Contextual Bandits for Hyper-Personalization based on User Behavior in Local Domain

Junhyung Kim[1], Kijun Kwon[1] and Jinho Kim[1]

[1] *AirSPACE Team, NAVER Corporation, Bundang-gu, Seongnam-si, Republic of Korea*

Abstract

Personalized platforms are providing services that strive to recommend personalized contents to their users by understanding individual user behavior in various context. In local domain, various contents (e.g. menus, themes, advertisements, etc.) are also recommended in this way. However, it is challenging to recommend the most relevant content by using one recommender system, as user contexts (e.g. area, time, etc.), user features (e.g. age, gender, preference, etc.), and contents are very various and complex. In this paper, we propose an empirical hyper-personalization problem reflecting user behavior in local domain, that is considered as contextual bandit problem with well-configured recommender system ensemble. We empirically introduce how to deal with insufficient user feedbacks in service by using feedbacks of other interfaces (e.g. search and feed platform app/web, map platform app/web), how to define user contexts and user features in local domain, and how to ensemble contextual bandits for optimization, called *LocalECB* (Local Ensemble Contextual Bandit). Furthermore, we show how well the model performs and how it works for hyper-personalization in service.

Keywords

Location based Services, Recommender Systems, Hyper-Personalization, Content Ranking, Online Learning Settings, Learning from Implicit Feedback, Sequential Decision Making, Multi-Armed Bandits, Contextual Bandits, User Behavior, Context-Aware, Ensemble Recommendation, Service Platform

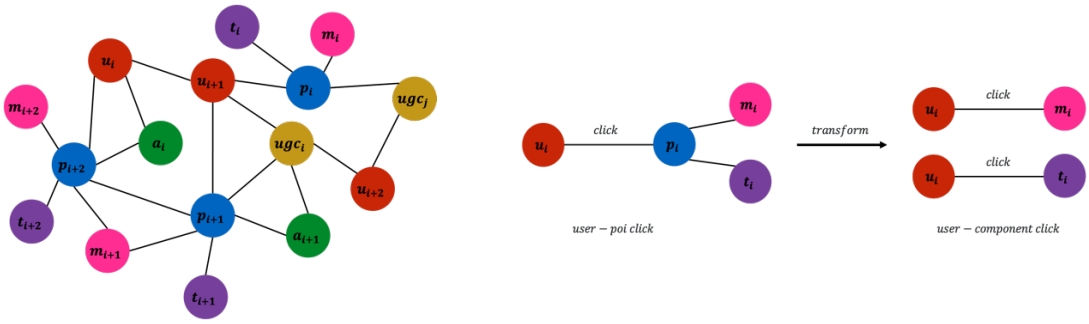
1. Introduction

Currently, various personalized platforms, such as news [1] and music streaming [2, 3] platforms, aim to recommend relevant and personalized contents to their users. These services are provided based on understanding user behavior in various contexts. It is crucial to provide satisfying contents in specific context. Because it can improve their user's engagements and experiences on platform and can generate revenue for service providers or clients. Since the local domain consists of various components, such as POI (Point-of-Interest), menus, themes, users, and UGC (User-Generated Contents) as illustrated in Figure 1(a), recommender systems in local domain also provide contents of these components in the same way. Furthermore, as most local components are connected around POI, these effective recommender systems can have a great economically impact on the local community [4] (e.g. increasing visit rate of POI, etc.) by helping users discover new contents preferred to them and providing users with their favorite contents in various context [5]. NAVER also has such services like A Appendix.

However, recommending the most relevant content to user in local domain is challenging for two reasons. First, depending on the characteristics of component to be recommended, various and complex user contexts (e.g. area, time, etc.), user features (e.g. age, gender, preference, etc.) and numerous contents of components should be considered. For example, it is about which data is needed to solve the problem of which menu a user might like in a particular area at that time. Second, since there are various models to properly utilize various data for each objective, we need a model that dominates in all performance indicators. Because it has long been found that a well-configured ensemble model can achieve better effectiveness than models separately [6].

Proceedings of BehavRec '23, the First International Workshop on Behavior Change and Persuasive Recommender Systems at the 17th ACM Conference on Recommender Systems (RecSys '23), Singapore, September 18-22, 2023

junhyung.kim@navercorp.com (Junhyung Kim); standard.kwon@navercorp.com (Kijun Kwon); jinho.kim@navercorp.com (Jinho Kim)



(a) Local Knowledge Graph (b) Transformation based on Local Connectivity

Figure 1: (a) Illustration of local knowledge graph. In this graph, we denote u, p, a, m, t, and ugc as user, POI, area, menu, theme, UGC, respectively. For example, user u_i frequently visits area a_i, follows user u_{i+1}, and visits on POI p_{i+2}. POI p_{i+1} has same category as POI p_{i+2}, is bookmarked by user u_{i+1}, has UGC ugc_i generated by u_{i+2}, belongs to area a_{i+1} and has menu m_{i+1}, thema t_{i+1}. In this way, local components are connected in this graph. (b) Transformation of user-POI click feedback into user-component (e.g. menu, theme) click feedback.

To address these two challenges, in this paper, we propose a model for hyper-personalization as contextual bandit problem with recommender system ensemble. We empirically show how to deal with insufficient user feedback in service by using various interfaces (e.g. search, feed, and map platform app/web, etc.), how to define user contexts and user features in local domain, how to ensemble contextual bandits for optimization, called *LocalECB* (Local Ensemble Contextual Bandit), and how well it performs and how it works for hyper-personalization in NAVER service cases.

2. Related Works

2.1. Contextual Bandits

In recent years, studies of multi-armed bandit have researched and developed. So, it has been applied to various service platforms in many domains. Multi-armed bandit has basically problem where it is to make the optimal recommendations by managing trade-off between exploitation and exploration based on user behavior. There are classic bandit algorithms like ϵ- *greedy* [7], *UCB* [8], and *Thompson Sampling* [9, 28], etc. Based on extending the basic bandit problem, there are many approaches for the optimal recommendations; contextual bandit approach [1, 10, 11], non-stationary bandit approach [12, 13, 14], and multi-play bandit approach [2, 15, 16, 17], etc. In addition, by combining different approaches, algorithms [18, 19] such as non-stationary contextual bandit with multi-play also exist. Because there are various service environments.

2.2. Ensemble Recommendation

Ensembles have been found that it is to make better performance than separate recommendation algorithms. In many cases, this method combines several algorithms [20, 21] to make a hybrid system by applying weighted sum, which is done only once, statically. However, despite of advantage of the combination method, there are approaches [22, 23, 24, 25] to dynamically control ensembles. These approaches can be used appropriately for environments of each service.

3. Background

In this section, we introduce how to make dataset that is added to insufficient user's service feedback and show various contexts in local domain.

3.1. Transformation of Multi-Interface Feedback from Local Connectivity

Multi-armed bandit is a ranker, which is to re-rank arms (i.e. items, components). This ranker has objective that users would show actions based on expected rewards. In other words, multi-armed bandit is related to service optimization, and requires user's service feedbacks for actions. However, if the dataset is not enough large, bias or uncertainty will increase. Therefore, it is necessary to transform feedback in other interface to service feedback for enough large dataset.

Since there are many channels related to POI recommendations in NAVER (e.g. search and feed platform app/web, map platform app/web), we use feedback related to POI recommendations and transform it into a form of service feedback. As illustrated in Figure 1(b), this transformation is done based on connectivity between components within local domain. For example, we transform user-POI feedback (e.g. click, display data) into user-menu feedback or user-theme feedback.

However, as it uses feedback from other interfaces, if we use it by itself, problems with reliability and bias should occur. We address these problems as follows. First, we give reliability for the connection between POI and component through UGC (e.g. review, etc.) related to POI. This means that the UGC left by users must contain identifying data about the corresponding component. Second, since we set CTR (Click-Through Rate) as reward, we adjust the display rank of other interface's feedbacks, and then change them into service feedback to control effect of reward, for bias problem. Third, the effect of click is adjusted by using *Bernoulli Distribution* [34]. Because it is click feedback of other interfaces.

3.2. Contextual Information in Local Domain

Contextual bandit is a model that utilizes contexts, that are observable information about user. These contexts consist of user context, which is user's environment (e.g. area, time, etc.), and user feature, which is observable user information (e.g. age, gender, preference, etc.). This subsection introduces what kinds of these contexts exist in local domain.

3.2.1. Spatio-Temporal Data

In local domain, there are user contexts, that are environment's conditions such as area (region) and time that user can be in. For example, a user visiting Texas would search menus such like barbecue which is a signature menu of Texas, and a user would search salad for lunch or beer for dinner. Like this, users show different behaviors depending on the context, which is spatio-temporal data.

3.2.2. User Profile Data

As like other domains, there are also user features such as age and gender that users have, in local domain. For example, users of 20s would usually search for popular local themes, and users of 40s would search for local themes to enjoy with their families. And men search for local themes related to pub, beer, and women search for local themes related to brunch cafe. In other words, user behavior changes depending on contexts like user profile data.

3.2.3. Understanding User Behavior Data

Through user behaviors in services related to local domain, user features can be extracted. These features aim to summarize user behavior (e.g. preference, user familiar area, etc.). And depending on type of user behavior data (e.g. implicit feedback, explicit feedback), these features can be classified into two types as follows.

First, by using implicit feedback, like click, we can extract features related to user preference (imf-preference). From the philosophy of *Collaborative Filtering* [26] between user and

components to be recommended, we use the user feature as expected display-to-click probabilities for components per user [2], based on logistic matrix factorization [37].

Second, through explicit feedback, such as visit (check-in), save (bookmark) and like, we extract features about the user preference (exf-preference) and user familiar area. In case of user exf-preference feature, we obtain the user feature from RFM (recency, frequency and monetary value) model [27] and representation learning (encoder) model. Encoder projects output labels of RFM model into a specific $Euclidean\ Space$ by an element-wise average of labels' embedding. And in case of user familiar area feature, we extract this feature from user's base data (e.g. home, school, and company), visit statistics, and user regional intent classification model [38, 39, 40, 41] which classifies daily life category. For example, when searching for a certain area, this feature can be used as a user context, depending on whether or not the user is familiar with this area.

4. Methodology

In this section, after introducing usage of contextual information and policies of $top\text{-}k$ recommendations on contextual bandits, we show our ensemble model, called $LocalECB$ (Local Ensemble Contextual Bandit) for hyper-personalization.

4.1. Leveraging Contextual Information

Since in 3.2 section we show that user behavior is different depending on contexts, we introduce how bandits use these contexts.

4.1.1. Semi-Personalization

Semi-personalization is recommendation in segment (group) units according to combinations of contexts. It is based on assumption that each user belongs to various segments according to their behavior. In other words, a user is simply partitioned to segments by the contexts of feedback. For example, for a same user, user searching for dinner in Singapore and user searching for lunch in Singapore belong to different segment. And we propose that users in same segment have identical expected probabilities for each component to be recommended as follows:

$$\forall s \in \{S_1, \dots, S_P\}\ (P\ segments \ll N\ users), \forall u \in \{s_1, \dots s_p\}, \forall i \in \{1, \dots, K\ arm\}$$
$$p_{ui} = p_{s_a i}\ (\ if\ u\ belongs\ to\ s_a\ by\ contexts\ ) \tag{1}$$

Then, we run contextual bandit algorithms by one of segments to recommend $top\text{-}k$ components.

4.1.2. Full-Personalization

Unlike semi-personalization, full-personalization is recommendation in user (persona) units according to combinations of contexts. Based on assumption that each user can have various personas according to their behavior, each user is partitioned to personas from the contexts of feedback. For example, even for a same user who is male and 20s, user searching for local themes in Seoul and user searching for local themes in Busan belong to different persona. This means that we directly use context feature. This context feature represents summarizing user and consists of user feature (e.g. age, gender, preference, etc.) and user context (e.g. area, time, etc.). We define that different personas of a same user have not identical expected probabilities for each component to be recommended as follows:

$$x_u \in \mathbb{R}^D, \forall i \in \{1, \dots, K\ arm\}, p_{u_c i} = f(x_{u_c}^T \theta_i)\ (\ if\ u\ is\ in\ user\ context\ c\ ) \tag{2}$$

where, $f(\cdot)$ is the function for probability, x_{u_c} means context feature that user feature is concatenated by user context c, and $\theta_1, \ldots, \theta_K \in \mathbb{R}^D$ means weight feature to learn for K arms. Specially, using only user feature (personalization) or using both user feature and user context (hyper-personalization) depends on policy of contextual bandit. Then, we run contextual bandit algorithms by context feature, and it recommends $top\text{-}k$ items.

4.2. Contextual Bandits for Top-K Recommendation

In this section, the methods of leveraging contextual information are separated into three personalization stages by followed policies; $semi\text{-}ts$ for semi-personalization, $full\text{-}cf$ for personalization, and $full\text{-}linucb$ for hyper-personalization.

4.2.1. Semi-Personalization

In the case of semi-personalization, contextual bandit recommends $top\text{-}k$ components based on $Thompson\ Sampling$ strategy ($semi\text{-}ts$) [9, 14, 15, 28, 29] as sequential decision-making algorithm, with using contextual information by the method described in section 4.1.1. This algorithm is a policy that selects the optimal action through sampling of $Beta\ Distribution$ [28], assuming that alpha and beta, which are estimates of reward for each action, follow the distribution.

4.2.2. Personalization

In personalization stage, contextual bandit uses Collaborative Filtering strategy ($full\text{-}cf$) [26, 30] as policy based on method of leveraging contextual information as described in section 4.1.2 with using only user feature, not using user context. This policy combines the characteristics of stochastic bandit algorithm [31, 32], which makes recommendations tailored to the continuously changing user preferences, and the characteristics of Collaborative Filtering algorithm [26]. In other words, it is a decision-making algorithm that can contain dynamic changes of user's preference with estimating similar preferences, using concept that similar users prefer similar items.

4.2.3. Hyper-Personalization

In the case of hyper-personalization, contextual bandit recommends $top\text{-}k$ components based on LinUCB strategy ($full\text{-}linucb$) [1, 32] as policy, as using contextual information based on the method described in section 4.1.2 with using both user feature and user context. This decision-making algorithm is policy that selects the optimal action by estimating the coefficient vector through ridge regression method from expected reward, which is expressed as linear function and means average reward value of each arm observed so far. In other words, regardless of the group or other users with similar preferences, this policy could reflect dynamic user behavior change according to various contexts.

4.3. Ensemble Contextual Bandits for Optimization

Each contextual bandit according to sequential decision-making algorithms/policies has the following benefits and costs:

1. $semi\text{-}ts$ has randomness to sample from $Beta\ Distribution$ [28], so it has benefits in terms of diversity and serendipity. And it provides a solution to $Cold\ Start$ by recommending components per segment (group) unit. However, there is cost that the degree of personalization is weak compared to other algorithms/policies.

2. $full\text{-}cf$ has benefit of recommending components per personalization unit that reflects philosophy of Collaborative Filtering [26] and the characteristic of the bandit algorithm [31, 32]. But there are costs. Although there is similar clustering concept that recommends similar items to users with similar preferences, it has relatively low diversity and serendipity compared to $semi\text{-}ts$. And since it is not a hyper-personalization unit, it has a relatively low precision and recall compared to $full\text{-}linucb$.
3. $full$ - $linucb$ has benefits in terms of precision and recall in that it recommends components per hyper-personalization unit. It makes combinations between various user features and user contexts. In other words, it is possible to personalize for every context. However, compared to other algorithms/policies, there is cost that diversity is low as uncertainty is low. And it is also not possible to recommend for users in $Cold\ Start$.

As above, each contextual bandit has various benefits and costs. In other words, combination of algorithms/policies could improve the performance separately obtained by individual algorithm/policy [6]. So, we use ensemble method, which multiplies term weight to policy of each bandit for balancing. This model is called $LocalECB$ (Local Ensemble Contextual Bandit) as followed:

$$\pi_{LocalECB}(A \mid x) \approx \alpha * f(\pi_{semi-ts}(A \mid x)) + \beta * f(\pi_{full-cf}(A \mid x)) + \gamma * f(\pi_{full-linucb}(A \mid x)) \quad (3)$$

where π is policy that is a probability distribution over actions conditioned on context x, A is arm, $f(\cdot)$ means function of normalizing and α, β, γ is term weight ($\alpha + \beta + \gamma = 1$).

5. Case Study: In Various Recommendations in Local Domain

We explain about how the contextual bandits explained in the section 4 will be applied to menu/theme recommendation in NAVER service. We firstly show the common techniques, and then introduce the details in the subsection.

Since service feedback is not sufficient, in each recommendation, we transform feedback related to POI recommendations in NAVER (e.g. search, feed and map platform app/web) into a form of service feedback (e.g. user-menu feedback, user-theme feedback).

In each recommendation, CTR is used as reward for contextual bandits. We apply 3 techniques to the reward as followed. First, unlike general CTR reward, we make reward discounted [14] to preserve trend of dynamic user behavior. This means that higher weight is given to the latest feedback and lower weight is given to past feedback. Second, since supplemented feedback is not actual service feedback, we apply a logic to determine if click feedback is made according to the $Bernoulli\ Distribution$ [34] to reduce bias. Third, for quantity of click feedback, diversity, serendipity and bias, we use sliding window method [33] for enough number of feedbacks. This method is applied by smoothed CTR, that average of click/display count for a specific key (e.g. per segment, per user) is added to the click/display count, respectively.

5.1. Case 1: Menu Recommendation

We show the features used as context feature and explain how this context feature is applied to the contextual bandits in menu recommendation as followed:

- (Scenario) Contextual bandit is applied as a ranker considering contexts based on the user/query in platform, as illustrated in B Appendix.
- (Context Feature) Area, time, and user familiar area are used as user context, and imf-preference and exf-preference are used as user feature. And we check whether the corresponding contexts are valid in C.1 Appendix.

- (Contextual Bandit) $semi\text{-}ts$ uses area, time and user familiar area (user context) and don't use user feature, as segment recommendation. $full\text{-}cf$ uses only imf-preference (user feature), as a personalization with Collaborative Filtering [26]. $full\text{-}linucb$ uses area, time, user familiar area (user context) and exf-preference (user feature), as hyper-personalization recommendation. And $LocalECB$ is applied.

5.2. Case 2: Theme Recommendation

In theme recommendation, we introduce the features used as context feature and explain how this context feature is applied to the contextual bandits as followed:

- (Scenario) Contextual bandit is applied as ranker considering contexts based on the user/query or user/location in platform, as illustrated in B Appendix.
- (Context Feature) Area, and user familiar area are used as user context, and age, gender, imf-preference and exf-preference are used as user feature. And we prove whether the contexts are valid in C.2 Appendix.
- (Contextual Bandit) $semi\text{-}ts$ uses area, user familiar area (user context), age and gender (user feature), as segment recommendation. $full\text{-}cf$ uses only imf-preference (user feature), as a personalization with Collaborative Filtering [26]. $full\text{-}linucb$ uses area, user familiar area (user context), age, gender and exf-preference (user feature), as hyper-personalization recommendation. And ensemble model, called $LocalECB$, is applied.

6. Experiments

In the following, we empirically evaluate and discuss the performances of our models.

6.1. Experimental Setting

6.1.1. Menu/Theme Recommendation on a NAVER Search/Feed Platform

Menu/theme recommendations are conducted by extracting context feature based on user's information (user feature), query or location (user context). We set the number of K arms with considering various service factors. K = 3,215 menus and K = 1,549 themes are set as the total number of arms, respectively. All models will recommend 10 arms in service. And, in each recommendation, cover images, that are extracted through thumbnail extraction model of UCG (e.g. review, blog, etc.), constitute recommendation's cards in the form of carousel for each menu or theme, as in B Appendix.

6.1.2. Environment and Dataset for Evaluation

In experiment, we measure various indicators of recommendation (e.g. precision, recall, and diversity) by using user feedback in service, rather than measuring which method shows good performance with virtual indicator, such as expected cumulative reward/regret. We conduct experiments in 2 steps according to service strategy.

The first experiment (Initial Evaluation, see Section 6.2.1) is run by updating through user feedback for past 4 weeks, and precision [35], recall [35], and diversity [36] are measured using user feedback from the day after the past 4 weeks, as D Appendix; Precision is ratio of components clicked by user among K recommended components, recall is ratio of how many K recommended components are included among all the components clicked by user, and diversity is ratio of the number of unique components recommended to the total number of components. Through this experiment, we check whether our models are trained well for each its objective (benefit).

Table 1: Performance of precision on menu case. The best score is **bold**, second score is underlined.

(a) Initial Evaluation: Precision

Policy Name	Prec@1	Prec@5	Prec@10
$random$	0.00031	0.00031	0.00031
$base_{utm}$	0.00466	0.00209	0.00108
$base_{rsm}$	0.00076	0.00055	0.00036
$semi\text{-}ts$	0.00481	0.00333	0.00269
$full\text{-}cf$	0.00543	0.00497	0.00420
$full\text{-}linucb$	**0.01432**	**0.01064**	**0.00916**
$LocalECB$	0.00719	0.00609	0.00548

(b) Online Evaluation: Precision

Policy Name	Prec@1		Prec@5		Prec@10	
	deploy	learn	deploy	learn	deploy	learn
$random$	0.00032	0.00032	0.00030	0.00030	0.00031	0.00031
$base_{utm}$	0.00471	0.00471	0.00116	0.00116	0.00004	0.00004
$base_{rsm}$	0.00067	0.00067	0.00051	0.00051	0.00033	0.00033
$semi\text{-}ts$	0.00761	0.01108	0.00620	0.00909	0.00409	0.00798
$full\text{-}cf$	0.00574	0.00877	0.00448	0.00772	0.00294	0.00755
$full\text{-}linucb$	**0.01435**	**0.01440**	**0.01067**	**0.01072**	**0.00916**	**0.00922**
$LocalECB$	0.00795	0.01139	0.00769	0.00996	0.00395	0.00833

Table 2: Performance of recall on menu case. The best score is **bold**, second score is underlined.

(a) Initial Evaluation: Recall

Policy Name	Recall@1	Recall@5	Recall@10
$random$	0.00031	0.00156	0.00311
$base_{utm}$	0.00542	0.01185	0.01226
$base_{rsm}$	0.00079	0.00271	0.00348
$semi\text{-}ts$	0.00497	0.01692	0.02711
$full\text{-}cf$	0.00642	0.02702	0.03381
$full\text{-}linucb$	**0.01653**	**0.05830**	**0.09865**
$LocalECB$	0.00795	0.03035	0.04456

(b) Online Evaluation: Recall

Policy Name	Recall@1		Recall@5		Recall@10	
	deploy	learn	deploy	learn	deploy	learn
$random$	0.00032	0.00032	0.00156	0.00156	0.00311	0.00311
$base_{utm}$	0.00551	0.00551	0.01205	0.01205	0.01247	0.01247
$base_{rsm}$	0.00070	0.00070	0.00244	0.00244	0.00318	0.00318
$semi\text{-}ts$	0.00826	0.01214	0.03287	0.04389	0.05308	0.07796
$full\text{-}cf$	0.00806	0.00954	0.03907	0.04167	0.04237	0.05041
$full\text{-}linucb$	**0.01676**	**0.01681**	**0.05871**	**0.05885**	**0.09918**	**0.09952**
$LocalECB$	0.00874	0.01261	0.04343	0.04423	0.05348	0.06094

Table 3: Performance of diversity on menu case. The best score is **bold**.

Policy Name	Div@1	Div@5	Div@10
$random$	**1.00000**	**1.00000**	**1.00000**
$base_{utm}$	0.79565	0.87745	0.88709
$base_{rsm}$	0.04261	0.10887	0.12131
$semi\text{-}ts$	**1.00000**	**1.00000**	**1.00000**
$full\text{-}cf$	0.98612	1.00000	1.00000
$full\text{-}linucb$	0.09611	0.28771	0.42519
$LocalECB$	**1.00000**	**1.00000**	**1.00000**

In the second experiment (Online Evaluation, see Section 6.2.2), updating/simulation is run through daily user feedback, and indicators like the first experiment are measured using user feedback of the next two days through buckets, which are fraction of traffic. Buckets are divided into two types; Learning buckets consist of fraction of traffic where algorithms/policies are run to estimate CTR of service, and deployment buckets consist of fraction of traffic where bandits are run by previous bandit parameters. This experiment is intended to determine the effectiveness of online learning for quickly reflecting user feedback and to compare initial and online learning for indicator's variation that evaluates whether model is converged by user behavior as D Appendix.

Additionally, we test each experiment in Python environment with feedbacks of completely anonymized users in service. In the case of menu recommendation, about 3,100,000 anonymous users are tested, and in the case of theme recommendation, about 2,900,000 anonymous users are tested.

Since we conducted various experiments about two cases with large-scale data, it could be applicable to other domains.

Table 4: Performance of precision on theme case. The best score is **bold**, second score is underlined.

(a) Initial Evaluation: Precision

Policy Name	Prec@1	Prec@5	Prec@10
$random$	0.00065	0.00065	0.00065
$base_{popular}$	0.00436	0.00559	0.00371
$semi\text{-}ts$	0.01241	0.00885	0.00647
$full\text{-}cf$	0.04083	0.03165	0.02707
$full\text{-}linucb$	**0.05746**	**0.04522**	**0.03482**
$LocalECB$	0.04220	0.03989	0.02714

(b) Online Evaluation: Precision

Policy Name	Prec@1		Prec@5		Prec@10	
	deploy	learn	deploy	learn	deploy	learn
$random$	0.00066	0.00066	0.00065	0.00065	0.00066	0.00066
$base_{popular}$	0.00429	0.00429	0.00548	0.00548	0.00364	0.00364
$semi\text{-}ts$	0.01282	0.04782	0.00911	0.03686	0.00662	0.02663
$full\text{-}cf$	0.04455	0.04698	0.04123	0.04252	0.02806	0.03108
$full\text{-}linucb$	**0.05811**	**0.05816**	**0.04566**	**0.04605**	**0.03503**	**0.03530**
$LocalECB$	0.04573	0.05497	0.04137	0.04258	0.02841	0.03253

Table 5: Performance of recall on theme case. The best score is **bold**, second score is underlined.

(a) Initial Evaluation: Recall

Policy Name	Recall@1	Recall@5	Recall@10
$random$	0.00073	0.00340	0.00663
$base_{popular}$	0.00421	0.02358	0.03172
$semi\text{-}ts$	0.01428	0.04807	0.06596
$full\text{-}cf$	*0.05738*	*0.23438*	*0.31416*
$full\text{-}linucb$	**0.07344**	**0.28188**	**0.39721**
$LocalECB$	0.05951	0.27248	0.37222

(b) Online Evaluation: Recall

Policy Name	Recall@1		Recall@5		Recall@10	
	deploy	learn	deploy	learn	deploy	learn
$random$	0.00075	0.00075	0.00340	0.00340	0.00663	0.00663
$base_{popular}$	0.00417	0.00417	0.02300	0.02300	0.03091	0.03091
$semi\text{-}ts$	0.01478	0.05742	0.04967	0.22320	0.06778	0.29842
$full\text{-}cf$	0.05906	*0.06186*	0.22832	*0.24326*	0.35373	*0.37074*
$full\text{-}linucb$	**0.07408**	**0.07421**	**0.28403**	**0.28693**	**0.39915**	**0.40354**
$LocalECB$	0.06403	0.06956	0.27472	0.27743	0.39466	0.39708

Table 6: Performance of diversity on theme case. The best score is **bold**.

Policy Name	Div@1	Div@5	Div@10
$random$	**1.00000**	**1.00000**	**1.00000**
$base_{popular}$	0.01098	0.04067	0.07037
$semi\text{-}ts$	**1.00000**	**1.00000**	**1.00000**
$full\text{-}cf$	0.97557	**1.00000**	**1.00000**
$full\text{-}linucb$	0.17366	0.30923	0.44287
$LocalECB$	**1.00000**	**1.00000**	**1.00000**

6.2. Experimental Results

In this section, we show the results of experiments described in the previous section.

Additionally, in menu recommendation, $base_{utm}$ which is extracted based on menu preference of each user (exf-preference) described in section 3.2.3, and $base_{rsm}$ which means special menus for each area are also measured in experiments in the same way. Because these $base$ ranking collections are recommended in initial stage of service. And these collections are used as candidate for contextual bandit in service. Similarly, in theme recommendation, $base_{popular}$ which is popular themes among users in subscribe data is additionally measured. Since $base_{utm}$ is mainly extracted based on reviews left by users, this collection has user preferences for long-term and short-term. Therefore, the results are not good as the bandit models, as it does not reflect various user feedback, such as implicit feedback and does not directly reflect user behavior. And $base_{rsm}$ does not reflect user preferences, it shows lower results than the bandit models. Likewise, $base_{popular}$ shows the same results.

And compared to case of using only feedback of service, case of using service and other feedback is increased by about 15% in all indicators. And the results in the above tables are measured when the policy is updated with all feedback, which means that transformation has occurred.

Since we mention other factors in above, in this subsection we only compare performance between bandit models.

6.2.1. Initial Evaluation

First, we introduce the results of the precision experiment. According to Table 1(a), in menu recommendation, performance of precision shows better depending on the degree of personalization. Therefore, $full\text{-}linucb$ implemented with hyper-personalization shows the highest performance, and $LocalECB$ shows relatively lower performance than $full\text{-}linucb$. Because $LocalECB$ balances the benefits of the three contextual bandits. Similarly, theme recommendation shows the same results as Table 4(a).

Second, Table 2(a) and Table 5(a) present that performance of recall has the same order as performance of precision. This can also be interpreted as equally attributed to the degree of personalization. And compared to performance of precision, these results mean that our models are well-trained according to user behavior, as our models are robust. In other words, precision and recall have often inverse relationship according to confusion matrix, but if model is robust, precision and recall has specially some direct relationship. This is because the overall user behavior sequence has been well reflected as intended, using 4 weeks of user feedback.

Third, performance of diversity is measured as Table 3 and Table 6. These results are made by characteristic of each model. $seg\text{-}ts$, which has some randomness to recommend through sampling of beta distribution, and $LocalECB$, which ensembles the characteristics of the three models, show the best performance. In the case of other models, since these models use user feedback of service and user feedback of other interfaces at the same time, it can be interpreted as there is some data bias in personalization. So other models show relatively low performance than $seg\text{-}ts$, $LocalECB$.

Through this experiment, we check our models are trained well for each objective.

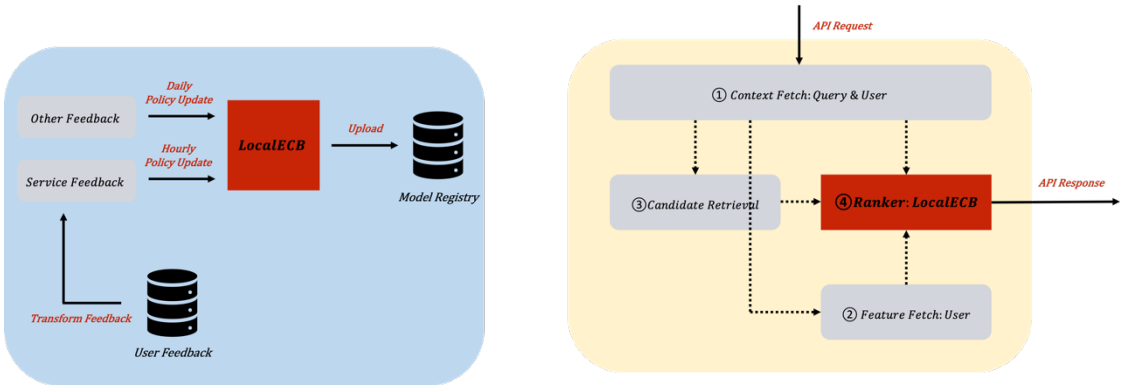
6.2.2. Online Evaluation

Comparing deployment buckets and learning buckets, we can see that performance of learning buckets is better than performance of deployment buckets in all indicators of evaluation, as Table 1(b) and Table 4(b) and recall in Table 2(b) and Table 5(b). This proves effectiveness of online learning, so we accept strategy of quickly simulations/updating algorithms/policies of contextual bandits.

To evaluate whether model is well converged according to user behavior, that is, whether there is little uncertainty, we compare the difference between offline evaluation's value and online evaluation's value. Since $seg\text{-}ts$ has randomness and is segment-based recommendation, it shows big difference of indicator depending on user behavior compared to other models. And in contrast, policy of $full\text{-}linucb$ does not change significantly according to the new user's behavior but shows good performance. Therefore, it proves that $LocalECB$ made by ensemble with above two models and $full\text{-}cf$ appropriately utilize uncertainty of models.

6.2.3. Entire Evaluation

Contextual bandit should consider precision and recall, which are related to reliability. And it also should consider diversity and model's uncertainty, which are related to discover behavior of user. In this respect, $LocalECB$, which balances benefits of other models, shows good performance in all indicators. So, it can be seen that $LocalECB$ is modeled as we intended. In other words, $LocalECB$ better matches the trade-off between exploration and exploitation than each single contextual bandit. Also, since each model used in ensemble has hyper-personalization, personalization, and semi-personalization unit, this fact shows that hyper-recommendation is possible in local domain.



(a) Feature Store Side (b) API Server Side

Figure 2: (a) Updating strategy for $LocalECB$ on feature store side. (b) Procedure of hyper-personalization by $LocalECB$ on API.

7. Contextual Bandit System

In this section, we introduce how contextual bandit works on the recommender system, as illustrated in Figure 2.

First, we explain how to update parameters of $LocalECB$. We transform service feedback and other interface's feedback into data format, which used in contextual bandit for updating. According to the development environment, service feedback is used as hourly updating, and other interface's feedback is used as daily updating. This method quickly optimizes recommendations per individual user through service feedback and creates a basic user behavior sequence using other feedback. After that, API server updates and serves as the latest model.

Second, we show how recommender system works for hyper-personalization, including contextual bandit system as ranker. Backend system of platform transmits data about user and result of query analysis through parser (or location through context-aware interface). And in API server, context fetch (user context), feature fetch (user feature), candidate retrieval (personalization level), and ranker (re-rank for hyper-personalization level) stage are followed in order.

8. Conclusion and Future Works

This study aims to introduce contextual bandits dealing with various and complex contexts in the local domain and show ensemble method for optimized recommendation with combining effective contextual bandits for preserving each benefit. And we explain how hyper-personalization is possible in this method. Since we share empirical hyper-personalization problem with contextual bandit, it may be applicable to other domains.

However, there are limitations of this work. First, it is context representations that is not expressive. Because current context features are extracted and concatenated through related labels. So, we will make context representations expressed by rich natural language through using a LLM (Large Language Model) [42, 43, 44, 45, 46, 47]. Second, it is computational cost (resource) problem. As there are three models, there are many resources used to learn and to sustain models. It is necessary to develop a state-of-the-art model.

Furthermore, in the future, by analyzing other user's behavior in platform, we make more detailed contexts features (e.g. weather, vehicles, and advanced summarizing feature of user, etc.). Through the expanded context, we will make more advanced hyper-personalization recommendation.

Acknowledgments

The authors would like to thank Jaehun Kim, Donggeol Shin, Changhoe Kim, Jihoon Choi and project members in NAVER for devoted support and discussion. And authors thank for Junguel Lee, Jungseok Lee and Mario Choi for valuable comments. Finally, the authors thank anonymous reviewers of BehavRec Workshop 2023 for their valuable comments.

Speaker BIO

Junhyung Kim is a ML engineer in the AirSPACE team of Naver, Republic of Korea. He current focuses on problem of hyper recommendations related to user behavior that connects various components in local domain. Previously, he was SW engineer at eBay Korea and got bachelor's degree in computer science from University of Yonsei.

Kijun Kwon is a ML engineer in the AirSPACE team of Naver, Republic of Korea. His focus is on AI modeling for personalized recommender system at hyper-local level. Previously, he was ML engineer at LG CNS and got bachelor's degree in industrial engineering from Korea University.

References

[1] Lihong Li, Wei Chu, John Langford, and Robert E Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In Proceedings of the 19th International Conference on World Wide Web. April 2010. Pages 661–670. https://doi.org/10.1145/1772690.1772758.

[2] Walid Bendada, Guillaume Salha and Théo Bontempelli. 2020. Carousel Personalization in Music Streaming Apps with Contextual Bandits. In Proceedings of the 14th ACM Conference on Recommender Systems. September 2020. Pages 420–425. https://doi.org/10.1145/3383313.3412217.

[3] James McInerney, Benjamin Lacker, Samantha Hansen, Karl Higley, Hugues Bouchard, Alois Gruson and Rishabh Mehrotra. 2018. Explore, exploit, and explain: personalizing explainable recommendations with bandits. In Proceedings of the 12th ACM Conference on Recommender Systems. September 2018. Pages 31–39. https://doi.org/10.1145/3240323.3240354.

[4] Junhyung Kim and Yeonghwan Jeon. 2022. Next POI Recommender System: Multi-view Representation Learning for Outstanding Performance in Various Context. In Proceedings of 2022 IEEE International Conference on Data Mining Workshops (ICDMW). November 2022. http://doi.org/10.1109/ICDMW58026.2022.00150.

[5] Jesús Bobadilla, Fernando Ortega, Antonio Hernando, and Abraham Gutiérrez. 2013. Recommender systems survey. Knowledge-Based Systems Volume 46. July 2013. PP 109–132. https://doi.org/10.1016/j.knosys.2013.03.012.

[6] Rocío Cañamares, Marcos Redondo and Pablo Castells. 2019. Multi-Armed Recommender System Bandit Ensembles. In Proceedings of the 13th ACM Conference on Recommender Systems. September 2019. Pages 432–436. https://doi.org/10.1145/3298689.3346984.

[7] Michel Tokic. 2010. Adaptive ε-greedy Exploration in Reinforcement Learning Based on Value Differences. In Proceedings of the 33rd annual German conference on Advances in artificial intelligence. September 2010. Pages 203–210. https://dl.acm.org/doi/10.5555/1882150.1882177.

[8] Volodymyr Kuleshov and Doina Precup. 2014. Algorithms for multi-armed bandit problems. Journal of Machine Learning Research. 1.

[9] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In Proceedings of the 24th International Conference on Neural Information Processing Systems. December 2011. Pages 2249–2257. https://dl.acm.org/doi/10.5555/2986459.2986710.

[10] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In Proceedings of the 30th International Conference on International Conference on Machine Learning. June 2013. Pages III-1220–III-1228. https://dl.acm.org/doi/10.5555/3042817.3043073.

[11] Branislav Kveton, Csaba Szepesvári, Zheng fu Wen, Mohammad Ghavamzadeh and Tor Lattimore. 2019. Garbage In, Reward Out: Bootstrapping Exploration in Multi-Armed Bandits. In Proceedings of the 36th International Conference on International Conference on Machine Learning. June 2019. PMLR 97:3601-3610. https://proceedings.mlr.press/v97/kveton19a.html.

[12] Garivier, Aurélien and Moulines, Eric. 2008. On Upper-Confidence Bound Policies for Non-Stationary Bandit Problems. https://doi.org/10.48550/arXiv.0805.3415.

[13] Fang Liu, Joohyun Lee, Ness Shroff. 2018. Change-Detection based Framework for Piecewise-stationary Multi-Armed Bandit Problem. In Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence. February 2018. Article No.: 447. Pages 3651–3658. https://dl.acm.org/doi/abs/10.5555/3504035.3504482.

[14] Vishnu Raj and Sheetal Kalyani. 2017. Taming Non-stationary Bandits: A Bayesian Approach. https://doi.org/10.48550/arXiv.1707.09727.

[15] Junpei Komiyama, Junya Honda, Hiroshi Nakagawa. 2015. Optimal Regret Analysis of Thompson Sampling in Stochastic Multi-armed Bandit Problem with Multiple Plays. In Proceedings of the 32nd International Conference on International Conference on Machine Learning. July 2015. Pages 1152–1161. https://dl.acm.org/doi/10.5555/3045118.3045241.

[16] Branislav Kveton, Csaba Szepesvári, Zheng Wen, Azin Ashkan. 2015. Cascading Bandits: Learning to Rank in the Cascade Model. In Proceedings of the 32nd International Conference on International Conference on Machine Learning. July 2015. Pages 767–776. https://dl.acm.org/doi/10.5555/3045118.3045201.

[17] Shuai Li, Tor Lattimore, Csaba Szepesvari. 2019. Online Learning to Rank with Features. In Proceedings of the 36th International Conference on Machine Learning. June 2019. PMLR 97:3856-3865. https://proceedings.mlr.press/v97/li19f.html.

[18] Robin Allesiardo, Raphaël Féraud and Djallel Bouneffouf. 2014. A Neural Networks Committee for the Contextual Bandit Problem. Neural Information Processing. ICONIP 2014. Lecture Notes in Computer Science, vol 8834. Springer, Cham. https://doi.org/10.1007/978-3-319-12637-1_47.

[19] Qingyun Wu, Naveen Iyer and Hongning Wang. 2018. Learning Contextual Bandits in a Non-stationary Environment. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval. June 2018. Pages 495–504. https://doi.org/10.1145/3209978.3210051.

[20] Bell Robert M, Yehuda Koren. Chris Volinsky. 2008. The BellKor 2008 Solution to the Netflix Prize. KorBell Team's Report to Netflix.

[21] Robin Burke. 2002. Hybrid Recommender Systems: Survey and Experiments. User Modeling and User-Adapted Interaction 12(4). November 2002. 10.1023/A:1021240730564.

[22] Fatih Aksel and Aysenur Akyuz Birtürk. 2010. An Adaptive Hybrid Recommender System that Learns Domain Dynamics. In International Workshop on Handling Concept Drift in Adaptive Information Systems: Importance, Challenges and Solutions (HaCDAIS-2010) at the European Conference on Machine Learning and Principles and Practice of Knowledge Discovery in Databases (ECML PKDD 2010). Barcelona, Spain, 49–56.

[23] Ariel Bar, Lior Rokach, Guy Shani, Bracha Shapira and Alon Schclar. 2013. Improving Simple Collaborative Filtering Models Using Ensemble Methods. Multiple Classifier Systems. MCS 2013. Lecture Notes in Computer Science, vol 7872. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-38067-9_1.

[24] Simon Dooms. 2013. Dynamic generation of personalized hybrid recommender systems. In Proceedings of the 7th ACM Conference on Recommender Systems. October 2013. Pages 443–446. https://doi.org/10.1145/2507157.2508069.

[25] Pigi Kouki, Shobeir Fakhraei, James Foulds, Magdalini Eirinaki and Lise Getoor. 2015. HyPER: A Flexible and Extensible Probabilistic Framework for Hybrid Recommender Systems. In Proceedings of the 9th ACM Conference on Recommender Systems. September 2015. Pages 99–106. https://doi.org/10.1145/2792838.2800175.
[26] Y. Hu, Y. Koren and C. Volinsky. 2008. Collaborative Filtering for Implicit Feedback Datasets. 2008 Eighth IEEE International Conference on Data Mining. December 2008. Pages 263-272. 10.1109/ICDM.2008.22.
[27] Musadig Aliyev, Elvin Ahmadov, Habil Gadirli, Arzu Mammadova and Emin Alasgarov. 2020. Segmenting Bank Customers via RFM Model and Unsupervised Machine Learning. https://doi.org/10.48550/arXiv.2008.08662.
[28] William R Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika 25, 3/4 (1933), 285–294.
[29] Bianca Dumitrascu, Karen Feng and Barbara E Engelhardt. 2018. PG-TS: Improved Thompson Sampling for Logistic Contextual Bandits. In Advances in Neural Information Processing Systems 30. 4629–4638. https://doi.org/10.48550/arXiv.1805.07458.
[30] Shuai Li, Alexandros Karatzoglou and Claudio Gentile. 2016. Collaborative Filtering Bandits. In Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval. July 2016. Pages 539–548. https://doi.org/10.1145/2911451.2911548.
[31] Richard S Sutton and Andrew G Barto. 2018. Reinforcement learning: An introduction. MIT press.
[32] Wei Chen, Yajun Wang, Yang Yuan, and Qinshi Wang. 2016. Combinatorial multi-armed bandit and its extension to probabilistically triggered arms. Journal of Machine Learning Research. January 1. pp. 1746–1778. https://dl.acm.org/doi/10.5555/2946645.2946695.
[33] Francesco Trovò, Stefano Paladino, Marcello Restelli and Nicola Gatti. 2020. Sliding-Window Thompson Sampling for Non-Stationary Settings. Journal of Artificial Intelligence Research. 68. 311-364. 10.1613/jair.1.11407.
[34] Aleksandrs Slivkins. 2019, Introduction to Multi-Armed Bandits. https://doi.org/10.48550/arXiv.1904.07272.
[35] Yan-Martin Tamm, Rinchin Damdinov and Alexey Vasilev. 2021. Quality Metrics in Recommender Systems: Do We Calculate Metrics Consistently?. In Proceedings of the 15th ACM Conference on Recommender Systems. September 2021. Pages 708–713. https://doi.org/10.1145/3460231.3478848.
[36] Mouzhi Ge, Carla Delgado-Battenfeld and Dietmar Jannach. 2010. Beyond Accuracy: Evaluating Recommender Systems by Coverage and Serendipity. In Proceedings of the fourth ACM conference on Recommender systems. September 2010. Pages 257–260. https://doi.org/10.1145/1864708.1864761.
[37] Christopher C. Johnson. 2014. Logistic Matrix Factorization for Implicit Feedback Data. In Advances of the 28th Neural Information Processing Systems. December 2014.
[38] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton and Jure Leskovec. 2018. Graph Convolutional Neural Networks for Web-Scale Recommender Systems. In Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. July 2018. Pages 974–983. https://doi.org/10.1145/3219819.3219890.
[39] Yong Zheng, Bamshad Mobasher and Robin Burke. 2014. Context Recommendation Using Multi-label Classification. 2014 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technologies. 2014. pp. 288-295. doi: 10.1109/WI-IAT.2014.110.
[40] Paul Covington, Jay Adams, Emre Sargin. 2016. Deep Neural Networks for YouTube Recommendations. In Proceedings of the 10th ACM Conference on Recommender Systems. September 2016. Pages 191–198. https://doi.org/10.1145/2959100.2959190.
[41] Emanuel Ben Baruch, T. Ridnik, Nadav Zamir, Asaf Noy, Itamar Friedman, M. Protter and Lihi Zelnik-Manor. 2020. Asymmetric Loss For Multi-Label Classification. IEEE/CVF International Conference on Computer Vision (ICCV). https://doi.org/10.48550/arXiv.2009.14119.

[42] Zhaopeng Qiu, Xian Wu, Jingyue Gao and Wei Fan. 2021. U-BERT: Pre-training User Representations for Improved Recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence. May 2021. Pages 4320-4327. https://doi.org/10.1609/aaai.v35i5.16557.

[43] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge and Yongfeng Zhang. 2022. A Unified Pretrain, Personalized Prompt & Predict Paradigm (P5). In Proceedings of the 16th ACM Conference on Recommender Systems. September 2022. Pages 299–315. https://doi.org/10.1145/3523227.3546767.

[44] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou and Hongxia Yang. 2022. M6-Rec: Generative Pretrained Language Models are Open-Ended Recommender Systems. https://doi.org/10.48550/arXiv.2205.08084.

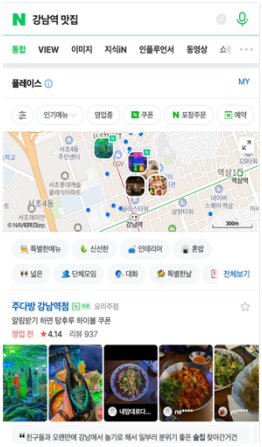
[45] Yupeng Hou, Shanlei Mu, Wayne Xin Zhao, Yaliang Li, Bolin Ding and Ji-Rong Wen. 2022. Towards Universal Sequence Representation Learning for Recommender Systems. In Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining. August 2022. Pages 585–593. https://doi.org/10.1145/3534678.3539381.

[46] Kyuyong Shin, Hanock Kwak, Su Young Kim, Max Nihlen Ramstrom, Jisu Jeong, Jung-Woo Ha and Kyung-Min Kim. 2023. Scaling Law for Recommendation Models: Towards General-purpose User Representations. In Proceedings of the AAAI Conference on Artificial Intelligence. October 2023. https://doi.org/10.48550/arXiv.2111.11294.

[47] Kyuyong Shin, Hanock Kwak, Wonjae Kim, Jisu Jeong, Seungjae Jung, Kyung-Min Kim, Jung-Woo Ha and Sang-Woo Lee. 2023. Pivotal Role of Language Modeling in Recommender Systems: Enriching Task-specific and Task-agnostic Representation Learning. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics. July 2023. Pages 1146–1161. 10.18653/v1/2023.acl-long.64.

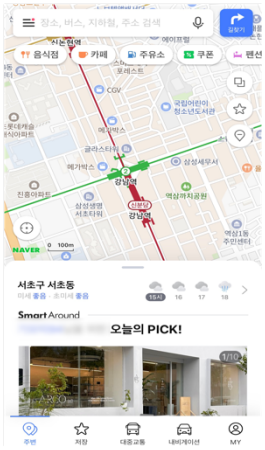
A. Examples of Local Search & Recommendation in NAVER Service

In the appendix A section, we provide examples of NAVER services for understanding structure and functionalities of NAVER.



(a) Search Platform

(b) Feed Platform

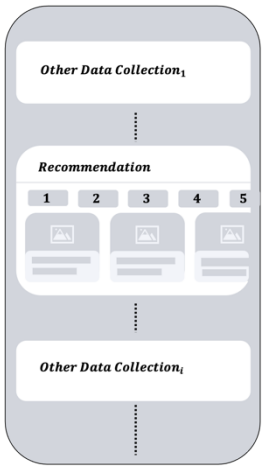


(c) Map Platform

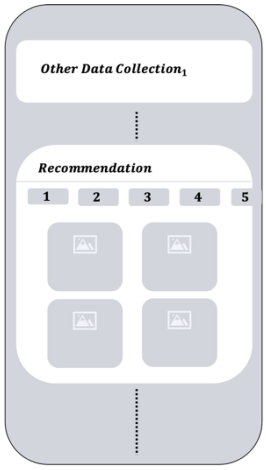
In (a) search platform, it basically works like this: location and context are set according to search query or user context, and then POI is recommended. And then, as user scrolls down, other collections tied to specific topic (e.g. menu, theme, etc.) are provided to users. In (b) feed platform, it provides collections related to topics that users would be interested in and allows users to discover local information. And in (c) map platform, it allows users to directly find local information on the map based on location.

B. Examples of Recommendation in Service Platforms

In the appendix B section, we show examples of forms where recommendations for each case are made in service platforms.



(a) Horizontal Carousel Type 1



(b) Horizontal Carousel Type 2

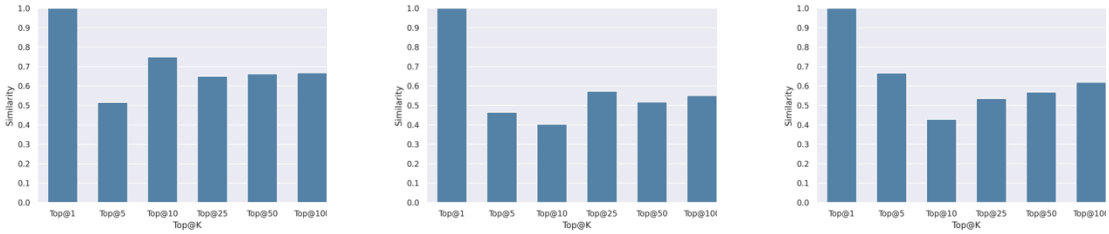
(c) Vertical Carousel

These are (a), (b) simplified illustration of swipeable horizontal carousel recommendation and (c) vertical carousel recommendation by scrolling. These forms will be applied to the NAVER search/feed platform. And these are recommendation in the form of UGC of related POI appeared according to the re-ranked components (e.g. menus, themes, etc.) for each user, based on user/query or user/location.

C. Validations of Contexts

In the appendix C section, we show whether the contexts used in each recommendation are valid. In the graphs of figure below, similarity means Jaccard Similarity calculated by $Top@k$, which is ranked by the number of clicks on between components (e.g. menu, theme) and key, such as area, time, (user) familiar (area), age, gender.

C.1 Validations of Contexts in Menu Recommendation

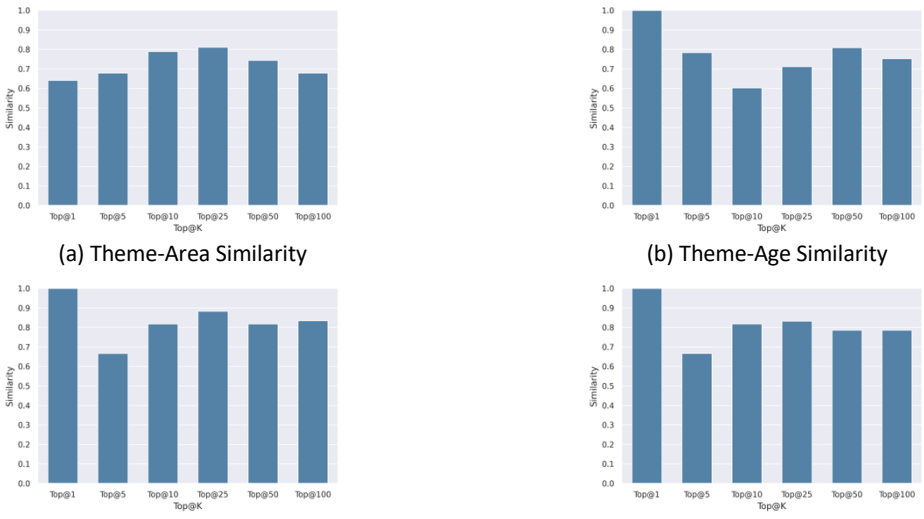


(a) Menu-Area Similarity (b) Menu-Time Similarity (c) Menu-Familiar Similarity

These are (a) menu-area, (b) menu-time, and (c) menu-(user area) familiar similarity based on Jaccard Similarity, using user feedback for a week.

We confirm that these are valid contexts because similarity is not constant in the menu-context similarity up to $Top@100$. That is, area, time, and familiar (user context) are valid contexts for menu recommendation. In the case of preferences, it is difficult to express graphically, but we make the features (user feature, e.g. imf-preference for menu, exf-preference for menu) to be different for each context.

C.2 Validations of Contexts in Theme Recommendation



(a) Theme-Area Similarity (b) Theme-Age Similarity

(c) Theme-Gender Similarity (d) Theme-Familiar Similarity

These are (a) theme-area, (b) theme-age, (c) theme-gender, and (d) theme-(user area) familiar similarity based on Jaccard Similarity, using user feedback for a week.

Like the menu recommendation, theme recommendation also selects contexts with similar results, which means similarity is not constant in the theme-context similarity up to $Top@100$. And preferences (user feature, e.g. imf-preference for theme, exf-preference for theme) are also made in the same way.

D. Formulas for Evaluation Metrics

D.1 Precision and Recall

$$Precision@k(u) = \frac{|click(u) \cap rec_k(u)|}{k}$$

$$Recall@k(u) = \frac{|click(u) \cap rec_k(u)|}{|click(u)|}$$

where u is user, $click$ is list of click item and rec_k is recommendation list.

D.2 Diversity

$$Diversity@k = \frac{total\ number\ of\ \sum_{u \in U} rec_k(u)}{total\ number\ of\ K\ arm}$$

where u is user and rec_k is recommendation list.

D.3 Variation

$$Variation@k = difference\ between\ value_{online@k}\ and\ value_{offline@k}$$

where $value$ is indicator's value of $Top@K$.