Modeling Position Bias Ranking for Streaming Media Services

Matteo Ruffini
ruffinim@amazon.com
Amazon Music ML
Germany, Berlin

Vito Bellini
vitob@amazon.com
Amazon Music ML
Germany, Berlin

Alexander Buchholz
buchhola@amazon.com
Amazon Music ML
Germany, Berlin

Giuseppe Di Benedetto
bgiusep@amazon.com
Amazon Music ML
Germany, Berlin

Yannik Stein
syannik@amazon.com
Amazon Music ML
Germany, Berlin

ABSTRACT

We tackle the problem of position bias estimation for streaming media services. Position bias is a widely studied topic in ranking literature and its impact on ranking quality is well understood. Although several methods exist to estimate position bias, their applicability to an industrial setting is limited, either because they require ad-hoc interventions that harm user experience, or because their learning accuracy is poor. In this paper, we present a novel position bias estimator that overcomes these limitations: it can be applied to streaming media services without manual interventions while delivering best in class estimation accuracy. We compare the proposed method against existing ones on real and synthetic data and illustrate its applicability to Amazon Music.

CCS CONCEPTS

• **Information systems → Learning to rank**.

KEYWORDS

Position Bias, Music Recommendation, Multi Armed Bandits

ACM Reference Format:
Matteo Ruffini, Vito Bellini, Alexander Buchholz, Giuseppe Di Benedetto, and Yannik Stein. 2022. Modeling Position Bias Ranking for Streaming Media Services. In *Companion Proceedings of the Web Conference 2022 (WWW '22 Companion), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/3487553.3524210

1 INTRODUCTION

Streaming media services are used by billions of people on a daily basis. Despite their content variety – movies, music, sports, etc. – they share the issue of how to enable users to explore their catalogs, whose size is typically of millions of items. A common solution is to select user-personalized lists of items and to present them in a series of widgets, which are displayed on the application home page (see Fig. 1). Each widget typically contains items sharing some characteristics; for example, a music service may have a widget containing songs from an artist that the user likes. The content of each widget is ranked from left to right to match user preferences.

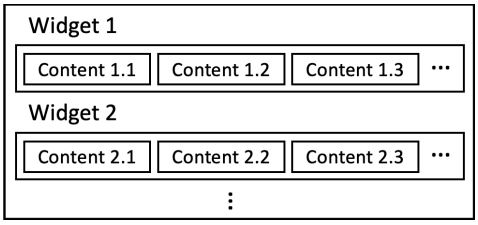

Figure 1: Homepage organization: widgets are listed vertically; the content of each widget is ranked horizontally.

Online learning-to-rank algorithms trained on implicit feedback are the industry standard for ranking items inside a widget, due to their practical advantages [1–4]. First, online methods trained one interaction after the other, allow to deploy a widget in production without the need of pretraining its ranker – a key advantage in a scenario where the lifespan of a widget is typically short[1], and no data for pretraining is available before the launch of a widget. Also, online ranking algorithms are equipped with exploration / exploitation policies [2, 3, 5], that overcome the selection bias arising in the top-k ranking scenario typical of this setting[2]. Last, training on implicit feedback [6] – i.e. deeming as relevant the items that were clicked by the user – allows to collect vast amounts of data with no need of any explicit user-provided relevance signal.

Implicit feedback, however, is frequently biased by the application interface [7], with *position bias* being a well known confounding factor [8]: items ranked in low-visibility positions are less likely to be examined, and they thus observe less interactions, even if they are relevant. An algorithm trained on raw click-data will thus be suboptimal [9]: low ranked items will continue to stay at the bottom just as a consequence of their lower visibility - which will lead users to examine them less and hence to be less likely to click on them. Several papers study how to correct the training procedure of ranking algorithms from position bias [3, 10–12]. These methods typically require a preliminary estimate of the position bias in terms of probability for a user to examine an item ranked in a given position, a task that is theoretically well understood and with several methods available in the literature [8, 13, 14]. Position

WWW '22 Companion, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9130-6/22/04…$15.00
https://doi.org/10.1145/3487553.3524210

[1] For example, a music service could have a widget like "Christmas Songs", that is only available during the Christmas period

[2] To avoid overloading the application interface, widgets can only display a limited amount of items, even if much more are available.

bias estimation methods, divide into two categories: a first class designs consistent estimators by executing additional randomization on the top of a production ranker [8, 14]; a second category formulates a maximum likelihood problem, which is solved using Expectation Maximization (EM) [13, 15]. These methods however, can be difficult to apply in the streaming media service scenario described above. The interventions required by the methods of the first family can harm customer experience [13]. EM instead, lacks of theoretical guarantees, often delivering suboptimal results [14].

In this paper we provide a novel position bias estimator that overcomes the limitations above when implemented in the streaming media services scenario. Our method, generalizes the estimators of the first class, but does not require additional randomization on the top of the production ranker. Instead, we leverage on the randomization naturally introduced by the exploration / exploitation policy typical of the online ranking systems described above. We present our approach in Section 2. Section 3 presents experimental results, where we show on real data that our method achieves best in class performance in terms of position bias estimation; also we experimentally prove 1) the harm caused by the manual interventions prescribed by the method from [8] and 2) the benefit of correcting a ranking algorithm with the estimated position bias.

2 ESTIMATING POSITION BIAS

The position bias model [16] is a click model describing user behavior when selecting items from a list. It assumes that a click on position i happens as the combination of two independent events: 1) the user examined the item in position i and 2) the item was relevant to the performed query. Formally, a click on the item in position i is modeled with the random variable C_i defined as $C_i = E_i Y_i$ where, $E_i \sim B(e_i)$ is a binary variable that is 1 if the user examines the item ranked in position i and 0 otherwise; e_i is the examination probability for the item ranked at position i. The parameter e_i only depends on the position i. $Y_i \sim B(rel(x_i))$, is also a binary random variable whose expected value is $rel(x_i)$, representing the relevance of the item ranked at position i. Here, x_i is the context of item i, and $rel(x_i)$ is the click probability if the item is observed.

We define the position bias at position i as e_i, and we call *position bias curve* the vector $(e_1, ..., e_k)$. Estimating the position bias from implicit feedback is crucial to use learning-to-rank algorithms that correct their training procedures from such confounder [3, 8, 10].

2.1 Existing Estimation Methods

The seminal work on position bias estimation is [8]. The method works by perturbing a base ranker by randomly swapping pairs of positions with 50% probability. The randomization introduced by the swaps allows to calculate consistent estimators of position bias; random swaps however, seriously impact ranking quality [13] – as we experimentally confirm in Section 3. The Intervention Harvesting method [14] introduces position bias estimators requiring much less intrusive interventions: they suggest to have $m \geq 2$ base rankers: at each query, one ranker is randomly chosen and used for ranking. The randomness needed to deliver position bias estimators here comes from the random selection of the ranker and from the fact that rankers are expected to have certain amount of disagreement. The applicability of this method in an industrial scenario is limited. First, maintaining m rankers for one task has non-negligible engineering costs. Second, customer experience will not be harmed only if all the rankers deliver the same ranking quality – but having m equally performing but disagreeing rankers is not trivial in practice. [13] proposed an approach not requiring any intervention; the method formulates a maximum likelihood problem, and uses Expectation Maximization (EM) to jointly estimate the position bias and a relevance function. EM works without the need of interventions on the ranking behavior, but, unlike the methods from [8, 14], it has no guarantees of learning accuracy. Experiments in Sec. 3 show that EM delivers suboptimal results in comparison with other methods.

2.2 Policy Aware Intervention Harvesting

In this section we present our approach. Our intuition is that in the streaming media service scenario described in the introduction, there is no need to artificially add randomness to develop reliable position bias estimators like those in [14]. In this setting in fact, stochastic policies are frequently used to solve the exploration / exploitation dilemma typical of online top-k ranking. Multi-armed bandits for example are used frequently [1, 2, 4] where exploration can be conduced using the well known Linear Thompson Sampling mechanism [3] or a generalization of $\epsilon-$greedy to the ranking use case [5]. The randomness introduced by a stochastic policy is enough to develop reliable position bias estimators that generalize those from [14], without the need of ad-hoc interventions.

At step t, the ranker observes k items together with their features $x_1, ..., x_k$ and proposes a ranking that depends on the observed features. After a user interaction, an implicit feedback (e.g. a click) signal is collected by the ranker, and used at a later stage for retraining. We will denote with $c_h^{(t)}$ a binary variable indicating if a click[3] happened at time t on the item ranked in position h. The behavior of the ranking policy is assumed to be known and $\mathbb{P}(rank_t(i) = h)$ will denote the probability that at iteration t, the item that was ranked at position i would have been ranked at position h. Similarly to [14], we define as $S_{h,l}^{(t)}$ as the set of the items that at step t can be ranked at both positions h and l from our randomized ranker:

$$S_{h,l}^{(t)} = \{i \in \{1, ..., k\} : \mathbb{P}(rank_t(i) = h)\mathbb{P}(rank_t(i) = l) > 0\} \quad (1)$$

Note that at time t the context vector associated to each of them depends on the user, and that consequently, the behavior of the ranker will differ from user to user. This means that for $t \neq t'$, the set $S_{h,l}^{(t)}$ might be different from the set $S_{h,l}^{(t')}$, because the user examining the content at step t might be different from that at step t'. Using these sets we can define $c_{h,l}^{(T)}$ and $\bar{c}_{h,l}^{(T)}$ as follows:

$$c_{h,l}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} \chi(h \in S_{h,l}^{(t)}) \frac{c_h^{(t)}}{\mathbb{P}(rank_t(h) = h)}, \quad (2)$$

$$\bar{c}_{h,l}^{(T)} = \frac{1}{T}\sum_{t=1}^{T} \chi(h \in S_{h,l}^{(t)}) \frac{1 - c_h^{(t)}}{\mathbb{P}(rank_t(h) = h)}, \quad (3)$$

where $\chi(.)$ is the indicator function. Note that the following holds:

[3] We remark that having a binary feedback is not a strict requirement, and the method we present works identically for any other kind of feedback signal.

Theorem 2.1. *Let $c_{h,l}^{(T)}$ and $\bar{c}_{h,l}^{(T)}$ as in Eq. (2) and (3); we have:*

$$\mathbb{E}[c_{h,l}^{(T)}] = e_h \mathbb{E}[\sum_{h \in S_{h,l}} rel(x_h)], \tag{4}$$

$$\mathbb{E}[\bar{c}_{h,l}^{(T)}] = \mathbb{E}[|S_{h,l}|] - e_h \mathbb{E}[\sum_{h \in S_{h,l}} rel(x_h)] \tag{5}$$

where $|S_{h,l}|$ is the cardinality of $S_{h,l}$ and all the expectations are taken over the randomness of the click event, and across the distribution of the users, on which the context vectors of each item depend.

The proof is identical to that of Proposition 1 in [14]. Theorem 2.1 says that it is not necessary to estimate the relevance model to estimate the position bias, as done in [13]; conversely, it states that is enough to estimate the average relevance $\mathbb{E}[\sum_{h \in S_{h,l}} rel(x_h)]$ of the ranked items, and shows – with Equations (4) and (5) – how this value relates with the position bias and with the expected value of the estimators introduced at Equations (2) and (3). The average relevance can be normalized to lay in the $[0, 1]$ interval, obtaining a variable $s_{h,l}$ defined as $s_{h,l} = \mathbb{E}[\sum_{h \in S_{h,l}} rel(x_h)]/\mathbb{E}[|S_{h,l}|]$. This allows to estimate the position bias by minimizing the weighted cross-entropy loss for the distribution $e_h s_{h,l}$, using the samples $c_{h,l}^{(T)}$ an $\bar{c}_{h,l}^{(T)}$, exactly as done in [14, Sec 4.2]. Concretely, this is accomplished by minimizing the following cost-function:

$$C(e, s) = -\left(\sum_{h \neq l} c_{h,l}^{(T)} log(e_h s_{h,l}) + \bar{c}_{h,l}^{(T)} log(1 - e_h s_{h,l})\right) \tag{6}$$

where $e = (e_1, ..., e_k)$ is the position bias and $s = (s_{h,l})_{h,l \in \{1,..,k\}}$. The function $C(e, s)$ is differentiable with respect to e and s and can thus be minimized using any variant of gradient descent [17], obtaining $e, s = \arg\min_{\bar{e}, \bar{s}}(C(\bar{e}, \bar{s}))$. The optimization procedure returns an estimate for both, the position bias curve e and the average relevance s, which can be considered as a dummy variable and discarded. We call this approach *Policy Aware Intervention Harvesting*. Unlike the Swap approach from [8], the method can be used without any intervention on the ranking. Unlike EM [13], our method does not require to estimate the relevance model $rel(.)$ from scratch, solving a much easier learning problem, which results in more accurate results, as shown in Section 3. The method proposed in this section is a generalization of the Intervention Harvesting method from [14]; instead of working with m rankers one of which is chosen randomly at each iteration – which is expensive under the engineering perspective and difficult to achieve without losing ranking quality – we work with a single stochastic ranker, as it is usually already the case for streaming media services.

3 EXPERIMENTS

3.1 Synthetic Data: Yahoo LTR Challenge

In this section we assess the robustness of the Policy Aware Intervention Harvesting method, comparing it with other methods from the literature. We generate synthetic click data – where the position bias curve is known in advance – and compare the various methods on the ability to reconstruct that curve. Following the same approach as in [14], we generate the synthetic data starting from the Yahoo LTR Challenge dataset [18]. The dataset contains $n = 14,000$ user queries; for each query a variable number of items to be ranked is present, together with their features and a manually annotated relevance score. We generated click data as follows: first we pre-trained a ranking algorithm[4] on a random hold-out subset of data; then, for each query, we first obtained a preliminary ranking using the pre-trained ranker; then, we perturbed this ranking using the randomized interventions[5] prescribed by [8]. Once a ranking is obtained, we generated clicks simulating the position bias model described in Section 2, with the addition of some noise to make the experiment more realistic. Concretely, we assumed a ground-truth position bias as $e_h = \frac{1}{h}$. Given an item ranked in position h, we simulated a click with probability e_h if the relevance score of the item is greater or equal to 3; if not, we generate a click with probability $e_h \epsilon$, where ϵ is a parameter controlling the noise in generating the data, which we set to $\epsilon = 0.1$. The data generated in this way can be used to calculate the position bias using the various competing methods: the algorithm presented in Section 2.2, EM [13], Intervention Harvesting [14] and the Swaps method [8]. For each method, we compute the position bias curve on the generated dataset and compare it against the ground-truth curve using the Mean Absolute Deviation (MAD)

$$MAD(e, \hat{e}) = \sum_{i=1}^{k} |e_i - \hat{e}_i|/k \tag{7}$$

where e_i is the ground-truth curve at position i and $\hat{e}_i$ is the estimated value. In Table 1, we compare the reconstruction error of the various methods over 5 runs of the experiment. [6] The Swaps and the Policy Aware Intervention Harvesting methods have the highest accuracy, with the latter having the advantage of not requiring intrusive interventions for its execution. Their variance is also small. EM has the lowest variance, but also has a high bias. The lower performance of EM is understandable: unlike the other methods it tries to reconstruct the full relevance model [13], a task that is not required by the other approaches. Also the Intervention Harvesting method is less accurate.

3.2 A Real World Use-Case: Amazon Music

3.2.1 Amazon Music. Amazon Music is a streaming service, where users can listen to more than 60 million tracks available in the catalog. Every time a user opens Amazon Music, the Home Page is displayed. As described in Section 1, the Home Page is organized as a vertical list of *widgets*. Each widget is a collection of music entities (songs, albums, playlists, etc.). Each widget is rendered as a horizontally ranked list of icons, each icon being associated to

[4] We used here a linear regressor trained to predict the ground-truth relevance given the features on the holdout dataset.

[5] More specifically, for each query we first randomly decided a treatment between *even treatment* and *odd treatment*. If the query is in the even treatment, we were randomly swapping, with 50% probability, items with even positions with those immediately after. If the query is in the odd treatment, we were randomly swapping items at odd positions with those immediately after. This additional randomization enables to run the Swaps method from [8] on the generated data. Also, it adds the randomization needed to run the Policy Aware Intervention Harvesting method from this paper. Last, by splitting the queries between Odd and Even treatments, it simulates the scenario where two rankers are present, thus enabling the usage of vanilla Intervention Harvesting [14] on the generated data.

[6] Experimental Setting: All the experiments were ran in Python 3.6. EM was implemented using a scikit-learn [19] Gradient Boosted Decision Tree to learn the relevance function, as in [13], iterating the method over 50 epochs. Policy Aware Intervention Harvesting was implemented using MxNet for optimization [20]; for minimizing the equation (6) we used Adam with a learning rate of 0.1.

Table 1: Mean and Standard deviation of the estimation errors of the various estimators across 5 experiment runs.

Method	Error	
	Mean	STD
Expectation Maximization	0.3	**0.0025**
Intervention Harvesting	0.18	0.007
Swaps	0.0085	0.0029
Policy Aware Intervention Harvesting	**0.0083**	0.003

one specific item of the widget. A user wanting to stream content within a widget, clicks on the corresponding icon to begin playback. Online ranking algorithms can be used for horizontally ranking the order of the items inside each widget. Each ranker takes a series of n items and displays the top-k items [7] in an ordered way inside the widget. Position bias is an important issue: for each widget only 2-3 items are visible, even if much more are present; to observe the items that are not immediately observable, the user has to swipe the content, unveiling new items.

3.2.2 Position Bias Estimation. We first validate on Amazon Music data that the Swaps and the Policy Aware Intervention Harvesting methods deliver similar results. In order to run the Swaps method, we ran a data collection executing the interventions described in [8], analogous to those described in the previous Section 3.1, footnote 5. As a base ranker, we adopted a bandit-based ranker, similar to the one described in [3], with Linear Thompson Sampling for exploration. We ran this data collection gathering data for 8 widgets. We thus obtained a series of datasets – one per widget. For each dataset, each data point contains one ranking request, triggered by the home page at the time when widgets are rendered. For every request, one log entry contains: information about the features of the items to be ranked $x_1, ..., x_k$; the order in which the items were presented to the user; the click/non-click signals $c_1, ..., c_k$ for each item; and the propensities of the production policy $\mathbb{P}(rank(i) = h)$, containing the probability that the item ranked in position i would have been ranked at position h for every $h \in \{1, ..., k\}$. Similarly to Section 3.1, we use the collected data to calculate the position bias using the Swap method from [8], the Intervention Harvesting method [14], EM [13] and the Policy Aware Intervention Harvesting method from this paper. In Table 2, for each pair of methods we present the average MAD (as in Eq. (7)) between the estimated position biases. We can observe that the Swaps and the Policy Aware Intervention Harvesting (PA-IH) methods deliver similar estimates; this means that the Policy Aware Intervention Harvesting is able to achieve the same performance of the Swap method, without the need of intrusive interventions, which negatively impact ranking quality, as we will see in the next section. The curves estimated with EM and with Intervention Harvesting are instead different.

3.2.3 Impact of Manual Interventions. We now demonstrate empirically that the interventions required by the Swap method do impact ranking performance. While collecting the data analyzed in this chapter, we monitored our production rankers, comparing the performance of the portion of traffic where the interventions described in [8] were performed against those of the portion of traffic where no such randomization was being executed (the base ranker however was the same, making the presence of the swaps the only difference between the two scenarios). Table 3 compares the performance of the two portions of traffic in terms of play seconds – i.e. the average number of seconds a user plays a content from a widget – and click through rate. Results are averaged across all the widgets. The swaps required to implement the method from [8] caused a loss in terms of both metrics. The consequence of this, is that the swap approach can not be adopted as a production solution, given the negative impact of position bias estimation. Overall, we can conclude that while the Swaps and the Policy Aware Intervention Harvesting methods have similar accuracy, the latter is preferable, due to the practical advantage of not requiring any user intervention – which we showed harms user experience.

[7] n depends on the widget and spans from 20 to 1000. k is never larger than 50.

Table 2: MAD (Eq. (7)) between the position biases estimated by different methods, averaged across the 8 widgets. Low values indicate that two methods deliver similar estimates.

	PA-IH	IH	EM	Swaps
PA-IH	0	*	*	*
IH	0.35	0	*	*
EM	0.37	0.31	0	*
Swaps	0.07	0.32	0.34	0

Table 3: Relative loss in performance for the portion of traffic where the interventions from [8] were executed.

	Play seconds	Click Through Rate
% Variation	-2.7%	-2.11%
p-value	0.03	0.005

Table 4: Relative improvements of Position Bias-aware Bandit-Based Ranker over a regular Bandit-Based Ranker

	Play seconds	Click Through Rate
% Improvement	+12.21%	+6.99%
p-value	0	0.001

3.2.4 Impact on Ranking Quality. The goal of an accurate position bias estimation is to improve ranking quality. Table 4 reports the relative increase in platform metrics observed as a consequence of the online usage of a position-bias aware ranker. We compared online the method from [3], using the position bias curves calculated with the approach from Section 2.2 – against the scenario where the same ranker is used, but without any notion of position bias – which was thus set to be 1 for every position. The adoption of position bias dramatically improves ranking quality, providing boosts in the analyzed metrics and thus a better user experience.

4 CONCLUSION

We presented a method to calculate position bias from ranking data. Our approach leverages on the randomness of a stochastic policy to calculate reliable position bias estimates, which makes it suitable for streaming media services, where stochastic policies are used to solve the exploration/exploitation dilemma. Next steps consist of exploring the impact of user context on position bias curves, expanding the work from [21].

REFERENCES

[1] Jaya Kawale and Elliot Chow. A multiarmed bandit framework for recommendations at netflix. *https://www.slideshare.net/JayaKawale/a-multiarmed-bandit-framework-for-recommendations-at-netflix*, 2018.

[2] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson, and Rishabh Mehrotra. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In *Proceedings of the 12th ACM conference on recommender systems*, pages 31–39, 2018.

[3] Beyza Ermis, Patrick Ernst, Yannik Stein, and Giovanni Zappella. Learning to rank in the position based model with bandit feedback. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 2405–2412, 2020.

[4] Walid Bendada, Guillaume Salha, and Théo Bontempelli. Carousel personalization in music streaming apps with contextual bandits. In *Fourteenth ACM Conference on Recommender Systems*, pages 420–425, 2020.

[5] Katja Hofmann, Shimon Whiteson, and Maarten de Rijke. Balancing exploration and exploitation in listwise and pairwise online learning to rank for information retrieval. *Information Retrieval*, 16(1):63–90, 2013.

[6] Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, and Geri Gay. Accurately interpreting clickthrough data as implicit feedback. In *ACM SIGIR Forum*, volume 51, pages 4–11. Acm New York, NY, USA, 2017.

[7] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142, 2002.

[8] Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. Unbiased learning-to-rank with biased feedback. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 781–789, 2017.

[9] Marco Morik, Ashudeep Singh, Jessica Hong, and Thorsten Joachims. Controlling fairness and bias in dynamic learning-to-rank. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 429–438, 2020.

[10] Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. A general framework for counterfactual learning-to-rank. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 5–14, 2019.

[11] Ziniu Hu, Yang Wang, Qu Peng, and Hang Li. Unbiased lambdamart: an unbiased pairwise learning-to-rank algorithm. In *The World Wide Web Conference*, pages 2830–2836, 2019.

[12] Harrie Oosterhuis and Maarten de Rijke. Policy-aware unbiased learning to rank for top-k rankings. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 489–498, 2020.

[13] Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, pages 610–618, 2018.

[14] Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li, Marc Najork, and Thorsten Joachims. Estimating position bias without intrusive interventions. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 474–482, 2019.

[15] Oriol Barbany Mayor, Vito Bellini, Alexander Buchholz, Giuseppe Di Benedetto, Diego Marco Granziol, Matteo Ruffini, and Yannik Stein. Ranker-agnostic contextual position bias estimation. *arXiv preprint arXiv:2107.13327*, 2021.

[16] Matthew Richardson, Ewa Dominowska, and Robert Ragno. Predicting clicks: estimating the click-through rate for new ads. In *Proceedings of the 16th international conference on World Wide Web*, pages 521–530, 2007.

[17] Sebastian Ruder. An overview of gradient descent optimization algorithms. *arXiv preprint arXiv:1609.04747*, 2016.

[18] Olivier Chapelle and Yi Chang. Yahoo! learning to rank challenge overview. In *Proceedings of the learning to rank challenge*, pages 1–24. PMLR, 2011.

[19] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *the Journal of machine Learning research*, 12:2825–2830, 2011.

[20] Tianqi Chen, Mu Li, Yutian Li, Min Lin, Naiyan Wang, Minjie Wang, Tianjun Xiao, Bing Xu, Chiyuan Zhang, and Zheng Zhang. Mxnet: A flexible and efficient machine learning library for heterogeneous distributed systems. *arXiv preprint arXiv:1512.01274*, 2015.

[21] Zhichong Fang, Aman Agarwal, and Thorsten Joachims. Intervention harvesting for context-dependent examination-bias estimation. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 825–834, 2019.