Non-Stationary Multi-Armed Bandits for News Recommendations

Noah Daniëls[1,2,*], Bart Goethals[1,2,3]

[1]University of Antwerp, Antwerp, Belgium
[2]Froomle, Antwerp, Belgium
[3]Monash University, Melbourne, Australia

Abstract

In the dynamic landscape of online news, recommending relevant articles is key to enhancing user engagement and satisfaction. Traditional Multi-Armed Bandit (MAB) approaches, however, struggle with the specific non-stationary nature of news consumption, where article relevance shifts rapidly due to emerging trends and breaking news. Through simulations, we explore the effectiveness of non-stationary adaptations such as sliding windows, exponentially weighted averages, and discounting, revealing some limitations in their performance. Furthermore, we introduce a novel algorithm, Optimistic/Pessimistic-Greedy (OP-Greedy), which leverages external signals such as article page views to optimize exploration. Our findings demonstrate that tailoring MAB algorithms to the unique dynamics and opportunities of news recommendations can significantly improve their adaptability and effectiveness.

Keywords

news, recommender system, multi-armed bandit problem, non-stationarity

1. Introduction

In the rapidly evolving landscape of online news consumption, the ability to effectively recommend relevant articles has become crucial for user engagement and satisfaction. This task, characterized by the inherent dilemma of balancing exploration (discovering new, potentially relevant articles) and exploitation (leveraging known popular articles), is well-modeled by the Multi-Armed Bandit (MAB) problem. Based on our experience in running recommendation systems for several large online news media sites, we noticed that traditional MAB approaches fall short in addressing the unique challenges presented by the non-stationary nature of news consumption, where article relevance fluctuates drastically over short periods due to emerging trends, breaking news, and shifting user interests.

This study introduces practical enhancements to conventional MAB algorithms, specifically tailored to the domain of news recommendations. Unlike existing methodologies that predominantly focus on contextual [1, 2, 3, 4] or general non-stationary settings [5, 6, 7, 8], our approach deals explicitly with the specific non-stationary dynamics inherent in online news recommendations. For this study, we focus on the non-contextual setting before investigating the more complex contextual version. As the largest proportion of traffic on a typical news media site is from anonymous, or cold-start, visitors, any improvement might already have a significant impact on the news website.

Our contributions are twofold: First, we evaluate established MAB algorithms in their ability to handle the dynamic nature of news content and show some surprising results. We run experiments in a simulated environment and investigate how mechanisms such as sliding windows can help or hinder algorithms in their ability to deal with the non-stationary behavior of news. Second, we describe a novel algorithm called Optimistic/Pessimistic-Greedy (OP-Greedy) which leverages external signals,

INRA'24: International Workshop on News Recommendation an Analytics, October 14–18, 2024, Bari, Italy
*Corresponding author.
noah.daniels@uantwerpen.be (N. Daniëls)
0009-0001-7778-4769 (N. Daniëls); 0000-0001-9327-9554 (B. Goethals)

such as article page views, to guide the exploration process more efficiently, reducing the need for indiscriminate exploration and focusing efforts on articles with higher potential for user engagement.

By addressing the unique challenges of the news recommendation domain, our study seeks to enhance the practical utility of MAB algorithms, ultimately contributing to the development of more adaptive, effective, and user-centric news recommendation systems.

2. Background

In the classical multi-armed bandit problem [9], an agent is faced with a row of slot machines (arms), each with an unknown reward distribution. The goal is to maximize the total reward gained over a series of pulls. This requires balancing the exploration of different arms to learn their reward distributions with the exploitation of arms that seem to provide the highest rewards. When applied to news recommendation systems, the arms represent different news articles and the act of pulling an arm corresponds to recommending the associated article. An article's reward distribution is characterized by its click-through-rate (CTR) and the agent typically receives binary feedback indicating whether the article was clicked or not. The agent must simultaneously identify articles with high CTRs and maximize aggregate clicks.

Our investigation is anchored in the non-stationary multi-armed bandit (NS-MAB) problem [5, 6], where the unknown reward distributions change over time. This introduces additional complexity as now there isn't a single optimal arm anymore. In general, when the rewards can change arbitrarily, no theoretical guarantees can be provided. For this reason, researchers study classes of problems in which the amount of change is bounded, for example by having a limited number of time points at which the rewards change [7, 8] or a limited total variation of the expected rewards [5, 8].

While these research efforts provide interesting theoretical guarantees for large classes of non-stationary problems, they sometimes fail to produce effective algorithms in practice. Our work diverges from this existing literature by addressing the specific non-stationary dynamics inherent in online news recommendations. By adapting MAB algorithms to this context, we contribute to closing the gap between theoretical bandit solutions and their applicability in the dynamic, real-world scenario of news recommendations.

Concretely, we make the following key assumptions about the characteristics of the news recommendation environment:

- **Decreasing CTRs**: due to the rapid turnover inherent to news content, article CTRs tend to decrease as more recent news becomes available and old news loses relevancy [10].
- **Large and dynamic item pool**: the number of news articles is large but not all articles are available from the start. Instead, new articles are introduced periodically while old articles are removed after their fixed lifetime.
- **Low signal-to-noise ratio**: The large item pool and short lifespans result in few recommendation that actually receive clicks. The CTRs for most items is lower than 1%, which necessitates extensive data collection.
- **External signals**: A news site might share their articles on social-media, send push-notification or news letters, and generate clicks through other curated or automated recommendations. These external sources of article clicks and page view lead to signals about an article's popularity which we should take advantage of.

These assumptions motivate the design of our simulated environment and they are crucial for explaining and understanding our results. Furthermore, the opportunity represented by external signals is the motivation behind our novel OP-Greedy algorithm.

2.1. Related Work

The theoretical foundations regarding different variations of the multi-armed bandit problem have been extensively studied, from the classical problem [11, 12] to variants including contextual [13, 14, 15] and

non-stationary [5, 6, 7, 16, 8] problems.

Besides these theoretical works, a lot of research has also been done on applying these technique to recommendation systems. The most well-known of these efforts is linUCB [17] which was conceptualized in the domain of personalized news recommendations, but did not consider the non-stationarity of news. In general, the focus of recommendation system bandit research has been on contextual bandits where user and or item features can be used to create personalized recommendations. For this reason, recent research considering non-stationarity in recommendation systems is applied to contextual bandits [1, 2, 3, 4]. Moreover, this research primarily focuses on changing user interests rather than changing item relevance

We believe that the non-contextual case can be equally important for the news domain as a large portion of news traffic is often from anonymous users. Therefore, we decided to diverge from existing literature and focus on this overlooked case, and leave the much more complex contextual version for further work.

3. Methodology

In this section, we describe popular MAB algorithms in the context of news recommendations and highlight possible adaptations to make them more suitable for the news NS-MAB problem. We also detail how our experiments were conducted, including a detailed description of our simulation environment.

3.1. Base Algorithms

ϵ-Greedy (EG), Upper Confidence Bound (UCB) [11], and Thompson Sampling (TS) [18, 19], are three popular MAB algorithms, all of which base their decisions on estimates computed from observed clicks and the number of impressions (recommendations). To ease notation, let $n_i(t)$ and $c_i(t)$ denote the number of times article i has been recommended and clicked respectively, up until time t. The estimated CTR of item i at time t is then given by $\bar{\mu}_i(t) = n_i^{-1}(t)c_i(t)$.

During each time step, after the agent has chosen a set I_t of items to recommend and obtained feedback for them in the form of boolean rewards indicating clicks $\{r_{i,t}|i \in I_t\}$, the statistics are updated as follows:

$$n_i(t) = n_i(t-1) + \mathbb{1}[i \in I_t]$$
$$c_i(t) = c_i(t-1) + r_{i,t}$$

with $n_i(0) = c_i(0) = 0$ and $r_{i,t}$ being zero if item i was not recommended at time t.

3.1.1. Epsilon Greedy

The most basic, yet surprisingly effective algorithm is ϵ-greedy. It handles the exploration/exploitation trade-off by picking a random set of items with probability ϵ and greedily picking the items with the highest estimated CTR otherwise. If ϵ is chosen properly, this approach will exploit the best items often while not getting stuck on sub-optimal items.

3.1.2. Upper Confidence Bound

Instead of exploring randomly as with EG, the UCB algorithm will guide exploration by considering the uncertainty of its estimates. Concretely, UCB looks at the confidence intervals for the CTR estimates and always select the article which has the highest upper confidence bound. This way, items with high CTR will get recommended, but also those that have not been explored sufficiently and have the potential for high CTR. In practice, UCB assigns a score $q_i(t)$ to each item and picks the highest scoring items. The scores are calculated as follows:

$$q_i(t) = \bar{\mu}_i(t) + c\sqrt{\frac{\ln n(t)}{n_i(t)}}$$

where c is a hyper-parameter regulating the required confidence and $n(t)$ it the total impressions count before time t of all available items.

3.1.3. Thompson Sampling

TS guides its exploration by recommending each item in proportion with the probability that they are optimal based on previous evidence. Similarly to UCB, this will cause more exploration for items with uncertain and promising estimates. In practice, the probability of item i being optimal is modeled using a beta distribution $Beta(1 + c_i(t), 1 + n_i(t) - c_i(t))$. At recommendation time, each item's distribution is sampled and those with the highest samples are recommended.

3.2. Adaptations for Non-Stationary Settings

Three notable ways to adapt MAB algorithms for non-stationary settings are to use sliding windows, discounting and weighted averages. All these approaches aim to increase the accuracy of the CTR estimates by either discarding old observations (sliding windows) or placing more importance on recent observations (weighted averages and discounting).

3.2.1. Sliding windows

A global (or time-based) sliding window will only consider the last τ interactions (or last τ minutes of interactions) when determining the click and impression counts [20, 8, 7]. For UCB, this also means changing the calculation of $n(t)$. This is the most common form of sliding windows because it ensures that items which have stopped being recommended will eventually be tried again as their observations fall outside the window and they are treated as new items again. This is necessary in cases where the relevance of items can change arbitrarily. However, in news, the relevance of items is expected to decrease, which would lead a global sliding window to keep exploring unnecessarily.

To resolve this, we can apply the sliding windows locally, making the algorithm only consider the last τ interactions of each item individually. When an item is deemed irrelevant by an algorithm using this approach, it is far less likely to be recommended again because the window will stop advancing as the item stops receiving recommendations. On the other hand, relevant items which receive many recommendations do advance the window, allowing the estimates to update quickly if they start to become less relevant.

3.2.2. Weighted averages

With exponential recency-weighted averages [9], the estimated CTR $\bar{\mu}_{i,t}$ of item i is not calculated from the click and impression counts but instead iteratively updated each time item i is recommended according to the following update rule:

$$Q_i(n) = Q_i(n-1) + \alpha(r_{i,n} - Q_i(n-1))$$

where α is the learning rate, and $r_{i,n}$ indicates the reward obtained from the n-th recommendation. The estimated CTR is always set to the latest value: $\bar{\mu}_i(t) = Q_i(n_i(t))$.

With this update rule, more weight is given to more recent observations and similar to the local sliding windows, this technique does not change CTR estimates when an item is not recommended.

For TS, where the beta distribution is parameterized based on the click and impression counts, and not the estimated CTR, we first calculate the number of clicks that would achieve the estimated CTR given the number of impressions:

$$Beta(1 + n_i(t)\bar{\mu}_i(t), 1 + n_i(t)(1 - \bar{\mu}_i(t))$$

A possible extra hyper-parameter when using weighted averages is the starting estimate $Q_i(0)$. When it is set higher than the expected true CTR we call it an *optimistic* starting value while if it is

set lower we call it a *pessimistic* starting value. Sometimes, optimistic weighted averages are used in combination with ϵ-greedy in which case ϵ can be set to zero as the optimism can drive the exploration for new items.

3.2.3. Discounting

A related technique to weighted averages is that of discounting [7, 16], where the impression and click counts are individually updated after each time step with the following update rule:

$$n_i(t) = \alpha n_i(t-1) + \mathbb{1}[i \in I_t]$$
$$c_i(t) = \alpha c_i(t-1) + r_{i,t}$$

While this technique gives more weight to more recent observations like weighted averages, it does so in way where all estimates are updated after each time step, which makes it more similar to global sliding windows.

3.3. Optimistic/Pessimistic-Greedy

On typical news portals, bandit-based recommender system will only be a small part of a larger ecosystem. A news site might share their articles on social-media, send push-notification or news letters, and generate clicks through other curated or automated recommendations. These other sources of article clicks and page view leads to signals about an article's popularity which we should take advantage of.

An interesting option presents itself when we use such popularity signals to decide whether an article warrants exploration or not. In this subsection, we describe a novel algorithm for doing so called *optimistic/pessimistic*-greedy (OP-greedy) which is based on weighted averages and greedy item selection. The idea behind our approach is to use external popularity information to retroactively decide whether we should use optimistic or pessimistic starting values.

Our algorithm expects as input the ranking of items based on the popularity signal. Note that we expect this ranking to contain valuable information, but not be good enough to use without modifcation. Based on an item's ranking $\text{rank}(i)$, we interpolate between an optimistic starting value $\hat{Q}(0)$ and a pessimistic starting value $\check{Q}(0)$ as follows:

$$Q_i(0) = \check{Q}(0) + (\hat{Q}(0) - \check{Q}(0)) \exp\left(-\frac{\text{rank}(i)}{\gamma}\right)$$

where γ is a hyper-parameter controlling how large the rank can be before being seen as pessimistic.

By writing the weighted average update rule as a direct formula, we see how the starting value can be retroactively changed:

$$Q_i(n) = (1-\alpha)^n Q_i(0) + \sum_{k=0}^{n-1} (1-\alpha)^k \alpha r_{i,t-k}$$

The effect of the starting value can be included in the CTR estimate by using weighted averages as normal (without a special starting value) and adding $Q_i(0)$ multiplied by $(1-\alpha)^{n_i(t)}$ to determine the final result.

3.4. Experimental Setup

Each experiment operates on a fixed number of episodes $T = 2000$. During each episode, the agent under evaluation must make a number of top-3 recommendation from a set of available items. After each recommendation the simulator provides binary feedback for *each* of the 3 recommended items, indicating whether each item was clicked or not. The feedback is based on simulated ground truth CTRs which remain constant during the episode. After P recommendations have been made (episode length P depends on the simulation), the next episode starts with possibly changed CTRs and item availability.

The main metric we look at is the average CTR, which is the total number of obtained clicks divided by the total number of impressions. Since the number of impressions is constant for all agents, this effectively measures how many clicks each agent obtained. To normalize the results across environments, we report the relative CTR compared to an omniscient agent which always picks the 3 articles with highest ground truth CTR. Thus, a relative CTR of 80% means the agent missed out on 20% of clicks that it could have received if it had picked optimally.

We repeat each experiment N times with different seeds and report the 95% confidence intervals of the results. The seed determines the evolution of the CTRs and thus each run of the experiment corresponds to an entirely different environment with different items. To further improve the accuracy of our results, we count the expected number of clicks instead of the obtained number of clicks to calculate the CTR. In the simulations we have access to the ground truth CTR which is equivalent to the click probability. While the agent receives boolean feedback based on the probabilities, data collection can use the CTRs directly for more accurate results.

All algorithms were tuned by evaluating each parameter combination on 15 runs. The best performing combinations were selected and used for the final 100 runs on different seeds. Those final results are the ones we report.

3.5. Simulation

In this subsection, we describe how the ground truth CTRs and their dynamics are determined in our simulation. As a reminder, the CTR of an item should be highest in the first half of its lifetime and decrease as time goes on. Nevertheless, the simulation should leave room for variation. Some items may keep their CTR constant or slightly rising for a while and some may exhibit multiple peaks of high CTR or reach their maximum near the middle of their lifetime.

To achieve this behavior, we model the base relevance $f_{i,t}$ of item i during episode t of its lifetime using exponential decay as follows

$$f_{i,t} = b_i \exp(-(\sigma_i t)^2)$$

where b_i and σ_i are item dependent simulation parameters controlling the item's maximum relevance and evolution speed respectively.

Only using the base relevance would result in all CTR curves simply trending toward 0 which is not what we want. Instead, the base relevance is modified by looking at the rank $\text{rank}_t(i)$ of item i's relevance w.r.t. all other items. This gives rise to the following formula for the ground truth CTR of item i

$$\mu_{i,t} = g\left(\frac{f_{i,t}}{\ln(a \cdot \text{rank}_t(i) + e)}\right)$$

where $a \geq 0$ is a constant regulating how much relevant articles are boosted compared to less relevant articles and $g(\cdot)$ denotes Gaussian smoothing. Using this formula, the items with highest base relevance have boosted CTR, but can suddenly drop when their underlying relevance falls below the relevance of other items. Figure 1 shows that the CTR curves described above indeed follow the desired behaviour. The CTRs are generally highest in the beginning and generally decrease, but some items have multiple peaks, no peaks at all, or only peak after a while.

To define sensible simulation parameter values, we assume an episode corresponds to 5 minutes of real time, meaning the time horizon covers about a week. We set the lifespan of items to 900 episodes (3 days) and use 300 items in total (135 items on average available at any time). To achieve a low signal to noise ratio, we set the maximum relevance to 0.005 ± 0.002 uniformly distributed across items. The evolution speed range, a and smoothing parameters were chosen by trial and error to achieve the desired CTR curve shapes.

Regarding the episode length P, we consider two simulation versions with identical parameters as described above but different amounts of throughput. This allows us to study how algorithms are affected by different amounts of available information.

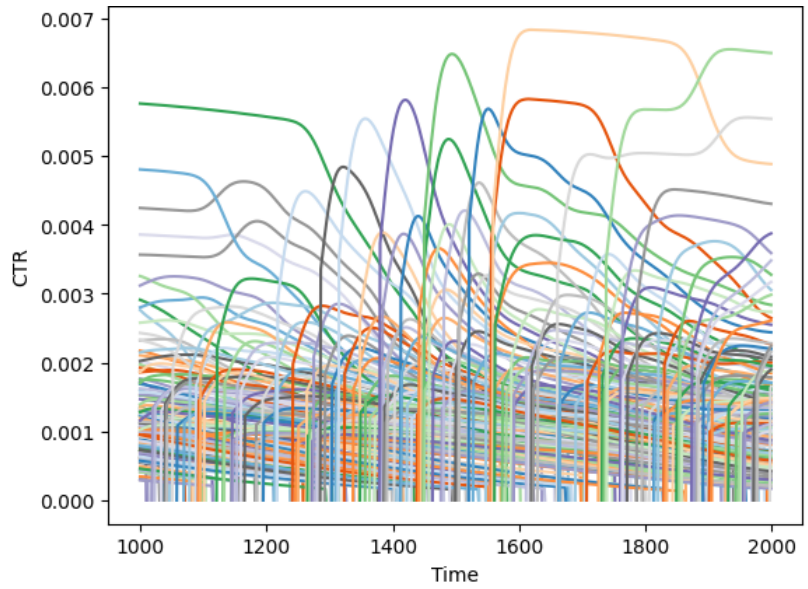

Figure 1: Simulated CTR curves.

- **Simulation A**: we set $P = 100$ corresponding to 1 item recommendation per second of real time (600,000 in total).
- **Simulation B**: we set $P = 1000$ corresponding to 10 item recommendations per second of real time (6,000,000 in total).

4. Results

4.1. Evaluating Non-stationary Adaptations

Table 1 shows the performance on Simulation A of various algorithms combined with various non-stationarity adaptations. The best combination for each algorithm is highlighted in bold. Table 2 shows the same but for Simulation B.

Table 1
Relative CTR of various algorithms for Simulation A.

Adaptation	Base algorithm			
	ϵ-Greedy	Greedy ($\epsilon = 0$)	UCB	TS
—	0.511 ± 0.012	0.276 ± 0.012	0.580 ± 0.008	$\mathbf{0.469 \pm 0.006}$
sliding window (global)	0.536 ± 0.011	0.457 ± 0.009	0.602 ± 0.009	0.293 ± 0.001
sliding window (local)	0.541 ± 0.012	0.520 ± 0.012	0.664 ± 0.012	$\mathbf{0.468 \pm 0.005}$
weighted (optimistic)	0.495 ± 0.007	0.488 ± 0.009	0.503 ± 0.005	0.419 ± 0.005
weighted (pessimistic)	$\mathbf{0.656 \pm 0.011}$	$\mathbf{0.709 \pm 0.014}$	$\mathbf{0.690 \pm 0.011}$	0.440 ± 0.004
discounting	0.551 ± 0.012	0.279 ± 0.012	0.502 ± 0.009	0.291 ± 0.001

4.1.1. Sliding windows

From the results, we can see that global sliding windows are not beneficial for algorithms which employ guided exploration, such as UCB and TS. The reason for this is that—by design—such sliding windows forget previously explored items, resulting in those items being explored again. In arbitrary non-stationary environments, this can be desirable as those items could have become relevant since

Table 2
Relative CTR of various algorithms for Simulation B.

Adaptation	Base algorithm			
	ϵ-Greedy	Greedy ($\epsilon = 0$)	UCB	TS
—	0.659 ± 0.010	0.271 ± 0.014	0.816 ± 0.006	0.778 ± 0.005
sliding window (global)	0.784 ± 0.006	0.483 ± 0.012	0.626 ± 0.005	0.337 ± 0.002
sliding window (local)	0.784 ± 0.008	0.654 ± 0.010	0.868 ± 0.004	$\mathbf{0.814 \pm 0.003}$
weighted (optimistic)	0.762 ± 0.004	0.828 ± 0.004	0.799 ± 0.004	0.738 ± 0.004
weighted (pessimistic)	$\mathbf{0.824 \pm 0.004}$	$\mathbf{0.873 \pm 0.006}$	$\mathbf{0.879 \pm 0.005}$	0.792 ± 0.004
discounting	0.775 ± 0.007	0.281 ± 0.014	0.538 ± 0.005	0.277 ± 0.001

they were last recommended. In the case of news however, the relevancy of items tends to decrease, making such exploration efforts unnecessary. For random exploration strategies such as ϵ-greedy, this is not a concern because the exploration is purely random. Instead, global sliding windows provide an advantage in this case as items that are starting to perform poorly get identified more quickly.

Local sliding windows are beneficial to all algorithms, especially in Simulation B. Local sliding windows enable algorithms to more quickly detect when the items they are recommending start to under-perform, without incurring excessive exploration as global sliding windows do.

4.1.2. Weighted averages

The results also show that weighted averages are beneficial for news. Surprisingly, pessimistic starting values perform the best with optimistic starting values even being slightly worse than most non-adapted algorithms in Simulation A. The success of pessimistic starting values further supports the view that limiting exploration is the best strategy in this environment. In fact, with pessimistic starting values, new items will generally not be explored as much when they are added. Only when the best performing items (which are being recommended) start to lose relevance and drop below the starting value, will more items be explored. There are too many new items to explore effectively so algorithms should only explore when necessary.

4.1.3. Pessimistic greedy

Pessimistic greedy (ϵ-greedy with $\epsilon = 0$ and pessimistic weighted averages) takes the previous one step further and actually does not explore new items at all, until the currently best performing items perform worse than the starting value. Despite presumably missing many good items this way, pessimistic greedy is the best performing algorithm tied with UCB in both simulations.

4.1.4. Discounting

The results show that discounting is the least effective strategy. As with global sliding windows, discounting encourages unwanted continued exploration in guided strategies. However, unlike global sliding windows which caused the greedy algorithm to perform better, discounting has no such benefit. The reason for this is that with global windows the estimated CTR of items which are not being recommend can increase when impressions fall outside the window, which can lead to the greedy action changing. With discounting, the estimates only ever decrease if the item is not being recommended, causing the algorithm to get stuck on suboptimal items as in the non-adapted version of greedy.

4.2. Optimistic/Pessimistic-greedy

Table 3 shows the performance of optimistic/pessimistic-greedy (OP-greedy). To simulate the external popularity signal, we apply varying amounts of noise to the ground truth CTRs. The items are then ranked by their perturbed CTR and this ranking is provided as popularity information to the algorithm.

Table 3
Relative CTR of popularity and optimistic/pessimistic-greedy with varying amounts of noise.

Simulation A

	low	medium	high
Popularity	0.735 ± 0.014	0.510 ± 0.015	0.375 ± 0.009
OP-Greedy	0.814 ± 0.008	0.731 ± 0.011	0.690 ± 0.012

Simulation B

	low	medium	high
Popularity	0.727 ± 0.013	0.523 ± 0.014	0.375 ± 0.010
OP-Greedy	0.924 ± 0.003	0.884 ± 0.005	0.867 ± 0.006

To validate that this popularity signal is not too accurate (which would make increased performance trivial), we report the performance of a pure popularity algorithm which simply recommends the top-3 items according to the signal. We calibrated the amount of noise based on the performance of this popularity algorithm and report results for low, medium and high amounts of noise representing good, medium and poor quality signals.

The results show that leveraging popularity signals to guide exploration can be very beneficial. Even with high amounts of noise the performance is comparable to that of the best algorithms in our other results, showing that OP-greedy is robust to poor signal quality. Furthermore, when the amount of noise is low, OP-greedy significantly outperforms the other algorithms we tested. Knowing that pure popularity approaches already typically perform surprisingly well in practice, this approach is very promising.

4.3. Impression plots

To gain further insight into why these algorithms are performing the way they do, we investigated which items are being recommended across times. To facilitate this, we evaluated some of the algorithms on a fixed simulation seed (the same seed as in Figure 1) so that the items are the same across runs. From these runs, we created plots showing the impression rate of the items. This was done by plotting a bar chart for every 10 episodes which shows how often each item is recommended in that time span. By placing those charts next to each other, we get a plot where the x-axis represents time and the height of items on the y-axis represents their impression rate.

As there are hundreds of items, we identify them using a sequence of 20 repeating colors. The ordering of items is fixed and based on the order they were added to the item pool, with older items appearing lower in the bar charts. To help interpret the plot, we also show the optimal impression distribution according to the omniscient agent above the plots. As each recommendation consists of three items, the optimal distribution shows three items on average. How close an algorithm's impression plot matches with the optimal impressions, directly determines its performance.

Figure 2 shows an example impression plot for the random algorithm which always selects random available items. The plot shows how each item gets an equal share of the impressions. The colouring gives a striped appearance evoking downwards movement. This is explained by the introduction of new items, which are added to the top of the bars, and old items leaving the item pool, which are removed from the bottom. This highlights that only an item's height shows the impression rate; its position is irrelevant.

4.3.1. Sliding windows

Figure 3 shows impressions plots for ϵ-greedy and TS and their adapted versions with global and local sliding windows on Simulation B.

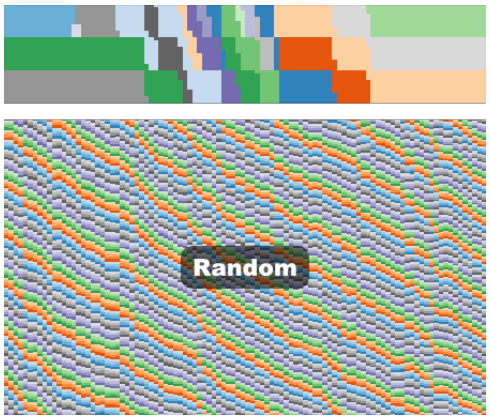

Figure 2: Impression plot for random recommendations. Optimal distribution is also shown (unlabeled, top). Each vertical slice shows how the impressions for that time interval were distributed across the items. Each color represents an item, with colors repeating and items alwyas occuring in the same order from oldest to newest, top to bottom.

The plots show how the non-adapted versions suffer from not being able to detect when an item stops being relevant. For example, both algorithms continue to recommend the dark green item (optimal in the beginning) long after it stopped being optimal. In contrast, the local sliding window versions adapt much more quickly and are able to switch more quickly to recommending relevant items.

As explained previously, the global sliding window adaptation works well for the unguided ϵ-greedy algorithm, but falls apart for TS because it needs to keep exploring all items. This results in an impressions plot which resembles random recommendations. On the other hand, the ϵ-greedy plots of both sliding window types look nearly identical because the exploration is unaffected and so both versions give the same benefit of faster change detection.

4.3.2. Greedy strategies

Figure 4 shows impression plots for various greedy strategies using weighted averages on Simulation B. The first two plots are for optimistic and pessimistic starting values and the last plot is for the optimistic/pessimistic-greedy algorithm (OP-greedy) which decides between these variants based on an external signal. The plots shows that all variant perform well and match the optimal distribution closely, but there are still some differences.

The most obvious difference is that optimistic greedy allocates a considerable number of impressions to new items. This can be seen in the plots by the items at the top of the bar charts extending down much farther for optimistic greedy than for the other algorithms. These impressions are "wasted" in the sense that the other algorithms are able to gather enough information to determine the optimal items, without needing so much exploration.

The difference between pessimistic- and OP-greedy is more subtle. The amount of exploration of new items is about the same for both algorithms as most items are seen pessimistically. However, items which the external signal indicates are popular will be explored more by OP-greedy. In some cases this is unwarranted and results in impressions being wasted on new items (the gray item in the top right). On the other hand, sometimes the extra exploration pays off and results in optimal items being discovered slightly sooner (the optimal dark blue item in the middle receives more attention sooner).

4.3.3. Discounting

Figure 5 shows impression plots for UCB and TS using discounting on Simulation B. We can see that the effect of discounting is similar to that of a global sliding window as it improves the change detection speed, but also incurs extra exploration costs. These graphs further make it clear that the extra exploration happens for old items as the bottom parts shows old items being recommended often.

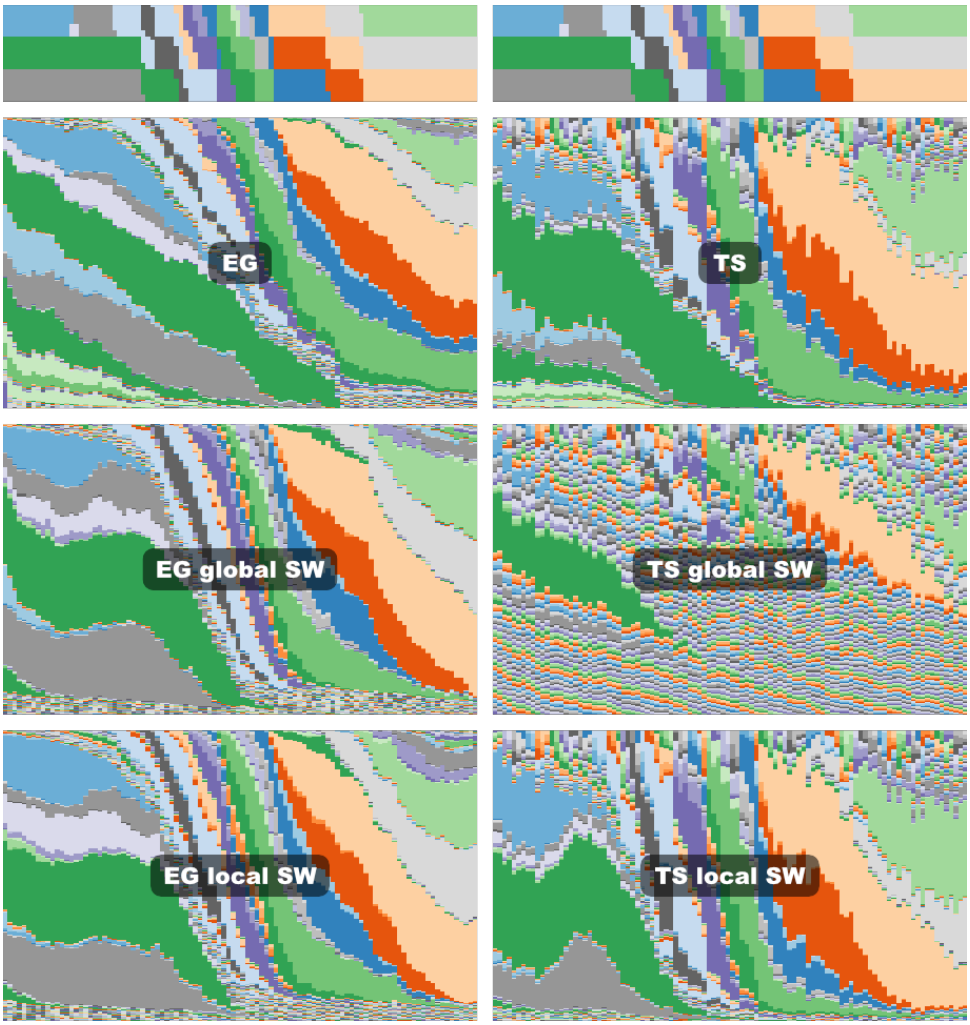

Figure 3: Impression plots for ϵ-greedy (left) and TS (right). Standard version (center top), global sliding windows (center bottom) and local sliding windows (bottom) are shown, as well as the optimal distribution (unlabeled, top).

4.4. Conclusion

In terms of the exploration/exploitation trade-off for news recommendation, it seems that the best strategies consist of being pessimistic and doing as little exploration as possible. The first reason for this is that the CTR of articles generally decreases. This means that exploring old articles which previously started to perform poorly is not needed. The second reason is that there are just too many new items to explore. You are better off searching for good performing items instead of chasing the optimal items, as they will quickly lose relevancy anyway.

In terms of handling the non-stationary behavior of news, we conclude that the focus should be on quickly detecting when the best performing articles stop performing well. Weighted average updates and local sliding windows are excellent for this. Importantly, sliding windows applied globally or discounting can be counterproductive in combination with guided exploration strategies as they lead to excessive continued exploration, which is undesirable.

When there is external information available indicating article popularity, we can make use of this by deciding which articles warrant exploration. We introduced optimistic/pessimistic-greedy, which uses this data effectively and outperformed all other algorithms in our simulated experiments.

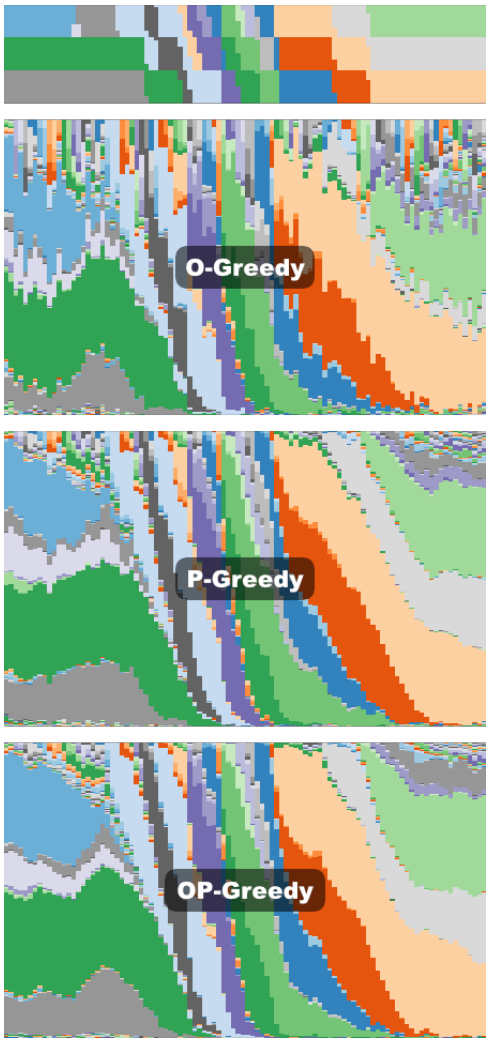

Figure 4: Impression plots for optimistic greedy (center top), pessimistic greedy (center bottom) and optimistic/pessimistic-greedy (bottom). Optimal distribution is also shown (unlabeled, top).

5. Discussion and Future Work

Our findings highlight the importance of considering decreasing relevancy and article lifespan in the news recommendation domain. The superior performance of local sliding windows and weighted averages compared to global sliding windows or discounting, underscores the potential benefits of investigating domain-tailored adaptations into recommendation strategies. However, while these results are encouraging, they also open up several avenues for further investigation.

The evaluation is based on a simulated environment that, despite being designed to mimic real-world scenarios, cannot capture the full complexity of actual user interactions. Another limitation of our experiments is the idealized evaluation procedure. In real recommendation systems, feedback is not obtained immediately and updates are performed in batches. For these reasons, one of the primary discussions arising from this study concerns the applicability of these results in live settings. Future investigations and especially deployments of these algorithms on real-world platforms would provide valuable insights into their practical effectiveness and user satisfaction.

Another point for discussion is the computational efficiency of the adapted algorithms. While local sliding windows demonstrate comparable and in some cases superior performance to weighted averages, the optimal window size for most algorithms was quite large, which can pose challenging infrastructure problems to accommodate. Weighted averages on the other hand, are in principle easier to implement due to their iterative nature, but can still present difficulties in large distributed systems.

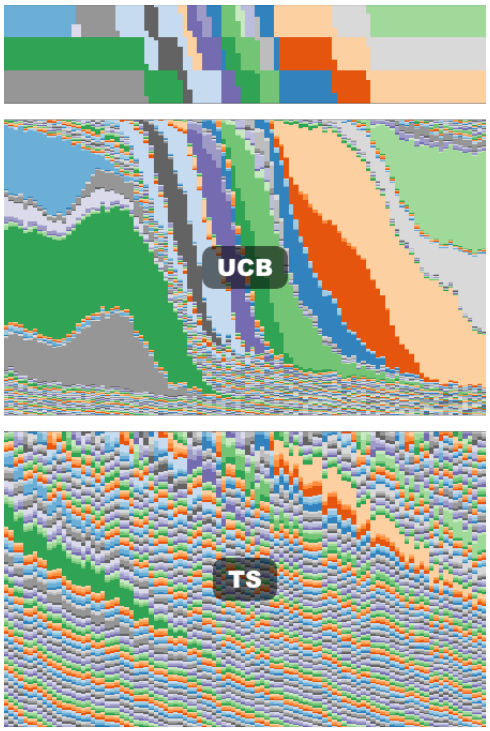

Figure 5: Impression plots for UCB (center) and TS (bottom) with discounting. Optimal distribution is also shown (unlabeled, top)

5.1. Future Work

Building on the foundation laid by this study, several directions for future work can be identified:

- **Real-world Implementation:** Deploying the adapted algorithms in a live news recommendation system would allow for the collection of empirical data, providing a clearer picture of their performance and user reception.
- **Personalization:** Integrating user-specific context, such as historical interactions and preferences, could enhance the personalization of recommendations, potentially leading to even higher engagement rates.
- **Longitudinal Studies:** Conducting long-term studies on the impact of adapted MAB algorithms on user engagement and satisfaction would provide deeper insights into their effectiveness over time.

In conclusion, this study contributes valuable insights into the application of multi-armed bandit algorithms in the context of online news recommendation. The promising results pave the way for future research and development in this area, with the potential to significantly enhance the user experience on news platforms.

References

[1] C. Li, Q. Wu, H. Wang, When and whom to collaborate with in a changing environment: A collaborative dynamic bandit solution, in: Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, ACM, 2021, pp. 10–12.

[2] Q. Wu, N. Iyer, H. Wang, Learning contextual bandits in a non-stationary environment, in: The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR '18, Association for Computing Machinery, New York, NY, USA, 2018, p. 495–504. URL: https://doi.org/10.1145/3209978.3210051. doi:10.1145/3209978.3210051.

[3] S. Ghoorchian, E. Kortukov, S. Maghsudi, Non-stationary linear bandits with dimensionality reduction for large-scale recommender systems, IEEE Open Journal of Signal Processing 5 (2024) 548–558. doi:10.1109/OJSP.2024.3386490.
[4] J. Hong, B. Kveton, M. Zaheer, Y. Chow, A. Ahmed, M. Ghavamzadeh, C. Boutilier, Non-stationary latent bandits, 2020. URL: https://arxiv.org/abs/2012.00386. arXiv:2012.00386.
[5] O. Besbes, Y. Gur, A. Zeevi, Stochastic multi-armed-bandit problem with non-stationary rewards, in: Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 1, NIPS'14, MIT Press, 2014, p. 199–207.
[6] R. Allesiardo, R. Féraud, O.-A. Maillard, The non-stationary stochastic multi-armed bandit problem, International Journal of Data Science and Analytics 3 (2017) 267–283.
[7] A. Garivier, E. Moulines, On upper-confidence bound policies for non-stationary bandit problems, arXiv preprint arXiv:0805.3415 (2008).
[8] F. Trovo, S. Paladino, M. Restelli, N. Gatti, Sliding-window thompson sampling for non-stationary settings, Journal of Artificial Intelligence Research 68 (2020).
[9] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, MIT press, 2018.
[10] D. Agarwal, B.-C. Chen, P. Elango, N. Motgi, S.-T. Park, R. Ramakrishnan, S. Roy, J. Zachariah, Online models for content optimization, in: Proceedings of the 21st International Conference on Neural Information Processing Systems, NIPS'08, Curran Associates Inc., 2008, p. 17–24.
[11] P. Auer, N. Cesa-Bianchi, P. Fischer, Finite-time analysis of the multiarmed bandit problem, Mach. Learn. 47 (2002) 235–256.
[12] J. C. Gittins, Bandit processes and dynamic allocation indices, Journal of the Royal Statistical Society Series B: Statistical Methodology 41 (1979) 148–164. URL: http://dx.doi.org/10.1111/j.2517-6161.1979.tb01068.x. doi:10.1111/j.2517-6161.1979.tb01068.x.
[13] N. Abe, P. M. Long, Associative reinforcement learning using linear probabilistic concepts, in: Proceedings of the Sixteenth International Conference on Machine Learning, ICML '99, Morgan Kaufmann Publishers Inc., 1999, p. 3–11.
[14] P. Auer, Using upper confidence bounds for online learning, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, 2000, pp. 270–279. doi:10.1109/SFCS.2000.892116.
[15] W. Chu, L. Li, L. Reyzin, R. Schapire, Contextual bandits with linear payoff functions, in: G. Gordon, D. Dunson, M. Dudík (Eds.), Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics, volume 15 of *Proceedings of Machine Learning Research*, PMLR, 2011, pp. 208–214.
[16] V. Raj, S. Kalyani, Taming non-stationary bandits: A Bayesian approach, arXiv preprint arXiv:1707.09727 (2017).
[17] L. Li, W. Chu, J. Langford, R. E. Schapire, A contextual-bandit approach to personalized news article recommendation, in: Proceedings of the 19th International Conference on World Wide Web, WWW '10, ACM, 2010, p. 661–670. doi:10.1145/1772690.1772758.
[18] W. R. Thompson, On the likelihood that one unknown probability exceeds another in view of the evidence of two samples, Biometrika 25 (1933) 285. URL: http://dx.doi.org/10.2307/2332286. doi:10.2307/2332286.
[19] S. L. Scott, A modern Bayesian look at the multi-armed bandit, Appl. Stoch. Model. Bus. Ind. 26 (2010) 639–658. doi:10.1002/asmb.874.
[20] O. Chapelle, L. Li, An empirical evaluation of Thompson sampling, in: Proceedings of the 24th International Conference on Neural Information Processing Systems, NIPS'11, Curran Associates Inc., 2011, p. 2249–2257.