Clickbait Mitigation in Industrial Recommender System

Václav Blahut[1,*], Karel Koupil[1], Michal Chudoba[2,†] and Radek Tomšů[1]

[1]Seznam.cz, a.s., Praha, CZ
[2]Praha, CZ

Abstract

Many online platforms struggle to control the massive increase in clickbait content. Publishers and professional content creators strive to get users attention in a highly competitive online environment, thus trying to write catchy, superlative, and misleading titles. In our setting, we run a semi-open on-line content platform with hundreds of professional publishers and thousands of content creators. We examined several different approaches, how to systematically penalize a clickbait item in industrial recommender system. To compare the efficiency of selected approaches, we conducted large-scale online AB experiments, reported results indicating KPI shifts, and shared several insights from application of the approach on our platform.

Keywords

Recommender systems, Clickbait, Debiasing, Auxiliary task, Industry, Post-processing

1. Introduction

Our content platform serves millions of active users each day, delivering personalized recommendations from a wide range of content creators. Users receive a customized mix of on-line media, including news, entertainment articles, videos, and podcasts. Clickbait (CB) often employs sensationalist headlines or descriptions that exaggerate the content's value or importance to lure users into clicking. These headlines often create a "curiosity gap" by providing just enough information to spur interest, but not enough to satisfy it, forcing users to click to learn more [1]. Without adequate control mechanisms, some publishers take advantage of the situation, leading to a surge in clickbait exposure and causing individual items to become excessively dominant in the content stream. Clickbait preys on human curiosity and emotional triggers, such as surprise, fear, or excitement. Using these psychological factors, clickbait aims to maximize click-through rates (CTR) at the expense of content quality [2, 3]. Therefore, it has a massive impact on industrial recommenders: erosion of user trust [1], degradation of content quality[2], challenges for content moderation [3], and also an economic impact, as clickbait is often driven by the economic incentives of online advertising [1], where platforms and content creators are rewarded for maximizing clicks and engagement. This creates a conflict between the goals of user satisfaction and revenue generation, and also between the short-term gains of publishers and the long-term sustainability of the content platform.

In this work, we investigate several approaches to integrate clickbait mitigation into an industrial recommendation system, ranging from heuristic methods to reinforcement learning, validated through extensive online A/B testing.

2. Clickbait mitigation

The initial step in mitigating clickbait involves using a model that assesses the degree of clickbaitiness in article titles. In our setting, we employ an internal BERT-like model trained on a large, carefully

INRA 2025: 13th International Workshop on News Recommendation and Analytics in Conjunction with ACM RecSys 2025, September 22–26, 2025, Prague, CZ

[*]Corresponding author.

[†]Work done while the author was an employee at Seznam.cz, a.s.

vaclav.blahut@firma.seznam.cz (V. Blahut); karel.koupil@firma.seznam.cz (K. Koupil); michal.chudoba.praha@gmail.com (M. Chudoba); radek.tomsu@firma.seznam.cz (R. Tomšů)
0009-0000-3026-2616 (V. Blahut); 0009-0001-2233-4074 (M. Chudoba)

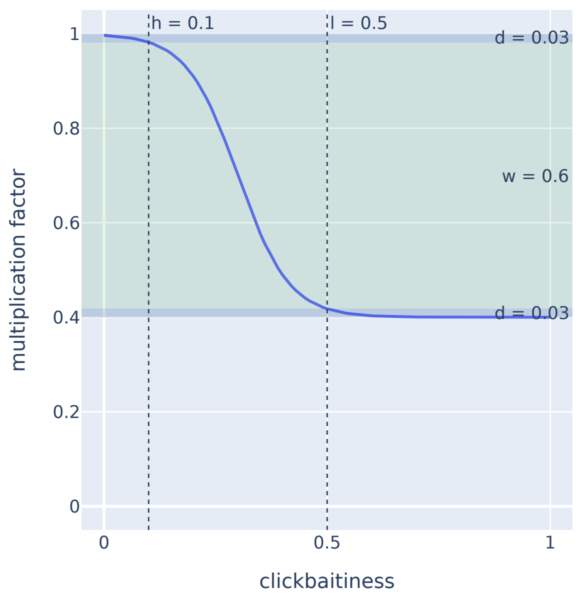

Figure 1: Customized sigmoid function

annotated dataset. The model classifies each article title and outputs a continuous score between 0 and 1, where 0 means that there is no clickbait and 1 signals a clickbait title. This model is static and used in all of our experiments.

We explore several methods to integrate clickbait mitigation into the recommendation system. First, we introduce a model-agnostic approach that requires no changes to the model architecture or the training process. Next, we modify the model itself by incorporating clickbait signals, both as an input feature and as an auxiliary learning objective, allowing the model to account for undesirable behavior through its loss function. Finally, we apply an actor-critic framework from reinforcement learning to guide the model's output distribution to more desirable recommendations. Each of these approaches is described in detail below.

2.1. Heuristic post-processing

One of the most straightforward, model-agnostic solutions to applying any kind of promotion or penalization logic based on some item attribute is to modify the score of an item that is used for ranking. In our case, we decided to reduce the item score by multiplying by a factor derived from the clickbaitiness of the item. To compute this factor, we use a custom, adjustable sigmoid function:

$$f(c,h,l,d,w) = (1-w) \ + \ \frac{w}{1+\left(\frac{d}{1-d}\right)^{1-\frac{2\,(c-h)}{l-h}}} \tag{1}$$

where c is the clickbaitiness of given item, h and l are the locations of the higher and lower control points, d is the relative drop of the value at the control points, and w is the weight, controlling the lowest value of the curve and therefore the overall effect of clickbait penalization. The c parameter is tied to a given item, and all the other parameters are treated as hyperparameters of the solution. The function, including a visualization of the hyperparameters and their values used in the experiment, can be seen in Figure 1.

2.2. Debiasing via shallow tower

The presence of clickbait in recommendation can be considered as a bias. We decided to experiment with an existing model debiasing technique that employs a shallow tower architecture[4], which bypasses the main model to isolate biases from genuine user utility. We used the clickbaitiness of an item as a dense feature connected to the shallow tower, bypassing the core of the model, and added the logit of a single dense layer with 0.1 dropout rate to the logits of multitask prediction heads. At prediction time, the clickbaitiness was zeroed. Previously, we also experimented with other baselines from [4] such as debiasing via simple input feature or adversarial learning via gradient negation of auxiliary task, but the shallow tower technique worked best for us, consistently with [4].

2.3. Auxiliary task – Clickbait reward

Even more direct solution how to penalize the model for recommending clickbait was to handle the clickbaitiness at loss level. While we tried reducing the weight of training data rows for clickbait items, the most promising approach turned out to be the utilization of the predicted CTR and item's clickbaitiness into an additional auxiliary task loss [5].

$$L_R = \log(\frac{1-c}{c}) * (2 * \bar{p}_{CTR} - 1) \tag{2}$$

This allows the model to directly learn which items to promote over others based on their clickbaitiness. The effect of this approach can be adjusted by modifying the loss weight.

2.4. Reinforcement learning

While auxiliary task solution does properly punish high-clickbait articles, it still needs interactions to learn such punishment from. We therefore looked for solutions that are more in the realm of model alignment rather than only standard supervised learning from implicit interactions. Inspired by RLHF [6] and other methods used in LLMs [7], we came up with a method for recommendation models.

We sample from a catalogue of available articles and perform an actor-critic cycle with KL divergence bounds over frozen model with the previously mentioned clickbait loss to align the model.

$$L_{\text{critic}} = MSE(\bar{R}; -L_R) \tag{3}$$

$$L_{KL} = D_{KL}(p_{\text{frozen}}; p_{\text{actor}}) \tag{4}$$

$$L_{\text{actor}} = w_{KL} * L_{KL} - w_R * \bar{R} \tag{5}$$

The impact of this method can be regulated by adjusting the loss weight as well as by modifying the way the items are sampled from catalogue.

3. Experimental setup

The main pipeline of our recommender system follows the 4-stage framework as described by Higley et al. [8]. At ranking stage, the system utilizes Deep and Cross Network V2[9] model from TensorFlow Recommenders library[1]. We use multitask learning to predict the click-through rate $\bar{p}_{CTR}$ and the time $\bar{p}_{time}$ spent consuming the item. For final ranking, the predictions are then combined using the formula

$$s(\bar{p}_{CTR}, \bar{p}_{time}) = \bar{p}_{time}^{\;\bar{p}_{CTR}^{\;0,8}} \tag{6}$$

as further described in [10]. The model is trained incrementally every 5 minutes. An endless feed of recommended items consists of 20-item slates, incrementally generated online as user scrolls through the feed. All hyperparameters of the system, including those of the variants discussed in this work, were selected based on preceding exhaustive series of experiments.

[1]https://www.tensorflow.org/recommenders/

Table 1
Experimental results

Variant	Impressions			CTR	Clicks	Spent time	∇Bounce rate
	Non-CB	Light CB	Heavy CB				
Post-process	+32.9 %	-20.1 %	-77.9 %	-4.2 %	-2.9 %	-1.5 %	-3.3 %
Shallow tower	+3.3 %	-3.4 %	-12.1 %	+0.9 %	-0.6 %	+0.8 %	-0.3 %
Auxiliary task	+47.5 %	-35.9 %	-81.4 %	-22.0 %	-14.7 %	+2.2 %	-13.1 %
Reinforcement learning	+55.0 %	-37.3 %	-43.5 %	-27.6 %	-14.6 %	-3.2 %	-6.1 %

3.1. A/B testing

In order to assess the performance of our recommender system, we perform randomized A/B tests on live traffic. For each of the tested variants, we include A/A variant. The size of each variant as well as test duration are chosen empirically w.r.t. statistical significance. The unit of randomization in our AB testing system is the user.

3.2. Metrics

To evaluate the effectiveness of the tested methods in reducing the amount of clickbait items in recommendations, and to see its impact on the KPIs, we report these user-level metrics:

- Number of impressions per clickbait category - to show the shifts in the amount of recommended CB, we categorize items into three categories according to their clickbait score corresponding to non-CB, light CB and heavy CB.
- Click-through rate - the number of clicks divided by number of item impressions.
- Number of clicks
- Spent time - the sum of amount of time user spent consuming clicked items.
- Bounce rate - the ratio of clicks resulting in immediate leave after the content of article is displayed. Unlike other mentioned metrics, reducing bounce rate means improvement.

All metrics are computed per user on personalized portion of traffic and their reported changes are in relative scale.

4. Results

Clickbait mitigation can lead to various outcomes. In the short-term, it most likely leads to a reduced number of clicks in the system. Clickbait titles have generally higher click-through-rate than other article titles. The same may not be true for other metrics that try to measure user satisfaction, such as spent time, the overall number of user interactions, etc. Our findings suggest that some clickbait titles are connected with a high number of bounces, i.e. users leaving the article in a short time and not reading a significant part of it. However, there are also clickbait titles that keep the users interested in the articles. Therefore, it is not straightforward to predict the outcomes of the mitigation on these metrics.

Examining the experimental results reveals that each tested approach exhibited distinguishably different behavior. All of the methods successfully reduced the exposure of CB, some of them more aggressively than others and with varying impacts on KPIs. Without any doubt, the heuristic post-process method is the overall favorite, substantially reducing the exposure of heavy CB similarly to the most aggressive Auxiliary task method, but with much lower, manageable impact on KPIs. Another positive aspect is that the method is easy to set up and fine-tune to desired trade-off, model-agnostic and therefore potentially applicable to other than personalized recommendation scenarios.

The debiasing via shallow tower variant has also shown interesting results. It is not as powerful in suppressing CB as other methods, however the effect is still not negligible, especially considering the improvements in majority of KPIs. The slight drop in number of clicks is compensated by an increase in spent time, meaning that users spent more time consuming the items rather than browsing the recommendations, which is considered a positive change. One potential limitation of this method is the inability to control the extent of clickbait suppression.

The auxiliary task produced highly contradictory results. On the one hand, it reduced CB impressions more effectively than the other approaches. On the other hand, it led to a significant decline in click-related metrics. Interestingly, despite this drop, users spent more time consuming the content overall, and the bounce rate decreased significantly more than with the other methods. These mixed outcomes are intriguing and give us flexibility in choosing which KPIs to prioritize.

Reinforcement learning resulted in a strong reduction in clickbait, but this came at the cost of an overall decline in both click and time-based metrics. Its outcomes were somewhat similar to those of the auxiliary task, with the added downside that even time spent on site decreased slightly. This is the most challenging setup, and so far, it hasn't delivered the level of performance we were hoping for.

5. Conclusion

Although we were open to use machine learning approaches to mitigate the presence of clickbait in recommendations, the heuristic post-process method demonstrated the best trade-off - it effectively reduced the exposition of heavy clickbait and positively impacted bounce rate with a rather minor cost in terms of the performance in clicks and CTR. A natural continuation of this research will be to explore a combination of multiple approaches, such as shallow tower and post-process methods, to further enhance the results. This combination may leverage the strengths of both heuristic and ML solutions to improve clickbait mitigation and user satisfaction.

Acknowledgments

We thank Vít Líbal and Daniel Procházka for their extensive feedback on this paper. We thank Josef Florian and Petr Zelenka for their assistance with conducting the experiments. We thank Minh Duc Pham for sigmoid function reduction.

Declaration on Generative AI

During the preparation of this work, the authors used ChatGPT, Microsoft Copilot and Perplexity AI in order to: Paraphrase and reword, Improve writing style. Further, the authors used SciSpace in order to: Citation management. After using these tools, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] W. Wang, F. Feng, X. He, H. Zhang, T.-S. Chua, Clicks can be cheating: Counterfactual recommendation for mitigating clickbait issue, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR '21, ACM, 2021, p. 1288–1297. URL: http://dx.doi.org/10.1145/3404835.3462962. doi:10.1145/3404835.3462962.

[2] N. Immorlica, M. Jagadeesan, B. Lucier, Clickbait vs. quality: How engagement-based optimization shapes the content landscape in online platforms, 2024. URL: https://arxiv.org/abs/2401.09804. arXiv:2401.09804.

[3] D. Jácobo-Morales, M. Marino-Jiménez, Clickbait: Research, challenges and opportunities – a systematic literature review, 2024. URL: https://doi.org/10.30935/ojcmt/15267.

[4] Z. Zhao, L. Hong, L. Wei, J. Chen, A. Nath, S. Andrews, A. Kumthekar, M. Sathiamoorthy, X. Yi, E. Chi, Recommending what video to watch next: a multitask ranking system, in: Proceedings

of the 13th ACM Conference on Recommender Systems, RecSys '19, Association for Computing Machinery, New York, NY, USA, 2019, p. 43–51. URL: https://doi.org/10.1145/3298689.3346997. doi:10.1145/3298689.3346997.

[5] L. Liebel, M. Körner, Auxiliary tasks in multi-task learning, 2018. URL: https://arxiv.org/abs/1805.06334. arXiv:1805.06334.

[6] D. M. Ziegler, N. Stiennon, J. Wu, T. B. Brown, A. Radford, D. Amodei, P. Christiano, G. Irving, Fine-tuning language models from human preferences, arXiv preprint arXiv:1909.08593 (2019).

[7] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, S. Ermon, C. Finn, Direct preference optimization: Your language model is secretly a reward model, Advances in Neural Information Processing Systems 36 (2023) 53728–53741.

[8] K. Higley, E. Oldridge, R. Ak, S. Rabhi, G. de Souza Pereira Moreira, Building and deploying a multi-stage recommender system with merlin, in: Proceedings of the 16th ACM Conference on Recommender Systems, RecSys '22, Association for Computing Machinery, New York, NY, USA, 2022, p. 632–635. URL: https://doi.org/10.1145/3523227.3551468. doi:10.1145/3523227.3551468.

[9] R. Wang, R. Shivanna, D. Cheng, S. Jain, D. Lin, L. Hong, E. Chi, Dcn v2: Improved deep & cross network and practical lessons for web-scale learning to rank systems, in: Proceedings of the Web Conference 2021, WWW '21, Association for Computing Machinery, New York, NY, USA, 2021, p. 1785–1797. URL: https://doi.org/10.1145/3442381.3450078. doi:10.1145/3442381.3450078.

[10] J. Florian, J. Drdák, R. Tomsu, K. Koupil, V. Blahut, J. Kuchar, M. Rehor, Combining models for better user satisfaction in video recommendation., in: ORSUM@ RecSys, 2020.