Robust Learning to Rank Against Noisy Clicks with Residual-Based Correction

Md Aminul Islam
University of Illinois Chicago
mislam34@uic.edu

Abstract

Implicit feedback signals such as user clicks are widely used in learning-to-rank (LTR) systems because they are easy to collect and generally capture user preferences. However, click data collected in real-world LTR systems often contain substantial noise, and directly training on these imperfect signals can degrade ranking performance. Existing LTR methods primarily focus on correcting click bias while assuming noise-free labels or requiring some clean annotations, limiting their practical applicability. We propose a *Residual-based Noise-Aware Correction* (RNC) framework that enables robust LTR from noisy click data without relying on predefined noise assumptions or clean relevance judgments. RNC estimates instance-level noise probabilities by exploiting residual discrepancies between predicted relevance and observed clicks, and incorporates these estimates as inverse propensity weights to attenuate the influence of unreliable interactions during training. Experiments on benchmark datasets show that RNC improves robustness to noisy clicks and outperforms or matches state-of-the-art methods, demonstrating its effectiveness as a practical solution for real-world LTR systems.

1 Introduction

Learning-to-rank (LTR) models play a central role in modern search engines, recommender systems, and content ranking applications, where the goal is to order items by their estimated relevance to a user query. While LTR models can be trained with high-quality human-annotated relevance judgments, obtaining such annotations at scale is prohibitively expensive and time-consuming, as modern LTR systems often contain millions or even billions of documents. Consequently, real-world systems typically rely on implicit feedback signals, particularly user clicks, as a practical and scalable proxy for user preference.

However, real world click data are both noisy (Ovaisi et al., 2020) and biased (Joachims et al., 2017; Wang et al., 2018). Users may click on items accidentally or out of curiosity rather than genuine interest, producing unreliable click signals. Furthermore, click behavior is shaped by external influences such as position bias and selection bias (Joachims et al., 2017; Ovaisi et al., 2020), where higher-ranked or more frequently exposed items attract disproportionate attention regardless of true relevance. Learning directly from such raw click data can mislead model training, degrade ranking performance, and reinforce harmful feedback loops over time in LTR systems.

These challenges are amplified by the fact that real-world click logs contain substantial noise. Yet, existing robust LTR approaches often impose restrictive assumptions about the noise distribution—such as predefined noise structures or stationarity—or require access to clean labels. For example, prior work commonly assumes specific noise patterns (Agarwal et al., 2019b; Haddad, 2022) or depends on partially clean relevance judgments (Ding et al., 2015). Such assumptions rarely hold in practice, making these approaches difficult to apply reliably in large-scale operational systems.

Most unbiased LTR research has focused on correcting click bias (Joachims et al., 2017; Wang et al., 2018; Luo et al., 2024). State-of-the-art approaches for addressing position bias can be grouped into click models, propensity-based methods, and econometrics-inspired models. Click models (Chapelle and Zhang, 2009; Craswell et al., 2008; Dupret and Piwowarski, 2008; Wang et al., 2013) infer true document relevance by modeling user browsing behavior and maximizing the likelihood of observed clicks. Propensity-based methods treat clicks as counterfactual events and apply inverse propensity weighting (IPW) to recover relevance-equivalent training (Joachims et al., 2017; Wang et al., 2016), often requiring result

randomization (Joachims et al., 2017; Oosterhuis and de Rijke, 2020; Wang et al., 2016). Some approaches (Ai et al., 2018; Hu et al., 2019; Luo et al., 2024; Vardasbi et al., 2020; Zhu et al., 2020) jointly learn propensities and rankers, though misspecification of either model can compromise unbiasedness (Agarwal et al., 2019c; Fang et al., 2019; Tian et al., 2020). Econometrics-based approaches (Ovaisi et al., 2020, 2021), using Heckman's two-stage model (Heckman, 1979), avoid propensity estimation but assume linear ranking models. However, these bias correction methods often implicitly assume that observed clicks are noise-free (Agarwal et al., 2019b) and do not account for the presence of noisy clicks in the training data.

Several studies (Agarwal et al., 2019a; Wu et al., 2022; Haddad, 2022; Ding et al., 2015) have addressed the issue of noisy clicks, but most rely on strong simplifying assumptions. Some assume noise is only position-dependent (Agarwal et al., 2019b), others treat it as random noise (Wu et al., 2022) or fixed noise (Haddad, 2022) levels, and some require partially clean labels (Ding et al., 2015) for correction. Despite their potential, these methods are often impractical in practice because they depend on prior knowledge of noise patterns or additional information that which limit their deployment in real-world settings.

To overcome these limitations, we propose a *Residual-based Noise-Aware Correction* (RNC) method that learns effectively from noisy click data without requiring assumptions about the noise distribution or access to clean labels. The main idea is to leverage the discrepancy between predicted relevance and observed clicks, captured through model residuals, to estimate noise probabilities for individual query–document pairs. These residual-based noise estimates are then incorporated into training as inverse propensity weights, so that cleaner samples have greater influence while noisy interactions are appropriately down-weighted in the objective function.

Our approach draws inspiration from residual-based methods in econometrics literature. A well-established class of techniques uses *control functions* (Wooldridge, 2015) to correct for biases arising from unobserved variables, including classical models such as *Heckman selection* (Heckman, 1979), *endogenous switching regressions* (Maddala and Nelson, 1975), and *instrumental variables* (Reiersøl, 1941). More recently, methods based on *orthogonal moments*, as employed in *double machine learning* (Chernozhukov et al., 2018), have generalized these ideas to high-dimensional and machine-learning-based settings. Inspired by these generic principles, our framework provides a general, assumption-free, and scalable solution for robust LTR under severe click noise.

Our main contributions are:

- A novel residual-based framework for correcting noise in implicit feedback data.
- A robust LTR training pipeline that does not rely on assumptions about the noise distribution or require access to clean data.
- Empirical results demonstrating substantial performance gains across multiple datasets and noise levels.

2 Related Work

Previous research in LTR has primarily focused on correcting bias in click data rather than directly addressing noise. Position bias, extensively explored in the LTR literature, refers to the fact that users are more likely to interact with items shown at higher ranks even when those items are not truly relevant (Craswell et al., 2008; Joachims et al., 2007). Selection bias is another pervasive issue in LTR systems. It emerges when the system surfaces only a small portion of the relevant documents for a query, so feedback is observed only for this restricted set (Ovaisi et al., 2020).

Bias in LTR systems is commonly addressed through click models, propensity-based methods, and econometric approaches. Prior work (Chapelle and Zhang, 2009; Chuklin et al., 2016; Craswell et al., 2008; Dupret and Piwowarski, 2008) models user browsing behavior to infer true relevance from clicks. Notable examples include the Cascade Model (Craswell et al., 2008), User Browsing Model (Dupret and Piwowarski, 2008), and Dynamic Bayesian Network (Chapelle and Zhang, 2009), which respectively assume sequential viewing, allow skipping behavior, and model user satisfaction and abandonment. Wang et al. (Wang et al., 2018) jointly learn ranking and examination propensity using a graphical model.

Propensity-based methods apply counterfactual learning, treating bias as a counterfactual factor and debiasing clicks via IPW. Several works (Joachims et al., 2017; Oosterhuis and de Rijke, 2020; Wang

et al., 2016) estimate propensities through randomized results. Joachims et al. (Joachims et al., 2017) demonstrate that, with accurate propensity estimates, training on clicks yields the same model as training on true relevance. Other methods jointly learn the ranker and propensity model (Ai et al., 2018; Hu et al., 2019). Vardasbi et al. (Vardasbi et al., 2020) design a cascade-based IPW estimator, and Oosterhuis et al. (Oosterhuis and de Rijke, 2020) introduce policy-aware propensity scoring using stochastic ranking policies. Luo et al. (Luo et al., 2024) identify propensity overestimation issues when relevance correlates strongly with position and propose a causal, two-step unconfounded propensity estimator. Econometrics-based approaches by Ovaisi et al. (Ovaisi et al., 2020, 2021) use the Heckman model (Heckman, 1979) to correct both selection and position bias without explicit propensity estimation.

However, these traditional approaches, such as IPW methods (Joachims et al., 2017; Ai et al., 2018; Luo et al., 2024, 2023), assume that click data are noise-free (Agarwal et al., 2019b) and mainly target bias correction. Several studies (Agarwal et al., 2019a; Wu et al., 2022; Haddad, 2022; Ding et al., 2015) have attempted to model noisy clicks, but they rely on simplifying assumptions. For instance, TrustPBM (Agarwal et al., 2019b) assumes that noise depends only on position, PeerRank (Wu et al., 2022) models random noise through peer loss, and other works consider constant noise levels (Haddad, 2022) or require partially clean labels for noise correction (Ding et al., 2015). These noise-aware approaches, while promising, are limited in practice due to their reliance on prior knowledge of noise distributions or additional supervision unavailable in LTR systems.

3 Problem Description

We begin with the formal definition of LTR systems following Ai et al. (Ai et al., 2018). Let $\mathcal{Q}$ denote a universal set of independent and identically distributed (i.i.d.) queries drawn from an unknown distribution $P(q)$. For a ranking function $g \in \mathcal{G}$, let $l(g, q)$ represent the loss incurred by g on query q. The expected loss of g over the query distribution is then defined as:

$$\mathcal{L}(g) = \int_{q \in \mathcal{Q}} l(g, q)\, dP(q). \tag{1}$$

The goal of an LTR system is to identify the optimal ranking function $g \in \mathcal{G}$ that minimizes the expected loss $\mathcal{L}(g)$. Because the true query distribution $P(q)$ is unknown, $\mathcal{L}(g)$ cannot be computed directly. Instead, it is approximated by the empirical loss:

$$\hat{\mathcal{L}}(g) = \frac{1}{|\mathcal{Q}|} \sum_{q \in \mathcal{Q}} l(g, q). \tag{2}$$

Human-annotated relevance labels for query–document pairs are commonly treated as reliable indicators of true relevance. However, obtaining annotations for all possible query–document combinations is costly and infeasible for modern LTR systems, which typically involve millions of documents. An alternative is to leverage implicit user feedback (e.g., clicks), which is easy to collect and often reflects users' actual preferences (Joachims, 2002). Nonetheless, the real-world click data is inherently noisy, containing random or mistaken user clicks and skewed by presentation bias. Training a model directly on this unreliable signal can divert it from learning true user preferences, degrading ranking performance. This degradation is further amplified through the feedback loop inherent in LTR systems (Chaney et al., 2018), where poor rankings reinforce future noise, progressively undermining personalization and long-term performance. Consequently, it is essential for LTR models to explicitly account for noisy click data during training.

4 Methodology

We now introduce our RNC framework that enhances the robustness of LTR systems in the presence of noisy click data. The RNC framework consists of two stages: (1) computing residual signals using an auxiliary model and transforming these signals into interaction-level noise estimates, and (2) training the final ranker using a noise-aware weighted objective. This section presents the formulation in detail.

Let document i be associated with query q and have feature vector $\boldsymbol{x}_i^q$. We first train an auxiliary ranking model, g_0, using all available click data as the output and the corresponding features as input, in order to predict relevance signals.

$$c_i^q = g_0(\boldsymbol{x}) + \epsilon_i^q, \tag{3}$$

where ϵ_i^q, the error term for document i associated with query q. For document i associated with query q, the model produces a predicted click score for

the corresponding query–document pair:

$$p_i^q = P(c_i^q = 1 \mid \boldsymbol{x}_i^q; g_0), \tag{4}$$

where p_i^q denotes the predicted click score.

Using these predictions, we define the estimated residual for each query-document pair as:

$$r_i^q = c_i^q - p_i^q. \tag{5}$$

The intuition is that query–document features reflect a document's true relevance to a query. Any other factors affecting clicks that are not captured by these features correspond to click behavior unrelated to true relevance, i.e., noise, which is therefore captured by the estimated residual r_i^q. Residuals near zero correspond to interactions well explained by features, while large positive or negative values reflect behavior that is unlikely under the model, signaling potential noise in the click data.

To convert the estimated residuals into continuous noise estimates, we apply the sigmoid function: $\eta_i^q = \sigma(r_i^q)$, where $\sigma(\cdot)$ is the sigmoid function. Large positive residuals yield η_i^q close to 1, while negative residuals yield smaller values. We refer to this noise probability η_i^q as the propensity weight for each query–document pair.

The noise probabilities are then converted into inverse propensity weights using:

$$w_i^q = \frac{1}{\eta_i^q + \varepsilon}, \tag{6}$$

where $\varepsilon > 0$ ensures numerical stability. Higher values of η_i^q produce smaller weights, reflecting lower reliability of the click signal, whereas smaller noise probabilities produce larger weights. Thus, clicks that are strongly affected by noise receive low inverse propensity weights, while those less affected receive higher weights.

The final ranking model g_θ is trained using an IPW-weighted implicit-feedback objective. The RNC-adjusted objective is given by:

$$L_{\mathrm{RNC}}(\theta) = -\sum_{q \in Q} \sum_{i \in q} w_i^q \log P(c_i^q \mid x_i^q; \theta), \tag{7}$$

where clicks with higher reliability are assigned larger weights, and those more affected by noise receive smaller weights. This ensures that noisy interactions have less influence during training, allowing the model to focus more on informative, reliable signals.

RNC constructs residuals that quantify disagreement between observed and predicted click behavior, transforms these signals into noise probabilities, and integrates the resulting weights directly into the ranking objective. Thus, our framework does not require any assumptions about document position (Agarwal et al., 2019b), random noise distributions (Wu et al., 2022), fixed noise levels (Haddad, 2022), or partially clean labels (Ding et al., 2015) for correction, which makes it practical for real-world LTR systems.

5 Experimental Setup

5.1 Datasets

We evaluate all methods on two widely used LTR benchmark datasets: Yahoo! Learning to Rank Challenge (C14B) (Chapelle and Chang, 2011) [1] and MSLR-WEB10k (Qin and Liu, 2013) [2]. The datasets provide human-annotated relevance labels in the range $[0, 4]$. The Yahoo dataset includes features already normalized to $[0, 1]$, while for MSLR-WEB10K, we apply min–max normalization to map all features into $[0, 1]$. We use these two benchmark datasets because they provide unbiased test sets, allowing us to conduct unbiased evaluations of the methods.

- **Yahoo! Learning to Rank Challenge (C14B):** Contains 19,944 training queries with 473,134 documents, 2,994 validation queries with 71,083 documents, and 6,983 test queries with 165,660 documents. Each query has on average 24 documents, and each document is represented by a 700-dimensional feature vector.
- **MSLR-WEB10K:** We use Fold 1 of the dataset, which includes 6,000 training queries (723,412 documents), 2,000 validation queries (235,260 documents), and 2,000 test queries (241,521 documents). Each query contains on average 121 documents, each with a 136-dimensional feature vector.

A summary of the dataset statistics is provided in Table 1.

5.2 Evaluation metrics

We evaluate the performance of our method using common metrics in the LTR literature, including Expected Reciprocal Rank (ERR), Normalized

[1] https://webscope.sandbox.yahoo.com/catalog.php?datatype=c

[2] https://www.microsoft.com/en-us/research/project/mslr

Table 1: Dataset descriptions.

Dataset	Train queries	Train docs.	Valid queries	Valid docs.	Test queries	Test docs.	Avg. docs. per query	Feature
Yahoo (Set1)	19,944	473,134	2,994	71,083	6,983	165,660	24	700
MSLR-WEB10k	6,000	723,412	2,000	235,260	2,000	241,521	121	136

Discounted Cumulative Gain (NDCG), and Recall. $ERR@p$ measures the expected probability that a user finds a sufficiently relevant document within the top-p positions. Higher $ERR@p$ values indicate that higher-utility documents appear earlier in the ranking, reflecting better user satisfaction. $NDCG@p$ evaluates the quality of a ranked list by considering both the relevance of documents and their positions. Higher $NDCG@p$ values indicate more effective ranking of relevant documents within the top-p positions. $Recall@p$ measures the fraction of relevant documents retrieved within the top-p positions of a ranked list relative to the total number of relevant documents for the query. Higher $Recall@p$ indicates that more relevant documents appear within the top-p positions.

5.3 Click simulation

We generate synthetic click data using a two-step procedure, following the approach proposed by Ai et al. (Ai et al., 2018) and Joachims et al. (Joachims et al., 2017), which is widely adopted in LTR research. This procedure allows us to simulate realistic user interactions while controlling for click noises in the data.

In the first step, we train a RankSVM (Joachims, 2006) model on 1% of the training data using true relevance labels. The trained model produces initial ranked lists π_q for each query q over the remaining 99% of the training data. Using a trained ranker, rather than random ranking, ensures that the initial rankings are meaningful and better than random ranker, while still leaving room for improvement (Ai et al., 2018).

In the second step, we generate synthetic clicks for each ranked list π_q using the Position-Based Model (PBM) (Richardson et al., 2007). According to PBM, a document is clicked if it is both observed and perceived as relevant. Let O_q^y denote whether document y is observed and R_q^y denote whether it is perceived as relevant, then the click probability can be written as:

$$Pr(C_q^y = 1) = Pr(O_q^y = 1 \mid \pi_q) \cdot Pr(R_q^y = 1 \mid \pi_q).$$

Following Ovaisi et al. (Ovaisi et al., 2020, 2021), we then sample from a binomial distribution using the click probability to determine whether each document is clicked or not.

The observation probability is defined following Joachims et al. (Joachims et al., 2017) as:

$$Pr(O_q^y = 1 \mid \pi_q) = \left(\frac{1}{r(y \mid \pi_q)}\right)^{\eta},$$

where $r(y \mid \pi_q)$ is the rank of document y in the list π_q, and η controls the severity of position bias. Unless otherwise stated, we set $\eta = 0.0$, since our primary goal is not to account for position bias but to control for noise in the click data.

The relevance probability captures the likelihood that a document is perceived as relevant and is defined as in Chapelle et al. (Chapelle et al., 2009):

$$Pr(R_q^y = 1 \mid \pi_q) = \epsilon + (1 - \epsilon)\frac{2^{rel(q,y)} - 1}{2^{rel_{\max}} - 1},$$

where $rel(q, y) \in [0, 4]$ is the true relevance of document y for query q, and $rel_{\max} = 4$ is the maximum relevance score. The parameter ϵ models click noise, allowing irrelevant documents to be mistakenly clicked. We set $\epsilon = 0.10$ by default, which provides the proportion of noisy clicks in the training data. This parameter can be adjusted to generate varying levels of noisy clicks in the data.

We define one full pass of click generation over the training data as a *sampling pass*. Following Ovaisi et al. (Ovaisi et al., 2020, 2021), we perform 10 sampling passes to generate clicks for all training data. This multi-pass procedure simulates realistic user behavior, where some documents may receive multiple clicks while others may remain unclicked (Ovaisi et al., 2020). The clicks generated through this process are used in all of our experiments. This click generation process allows us to simulate noisy real-world clicks while still enabling unbiased evaluation of methods using the human-annotated, unbiased test data provided in these datasets.

5.4 Implementation details

To train the ranker g, we use a deep neural network (DNN)–based LTR model following the architecture in Ai et al. (Ai et al., 2018). We learn the initial ranker g_0 using supervised machine learning. We use Ridge, Logistic Regression, SVM, and XGBoost regression and find that XGBoost generally yields the best ranking performance once its residuals are incorporated into the final ranker as IPW. Therefore, all reported results use XGBoost to estimate g_0. We set the parameter $c = 20$ for the RankSVM (Joachims, 2006). For the DNN ranker, we follow the setup used by Ai et al. (Ai et al., 2018). Specifically, we use a three-layer network with hidden sizes $[128, 256, 512]$. The DNN is trained using stochastic gradient descent with the Adam optimizer, and the learning rate α is tuned in the range $[0.005, 0.05]$. We use a batch size of 256 and terminate training after 10,000 steps. For a fair comparison, all baselines and our RNC method are trained using the same configuration and the same DNN ranking architecture. All experiments are conducted on machines with 64GB RAM, 1TB storage, and a 16GB CUDA-enabled GPU. The click generation, first-stage estimation, and residual transformations run on CPU, while the DNN ranker is trained on GPU.

5.5 Baselines

We mainly compare our method against several baselines that address bias in LTR. This is because existing methods typically focus on correcting for biases in the data rather than handling noisy clicks. To ensure a fair comparison, we evaluate these methods by injecting click noise into the data and comparing their performance with our RNC method.

- **UPE:** Unconfounded Propensity Estimation (Luo et al., 2024) mitigates position bias using a causal approach that treats document relevance as a confounder and addresses overestimation of propensities.
- **MULTR:** Model-based Unbiased Learning to Rank (Luo et al., 2023) leverages a context-aware simulator to generate pseudo clicks and corrects the mismatch between pseudo and actual clicks via doubly robust IPW using pseudo labels.
- **PAL:** Position-bias Aware Learning (Guo et al., 2019) employs a two-tower model to disentangle relevance from bias factors.
- **REM:** Regression-based EM (Wang et al., 2018) uses a position-biased click model to estimate propensity scores and trains a ranker accordingly.
- **DLA:** Dual Learning Algorithm (Ai et al., 2018) simultaneously learns an unbiased propensity model and an unbiased ranker from biased click data.
- **DNN:** A Deep Neural Network is trained on biased click data without bias correction, following (Ai et al., 2018). Note that this method does not perform any bias correction. Because we use a DNN ranker as the backbone, this baseline allows a direct comparison to our RNC approach under the same DNN ranking architecture.

5.6 Difficulty of online evaluation

In the LTR literature, it is common to evaluate methods using benchmark datasets and synthetic click data that simulate real-world user behavior. Although A/B testing with live users would be ideal, such evaluations are typically infeasible in academic settings due to high costs and the requirement of an operational LTR system with real-time users. Moreover, unbiased evaluation requires access to true relevance or unbiased ground truth, which is generally unavailable in real-world data. As a result, many state-of-the-art LTR methods (Ai et al., 2018; Luo et al., 2023, 2024; Ovaisi et al., 2021, 2020) rely on benchmark datasets where true relevance is known. Following this practice, we evaluate our method using such datasets and synthetic click data. The use of synthetic clicks also allows us to systematically vary conditions of the level of click noise which are commonly observed in real-world systems.

6 Results and Analysis

We aim to answer the following research questions in our evaluation:

- **RQ1:** How does RNC compare with existing methods in the LTR literature?
- **RQ2:** How robust is RNC under varying levels of click noise?

Performance comparison with existing methods in the LTR literature (RQ1). For RQ1, we compare the performance of our RNC method against traditional bias-correction baselines using ERR@p, NDCG@p, and Recall@p for $p = 5$, 10, and 15. The results are shown in Table 2. Overall, the results demonstrate mixed but promising outcomes when training with 10% simulated click noise. On the Yahoo dataset, the DNN baseline

Table 2: A detailed comparison of RNC and baseline methods across datasets with DNN as the base ranker under 10% noisy clicks. Higher metric values indicate better performance. The best results are bolded, and the second-best are underlined. Percentage improvements (%) of RNC over the best baselines are also reported.

Datasets	Methods	ERR@p			NDCG@p			Recall@p		
		$p=5$	$p=10$	$p=15$	$p=5$	$p=10$	$p=15$	$p=5$	$p=10$	$p=15$
Yahoo	RNC	<u>0.447</u>	<u>0.462</u>	<u>0.467</u>	<u>0.759</u>	<u>0.797</u>	<u>0.819</u>	<u>0.268</u>	<u>0.508</u>	<u>0.636</u>
	UPE	0.435	0.451	0.454	0.740	0.781	0.805	**0.269**	**0.517**	**0.643**
	MULTR	0.441	0.457	0.460	0.752	0.792	0.815	0.267	0.507	0.635
	PAL	0.443	0.460	0.464	0.755	0.794	0.816	0.267	0.507	0.634
	REM	0.444	0.460	0.463	0.750	0.790	0.814	0.267	0.506	0.634
	DLA	0.444	0.460	0.463	0.757	0.796	0.818	0.266	0.506	0.635
	DNN	**0.450**	**0.465**	**0.468**	**0.762**	**0.799**	**0.822**	0.267	0.507	0.634
	Improvement (%)	-0.67	-0.65	-0.21	-0.40	-0.25	-0.37	-0.37	-1.77	-1.10
MSLR-WEB10k	RNC	**0.271**	**0.292**	**0.299**	**0.452**	**0.465**	**0.489**	**0.077**	**0.144**	**0.203**
	UPE	0.245	0.266	0.273	0.371	0.390	0.407	0.063	0.123	0.180
	MULTR	<u>0.269</u>	<u>0.289</u>	<u>0.295</u>	<u>0.446</u>	<u>0.460</u>	<u>0.474</u>	0.075	0.139	0.199
	PAL	0.267	0.286	0.292	0.418	0.438	0.455	0.073	0.140	0.199
	REM	0.268	0.288	0.294	0.427	0.446	0.462	0.075	0.141	0.201
	DLA	0.266	0.288	0.294	0.443	0.458	0.473	<u>0.076</u>	<u>0.142</u>	<u>0.202</u>
	DNN	0.261	0.283	0.290	0.429	0.449	0.466	0.075	0.141	0.201
	Improvement (%)	+0.74	+1.04	+1.36	+1.35	+1.09	+3.16	+1.32	+1.41	+0.50

achieves the best performance across ERR@p and NDCG@p, with RNC consistently ranking as the second-best method. For Recall@p, UPE performs best, while RNC again achieves the second-best performance. Although RNC does not outperform all baselines on Yahoo, its strong and consistent second-place performance across nearly all metrics highlights its effectiveness in mitigating noisy click behavior without relying on any assumptions. The relatively weaker improvements on the Yahoo dataset can be attributed to its smaller query set and lower average number of documents per query (Table 1), which provide fewer interactions and residual signals for RNC to reliably estimate noise probabilities, thereby limiting its corrective advantage.

In contrast, on the MSLR-WEB10k dataset, RNC clearly outperforms all baselines, including the DNN model, across every metric (ERR@p, NDCG@p, and Recall@p). These improvements confirm that RNC effectively leverages residual discrepancies between predicted relevance and observed clicks to estimate noise probabilities and down-weight unreliable interactions. The strong and consistent gains on MSLR-WEB10k demonstrate that RNC provides a robust solution for learning from noisy clicks in LTR systems.

Robustness Under Increasing Click Noise (RQ2). To answer RQ2, we investigate how the RNC framework behaves as the level of injected click noise increases. Assessing robustness across noise levels is essential for determining whether a method generalizes beyond moderately noisy environments. We vary the proportion of noisy clicks to 10%, 20%, and 30%, and report results in Table 3 using ERR@10 and NDCG@10 on both Yahoo and MSLR-WEB10k datasets.

On the Yahoo dataset, RNC maintains stable performance as noise increases for both ERR@10 and NDCG@10. Although RNC does not outperform the DNN baseline, it consistently ranks as the second-best method across all noise levels. This stability indicates that the residual-based noise estimation remains reliable under moderate to severe noise perturbations. The relatively smaller query set and lower average number of documents per query in Yahoo limit the amount of interaction data available for estimating residual signals, which likely constrains RNC's maximum achievable gains. Nevertheless, RNC's performance remains consistent and more robust across noise levels compared to most baselines, underscoring its ability to generalize effectively under noisy click conditions.

On the MSLR-WEB10k dataset, the benefits of RNC are more pronounced. RNC sustains robust-

Table 3: Comparison of NDCG@10 and ERR@10 across noise levels (10%, 20%, 30%) for each dataset. Higher metric values indicate better performance. The best results for each metric are bolded, and the second-best are underlined.

Datasets	Noise	ERR@10							NDCG@10						
		RNC	UPE	MULTR	PAL	REM	DLA	DNN	RNC	UPE	MULTR	PAL	REM	DLA	DNN
Yahoo	10%	<u>0.462</u>	0.451	0.457	0.460	0.460	0.460	**0.465**	<u>0.797</u>	0.781	0.792	0.794	0.790	0.796	**0.799**
	20%	<u>0.462</u>	0.451	0.458	0.460	0.459	0.460	**0.463**	<u>0.795</u>	0.773	0.791	0.793	0.791	0.793	**0.796**
	30%	<u>0.461</u>	0.437	0.454	0.459	0.457	**0.462**	0.460	<u>0.794</u>	0.744	0.785	0.792	0.786	0.792	**0.795**
MSLR-WEB10k	10%	**0.292**	0.266	<u>0.289</u>	0.286	0.288	0.288	0.283	**0.465**	0.390	<u>0.460</u>	0.438	0.446	0.458	0.449
	20%	**0.301**	0.272	0.285	0.296	0.289	0.293	<u>0.298</u>	**0.474**	0.383	0.448	0.453	0.452	<u>0.460</u>	0.450
	30%	**0.304**	0.252	0.284	0.290	0.286	0.292	<u>0.299</u>	**0.472**	<u>0.470</u>	0.443	0.441	0.447	0.454	0.449

ness under moderate and high noise levels, consistently surpassing all baseline methods. As the noise level increases, the performance of competing approaches varies substantially, whereas RNC remains the most stable and best-performing method across all noise conditions. Even at the 30% noise level—where all models naturally experience a drop in absolute performance, RNC continues to outperform its 20%-noise variant, demonstrating strong robustness to higher noise. These results demonstrate that RNC adapts effectively to increasing noise severity and leverages its instance-level noise probabilities to suppress noisy interactions, even when nearly one-third of observed clicks are corrupted.

Overall, the results across the three noise levels show that RNC is more robust to noisy clicks than existing LTR baselines, particularly on datasets with richer query–document interactions. On MSLR-WEB10k, RNC delivers consistent improvements across all noise settings, confirming that its residual-based weighting scheme generalizes effectively even under heavily distorted click environments. On Yahoo, RNC remains competitive and exhibits stable behavior, though the gains are smaller due to the dataset's limited query–document volume, which restricts the precision of residual estimation. These findings collectively demonstrate that RNC offers a principled and reliable solution for robust LTR under varying degrees of click noise.

7 Conclusion and Future Works

In this paper, we introduce the RNC framework for robust LTR from noisy click data. By leveraging residual discrepancies between predicted relevance and observed clicks, RNC estimates instance-level noise probabilities and incorporates them as inverse propensity weights during training. Experimental results demonstrate that RNC substantially improves performance on MSLR-WEB10k and achieves competitive second-best results on Yahoo under noisy-click settings. The relatively weaker improvements on Yahoo can be attributed to its smaller query set and shorter per-query document lists, which provide limited interactions and residual signals for reliable noise estimation.

Future work will explore several directions to further enhance the robustness and generality of the proposed method. First, we plan to integrate and compare RNC with state-of-the-art noise-correction baselines to more comprehensively evaluate its behavior under diverse noise conditions. Second, we aim to refine the residual estimation process, for example, by incorporating uncertainty-aware or adaptive residual models, to improve performance on datasets with limited query-document interactions such as Yahoo. Finally, extending RNC to jointly address both noise and traditional click biases, as well as applying it to large-scale real-world online systems, represents promising avenues for advancing robust LTR in practical environments.

References

Aman Agarwal, Kenta Takatsu, Ivan Zaitsev, and Thorsten Joachims. 2019a. 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.

Aman Agarwal, Xuanhui Wang, Cheng Li, Michael Bendersky, and Marc Najork. 2019b. Addressing trust bias for unbiased learning-to-rank. In *The World Wide Web Conference*, pages 4–14.

Aman Agarwal, Ivan Zaitsev, Xuanhui Wang, Cheng Li,

Marc Najork, and Thorsten Joachims. 2019c. Estimating position bias without intrusive interventions. In *Proceedings of the twelfth ACM international conference on web search and data mining*, pages 474–482.

Qingyao Ai, Keping Bi, Cheng Luo, Jiafeng Guo, and W Bruce Croft. 2018. Unbiased learning to rank with unbiased propensity estimation. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pages 385–394.

Allison JB Chaney, Brandon M Stewart, and Barbara E Engelhardt. 2018. How algorithmic confounding in recommendation systems increases homogeneity and decreases utility. In *Proceedings of the 12th ACM conference on recommender systems*, pages 224–232.

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

Olivier Chapelle, Donald Metlzer, Ya Zhang, and Pierre Grinspan. 2009. Expected reciprocal rank for graded relevance. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, page 621–630, New York, NY, USA. Association for Computing Machinery.

Olivier Chapelle and Ya Zhang. 2009. A dynamic bayesian network click model for web search ranking. In *Proceedings of the 18th international conference on World wide web*, pages 1–10.

Victor Chernozhukov, Denis Chetverikov, Mert Demirer, Esther Duflo, Christian Hansen, Whitney Newey, and James Robins. 2018. Double/debiased machine learning for treatment and structural parameters. *The Econometrics Journal*, 21(1):C1–C68.

Aleksandr Chuklin, Ilya Markov, and Maarten de Rijke. 2016. Click models for web search and their applications to IR: WSDM 2016 tutorial. In *Proceedings of the Ninth ACM International Conference on Web Search and Data Mining*, pages 689–690.

Nick Craswell, Onno Zoeter, Michael Taylor, and Bill Ramsey. 2008. An experimental comparison of click position-bias models. In *Proceedings of the 2008 international conference on web search and data mining*, pages 87–94.

Wenkui Ding, Xiubo Geng, and Xu-Dong Zhang. 2015. Learning to rank from noisy data. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 7(1):1–21.

Georges E Dupret and Benjamin Piwowarski. 2008. A user browsing model to predict search engine click data from past observations. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*, pages 331–338.

Zhichong Fang, Aman Agarwal, and Thorsten Joachims. 2019. 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.

Huifeng Guo, Jinkai Yu, Qing Liu, Ruiming Tang, and Yuzhou Zhang. 2019. PAL: a position-bias aware learning framework for CTR prediction in live recommender systems. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 452–456.

Dany Haddad. 2022. Noise tolerance of learning to rank under class-conditional label noise. *arXiv preprint arXiv:2208.02126*.

James J. Heckman. 1979. Sample selection bias as a specification error. *Econometrica*, 47(1):153–161.

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

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

Thorsten Joachims. 2006. Training linear SVMs in linear time. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 217–226.

Thorsten Joachims, Laura Granka, Bing Pan, Helene Hembrooke, Filip Radlinski, and Geri Gay. 2007. Evaluating the accuracy of implicit feedback from clicks and query reformulations in web search. *ACM Transactions on Information Systems (TOIS)*, 25(2):7–es.

Thorsten Joachims, Adith Swaminathan, and Tobias Schnabel. 2017. Unbiased learning-to-rank with biased feedback. In *Proceedings of the tenth ACM international conference on web search and data mining*, pages 781–789.

Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Chenliang Li, Dawei Yin, and Brian D Davison. 2024. Unbiased learning-to-rank needs unconfounded propensity estimation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1535–1545.

Dan Luo, Lixin Zou, Qingyao Ai, Zhiyu Chen, Dawei Yin, and Brian D Davison. 2023. Model-based unbiased learning to rank. In *Proceedings of the Sixteenth ACM International Conference on Web Search and Data Mining*, pages 895–903.

G. S. Maddala and F. Nelson. 1975. Switching regression models with exogenous and endogenous switching. In *Proceedings of the American Statistical Association, Business and Economics Section*, pages 423–426. American Statistical Association.

Harrie Oosterhuis and Maarten de Rijke. 2020. 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*, page 489–498. Association for Computing Machinery.

Zohreh Ovaisi, Ragib Ahsan, Yifan Zhang, Kathryn Vasilaky, and Elena Zheleva. 2020. Correcting for selection bias in learning-to-rank systems. In *Proceedings of The Web Conference 2020*, pages 1863–1873.

Zohreh Ovaisi, Kathryn Vasilaky, and Elena Zheleva. 2021. Propensity-independent bias recovery in offline learning-to-rank systems. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1763–1767.

Tao Qin and Tie-Yan Liu. 2013. Introducing LETOR 4.0 datasets. *arXiv preprint arXiv:1306.2597*.

Olav Reiersøl. 1941. Confluence analysis by means of lag moments and other methods of confluence analysis. *Econometrica*, 9(1):1–24.

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

Mucun Tian, Chun Guo, Vito Ostuni, and Zhen Zhu. 2020. Counterfactual learning to rank using heterogeneous treatment effect estimation. *Proceedings of ACM SIGIR Workshop on eCommerce (SIGIR eCom)*.

Ali Vardasbi, Harrie Oosterhuis, and Maarten de Rijke. 2020. When inverse propensity scoring does not work: Affine corrections for unbiased learning to rank. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1475–1484.

Hongning Wang, ChengXiang Zhai, Anlei Dong, and Yi Chang. 2013. Content-aware click modeling. In *Proceedings of the 22nd international conference on World Wide Web*, pages 1365–1376.

Xuanhui Wang, Michael Bendersky, Donald Metzler, and Marc Najork. 2016. Learning to rank with selection bias in personal search. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, page 115–124, New York, NY, USA. Association for Computing Machinery.

Xuanhui Wang, Nadav Golbandi, Michael Bendersky, Donald Metzler, and Marc Najork. 2018. Position bias estimation for unbiased learning to rank in personal search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining (WSDM)*, pages 610–618.

Jeffrey M Wooldridge. 2015. Control function methods in applied econometrics. *Journal of Human Resources*, 50(2):420–445.

Xin Wu, Qing Liu, Jiarui Qin, and Yong Yu. 2022. Peerrank: robust learning to rank with peer loss over noisy labels. *IEEE Access*, 10:6830–6841.

Ziwei Zhu, Yun He, Yin Zhang, and James Caverlee. 2020. Unbiased implicit recommendation and propensity estimation via combinational joint learning. In *Proceedings of the 14th ACM Conference on Recommender Systems*, pages 551–556.