Addressing Cold Start With Dataset Transfer In E-Commerce Learning To Rank

Paul Missault
Amazon
pmissaul@amazon.com

Arnaud de Myttenaere
Amazon
ademytt@amazon.com

Andreas Radler
Amazon
rdlrr@amazon.com

Pierre-Antoine Sondag
Amazon
pierreas@amazon.com

ABSTRACT

Learning To Rank (LTR) models are a crucial component for the relevance ranking in e-commerce applications. When properly trained, the LTR models yield state-of-the-art ranking performance. A popular approach to training the LTR models involves minimising a loss over historical customer interactions. Historical customer interactions are preferred over manually annotated data since the latter is costly to acquire and quickly becomes stale. Historical customer interactions are not always available when a new e-commerce platform is launched, or an existing platform is extended to new products. In this paper, we present the use of *inverse propensity weighting* as a strategy to create datasets where no historical customer interaction is available. We show results of an online A/B test which demonstrate the efficacy of our proposed strategy. Due to a simple practical application of the proposed strategy, we believe this can help LTR practitioners.

CCS CONCEPTS

• **Applied computing** → **Online shopping**; • **Information systems** → **Learning to rank**.

KEYWORDS

datasets, covariate shift, transfer learning, cold start, e-commerce LTR

ACM Reference Format:
Paul Missault, Arnaud de Myttenaere, Andreas Radler, and Pierre-Antoine Sondag. 2021. Addressing Cold Start With Dataset Transfer In E-Commerce Learning To Rank. In *Proceedings of Knowledge Management in e-Commerce (KMM-ecomm '21).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/XXXXXX.YYYYYY

1 INTRODUCTION

Product search is tasked with two distinct problems. First, a customer query needs to be *matched* to the set of documents that are related to the customer query. This matching is typically done using an inverted index [2]. Second it needs to rank those matched documents in such a way that users find what they are looking for without needing to sift through many pages of irrelevant results. Finding optimal strategies to do this ranking is an active research topic [3–5].

LTR models typically optimise a loss [1, 9] defined on a dataset of user interactions. In the context of e-commerce these interactions are typically clicks, add-to-carts or purchases. A user in this context is often referred to as a customer. Using a dataset of historical interactions that is representative of the online use-case is essential in achieving good results when training an LTR model. This raises the question on how an LTR model can be trained when data is unavailable. When a new e-commerce platform is launched, a dataset of customer interactions might not be available. Even within existing e-commerce platforms a *cold start* can occur when a new set of products is made available to customers.

This paper presents a strategy to efficiently transfer datasets to a new domain. The rest of the paper is organized as follows: first, we briefly sketch the context of a ranking model and present a scientific framework that enables usage of a dataset from another domain in Section 2. In Section 3 we show how to apply this framework, and highlight the simplicity of application. Then we discuss the results provided by an online experiment in Section 4, and discuss the made assumptions in Section 5. Section 6 concludes this document and mentions possible future works.

2 BACKGROUND

2.1 Context

Let us consider a product search engine context, where a user sends a query q and receives a sequence of products selected by a ranking algorithm in production.

When the user analyses some of the products, he/she may generate actions (click, add, purchase, ...). These actions lead to feedbacks (also called rewards) and can be used to optimize the ranking model in production, or to train a new model. To do so, user sessions are usually sampled from the logs to build a training dataset. Let's assume that:

- queries and resulting products hitting the search engine are i.i.d. observations drawn from a random variable Q,
- queries and resulting products in the training dataset $\mathcal{D}$ are i.i.d. observations drawn from Q,

KMM-ecomm '21, April 16, 2021,

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/XXXXXX.YYYYYY

- a query - product pair can be sampled from Q and represented in "feature" vector X,
- the ranking algorithm aims at minimizing a loss function ℓ (e.g. Normalized Discounted Cumulative Gain [2]),
- the class of LTR models considered is $\mathcal{G}$ (e.g. linear models, gradient boosted decision trees, etc),
- r_i is the reward associated to a query q_i (could be Sales, Clicks, etc).

Then the empirical loss of a model $g \in \mathcal{G}$ on the training dataset $\mathcal{D}$ is defined as

$$L_{\mathcal{D}}(g) := \sum_{(r,X)\in\mathcal{D}} \ell(g(X), r)$$

Training a model consists in finding the model $g^\star$ that minimizes the empirical loss on the training dataset, that is

$$g^\star = \arg\min_{g\in\mathcal{G}} \sum_{(r,X)\in\mathcal{D}} \ell(g(X), r) \tag{1}$$

This strategy is called empirical risk minimization and ensures that $g^\star$ will perform well once in production if queries hitting the index and queries in $\mathcal{D}$ are sampled from the same distribution.

2.2 Dataset Availability

As mentioned in section 1, the goal of this paper is to present a strategy to train a ranking model on a new domain, where training data $\mathcal{D}$ is not (yet) available. Although training a model without data seems impossible, in practice it is often possible to use data from other domains. For example, if the e-commerce platform is currently available for customers in some domain, the logs from that domain can serve as a dataset for related domains. The dataset gathered in the established domain can simply be sampled from the user interactions in that domain. When we want to use that data in a new domain, we hypothesize that the distribution of the ranking features vector X in the established domain do not match the distribution of the features in the new domain. Differences in feature distribution will be discussed in more detail in Sections 2.3 and 5.1.

Formally, training the ranking model for a domain of products on the dataset of a different domain introduces a covariate shift problem [7, 8], which can be solved by reweighting the datapoints in the training dataset.

2.3 Covariate shift problem

Training LTR models for new domains is challenging in practice since by definition there is no data available on a new domain. This problem could be addressed by sampling queries from an existing domain. However, if the training data $\mathcal{D}$ collected on the existing domain does not reflect the distribution we observe in the new domain Q, we can no longer assume that a model that minimizes Equation 1 will perform well on the new domain.

Figure 1 represents the marginal distribution of 4 different ranking features (i.e. dimensions of vector X) in different Amazon retail websites. A website in this case refers to the amazon retail websites for a country, examples are "amazon.de" for Germany or "amazon.com" for the US. We notice how in all 4 features, the Cumulative Distribution Function (CDF) of websites 2-4 are more similar as the outlier website 1.

Notice that Figure 1 represents the CDF of the feature. While we expect query-product pairs to have unique values in different retail websites, a difference in the CDF highlights that the distribution of the feature is different. This shows that if we were to sample data from website 1 and train a model to be used on website 3 we would introduce a covariate shift problem since: $P_{website_1}(X) \neq P_{website_3}(X)$.

In this case, the training set is biased and the model selected using equation 1 might perform poorly once in production. As mentioned in [7], one solution to train a model for a new domain consists in re-weighting each instance in the training dataset to mimic the expected distribution on new domains. More precisely, in our case covariate shift theory[1] shows that each sample X_i in the training dataset should be weighted by a factor $\frac{p_Q(X_i)}{p_{\mathcal{D}}(X_i)}$ where $P_{\mathcal{D}}$ is the distribution in the source domain (e.g. a retail segment where we can collect data) and P_Q is the distribution in the target domain (e.g. a retail segment where we test the LTR model).

This leads to a new model, defined by

$$g^\star = \arg\min_{g\in\mathcal{G}} \sum_{(r_i,X_i)\in\mathcal{D}} \omega(X_i)\cdot\ell(g(X_i), r_i) \quad \text{with } \omega(X_i) = \frac{p_Q(X_i)}{p_{\mathcal{D}}(X_i)}. \tag{2}$$

Note: in practice, any weight proportional to $\frac{p_Q(X_i)}{p_{\mathcal{D}}(X_i)}$ should fix the problem. Also notice how the denominator can not be 0 since it would be impossible to sample an X from $\mathcal{D}$ where $P_{\mathcal{D}}(X) = 0$

2.4 Addressing the covariate shift problem

Estimating the weight ω is non-trivial since $\mathcal{D}$ is a high dimensionality random variable and Q is not observed on new markets. Furthermore it is important to note that the dimensions of feature vector X are not guaranteed independent and so $P(X) \neq \prod_{i=1}^{N} P(x_i)$. For this reason looking at CDF's of individual features as presented in Figure 1 can give an intuition but the distribution of the individual feature does not tell us the relation of $P_Q(X)$ and $P_{\mathcal{D}}(X)$.

So while inverse propensity scoring is guaranteed to be optimal, we need to make some simplifying assumptions in order to use it in practice. We propose 2 strategies:

Strategy 1: We can assume that there is no bias, so $\omega(X) = 1$. In practice this assumption would hold if the distribution of features in the source domain is the same as in the target domain.

Strategy 2: In strategy 2, we make 3 assumptions.

Assumption 1. *The distribution of features for a query-product pair depend only on how often that query has been searched before.*

In other words: the distribution of features for a given *query frequency* is identical across Q and $\mathcal{D}$. Consider:

[1] This strategy, also known as Inverse Propensity Scoring, is valid since $\mathbb{E}_{\mathcal{D}}[\omega\ell] = \int \ell(g(X), r)\frac{dp_Q(X)}{dp_{\mathcal{D}}(X)}dp_{\mathcal{D}}(X) = \int \ell(g(X), r)dp_Q(X) = \mathbb{E}_Q[\ell]$

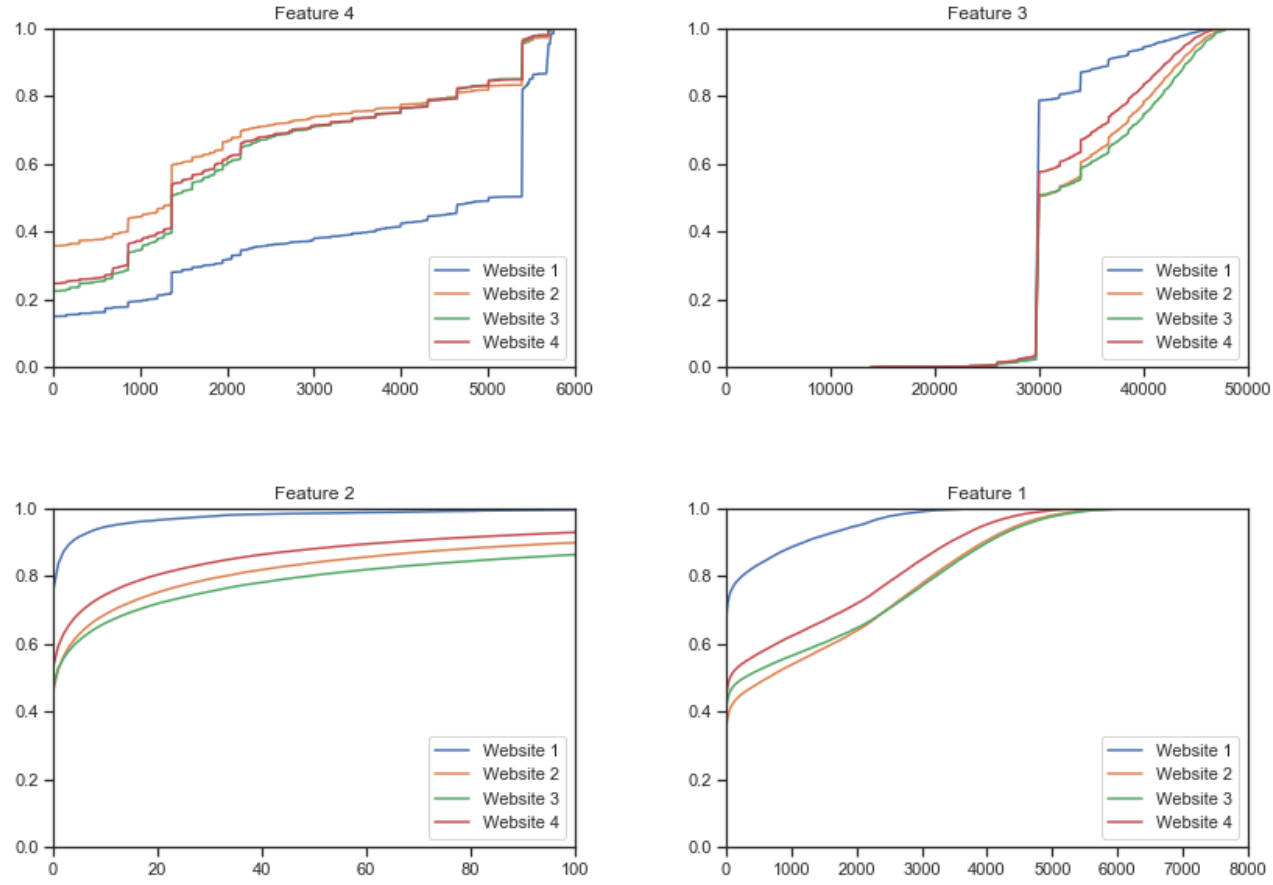

Figure 1: Comparing CDF's of ranking features

$$\omega(X_i) = \frac{p_Q(X_i)}{p_{\mathcal{D}}(X_i)} = \frac{p_Q(x_f)}{p_{\mathcal{D}}(x_f)} \frac{p_Q(X_i|x_f)}{p_{\mathcal{D}}(X_i|x_f)} \quad x_f \text{ is the query frequency} \quad (3)$$
$$\approx \frac{p_Q(x_f)}{p_{\mathcal{D}}(x_f)} \quad \frac{p_Q(X_i|x_f)}{p_{\mathcal{D}}(X_i|x_f)} \text{ assumed to be 1}$$

Assumption 2. *In any domain Q the query frequency will reach a maximum value f_M where $P_Q(x_f > f_M) = 0$.*

From Equation 3 we see this means $\omega(X_i) = 0$ or equivalently, we can filter out all X_i from the training dataset where query frequency $> f_M$.

Assumption 3. *We will assume $\frac{p_Q(x_f)}{p_{\mathcal{D}}(x_f)} = c$ for all $x_f < f_M$.*

This assumption implies that the distribution of the query frequency is identical (discarding the scaling factor c) below some threshold. In practice the scaling factor c it can be set to 1 since in Equation 2 the $g^\star$ that solves the optimisation will be identical for ω and $c\omega$. In what follows we will assume $c = 1$.

2.5 Discussion of assumptions

Neither of the 3 assumptions are guaranteed to be correct. In fact, it is possible to find counter-examples to all 3. However under these assumptions, strategy 2 only introduces 1 extra hyperparameter on your model design, f_M. Once f_M is chosen, you can sample from domain Q only those X where $x_f < f_M$. Alternatively, for an existing training dataset D, a modeller need only filter those datapoints where $x_f \geq f_M$.

Intuitively, this introduction of a parameter f_M can be explained when we refer back to the reason we make these assumptions. We are trying to transfer a dataset from an existing domain to train a model for a new domain. In a new domain all of the queries will be new, and so the behavior of features in a this new domain will be closer to the distribution of infrequent (with "infrequent" meaning $x_f < f_M$) queries from the established domain.

In the discussion of an on-line A/B test in Section 3 compelling proof will be given to show the efficacy of the proposed strategy

2. Section 4 we will deeper investigate the assumptions made in Strategy 2 and show that an ω generated by this strategy will outperform strategy 1 even though in reality $p_Q(X_i|x_f) \neq p_{\mathcal{D}}(X_i|x_f)$ and $\frac{p_Q(x_f)}{p_{\mathcal{D}}(x_f)} \neq c$.

3 APPLICATION

In what follows we will present an online A/B test that was run when a new Amazon retail segment was made available to customers. The A/B test compares the results of the proposed strategy against the LTR model previously used at new segment launch. The source training dataset $\mathcal{D}$ is sampled from comparable segments on which data is available.

We trained 2 LTR models, where each model followed one of the strategies mentioned in Section 2.4. (1) We sampled training data from training dataset $\mathcal{D}$ without any weighting, following strategy 1. (2) To train with strategy 2, we need to find the value of f_M.

The value of f_M is unknown in a new segment since we need data in order to compute the exact value at time of training. Since f_M has an intuitive meaning (the frequency of the most common query) it is possible to make some estimates based on the expected query frequency of the new segment. In our application, we used traffic estimates and offline experimentation of the new segment to set f_M.

The models trained with these strategies will use identical features and identical hyper-parameters to allow for a fair comparison.

4 RESULTS

The results of experimental A/B testing are shown in Table 1. The A/B test is issued in 3 treatments. The control group of the A/B test is an LTR model previously designed for that new segment launch (our baseline). The two treatments are the models trained using strategies 1 and 2. The table shows the relative changes in search-attributed revenue for the treatments following strategy 1 and 2 compared to the control group. In search-attributed revenue we only consider those purchases that have been made as the direct result of a search. I.e. purchases that happen without clicking through from a search result are excluded from search-attributed revenue.

Treatment	Search-attributed revenue
Strategy 1	+0.56% (p=0.87)
Strategy 2	**+6.87%** (p=0.06)

Table 1: A/B results after 28 days. The strategies are described in Section 2.4

Interestingly the LTR model trained using strategy 1 was unable to show a significant improvement (or deterioration) compared to the baseline model. The model trained using strategy 2 on the other hand was able to reach a significant improvement on the search-attributed revenue. This result will be discussed further in Section 5.2.

5 DISCUSSION

5.1 Hypothesis validation

Since the segment on which we ran the experiment was launched after we trained the model, we are able to capture data from new domain Q and validate some of our choices and assumptions. We can investigate if the training dataset generated with strategy 2 yields a feature distribution that is more similar to the source domain as one generated with strategy 1.

To assess the similarity of the training datasets with one sampled from the new domain, we set up a domain classifier. We use a random forest classifier from scikit-learn [6] using $max\_depth = 4$ and $n\_estimators = 1000$ that needs to distinguish between 3 classes of data. 1) Data pulled from the target domain Q, 2) Training dataset D 3) Training dataset D where $x_f < F_M$. The confusion matrix of this classifier can be seen in Figure 2.

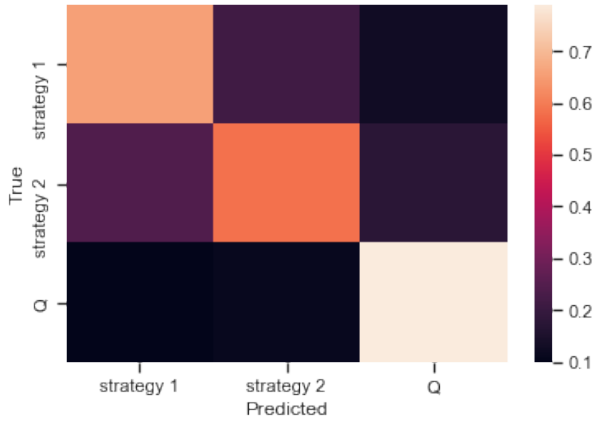

Figure 2: confusion matrix of dataset classifier

The confusion matrix shows that the model has the greater tendency to confuse datapoints from Q with those sampled using strategy 2. Since the confusion is still limited, the assumptions made in Section 2.4 are not fully accurate, but this confusion matrix shows that conditioning on the query frequency brings the distributions closer together.

5.2 Discussion of A/B testing results

The assumptions made in strategy 2 of Section 2.4 were quite strong and oversimplify the reality of feature distributions. However, using this strategy the results indicate that these assumptions improve results in an online A/B test.

The fact that a model trained with strategy 1 (i.e. naively take a dataset from other segments) could not achieve significant results shows that out-of-domain sampling without adjusting for the covariate shift is unlikely to perform well in an online test. By conditioning the training data on the query frequency, we were able to get a significant win compared to the baseline. This indicates the model trained using strategy 2 closer resembles the new domain in which it was tested.

Intuitively we believe these results make sense. In a new domain all of the queries are unique or infrequent. Our experiment and

discussion in Section 5.1 then showed that the distribution of ranking features in such a new domain are closer to the distribution of unique and infrequent queries of an existing domain.

We believe the ease at which this strategy can be applied means it can have large implications for all LTR practitioners. Filtering training datasets to mimic the expected query frequency of the target domain can often be easily done and as shown in the online results it can have a significant impact. The coupling of this strategy to Inverse Propensity Scoring means there is a theoretical background behind this choice, and when more information is known about the target domain a practitioner can chose to vary on the proposed strategy by removing one of the assumptions.

6 CONCLUSION AND FUTURE WORK

In this paper we have sketched a strategy surrounding transfer learning and covariate shift in product ranking at Amazon. We show using an online A/B tests that ranking models for new domains can be improved by conditioning the training dataset on the expected query frequency of the new domain. We have shown that even though we enforce some hard assumptions, the results are promising and the actual application is simple.

As future work we want to investigate how the out-of-domain dataset can be augmented by in-domain samples. The work described in this paper is essential when no data about the new domain is available, but once the new domain is visible to customers real data can be sampled. As you start collecting data, it can be used to estimate the expected distribution on the target domain and find a better f_M. Or even directly estimate the ratio $\frac{p_Q(X_i)}{p_{\mathcal{D}}(X_i)}$ and retrain a better model. It is currently unknown when we can best use this data, and at what point we have *enough* data to fully train on the new domain.

REFERENCES

[1] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[2] U. Cambridge. *Online edition (c) 2009 Cambridge UP An Introduction to Information Retrieval Christopher D.* Manning Prabhakar Raghavan Hinrich Schütze Cambridge University Press …, 2009.

[3] R.-C. Chen, L. Gallagher, R. Blanco, and J. S. Culpepper. Efficient cost-aware cascade ranking in multi-stage retrieval. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 445–454, 2017.

[4] S. et al. 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*, pages 373–382, 2015.

[5] T.-Y. Liu et al. Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331, 2009.

[6] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[7] H. Shimodaira. Improving predictive inference under covariate shift by weighting the log-likelihood function. *Journal of statistical planning and inference*, 90(2): 227–244, 2000.

[8] M. Sugiyama, M. Krauledat, and K.-R. MÃžller. Covariate shift adaptation by importance weighted cross validation. *Journal of Machine Learning Research*, 8 (May):985–1005, 2007.

[9] X. Wang, C. Li, N. Golbandi, M. Bendersky, and M. Najork. The lambdaloss framework for ranking metric optimization. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 1313–1322, 2018.