Sample Optimization For Display Advertising

Anonymous Authors

ABSTRACT

Sample optimization, which involves sample augmentation and sample refinement, is an essential but often neglected component in modern display advertising platforms. Due to the massive number of ad candidates, industrial ad service usually leverages a multi-layer funnel-shaped structure involving at least two stages: the candidate generation stage and the ranking stage. Given a user's feature vector and its context feature, an online approximate nearest neighbor (ANN) search module efficiently retrieves top-*N* most relevant ads by measuring the similarity between the user feature vector and ad feature vectors. With the top-*N* candidates, a ranking module generates a short ordered list and delivers it to the user. An offline candidate-generation model is often trained based on past click/conversion data to obtain the user feature vector and ad feature vector. However, there is a covariate shift problem between the user observed ads and all possible ones. As a result, the candidate generation model trained from the past click history cannot fully capture users' potential intentions or generalize well to unseen ads. In this paper, we utilize several sample optimization strategies to alleviate the covariate shift problem for training candidate generation models, including weighted random negative sampling, real-negative subsampling, sample refinement with positive-unlabeled learning, fuzzy positive sample augmentation, and sampling with noise contrastive estimation. We have launched these strategies in a commercial display ad platform and achieved considerable improvements in offline metrics, including both offline click-recall, cost-recall, and online metric cost per mille (CPM).

ACM Reference Format:
Anonymous Authors. 2020. Sample Optimization For Display Advertising. In *The 29th ACM International Conference on Information and Knowledge Management (CIKM'20), October 19–23, 2020, Online.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Display advertising is a primary source of revenue for online ad publishers like Youtube and Facebook, which sell ad space to advertisers and show their ads to end users. The expected revenue for publishers is often measured by cost per mille (CPM), which is calculated as the product of the bid price and click-through rate (CTR) or conversion rate (CVR). Therefore intensive efforts have been made to improve CTR/CVR and CPM model, meanwhile guarantee user experience.

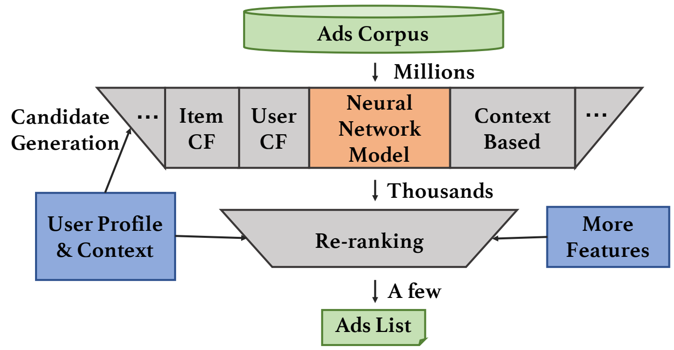

Figure 1: The illustration of a typical ads serving system. It mainly contains two stages: (i) candidate generation, which reduces the ads corpus from millions to thousands; (ii) re-ranking, which further ranks top candidates and generate the final ads list.

Due to the tremendous amount of ad corpus, industrial ads ranking systems usually leverages a multi-layer funnel-shaped structure with at least two stages [3], (i) the candidate generation stage, and (ii) the re-ranking stage, as shown in Figure 1. With a user's feature vector and its context feature vector, the candidate generation stage reduces the corpus size from millions to thousands or hundreds. And then, the re-ranking stage employs some complex models and extra features to generate the final ads list and shows it to users. The candidate generation process is vital since it is required to efficiently rank and produce a small but proper subset from a large corpus.

In real display ads systems, there are usually multiple generation methods running in parallel in the candidate generation stage to ensure a proper recall. Those methods include but not limited to neural network based ranking models [3, 7], user-based, and item-based Collaborative Filtering (CF) [1], context-based ranking, etc. Among them, neural network based models, such as DSSM [7] and Softmax models [3], usually dominate others. For example, in our ads system, we generate 60% of candidates using DSSM [7].

From a data perspective, there are several challenges to train a proper candidate generation model:

(1) The distribution of the training set and the inference set are dramatically different, and the ranking results cannot be fully evaluated. In a real display ads system, an approximate nearest neighbor search (ANN) [3] method is often used to get approximate top hundreds of candidates from millions of ads. And then, these results are merged with other sources and pass through the followed re-ranking procedure. Finally, only a few ads are survived and shown to users. Following the above procedures, the observed data at end-users (i.e., clicked or not clicked) are quite different from the inference set (i.e., the full ads set). As a result, the ranking model cannot fully capture users' potential

CIKM'20, October 19–23, 2020, Online

ACM ISBN 978-1-4503-XXXX-X/18/06…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

intentions and generalize well to unseen samples. This problem is also called sample selection (survival) bias or covariate shift.

(2) The real-world ads impression data is usually in a long-tail distribution and high-frequency ads only cover a tiny proportion of all population. However, they are more important or have higher bid prices than others. Since most exposed ads are not clicked, many high-frequency ads are included in the negative training set. During training, these ads may be suppressed, which leads to a revenue drop.

(3) Unclicked ads are not necessarily to be real negative samples. In display ads serving systems, most exposed ads are not clicked for various reasons. Therefore we cannot claim they do not match users' interests. Such uncertainty challenges the model training since it is difficult to differentiate which samples are true negatives. most ads are not clicked, the training data is very sparse. Especially the positive training samples are seriously inadequate.

Lots of papers focus on designing better ranking models [4, 7, 11, 14, 15], but very few works study how to train a better candidate generation model from a data perspective. In this paper, we design and study several sample optimization strategies, including weighted random negative sampling, real-negative subsampling, sample refinement with PU learning, fuzzy positive sample augmentation, and sampling with noise contrastive estimation. These strategies are designed to solve or alleviate the above-enumerated challenges, respectively. We have launched these strategies in a commercial display ads platform and achieved considerable improvements in offline metrics, including both offline click-recall, cost-recall, and online metric cost per mile (CPM).

In summary, our contributions of this paper are as below:

- To resolve the covariate shift problem, we firstly investigate various sample optimization strategies for training better candidate generation models. Although we focus on display ads in this paper, most of the strategies are also applicable in general personalized search and recommendation systems.
- We have conducted extensive offline and online experiments to evaluate these strategies and summarized each strategy's contribution to the overall result.

2 RELATED WORK

Closely related work to ours is [2, 12]. The Entire Space Multi-task Model [12] simultaneously estimated CVR and CTR with a shared embedding layer, which has a probabilistic interpretation that CVR is modeled over the entire input space of all impressions. Bron et al. [2] reduced potential biases of each user attribute after analyzing ad feedback data with the deviance statistic [6].

We are mainly motivated to combating the covariate shift problem from a data perspective. Unlike Ma et al. [12] that defined their entire space over all exposed ads, we explore a larger sample space over all ads. In particular, we focus on a set of sample optimization strategies to optimize the training set for candidate generation models. Compared with the "explore and exploit" method [10] that runs online experiments to improve data quality, our method is low cost and does not hurt the user experience or the revenue.

3 METHODOLOGY

In this paper, we denote the click history to be $\mathrm{S} = \{u_i, a_i, y_i\}|_{i=1}^{N}$, where u_i is the ith user, a_i is the showed ad, and $y_i \in \{0, 1\}$ is the user's action towards the ad.

3.1 Weighted Random Negative Sampling

A natural way to enrich the training set is to use negative sampling [13], which randomly selects k ads as negative samples from all ads for each positive sample. Since the ad impression frequency is long-tailed, our sampling strategy is different from Mikolov et al. [13] in that we adopt a piece-wise weighted negative sampling method.

In particular, we split all ads based on their impression frequency. Let $\mathcal{A} = \mathcal{A}_h \bigcup \mathcal{A}_l$, where $\mathcal{A}_h = \{a : f(a) > \alpha\}$, $\mathcal{A}_l = \{a : f(a) \leq \alpha\}$ and α is the splitting threshold. When sampling a random negative sample, we first generate a random number $p \sim U(0, 1)$. If $p < p_l$, we uniformly sample an ad from $\mathcal{A}_l$, where $p_l = \sum_{a_i \in \mathcal{A}_l} f(a_i)^{t_1} / \sum_{a_j \in \mathcal{A}} f(a_j)^{t_1}$. Otherwise, we follow Mikolov et al. [13] to sample an ad from $\mathcal{A}_h$ according to a "unigram distribution" defined as $P(a_i) = f(a_i)^{t_1} / \sum_{a_j \in \mathcal{A}_h} f(a_j)^{t_1}$.

Since $\|\mathcal{A}_l\| \gg \|\mathcal{A}_h\|$, compared with the original negative sampling method, our strategy significantly reduces memory usage while preserving the original properties. We set $k = 1$ and $t_1 = 0.75$ as suggested in [13], and $\alpha = 15$ according to the impression frequency distribution.

3.2 Real-Negative Subsampling

In our platform, the overall CTR is only around 0.03%. Due to the long-tailed distribution of impression frequency, a few top ads dominate the overall impressions, and they may occur in both the positive set (showed and clicked) and real negative set (showed but not clicked). We do not want the top ads occurring in the real-negative set to be over-suppressed since they usually have high business value. Instead of using all the negative samples for training, we perform subsampling similar to the procedure in dealing with frequent words in word2vec [13].

Specifically, each negative triplet with an ad having high impression frequency in the training set is discarded with probability computed by the formula: $p(i) = 1 - (\beta / \tilde{f}(a_i))^{t_2}$, where $\tilde{f}(a_i))$ is the normalized impression frequency $f(a_i)$ and β is a chosen threshold, typically around 10^{-5} and $0 < t_2 < 1$.

We use this subsampling formula because it only ignores ads whose frequency is greater than β according to a probability monotonic w.r.t. frequency. After subsampling, the expected number of top negative samples is $f(a) * (\beta / \tilde{f}(a))^{t_2}$ and we set $t_2 = 0.75$ in our experiments. Although this strategy is chosen heuristically, we find it works well in practice. It significantly improves offline recall and online CPM metrics, as shown in our experiment.

3.3 Sample refinement with PU Learning

Traditionally, ads showed but not clicked by users are treated as negatives. However, unclicked ads are not necessarily to be irrelevant to users. Hence we can view the click history data as a combination of positive (clicked) and unlabeled (reliable negative + potentially positive) and refine the negative set to include only those reliable negative samples.

Towards that end, we employed a positive unlabeled learning algorithm "spy technique" [9] to find reliable negative samples with the following steps:

- Randomly sample a "spy set" S from the positive set $\mathcal{P}$ and add $\mathcal{S}$ to the unlabeled set $\mathcal{U}$.
- Treat $\mathcal{P} \setminus \mathcal{S}$ as the positive set and $\mathcal{U} \bigcup \mathcal{S}$ as the negative set and train a biased SVM classifier [8] using the concatenation of user and ad feature vector.
- Use the trained classifier to score each element in $\mathcal{U}$ how likely it is a positive sample with $p(y=1|u,a)$.
- Compute the average probability $\bar{p} = \sum_{(u,a)\in\mathcal{S}} p(y=1|u,a)/|\mathcal{S}|$ over the spy set.
- Construct the reliable negative training set: $\mathcal{RN} = \{(u,a,0) : p(y=1|u,a) < \bar{p}\}$ *and* $(u,a) \in \mathcal{U}$.

With the refined negative set $\mathcal{RN}$ and positive set $\mathcal{P}$, we can train a candidate generation model.

3.4 Fuzzy Positive Sample Augmentation

To alleviate the data scarcity problem, we introduce a fuzzy logic to augment positive samples. In the final ads list, only the top few ads are shown to the user, while the rest may not get shown. Although those hidden ads cannot directly serve as training samples, they have passed candidate generation and re-ranking stage, and are more likely to match the user's interest.

To augment positive samples, we parse the undisplayed event log and collect all triplets of (user, ad, CPM) in the final list with CPM higher than a predefined threshold. We call these triplets (user, ad, CPM/bid) as "fuzzy positive samples" and add them to the positive training set. It is worth noting that the labels of fuzzy positive samples are less than 1 since they are not clicked records.

3.5 Sampling with Noise Contrastive Estimation (NCE)

NCE [5] is a sampling procedure typically used to train classifiers with an ample output space. Using NCE, we aim to discriminate between samples from the "real" distribution and an artificially generated noise distribution. NCE is very similar to Negative Sampling in implementation but with more theoretical guarantees.

To train a candidate generation model, we use $p_n(a) = f(a)^t$ as the noise distribution, where $f(a)$ is the ad a's display frequency and $t = 0.75$ is a hyper-parameter. Similar to the random negative sampling strategy introduced above, we divide all ads into two sets based on their display frequency with a threshold of 15. For each positive sample, we randomly sample $k = 5$ ads and construct 5 negative samples. Each negative ad is either sampled from the high-frequency region based on the noise distribution or from the low-frequency region uniformly.

4 EXPERIMENT

4.1 Dataset

We collect the impression events on 01/11/2020 from our display ads platform. Since click events are rare, we keep all of them as the positive set. We also random sample 1% of unclicked events, resulting in total 300M impressions for training DSSM [7].

To evaluate our sample optimization strategies, we collect two types of test data from impression logs on 01/12/2020:

(1) Skip the ranking stage and directly show the top ads from our candidate generation model. The purpose of skipping the ranking stage is to eliminate the effect of ranking models on the display order to users. Collect the clicked impressions with bidding information as ground truth to compute evaluation metrics. We refer to this test set as ***Unbiased Test Set***.
(2) Collect another test set similar to the *Unbiased Test Set* except that we keep the ranking stage and show the reranked ads to users. We refer this test set as ***Biased Test Set***.

Since the unbiased online experiment only covers a small group of users, we collect 2.7M samples in the unbiased test set compared with 200M in the biased set. Table 1 summarizes the statistics of the three datasets.

Table 1: Summary statistics of datasets.

Dataset	Total	Positive	Negative
Training Set	300M	18M	292M
Unbiased Test Set	2.7M	70K	2.63M
Biased Test Set	200M	12M	188M

4.2 Evaluation Metric

We conduct both offline and online evaluations. Since our focus is to improve the candidate generation model, recall is our primary metric to evaluate how relevant the returned ad candidates are. Given a user in the test set, we construct the user embedding vector using the trained DSSM model and apply ANN [3] to search its top $N = 240$ ad candidates. Then we check the returned top-N candidates against the user's actual clicked ads. For offline metrics, we compute the click-recall and the cost-recall, which measure the percentage of relevant ads and the percentage of the total cost related to the relevant ads among the returned top candidates.

The $click{-}recall$ is defined as follows:

$$\begin{aligned} click(L_u) &= \frac{count(L_u \cap B_u)}{count(B_u)} \\ click{-}recall &= \frac{\sum_{u\in U} click(L_u)}{|U|} \end{aligned} \tag{1}$$

where L_u is top-N ad candidates for user u, B_u is the set of ads with u's positive feedback, $|U|$ is the number of users in the positive set.

The $cost{-}recall$ is defined as follows:

$$cost{-}recall = \frac{\sum_{u\in U} G(L_u \cap B_u)}{\sum_{u\in U} G(B_u)} \tag{2}$$

where $G(L_u \cap B_u) = \sum_{a\in L_u\cap B_u} bid(a)$, and $G(B_u) = \sum_{a\in B_u} bid(a)$.

Besides recall, we also compute the AUC of CTR prediction. Since we have launched these strategies in our display ads system, we also report the relative improvement of online CPM as CPM↑= $(CPM_{new} - CPM_{old})/CPM_{old}$, where $CPM = bid * CTR$.

4.3 Results and Analysis

During our survey, we did not find any previous works focusing on sample optimization to improve candidate generation models. Therefore, we mainly present the offline and online results of our strategies and an ablation study.

Table 2: Experimental Results for the five sample optimization strategies, in unbiased and biased, offline and online tests.

Strategies	Offline Test						Online Test
	Unbiased Test			Biased Test			CPM↑
	AUC	click-recall	cost-recall	AUC	click-recall	cost-recall	
Base	0.8994	0.09%	0.44%	0.8836	0.71%	1.71%	0%
Base+(a)	0.8738	1.67%	0.59%	0.8825	8.95%	7.72%	+1.7%
Base+(a)+(b)	0.8763	2.28%	0.73%	0.8801	14.66%	14.30%	+1.7%+2.0%
Base+(a)+(b)+(c)	0.8758	2.10%	0.59%	0.8802	15.16%	14.56%	+1.7%+2.0%+0.5%
Base+(a)+(b)+(c)+(d)	0.8748	2.85%	2.42%	0.8777	15.54%	18.42%	+1.7%+2.0%+0.5%+2.0%
Base+(a)+(b)+(c)+(d)+(e)	0.8557	4.76%	4.45%	0.8745	23.98%	29.05%	+1.7%+2.0%+0.5%+2.0%+1.8%

Table 3: Ablation study for each optimization strategy.

Strategies	Offline Test						Online Test
	Unbiased Test			Biased Test			CPM↑
	AUC	click-recall	cost-recall	AUC	click-recall	cost-recall	
All	0.8557	4.76%	4.45%	0.8745	23.98%	29.05%	8%
All-(a)	0.8537	3.12%	2.65%	0.8774	17.28%	18.98%	-1.6%
All-(b)	0.8555	1.67%	2.74%	0.8777	19.59%	19.76%	-1.9%
All-(c)	0.8551	4.89%	4.24%	0.8751	22.86%	26.04%	-0.5%
All-(d)	0.8516	4.88%	3.49%	0.8752	24.07%	23.05%	-1.9%
All-(e)	0.8748	2.85%	2.42%	0.8777	15.54%	18.42%	-1.7%

We show the experimental results in Table 2 and Table 3. For simplicity, we denote the **Base** method: clicked data as positive and unclicked as negative. And then the five sample optimization strategies are: **(a)** weighted random negative sampling, **(b)** real-negative subsampling, **(c)** sample refinement with PU learning, **(d)** fuzzy positive sample augmentation, **(e)** sampling with noise contrastive estimation (NCE). **All** is for Base+(a)+(b)+(c)+(d)+(e).

In Table 2, we show different ways to ensemble the strategies and their corresponding performance. Note that the combination order is strictly following our product iteration. Our first observation is that AUC is not improved for both test sets. The reason is that our strategies enable the candidate generation model to explore more user preferences instead of purely fitting on historical ads exposure data, especially the top 1st ad that is used to compute AUC. Sacrificing a certain AUC, we significantly improve offline recall and online CPM↑ with a consistent trend, which demonstrates that our strategies can help recalling more relevant ads in the candidate generation stage. Note that recall on the unbiased test set is smaller than that of the biased one since the candidate generation model is trained on the ranked and exposed data. To summarize, when combining the five strategies, we obtain the best offline and online results, which is consistent for both test sets.

To study how much contribution each strategy can make to the full model, we conduct an ablation study in Table 3. We observe that the offline results are consistent between the Unbiased Test and the biased Test. For the online CPM improvement metric, **(a)**, **(b)** and **(e)** contribute most with similar trend for both offline and online experiment, while PU-learning contributes the least. We also observe that fuzzy positive samples **(d)** generates opposite results between click-recall and online CPM↑. One possible reason is that click-recall (ignoring bid) and online CPM (considering bid) have different objectives, which may cause inconsistent outcomes.

5 CONCLUSION

In this paper, we utilize several sample optimization strategies to alleviate the survival bias problem for the candidate generation model. Experiments on real offline datasets demonstrate the utility of the proposed strategy. We have launched these strategies in our display ads product line and achieved considerable improvement in cost per mille (CPM) online.

REFERENCES

[1] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. *Knowledge-based systems* 46 (2013), 109–132.

[2] Marc Bron, Ke Zhou, Andy Haines, and Mounia Lalmas. 2019. Uncovering Bias in Ad Feedback Data Analyses & Applications10033. In *World Wide Web Conference, WWW*. 614–623.

[3] Paul Covington, Jay Adams, and Emre Sargin. 2016. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*. 191–198.

[4] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W. Bruce Croft. 2016. A Deep Relevance Matching Model for Ad-hoc Retrieval. In *International Conference on Information and Knowledge Management (CIKM)*. Indianapolis, IN, 55–64.

[5] Michael Gutmann and Aapo Hyvärinen. 2010. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics, AISTATS*. 297–304.

[6] David W. Hosmer and Stanley Lemeshow. 2004. *Applied logistic regression* (2 ed.). Wiley-Interscience Publication.

[7] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2333–2338.

[8] Bing Liu, Yang Dai, Xiaoli Li, Wee Sun Lee, and Philip S. Yu. 2003. Building Text Classifiers Using Positive and Unlabeled Examples. In *IEEE International Conference on Data Mining (ICDM*. 179–188.

[9] Bing Liu, Wee Sun Lee, Philip S. Yu, and Xiaoli Li. 2002. Partially Supervised Classification of Text Documents. In *ICML 2002, University of New South Wales, Sydney, Australia*. 387–394.

[10] Honglei Liu, Anuj Kumar, Wenhai Yang, and Benoît Dumoulin. 2018. Explore-Exploit: A Framework for Interactive and Online Learning. *NIPS* (2018).

[11] Zhengdong Lu and Hang Li. 2013. A Deep Architecture for Matching Short Texts. In *Advances in Neural Information Processing Systems (NIPS)*. Lake Tahoe, NV, 1367–1375.

[12] Xiao Ma, Liqin Zhao, Guan Huang, Zhi Wang, Zelin Hu, Xiaoqiang Zhu, and Kun Gai. 2018. Entire Space Multi-Task Model: An Effective Approach for Estimating Post-Click Conversion Rate. In *ACM SIGIR*. 1137–1140.

[13] Tomas Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013. Distributed Representations of Words and Phrases and their Compositionality. In *Advances in Neural Information Processing Systems*. 3111–3119.

[14] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. Learning semantic representations using convolutional neural networks for web search. In *WWW*. ACM, 373–374.

[15] Hamed Zamani, Mostafa Dehghani, W. Bruce Croft, Erik G. Learned-Miller, and Jaap Kamps. 2018. From Neural Re-Ranking to Neural Ranking: Learning a Sparse Representation for Inverted Indexing. In *CIKM*. Torino, Italy, 497–506.