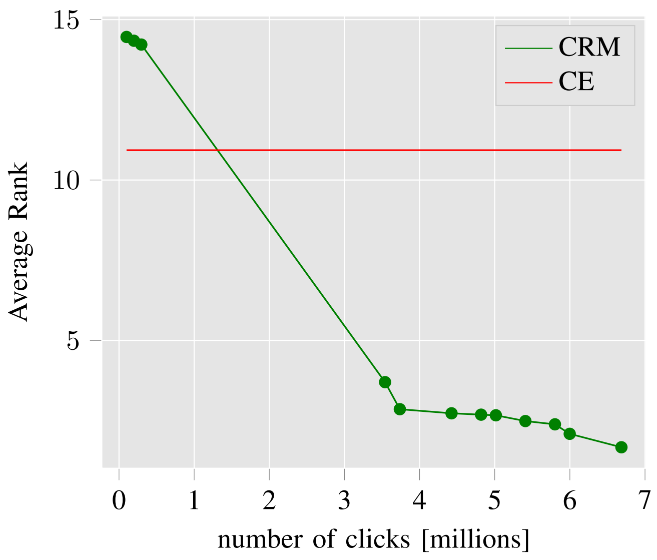

Fig. 1: Avg. rank of relevant product in click test set

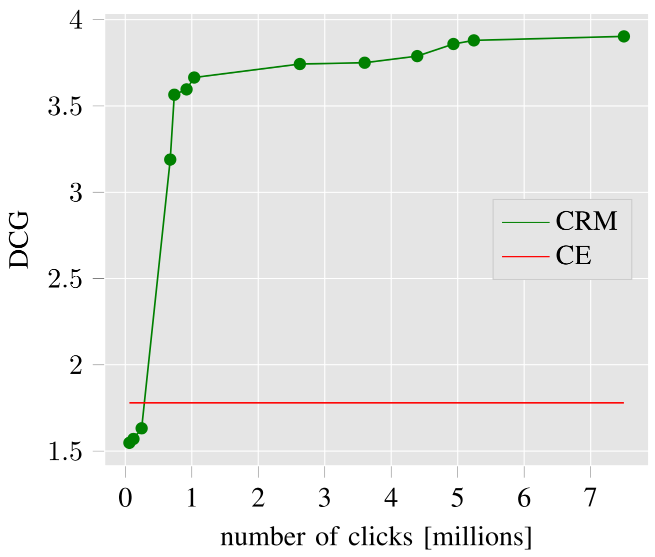

Fig. 2: Average DCG of relevant product in click test set

visualize how the ranking model learns with increasing number of bandit feedback (clicks), we stop the training after certain number of clicks and evaluate the model on clicks test set and the resume training.

In Figure 1 and Figure 2, we plot the values of average rank and average Discounted Cumulative Gain (DCG) scores of relevant products for all queries in clicks test set, depending on number of clicks processed at training time. These curves clearly show that there is a constant improvement in average rank of relevant (clicked) product as more bandit feedback is processed. Similarly, average DCG values rise monotonically with increasing number of click feedback. The red line in the figures represent the average rank and average DCG of relevant products in test set, as per the predictions of trained CE ranking model. This experiment confirms the ability of CRM approach to learn efficiently with increasing bandit feedback (number of clicks).

E. Comparison of Deep Learning Models

In order to investigate whether the improvement in performance is architecture agnostic or not, we performed comparison of CRM loss with full-information loss on several different DNN architectures. We choose these models from MatchZoo [29], an open-source codebase for deep text matching research. For a fair comparison with CNN model, we modified the models in MatchZoo and added a fully connected layer before the last layer. This layer takes dense features as input. We considered four models from MatchZoo, the results of the comparison are summarized in Table IV.

We note that CRM loss outperfoms full-information loss on all five deep learning models. Specifically, performance gain of CRM loss over full-information loss w.r.t NDCG@10 is 10.5% for ARCII, 13.3% for DRMMMTKS, 7.4% for ConvKNRM and 6.6% for DUET. It shows that performance gains of CRM loss is not limited to any specific architecture.

We also note from Table IV that, as already well-known in the literature, the deep learning model architecture has significant impact on the performance. For instance, best performing model with full-information loss, DUET [4] has 27.8% improvement over simple CNN [22] w.r.t MAP. For CRM loss, best performing model DUET [4] has 21% improvement over worst performing model ConvKNRM [28] w.r.t MAP.

The results of this experiment support our claim that mending the learning approach is beneficial for learning from logged data as compared to modifying the data to fit the supervised (full-information) learning approach.

VI. Conclusion

In E-Com platforms, user feedback signals are ubiquitous and are usually available in log files. These signals can be interpreted as contextual-bandit feedback, i.e. partial information which is limited to the actions taken by the logging policy and users' response to the actions. In order to learn effective ranking of the products from such logged data, we propose to employ counterfactual risk minimization approach. Our experiments have shown that CRM approach outperforms traditional supervised (full-information) approach on several DNN models. This proves empirically that reformulating the LTR problem to utilize the information contained in log files is better approach than artificial adapting of data for the learning algorithm.

Acknowledgment

The work of author 1 has been supported by the Bavarian Ministry of Economic Affairs, Regional Development and Energy through the *WoWNet* project #IUK 625/001. The work of author 2 has been supported by the European Regional Development Fund *VANDA*- project# 100272023.

Appendix A
Comparison of counterfactual risk estimators

We compare the performance of SNIPS estimator with two baseline estimators for counterfactual risk. We conduct the experiments on click training data of *Commercial dataset*. The inverse porpensity scoring (IPS) estimator is calculated by:

$$\hat{R}_{IPS}(\pi_w) = \frac{1}{n}\sum_{i=1}^{n}\delta_i\frac{\pi_w(a_i|c_i)}{\pi_0(a_i|c_i)}. \tag{4}$$

Second estimator is an empirical average (EA) estimator defined as follows:

$$\hat{R}_{EA}(\pi_w) = \sum_{(c,a)\in(\mathcal{C},\mathcal{A})}\bar{\delta}(c,a)\pi_w(a|c), \tag{5}$$

where $\bar{\delta}(c,a)$ is the empirical average of the losses for a given context and action pair. The results for these estimators are provided in Table V. Compared to SNIPS both IPS and EA perform significantly worse on all evaluated metrics. The results confirm the importance of equivariance of the counterfactual estimator and show the advantages of SNIPS estimator.

Appendix B
Choosing hyperparameter λ

One major drawback of SNIPS estimator is that, being a ratio estimator, it is not possible to perform its direct stochastic optimization [15]. In particular, given the success of stochastic gradient descent (SGD) training of deep neural networks in related applications, this is quite disadvantageous as one can not employ SGD for training.

To overcome this limitation, Joachims et. al. [15] fixed the value of denominator in Equation 3. They denote the denominator by S and solve multiple constrained optimization problems for different values of S . Each of these problems

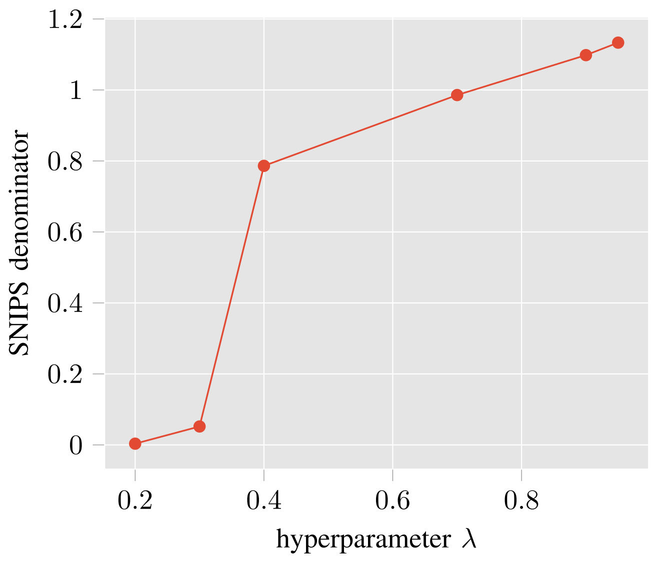

Fig. 3: SNIPS denominator vs λ on order logs (training set)

can be reformulated using lagrangian of the constrained optimization problem as:

$$\hat{w}_j = \underset{w}{\operatorname{argmin}}\ \frac{1}{n}\sum_{i=1}^{n}(\delta_i - \lambda_j)\frac{\pi_w(a_i|c_i)}{\pi_0(a_i|c_i)} \tag{6}$$

where λ_j corresponds to a fixed denominator S_j.

The main difficulty in applying the CRM method to learn from logged data is the need to choose hyperparameter λ. We discuss below our heuristics of selecting it. We also evaluate the dependence of λ on SNIPS denominator S, which can be used to guide the search for λ. To achieve good performance with CRM loss, one has to tune hyperparameter $\lambda \in [0,1]$. Instead of doing a grid search, we follow a smarter way to find a suitable λ. Building on the observations proposed in [15], we can guide the search of λ based on value of SNIPS denominator S. It was shown in [15] that the value of S increases monotonically, if λ is increased. Secondly, it is straightforward to note that expectation of S is 1. This implies that, with increasing number of bandit feedback, the optimal value for λ should be selected such that its corresponding S value concentrates around 1. In our experiments, we first select some random $\lambda \in [0,1]$ and train the model for two epochs with this λ. We then calculate S for the trained model; if S is greater than 1, we decrease λ by 10%, otherwise we increase it by 10%. The final value of λ is decided based on best performance on validation set.

In Figure 3, we plot the values of denominator S on order logs (training set) of *Commercial dataset* for different values of hyperparameter λ. On the figure below, Figure 4, we also plot performance on orders test set, in terms of MAP and NDCG@5 scores, of different rankers for these values of hyperparameter λ. It is to be noted that the values of SNIPS denominator S monotonicaly increase with increasing λ. The MAP and NDCG@5 reach its highest value for $\lambda = 0.4$, but decrease only slightly with increasing values of λ. Furthermore, it can also be seen from these two figures that the λ

TABLE V: Results on *Commercial dataset* with click relevance for IPS and empirical average estimators

Estimator	MAP	MRR	NDCG@5	NDCG@10
CNN (CRM) - SNIPS	**0.5993**	**0.8391**	**0.4332**	**0.4964**
CNN (CRM) - IPS	0.4229	0.7139	0.3426	0.3703
CNN (CRM) - EA	0.2320	0.3512	0.2083	0.2253

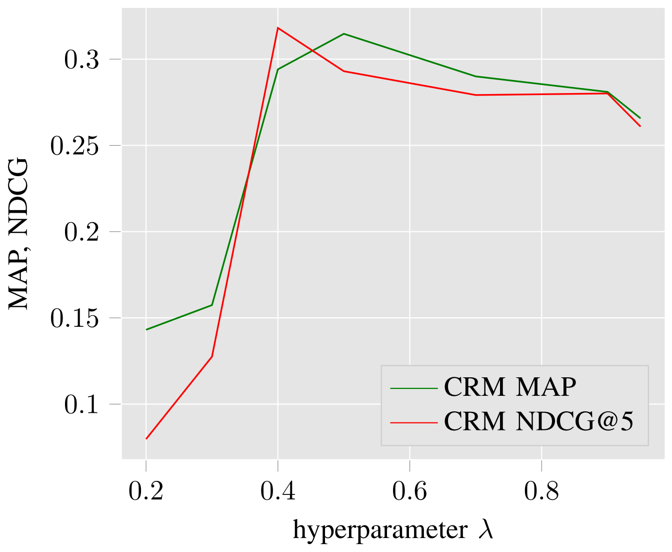

Fig. 4: Performance on orders test set of rankers trained with different λ

values with good performance on test set have corresponding SNIPS denominator values close to 1.

References

[1] O. Chapelle and Y. Chang, "Yahoo! learning to rank challenge overview," in *Proceedings of the Learning to Rank Challenge*, 2011, pp. 1–24.

[2] J. Xu and H. Li, "AdaRank: a boosting algorithm for information retrieval," ser. SIGIR '07. New York, NY, USA: ACM, 2007, pp. 391–398.

[3] L. Pang, Y. Lan, J. Guo, J. Xu, J. Xu, and X. Cheng, "Deeprank: A new deep architecture for relevance ranking in information retrieval," *CoRR*, vol. abs/1710.05649, 2017.

[4] B. Mitra, F. Diaz, and N. Craswell, "Learning to match using local and distributed representations of text for web search," *CoRR*, 2016.

[5] J. Guo, Y. Fan, Q. Ai, and W. B. Croft, "A deep relevance matching model for ad-hoc retrieval," in *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, ser. CIKM '16. New York, NY, USA: ACM, 2016, pp. 55–64.

[6] A. Schuth, K. Hofmann, S. Whiteson, and M. de Rijke, "Lerot: An online learning to rank framework," in *Proceedings of the 2013 workshop on Living labs for information retrieval evaluation*. ACM, 2013, pp. 23–26.

[7] Y. Hu, Q. Da, A. Zeng, Y. Yu, and Y. Xu, "Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application," in *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery &#38; Data Mining*, ser. KDD '18. New York, NY, USA: ACM, 2018, pp. 368–377. [Online]. Available: http://doi.acm.org/10.1145/3219819.3219846

[8] S. Chaudhuri and A. Tewari, "Online ranking with top-1 feedback," in *Artificial Intelligence and Statistics*, 2015, pp. 129–137.

[9] O. Alonso and S. Mizzaro, "Relevance criteria for e-commerce: a crowdsourcing-based experimental analysis," in *SIGIR '09: Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*. New York, NY, USA: ACM, 2009, pp. 760–761.

[10] S. K. K. Santu, P. Sondhi, and C. Zhai, "On application of learning to rank for e-commerce search," in *Proceedings of the 40th International ACM SIGIR*, ser. SIGIR '17. ACM, 2017.

[11] A. Swaminathan and T. Joachims, "Batch learning from logged bandit feedback through counterfactual risk minimization," *Journal of Machine Learning Research*, vol. 16, pp. 1731–1755, 2015.

[12] S. Sidana, C. Laclau, M. R. Amini, G. Vandelle, and A. Bois-Crettez, "Kasandr: A large-scale dataset with implicit feedback for recommendation," ser. SIGIR '17. ACM, 2017, pp. 1245–1248.

[13] E. P. Brenner, J. Zhao, A. Kutiyanawala, and Z. Yan, "End-to-end neural ranking for ecommerce product search," *Proceedings of SIGIR eCom*, vol. 18, 2018.

[14] K. Bi, C. H. Teo, Y. Dattatreya, V. Mohan, and W. B. Croft, "Leverage implicit feedback for context-aware product search," in *eCOM@SIGIR*, 2019.

[15] T. Joachims, A. Swaminathan, and M. d. Rijke, "Deep learning with logged bandit feedback," in *International Conference on Learning Representations*, May 2018.

[16] A. Swaminathan and T. Joachims, "The self-normalized estimator for counterfactual learning," in *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 2*, ser. NIPS'15. Cambridge, MA, USA: MIT Press, 2015, pp. 3231–3239.

[17] D. Dheeru and E. Taniskidou, "UCI machine learning repository," 2017.

[18] D. Chen, "Data mining for the online retail industry: A case study of rfm model-based customer segmentation using data mining," *Journal of Database Marketing and Customer Strategy Management*, vol. 19, no. 3, pp. 197–208, August 2012.

[19] C. Lucchese, F. M. Nardini, S. Orlando, R. Perego, and N. Tonellotto, "Speeding up document ranking with rank-based features," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: ACM, 2015, pp. 895–898.

[20] T. Qin, T.-Y. Liu, J. Xu, and H. Li, "LETOR: A benchmark collection for research on learning to rank for information retrieval," *Inf. Retr.*, vol. 13, no. 4, pp. 346–374, Aug. 2010.

[21] J. Pennington, R. Socher, and C. D. Manning, "Glove: Global vectors for word representation," in *In EMNLP*, 2014.

[22] A. Severyn and A. Moschitti, "Learning to rank short text pairs with convolutional deep neural networks," in *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR '15. New York, NY, USA: ACM, 2015, pp. 373–382.

[23] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2015.

[24] K. Järvelin and J. Kekäläinen, "Cumulated gain-based evaluation of IR techniques," *ACM Trans. Inf. Syst.*, vol. 20, no. 4, pp. 422–446, Oct. 2002.

[25] Q. Wu, C. J. Burges, K. M. Svore, and J. Gao, "Adapting boosting for information retrieval measures," *Inf. Retr.*, vol. 13, no. 3, pp. 254–270, Jun. 2010.

[26] S. Wan, Y. Lan, J. Guo, J. Xu, L. Pang, and X. Cheng, "A deep architecture for semantic matching with multiple positional sentence representations," *CoRR*, vol. abs/1511.08277, 2015. [Online]. Available: http://arxiv.org/abs/1511.08277

[27] Z. Yang, Q. Lan, J. Guo, Y. Fan, X. Zhu, Y. Lan, Y. Wang, and X. Cheng, "A deep top-k relevance matching model for ad-hoc retrieval," in *Information Retrieval*, S. Zhang, T.-Y. Liu, X. Li, J. Guo, and C. Li, Eds. Cham: Springer International Publishing, 2018, pp. 16–27.

[28] Z. Dai, C. Xiong, J. Callan, and Z. Liu, "Convolutional neural networks for soft-matching n-grams in ad-hoc search," in *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, ser. WSDM '18. New York, NY, USA: ACM, 2018, pp. 126–134. [Online]. Available: http://doi.acm.org/10.1145/3159652.3159659

[29] J. Guo, Y. Fan, X. Ji, and X. Cheng, "Matchzoo: A learning, practicing, and developing system for neural text matching," in *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, ser. SIGIR'19. New York, NY, USA: ACM, 2019, pp. 1297–1300. [Online]. Available: http://doi.acm.org/10.1145/3331184.3331403