Paper Replication and Improvement: A Sleeping, Recovering Bandit Algorithm for Optimizing Recurring Notifications

Angela Yang
Andrew ID : anqiyang
Carnegie Mellon University
anqiyang@andrew.cmu.edu

1 Executive summary

The project is divided into two parts: paper reproduction and improvement. The reproduction of "A Sleeping, Recovering Bandit Algorithm for Optimizing Recurring Notifications" validated the original study's claims by implementing the Recovering Difference Softmax Algorithm. Our final test result shows 0.09% increase in average reward compared to the baseline random selection for selecting a notification template. The algorithm tries to increase user engagements for online and mobile applications with daily email or push notifications. This algorithm optimizes recurring notification templates by addressing novelty effects and conditional eligibility constraints. The claims of the paper are:

- Users with different UI languages displays show different interest on notification templates;
- Users tend to click on notifications they were not commonly seen before.

1.1 Dataset Description

The dataset used in this study consists of historical log data from Duolingo, collected over a 34-day period, and was divided into 15 days of training data (80500 rows) and 19 days of test data (80500 rows). This dataset enabled the evaluation of the algorithm through offline experiments, where weighted importance sampling was applied to assess the performance of various policies [YS20b].

Table 1: Summary of the offline evaluation datasets

Dataset	Duration	Row Count
Train	15 days	80500
Test	19 days	80500

Table 2: Dataset Description

Name	Type	Description
Timestamp	DateTime	The time that the bandit decision was made.
User ID	UserID	Identifies the user that the decision applies to. This along with timestamp are used to compute reward when reconciling the data with the metrics database.
Eligible Templates.	Map[ArmID]	A list of the eligible tempaltes. This is used to determine which arms were eligible and to compute the importance weights for sleeping arm algorithm
Selected templates	ArmID	The arm that was selected.

The final evaluation, conducted using historical data, compared the baseline uniform random selection to the algorithm's variations. The template-based model improved the average reward by 1.39 percent, while the Template + UI Language model further increased it to 1.61 percent. These results align with the original paper, demonstrating that the proposed algorithm effectively increases user engagement.

As for the improvement part, we implement the UCB algorithm with regards to the eligible templates for each user. The result shows two times better than the baseline, random distribution, algorithm. We also update a reward function to consider historical using days during evaluation.

2 Analytical Model and Data

2.1 Problem Statement

The analysis focuses on identifying which notification templates lead to the highest user engagement, specifically encouraging active daily use of the Duolingo app. Duolingo maintains user activity by sending personalized notifications tailored to user-specific characteristics such as language preferences, learning history, and travel plans.

2.2 Baseline Model

The baseline model is a random selection policy, where each notification template (arm) is selected uniformly at random from the set of eligible templates available at each round t. Let $\mathcal{A}_t \subseteq \mathcal{A}$ denote the set of eligible arms at round t. The baseline policy assigns an equal probability to each arm in $\mathcal{A}_t$, such that the selection probability for any arm $a \in \mathcal{A}_t$ is given by:

$$b_t(a) = \frac{1}{|\mathcal{A}_t|}$$

where $|\mathcal{A}_t|$ represents the set of eligible arms at round t. This random selection ensures equal exploration of all eligible templates without bias toward specific arms. However, as it does not learn from historical rewards, the model lacks adaptability to user-specific preferences or the varying performance of templates.

2.3 Recovering and Sleeping Algorithm Summary

The paper proposes the sleeping arm score calculation to compare arms only during rounds where they are eligible, which avoiding biases from arms designed for specific user groups. The recovering arm mechanism addresses novelty effects by applying a recency penalty to frequently used arms, which decays over time to restore their relevance. The key components of the algorithm are as follows:

2.3.1 Arm Score Calculation

Sleeping Arm For each arm a, the algorithm computes a score s_a using historical data and then calculating the difference between the arm is chosen or not. Two metrics are estimated:

Using *weighted importance sampling*, these metrics are computed as[YS20a]:

$$\mu_a^+ = \frac{\sum_{t \in H_a, a_t = a} w_t^+ r_t}{\sum_{t \in H_a, a_t = a} w_t^+}, \quad w_t^+ = \frac{1}{b_t(a)}$$

$$\mu_a^- = \frac{\sum_{t \in H_a, a_t \neq a} w_t^- r_t}{\sum_{t \in H_a, a_t \neq a} w_t^-}, \quad w_t^- = \frac{1}{1 - b_t(a)}$$

- μ_a^+: The expected reward when arm a is chosen.
- μ_a^-: The expected reward when arm a is eligible but not chosen.
- H_a: represents the number of historical rounds for arm a
- $b_t(a)$ is the probability of selecting arm a in eligible arm lists.
- r_t is the observed reward (1 if the user engages within 2 hours, 0 otherwise)

The arm score is then calculated as the *relative difference*[YS20a]:

$$s_a = \frac{\mu_a^+ - \mu_a^-}{\mu_a^-}$$

Recovering Arm: Recency Penalty To model novelty effects, a recency penalty is applied to reduce the score of arms used recently. This penalty decays over time using an exponential decay function:

$$s_a^*(t) = s_a - \gamma \cdot 0.5^{d_{a,t}/h}$$

where $d_{a,t}$ is the number of days since arm a was selected at round t, γ and h are hyperparameters.

Arm Selection Using Softmax The algorithm selects arms using a *softmax policy*, which balances exploration and exploitation:[YS20a]

$$\pi(a|t) = \frac{\exp(s_a^*(t)/\tau)}{\sum_{a' \in A_t} \exp(s_{a'}^*(t)/\tau)}$$

where $\pi(a|t)$ is the probability of selecting arm a at round t, and τ is a parameter for controling exploitation and exploration.

2.4 Improved Algorithm: Constrained Upper Confidence Bound(UCB) Algorithm

For each arm a, the UCB score is calculated as:

$$\text{UCB}_a = \overline{r}_a + c\sqrt{\frac{\log T}{n_a}}$$

where:

- $\overline{r}_a = \frac{R_a}{n_a}$: The average reward for arm a, calculated as the total reward R_a divided by the number of times the arm has been played n_a.
- T: The total number of plays across all arms.
- n_a: The number of times arm a has been played.
- c: The exploration-exploitation tradeoff parameter. Larger values of c encourage more exploration.

The first term, $\overline{r}_a$, promotes exploitation by favoring arms with higher observed rewards, while the second term, $c\sqrt{\frac{\log T}{n_a}}$, promotes exploration by assigning higher confidence to arms with fewer plays. Arms that have not been played yet are given a UCB score of ∞ to ensure they are explored.

2.4.1 Algorithm Workflow

The steps of the UCB algorithm are as follows:

1. Initialize $R_a = 0$ and $n_a = 0$ for each arm a.
2. At each round t:
   - Calculate the UCB scores for all arms using the formula above.
   - Select the arm a with the highest UCB score.
   - Observe the reward r_t for the selected arm and update:

$$R_a \leftarrow R_a + r_t, \quad n_a \leftarrow n_a + 1$$

3. Repeat until termination criteria are met.

3 Evaluation Metric

This a weighted average reward metric, commonly using in offline test. It evaluate the new performance by calculating the probabilities versus the old policy. The formula is defined as follows:

$$\overline{r}_\pi = \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{\pi(a \mid t)}{b_t(a \mid t)} r_t$$

where:

- $\pi(a \mid t)$: The new algorithm of choosing arm a at time t.
- $b_t(a \mid t)$: The distribution of randomly select an arm a at time t (uniform distribution).
- r_t: The observed reward at time t.
- $\mathcal{T}$: The set of rounds in the test set.
- $|\mathcal{T}|$: The total number of rounds in the test set.

This metric computes a weighted reward, where the contribution of each reward r_t is scaled by the ratio of the probabilities from the new policy (π) and the old policy (b_t). This ensures that the evaluation accounts for changes in policy while normalizing for differences in probability distributions.

4 Analysis

4.1 UI languages Incorporation

4.1.1 UI languages Arms Analysis and Visualization

As shown in Table 3, after calculating the arm score using sleeping and recovering arm bandit algorithm, we discover that each template shows different reward for each UI language translate version. The negative arm scores indicate the arm perform poorly than the others. For instance, template A has better reward for those who using Hindi; template J in English version performs better than Spanish and Hindi. Duolingo could add some Hindi features to template A based on the running results.

From figure 1, the heatmap,it display the top five languages for each templates, and the correlation between languages and templates, template A in Hindi, and template J with Thai can increase active usage most. Figure 2 (Bar Chart) shows the top 3 languages for each template by arm score. We can conclude from the chart that template A works best for Romanian; template K works best for Thai and Japanese; template B works best for Polish, etc.

Table 3: Differences in arm scores by UI language

Template	en	es	hi
A	-0.64%	-0.71%	0%
J	-0.79%	-0.84%	-1%
B	-0.80%	-0.86%	-1%

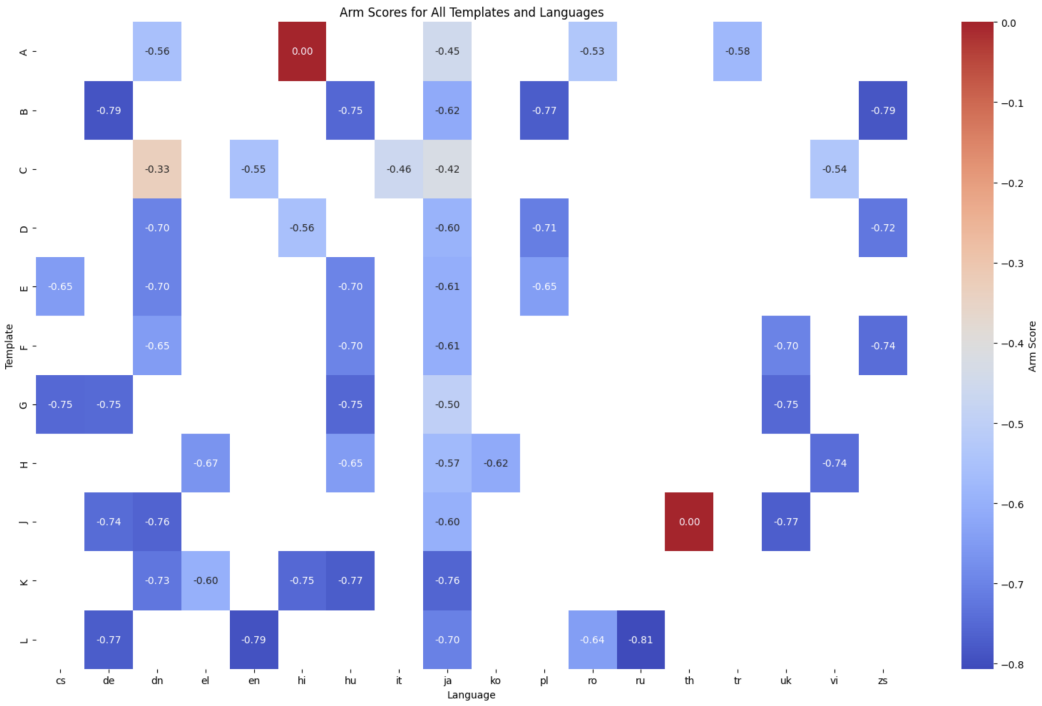

Figure 1: Differences in arm scores by UI language

4.1.2 Model Evaluation

After we apply the new algorithm, sleeping arm and recovering bandit algorithm can lift the award by 0.09%. The experiments with the UI language translation increase the award by 0.0001. We confirm that the result from the paper is meaningful, and Duolingo could consider tailoring the notification to different languages, cultural backgrounds.

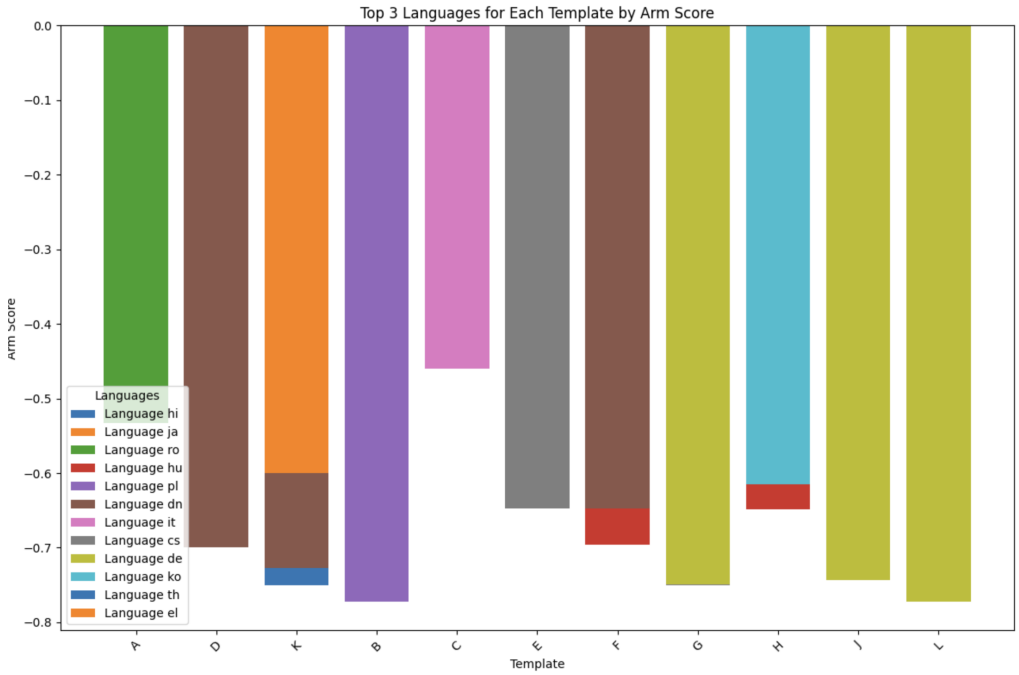

Figure 2: Top 3 Languages for Each Template by Arm Score

For further research, we could also consider other demographic factors such as region, gender, job titles, etc.

Table 4: Results from offline experiment 1 (UI language).

Bandit Algorithm	**Avg. Reward** ($\bar{r}$)	**Rel. Diff.**
Baseline (random)	0.1823	-
Template	0.1825	+0.09%
Template + UI Language	0.1824	+0.00%

4.2 Recency Penalty Experiment

The second experiment is testing how sensitive users are to previous templates. The paper run a comparison test between using sleeping and recovering arm bandit algorithm and reusing the template from the last one. However, the test result did not perform the same the decrease as the paper stated in experiment section. In the paper, their relatively difference was decrease by 0.5% compared to random selection. One of the reasons would be the dataset is 10 times smaller than paper's original datasets. Their dataset include 88 million training data and 114 million testing data, while the datasets we currently use are 10 times smaller than both training and testing datasets from original paper.

Table 5: Results from offline experiment 2 (recency).

Bandit Algorithm	**Avg. Reward** ($\bar{r}$)	**Rel. Diff.**
Baseline (random)	0.1823	-
Reuse Last Template	0.205	+11%
Template + UI Language	0.1824	+0.09%

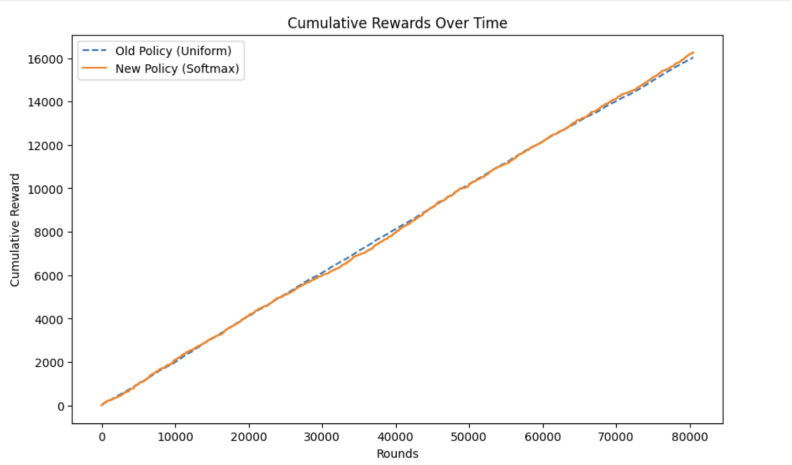

Figure 3: Comparison between random selection and paper algorithm

4.3 Visualization of Sleeping and Recovering Algorithm

As shown in figure 3, We make a line chart for visualizing every round of the random selection and recovering bandit algorithm. The x-axis indicate the total rounds, 80000, and the y-axis are cumulative reward, representing the number of users finish daily task after the notification sent in 2 hours. The new algorithm did not increase the total rewards very much from the randomly selection. Two lines are very close to each other. The result indicates that assigning templates to users based on the sleeping and recovering algorithm might not be a good method.

5 Improvement

5.1 Improved Arm Selection Algorithm

We already proved that the sleeping, recovering arm bandit algorithm did not improve the final rewards very much with only 0.8% increase from the baseline. Though the purpose of the paper is to test the recency penalty and UI language implementation, we could still optimize the cumulative reward to further improve user engagement. As a result, we purpose the UCB algorithm but considering the eligible templates.

Figure 4 shows the cumulative rewards of UCB but considering eligible templates for each round. The final reward is 0.4385. Since the final reward of UCB is surprisingly higher than recovering and sleeping bandit algorithm, we dive deeper to see what makes that happen. One reason is the evaluation metrics. The evaluation metric run offline on test dataset.

Table 6: Results from UCB algorithm.

Bandit Algorithm	**Avg. Reward ($\bar{r}$)**	**Rel. Diff.**
Baseline (random)	0.1823	-
UCB	0.4385	+ 2.38

5.2 Updated Reward Function

Since the reward function is only defined as sum of scores which took the language tasks in 2 hours, we plan to add the time decay into reward function. The purpose for adding the time decay is that we could have more accurate score when multi arm The reward function is :

$$r_t = r_t^{\text{base}} \times \prod_{\text{recent arms}} \text{decay}(n_{\text{days}})$$

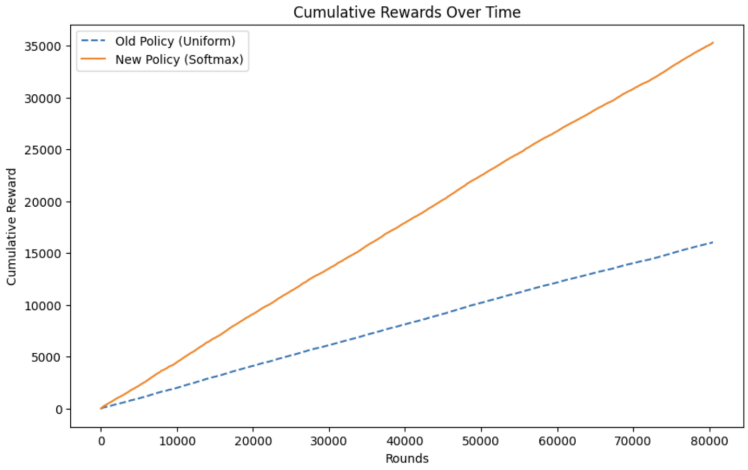

Figure 4: UCB reward VS baseline reward

- r_t^{base}: The base reward (1 for finishing task within two hours after notification is sent, 0 otherwise).
- $\text{decay}(n_{\text{days}}) = e^{-\alpha n_{\text{days}}}$: A decay factor that decreases exponentially with recency (n_{days}).
- α: A parameter controlling the decay rate.

6 Conclusion

In summary, we replicate the sleeping and recovering arm algorithm from the paper, run the offline experiments, and conclude that language version would help notification engage users more in Duolingo app. But for the recency penalty effect, we might need larger dataset to run the test for it. The replication code is inspired by the work described in [Maz23].

References

[Maz23] Jake Mazurkiewicz. *How I Re-Created Duolingo's Famous Notification Algorithm*. Accessed: 2024-12-09. 2023. URL: https://medium.com/@jakemazurkiewicz6/how-i-re-created-duolingos-famous-notification-algorithm-00fce580b84e.

[YS20a] K. Yancey and Burr Settles. "A Sleeping, Recovering Bandit Algorithm for Optimizing Recurring Notifications". In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020). URL: https://api.semanticscholar.org/CorpusID:219860828.

[YS20b] Kevin Yancey and Burr Settles. *Replication Data for: A Sleeping, Recovering Bandit Algorithm for Optimizing Recurring Notifications*. Version V1. 2020. DOI: 10.7910/DVN/23ZWVI. URL: https://doi.org/10.7910/DVN/23ZWVI.