Conversational Recommendation With Online Learning and Clustering on Misspecified Users

Xiangxiang Dai, *Graduate Student Member, IEEE*, Zhiyong Wang, Jize Xie, Xutong Liu, and John C.S. Lui, *Fellow, IEEE*

***Abstract*—In the domain of conversational recommendation systems (CRSs), the development of recommenders capable of eliciting user preferences through conversation has marked a significant advancement. These systems have been enhanced by incorporating conversational key-terms related to items, which streamline the recommendation process by reducing the extensive exploration that traditional interactive recommenders necessitate. Despite these advancements, CRSs still face significant challenges. The vast number of users and the difficulty in accurately capturing preferences lead to persistent inaccuracies, even when direct user interactions are employed to refine the understanding of user preferences. To tackle these challenges, we propose two innovative bandit algorithms: RCLUMB (Robust Clustering of Misspecified Bandits) and RSCLUMB (Robust Set-based Clustering of Misspecified Bandits). These algorithms employ dynamic graphs and evolving cluster sets, respectively, to represent the changing structure of user preferences, thus leveraging collaborative user preferences to accelerate the learning process. Our algorithms are designed to be resilient against errors in preference modeling and the resulting inaccuracies in clustering. We rigorously analyze the performance of our algorithms and establish regret upper bounds of $O(\epsilon_* T\sqrt{md\log T} + d\sqrt{mT}\log T)$ under milder assumptions than previous works, matching the state-of-the-art results in several degenerate cases. Through extensive experiments on synthetic and real-world datasets, our algorithms demonstrate superior performance over existing algorithms.**

***Index Terms*—Misspecified model, online learning, conversational recommendation, bandit feedback, clustering of bandits.**

I. Introduction

RECOMMENDATION systems have become a cornerstone of user experience in digital platforms, harnessing user behavior data to predict preferences with notable success across e-commerce and social networking services, as evidenced by giants like Amazon and TikTok [1]. The evolution of these systems has been further propelled by the integration of conversational capabilities, enabling a more dynamic elicitation of user preferences through conversation [2], [3]. Unlike traditional recommendation systems that rely on a predefined set of item attributes, conversational recommenders can query users with a variety of key-terms related to a broad spectrum of item characteristics, thereby streamlining the recommendation process [3], [4]. For instance, by inquiring about a user's favorite cuisine, a system can efficiently narrow down the selection of restaurants to recommend.

Contextual linear bandits have been instrumental in refining the personalization of recommendations by modeling the expected reward (e.g., click-through rate) as a linear function of the interaction between item features and user preferences [5], [6], [7]. By leveraging contextual information about both users and items, these bandit algorithms can tailor recommendations to individual tastes [8]. Furthermore, CRSs enhance this approach by soliciting explicit user feedback on certain key-terms, using this conversational data to more accurately infer user preferences [9], [10], [11]. However, in real-world recommendation platforms, user bases are typically very large. Classical conversational bandit methods designed for individual users struggle to scale effectively to such extensive user bases, significantly slowing the learning process in practical applications.

Collaborative filtering, through clustering of bandits (CB), has been employed to capitalize on user relationships, allowing platforms to provide faster and more precise recommendations [12], [13]. Despite the success of existing CB algorithms, they operate under the strong assumption that users within the same cluster share identical preferences–an assumption that often does not hold due to the diversity of individual tastes and interests [12], [14], [15]. Additionally, both conversational linear bandits and CB algorithms presuppose a *perfectly linear* reward model. Nevertheless, this assumption may not align with the dynamic reality of CRSs, where user preferences and environmental factors are subject to change and uncertainty, leading to what is termed as a "*misspecified*" model. Such misspecification can cause significant deviations in reward predictions at both the item and key-term levels [16], [17], emphasizing the need for accommodating the inherent unpredictability and variability of real-world data.

Recognizing the various, dynamic, and often misspecified nature of user preferences, we introduce a new problem framework for "*online learning and clustering misspecified users* (OLCMU)" within CRSs. In OLCMU, we acknowledge that

Manuscript received 28 February 2024; revised 28 May 2024; accepted 1 July 2024. Date of publication 4 July 2024; date of current version 13 November 2024. This work of John C.S. Lui was supported in part by the RGC's GRF under Grant 14202923. Recommended for acceptance by X. He. *(Xiangxiang Dai and Zhiyong Wang contributed equally to this work.) (Corresponding author: Zhiyong Wang.)*

Xiangxiang Dai, Zhiyong Wang, Xutong Liu, and John C.S. Lui are with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, Central Ave, Hong Kong (e-mail: xxdai23@cse.cuhk.edu.hk; zywang21@cse.cuhk.edu.hk; liuxt@cse.cuhk.edu.hk; cslui@cse.cuhk.edu.hk).

Jize Xie is with the Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Clear Water Bay, Hong Kong (e-mail: jxiebj@connect.ust.hk.).

This article has supplementary downloadable material available at https://doi.org/10.1109/TKDE.2024.3423442, provided by the authors.

Digital Object Identifier 10.1109/TKDE.2024.3423442

the expected reward model for each user within large user bases may not be a perfectly linear combination of user preferences and item features, allowing for additive uncertainty deviations. We posit that users within the same unknown interest cluster share common preferences, while permitting individual deviations to capture the diversity of user personalities. This nuanced approach to linearity and reward homogeneity within clusters presents a complex set of challenges for OLCMU. The framework must manage the uncertainty stemming from unknown user preference vectors and model misspecifications. Robust algorithms for the OLCMU problem must strike a delicate equilibrium: it should exhibit sufficient flexibility to accommodate model misspecifications, thereby ensuring that users with analogous preferences are clustered together to capitalize on collaborative advantages. Simultaneously, it must maintain a discerning approach to clustering, carefully distinguishing between users to avoid the erroneous grouping of individuals with markedly divergent preferences under the recommendation process.

In summary, this article presents the following four key contributions to the field of CRSs.

Innovative Model Formulation: We pioneer the formulation of the OLCMU problem for CRSs. This model is grounded in practicality, accommodating additive uncertainties in user interactions with both items and conversational key-terms, reflecting a more realistic scenario in user-system interactions.

Cutting-edge Algorithmic Design: We introduce two innovative algorithms, RCLUMB (Robust Clustering of Misspecified Bandits) and RSCLUMB (Robust Set-based Clustering of Misspecified Bandits). These algorithms effectively learn user clusters despite preference model misspecifications and use this collaborative data to quickly refine preference predictions. They incorporate a conversational query strategy, utilizing interaction history to select key-terms for exploration, and merge recommendation and conversational data streams. RCLUMB updates a user graph to reflect evolving clusters, while RSCLUMB handles clusters through adaptable sets that evolve with the learning process. To comprehensively address the OLCMU problem, we present both graph-based and set-based solutions, showcasing their effectiveness in handling misspecified users.

To navigate the complexities of user model misspecification, our algorithms implement: i) A more tolerant edge deletion rule, accounting for potential misspecifications, ensuring that similar users remain connected. ii) A discriminative filtering approach to cluster formation, which prevents the misclustering of users by considering only those with sufficiently similar preferences. iii) An expanded confidence radius that encompasses both the exploration bonus and the additional uncertainty from misspecification when recommending items.

Rigorous Theoretical Analysis: We establish regret upper bounds for our algorithms at $O(\epsilon_* T\sqrt{md\log T} + d\sqrt{mT}\log T)$ within the OLCMU problem, predicated on more relaxed and realistic assumptions than those typically found in bandit literature [12], [13], [14]. These bounds are competitive with the best results in specific scenarios. Our analysis diverges from conventional methods, particularly in addressing the challenge of regret due to misclustering users with similar but distinct preferences. A key lemma is introduced to bound this regret component (detailed in Section V), which may have broader implications. We also give a regret lower bound of $\Omega(\epsilon_* T\sqrt{d})$ for OLCMU, showing that our upper bounds are asymptotically tight with respect to T up to logarithmic factors.

Good Experimental Performance: Through comprehensive experiments on both synthetic and real-world datasets, our algorithms demonstrate superior performance compared to existing algorithms, showcasing their practical effectiveness in real-world CRS applications with misspecified users.

II. Related Work

Bandits utilizing User Relationships: The seminal paper by Gentile et al. [12] introduces the clustering of bandits (CB) problem, presenting a graph-based algorithm to address it. Subsequent work [18] expands on this by utilizing collaborative effects on items to inform user clustering strategies. Li et al. [14] further extend the CB framework to cascading bandits with random prefix feedback. Another work [15] differentiates users based on their varying arrival frequencies. Liu et al. [13] recently proposes a federated bandits clustering approach, balancing privacy concerns with communication efficiency. In [19], we discuss a misspecified contextual bandit model; however, we do not address the issue of dual misspecified feedback. Besides these CB works, a *pre-defined* user adjacency graph is exploited to share context among users [20], whereas CB typically requires the discovery of user relationships.

Conversational Recommendation: Conversational recommender systems (CRSs) have evolved to interact with users by inquiring about their preferences [21]. Enhancements in CRSs have been achieved through deep learning and reinforcement learning to generate dialogues and assist in recommendations, albeit without theoretical guarantees [22], [23]. Recent advancements include the utilization of conversation on key terms to refine user preferences [9]. Further developments involve incorporating additional information sources such as relative feedback [10], self-generated key terms [24], and knowledge graphs [25]. Beyond the bandit-based recommendation models, [26] explores policy learning by incorporating a meta-exploration policy and a Transformer-based state encoder. Additionally, [27] utilizes hierarchical information modeling. There are also works leveraging natural language processing; for instance, [28] integrates recommendations using knowledge-enhanced prompt learning and applies a task-specific pre-trained language model. [29] responds to evolving user preferences with large language models. Price is considered in [30] based on Attention. Unlike these works, our research focuses on learning unknown user relations via potentially misspecified bandit feedback at both the arm and key-term levels.

Misspecified Linear Bandits: The concept of misspecified linear bandits (MLB) is first proposed by Ghosh et al. [17], highlighting the susceptibility of linear bandit algorithms to deviations and introducing an algorithm robust to non-sparse deviations. Lattimore et al. [31] offer two algorithms to address general deviations, building upon the phased elimination algorithm [32] and LinUCB [7]. Recent studies [33], [34] have employed model selection techniques to manage the unknown maximum level of model misspecification. Notably, the work by Foster et al. [34] assumes access to an online regression oracle, and Pacchiano et al. [33] require knowledge of an upper bound on the model deviation level. Our work is distinct in that it explores the collaborative effect of clustering similar users under model misspecifications.

Collaborative Learning Paradigms: Research in multi-task learning [35], [36], [37], [38], meta-learning [39], [40], [41], and federated learning [42], [43] has focused on jointly solving multiple tasks and sharing information among them. Unlike these works, we assume an underlying unknown user clustering structure that the agent must infer to accelerate learning. Multi-task learning studies [35], [36], [37], [38] consider task-relatedness without user clustering, and to our knowledge, do not address model misspecifications. Meta-learning research [39], [40], [44] has introduced Bayesian hierarchical models for cross-task knowledge sharing and Thompson Sampling-based algorithms for optimizing Bayes regret. Federated learning works [42], [43] focus on privacy protection and communication among servers.

Our contribution is pioneering the study of the Online Learning and Clustering Misspecified Users (OLCMU) problem, proposing a comprehensive framework to handle model misspecifications in CB problems. Future work could integrate model selection methods [33], [34] into our framework to address the unknown maximum level of model misspecification. It would also be intriguing to apply our approach and insights on model misspecifications to multi-task learning, meta-learning, and federated learning domains.

III. System Model and Problem Formulation

This section formulates the problem framework of "*online learning and clustering misspecified users* (OLCMU)",[1] as illustrated in Fig. 1.

A. User Clusters With Potentially Misspecified Preferences

In the context of OLCMU, we consider a scenario with u users, represented by the set $\mathcal{U} = \{1, 2, \dots, u\}$. Each user $i \in \mathcal{U}$ has an associated preference vector $\boldsymbol{\theta}_i \in \mathbb{R}^d$, which is *unknown* and satisfies $\|\boldsymbol{\theta}_i\|_2 \le 1$. The users can be grouped into the same cluster based on the similarity of their preferences, which remains *unknown* to the agent. More precisely, the user set $\mathcal{U}$ is formulated to be partitioned into m clusters, with m being significantly smaller than u (i.e., $m \ll u$). These clusters are denoted by $V_1, V_2, \dots, V_m$, satisfying the conditions $\cup_{j\in[m]} V_j = \mathcal{U}$ and $V_j \cap V_{j'} = \emptyset$ for all $j \neq j'$. We refer to these clusters as *ground-truth clusters*, and we use the notation $V = \{V_1, V_2, \dots, V_m\}$ to represent the set of these clusters. Within this framework, users belonging to the same *ground-truth cluster* share a similar preference vector. Conversely, users from distinct *ground-truth clusters* have unique preference vectors.

[1]Notation: *lowercase* boldface letters denote vectors, and *CAPITALIZED* boldface letters denote matrices. The cardinality of a set $\mathcal{A}$ is denoted by $|\mathcal{A}|$, the set of the first m positive integers is denoted by $[m]$, and the matrix norm of a vector $\boldsymbol{x}$ with respect to a positive semi-definite (PSD) matrix $\boldsymbol{M}$ is denoted by $\|\boldsymbol{x}\|_{\boldsymbol{M}} = \sqrt{\boldsymbol{x}^\top \boldsymbol{M}\boldsymbol{x}}$.

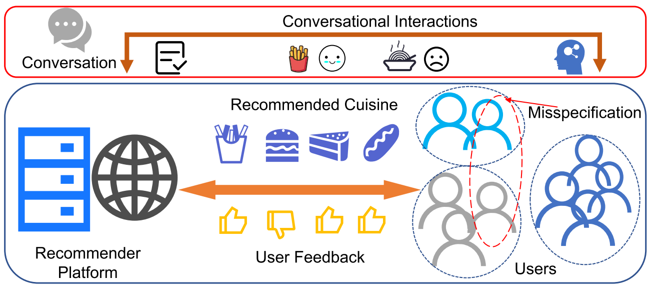


Fig. 1. Illustration of OLCMU. The recommender platform (agent) presents a selection of items to the served users, who in turn provide feedback on these offerings. The agent is also capable of engaging in direct conversations with users to gain a deeper insight into their preferences. Users exhibiting similar tastes are clustered together to facilitate the provision of more personalized services. However, inaccuracies from misspecification may arise from both direct conversations and feedback on item choices, potentially leading to erroneous assessments of user preferences and groupings.

The common preference vector for users in cluster V_j is denoted by $\boldsymbol{\theta}^j$. For any user i, let $j(i) \in [m]$ indicate the *ground-truth cluster* to which user i belongs.

At each round $t \in [T]$, a user $i_t \in \mathcal{U}$ arrives to be served. The learning agent is presented with a finite set of arms $\mathcal{A}_t$ from which to choose (i.e., item set), where each arm $a \in \mathcal{A}_t \subseteq \mathcal{A}$ is associated with a normalized feature vector $\boldsymbol{x}_a \in \mathbb{R}^d$. The learning agent (i.e., recommender) assigns user i_t to an appropriate cluster $\overline{V}_t$ and recommends an arm $a_t \in \mathcal{A}_t$ based on the aggregated historical information from cluster $\overline{V}_t$. Upon receiving the recommendation a_t, user i_t provides a stochastic reward $r_t \in [0, 1]$ to the agent. To more accurately reflect real-world recommendation scenarios, we posit that the reward r_t is generated by a misspecified linear function of the item's feature vector $\boldsymbol{x}_{a_t}$ and the *unknown* user preference vector $\boldsymbol{\theta}_{i_t}$. Formally,

$$r_t = \boldsymbol{x}_{a_t}^\top \boldsymbol{\theta}_{i_t} + \boldsymbol{\epsilon}_{a_t}^{i_t,t} + \eta_t \,, \tag{1}$$

where $\boldsymbol{\epsilon}^{i_t,t} = [\boldsymbol{\epsilon}_1^{i_t,t}, \boldsymbol{\epsilon}_2^{i_t,t}, \dots, \boldsymbol{\epsilon}_{|\mathcal{A}_t|}^{i_t,t}]^\top \in \mathbb{R}^{|\mathcal{A}_t|}$ represents the *unknown* misspecification from the expected linearity of rewards for the arms in $\mathcal{A}_t$ for user i_t at round t, and η_t is 1-sub-Gaussian noise. Note that even users within the same cluster can have different misspecification vectors, providing a more detailed representation of their preferences. Misspecifications in the user model can be static or change over time. Unlike sub-Gaussian noise, which typically has a zero-mean expectation, misspecifications do not have a fixed mean, allowing for perturbations in the expected reward model away from linearity.

Let $a_t^* \in \arg\max_{a\in\mathcal{A}_t} \boldsymbol{x}_a^\top \boldsymbol{\theta}_{i_t} + \boldsymbol{\epsilon}_a^{i_t,t}$ represent the optimal arm that yields the highest expected reward at round t. The agent's objective is to minimize the expected cumulative regret defined as

$$R(T) = \mathbb{E}\left[\sum_{t=1}^{T} (\boldsymbol{x}_{a_t^*}^\top \boldsymbol{\theta}_{i_t} + \boldsymbol{\epsilon}_{a_t^*}^{i_t,t} - \boldsymbol{x}_{a_t}^\top \boldsymbol{\theta}_{i_t} - \boldsymbol{\epsilon}_{a_t}^{i_t,t})\right] , \tag{2}$$

where the expectation accounts for the randomness of the algorithm and the environment, including the sequence of users $i_1, \dots, i_T$, the misspecifications, and the arm sets $\mathcal{A}_1, \dots, \mathcal{A}_T$.

B. Key-Term Feedback and User Interaction Management

In contrast to traditional recommendation systems, CRSs not only provide recommendations but also intermittently solicit direct feedback from users about specific "key-terms." These key-terms are keywords or topics linked to a subset of arms. For instance, the key-term "cuisine" could encompass a variety of food-related categories such as Italian, Chinese, vegetarian, etc. We denote the finite set of key-terms as $\mathcal{K}$.

The relationship between arms and key-terms is represented by a weighted bipartite graph $(\mathcal{A}, \mathcal{K}, \boldsymbol{W})$, where $\boldsymbol{W} \triangleq [w_{a,k}], a \in \mathcal{A}, k \in \mathcal{K}$ is the weight matrix. This matrix indicates the strength of association between each arm $a \in \mathcal{A}$ and key-term $k \in \mathcal{K}$. A non-negative weight $w_{a,k}$ reflects the level of association, with the stipulation that each key-term k is positively associated with at least one arm (i.e., $\sum_{a \in \mathcal{A}} w_{a,k} > 0$ for all $k \in \mathcal{K}$), and the weights for each arm sum to 1 (i.e., $\sum_{k \in \mathcal{K}} w_{a,k} = 1$ for each $a \in \mathcal{A}$). The feature vector for a key-term k is constructed as $\tilde{\boldsymbol{x}}_k = \sum_{a \in \mathcal{A}} \frac{w_{a,k}}{\sum_{a' \in \mathcal{A}} w_{a',k}} \boldsymbol{x}_a$. The feedback mechanism for a key-term k at time t from a user $i_t \in \mathcal{U}$, who may be either normal or corrupted, is mathematically expressed as:

$$\tilde{r}_{k,t} = \tilde{\boldsymbol{x}}_k^\top \boldsymbol{\theta}_{i_t} + \tilde{\boldsymbol{\epsilon}}_k^{i_t,t} + \tilde{\eta}_t \,, \tag{3}$$

where $\tilde{\boldsymbol{\epsilon}}^{i_t,t} = [\tilde{\boldsymbol{\epsilon}}_1^{i_t,t}, \tilde{\boldsymbol{\epsilon}}_2^{i_t,t}, \dots, \tilde{\boldsymbol{\epsilon}}_{|\mathcal{K}|}^{i_t,t}]^\top \in \mathbb{R}^{|\mathcal{K}|}$ denotes the *unknown* misspecification from expected linearity in the key-term rewards for user i_t at time t, and $\tilde{\eta}_t$ is 1-sub-Gaussian noise. Following previous research [9], [10], [25], the unknown user preference vector $\boldsymbol{\theta}_{i_t}$ is consistent across both the arm and key-term levels. Our model, however, uniquely accounts for the potential of misspecification in key-term feedback. In recommendation systems, it is imperative to account for misspecifications in key-term rewards, as a user's genuine preferences for specific categories, such as "desserts" or "spicy food", may deviate from the linear predictions due to the intricate and varied nature of individual palates.

To ensure a positive user experience, the agent must judiciously manage the frequency of conversation interactions. We introduce a conversation frequency function $b_{i_t}(t)$ for the user i_t currently being served. This function determines the number of conversational interactions initiated by the agent. At each round t, the system may engage in $q(t) = \lfloor b_{i_t}(t) - b_{i_t}(t-1) \rfloor$ conversations with user i_t, provided $b_{i_t}(t) - b_{i_t}(t-1) > 0$. Consequently, over the course of the interaction, the agent will have engaged in $b_{i_t}(t)$ conversations with user i_t.

Following works addressing misspecified linear bandits [31], we postulate that the infinity norm of the misspecification vector $\boldsymbol{\epsilon}^{i,t}$ is bounded by ϵ_* for all users $i \in \mathcal{U}$ and rounds $t \in [T]$, across both arm and key-term levels, which is not necessarily required to know by our algorithms. This cap is generally predetermined based on the expectation that misspecification will be minor [17], allowing to establish a relatively large value as a conservative upper bound. However, in scenarios where it is not predefined, one could potentially employ contemporary model selection techniques [34] to address this issue. For clarity, we do not delve into these methods.

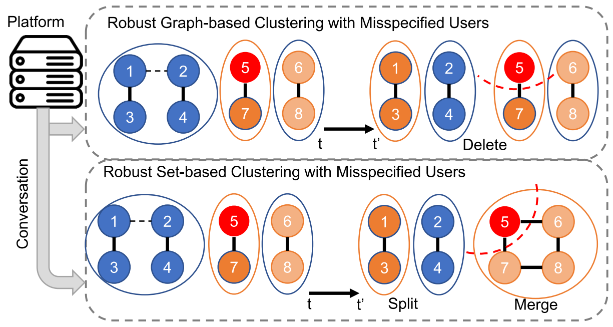

Platform
Conversation
Robust Graph-based Clustering with Misspecified Users
1
2
3
4
5
7
6
8
t
t'
Delete
Robust Set-based Clustering with Misspecified Users
t
t'
Split
Merge

Fig. 2. Illustration of RCLUMB and RSCLUMB. Graph-based RCLUMB primarily clusters using a deletion strategy, whereas set-based RSCLUMB utilizes splitting and merging strategies. Under the CRS platform, restricted user conversations can be directly initiated. Moreover, RCLUMB and RSCLUMB both strive to minimize the interference of misspecified user information during clustering and recommendation, as demonstrated in the 'red user 5' example.

IV. Algorithm Design

As shown in Fig. 2, this section introduces our two algorithms called "Robust Clustering of Misspecified Bandits" (RCLUMB, see in Algorithm 1) and "Robust Set-based Clustering of Misspecified Bandits"(RSCLUMB, see in Algorithm 3 to conduct systematic research on the OLCMU problem and show the generality of our ideas and techniques in addressing issues arising from misspecified users wthin CRSs.

To facilitate understanding, we introduce the coefficient ζ as the theoretical minimum gap between the preference vectors of two users that an algorithm can reliably discern their dissimilarity with a high probability, which is *unknown* to the agent. For clarity, we provide the following definition:

Definition IV.1 (ζ-close users and ζ-good clusters): Two users $i, i' \in \mathcal{U}$ are considered ζ-close if $\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_{i'}\|_2 \le \zeta$. At round t, a cluster $\overline{V}$ is deemed a ζ-good cluster if, for every user $i \in \overline{V}$, user i and the incoming user i_t are ζ-close.

Similarly, we describe two *ground-truth clusters* as "ζ-close" if the distance between their preference vectors is less than ζ.

A. Grapg-Based RCLUMB Algorithm Design

We first introduce the process and intuitions of RCLUMB in detail (refer to Algorithm 1). This algorithm constructs an undirected graph $G_t = (\mathcal{U}, E_t)$ across users, connecting users with edges when they are deemed to belong to the same cluster. The connected component in G_{t-1} that includes user i_t at round t is denoted as $\tilde{V}_t$.

Cluster Detection: We start with a complete graph, G_0, which is updated over time based on user feedback. When a user i_t arrives at round t, i_t is presented with a set of choices $\mathcal{A}_t$. Instead of trying to group users with exactly the same preferences, we cluster them based on their preferences being relatively close, within a certain distance ζ. Over time, it becomes very likely that users connected in the graph have preferences that are within this ζ distance from each other. Previous methods use larger clusters which could lead to errors because the preferences of users further away in the cluster could differ significantly from i_t. To improve accuracy, a refined method is utilized where

Algorithm 1: Robust Clustering of Misspecified Bandits Algorithm (RCLUMB).

```
1:  Input: Deletion parameter $\alpha_1, \alpha_2 > 0$,
    $f(T) = \sqrt{\frac{1+\ln(1+b_i(T)+T)}{1+b_i(T)+T}}$, $\lambda, \beta, \epsilon^* > 0$.
2:  Initialization:
      • $\boldsymbol{M}_{i,0} = 0_{d\times d}, \boldsymbol{b}_{i,0} = 0_{d\times 1}, T_{i,0} = 0, \forall i \in \mathcal{U}$;
      • A complete Graph $G_0 = (\mathcal{U}, E_0)$ over $\mathcal{U}$.
3:  for all $t = 1, 2, \ldots, T$ do
4:      Receive the index of the current user $i_t \in \mathcal{U}$, and the current feasible arm set $\mathcal{A}_t$;
5:      Select key-terms to by running Conversation (Algorithm 2) based on the specified key-term selection strategy $\boldsymbol{\pi}$;
6:      Determine the connected component $\tilde{V}_t$ in the current graph $G_{t-1} = (\mathcal{U}, E_{t-1})$, such that $i_t \in \tilde{V}_t$;
7:      Filter cluster $\tilde{V}_t$ to find the cluster $\overline{V}_t \subseteq \tilde{V}_t$ which only contains user $i_t$ and users $i \in \tilde{V}_t$ that are directly connected with user $i_t$ via edge $(i, i_t) \in E_{t-1}$;
8:      Compute the estimated statistics for cluster $\overline{V}_t$
          $\overline{\boldsymbol{M}}_{\overline{V}_t,t-1} = \lambda \boldsymbol{I} + \sum_{i\in\overline{V}_t} \boldsymbol{M}_{i,t-1}$,
          $\overline{\boldsymbol{b}}_{\overline{V}_t,t-1} = \sum_{i\in\overline{V}_t} \boldsymbol{b}_{i,t-1}$, $\hat{\boldsymbol{\theta}}_{\overline{V}_t,t-1} = \overline{\boldsymbol{M}}^{-1}_{\overline{V}_t,t-1}\overline{\boldsymbol{b}}_{\overline{V}_t,t-1}$;
9:      Recommend an arm $a_t$ with the largest UCB index as shown in (5), and receive the reward $r_t \in [0,1]$;
10:     Update the statistics for user $i_t$
          $\boldsymbol{M}_{i_t,t} = \boldsymbol{M}_{i_t,t-1} + \boldsymbol{x}_{a_t}\boldsymbol{x}_{a_t}^\top$,
          $\boldsymbol{b}_{i_t,t} = \boldsymbol{b}_{i_t,t-1} + r_t\boldsymbol{x}_{a_t}$, $T_{i_t,t} = T_{i_t,t-1} + 1$,
          $\hat{\boldsymbol{\theta}}_{i_t,t} = (\lambda \boldsymbol{I} + \boldsymbol{M}_{i_t,t})^{-1}\boldsymbol{b}_{i_t,t}$;
11:     Keep the statistics of other users unchanged
          $\boldsymbol{M}_{\ell,t} = \boldsymbol{M}_{\ell,t-1}, \boldsymbol{b}_{\ell,t} = \boldsymbol{b}_{\ell,t-1}, T_{\ell,t} = T_{\ell,t-1}$,
          $\hat{\boldsymbol{\theta}}_{\ell,t} = \hat{\boldsymbol{\theta}}_{\ell,t-1}$, for all $\ell \in \mathcal{U}, \ell \neq i_t$;
12:     Delete the edge $(i_t, \ell) \in E_{t-1}$, if
          $\left\|\hat{\boldsymbol{\theta}}_{i_t,t} - \hat{\boldsymbol{\theta}}_{\ell,t}\right\|_2 \geq \alpha_1\Big(f(T_{i_t,t}) + f(T_{\ell,t})\Big) + \alpha_2\epsilon^*$,
        and get an updated graph $G_t = (\mathcal{U}, E_t)$;
```

only users directly connected (1-hop away) to i_t are considered. This smaller, more focused group is more likely to have similar preferences, which helps in making better recommendations.

Cluster-based Recommendation: After identifying the appropriate cluster $\overline{V}_t$ for user i_t, the agent estimates the shared user preference using the historical data associated with cluster $\overline{V}_t$, which is formulated as a regularized least squares problem:

$$\hat{\boldsymbol{\theta}}_{\overline{V}_t,t-1} = \mathop{\arg\min}_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{\substack{s\in[t-1]\\ i_s\in\overline{V}_t}} (r_s - \boldsymbol{x}_{a_s}^\top\boldsymbol{\theta})^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{4}$$

Algorithm 2: Conversation.

```
1: if $b_{i_t}(t) - b_{i_t}(t-1) > 0$ for user $i_t$ then
2:     $q(t) = \lfloor b_{i_t}(t) - b_{i_t}(t-1)\rfloor$;
3:     while $q(t) > 0$ do
4:         Select a key-term $k \in \mathcal{K}$ using the selection strategy $\boldsymbol{\pi}$ (refer to (8) or (9)) to inquire about the user's preference.
5:         Receive the user's feedback $\tilde{r}_{k,t}$;
6:         $\boldsymbol{M}_{i_t,t} = \boldsymbol{M}_{i_t,t-1} + \tilde{\boldsymbol{x}}_k\tilde{\boldsymbol{x}}_k^\top$,
           $\boldsymbol{b}_{i_t,t} = \boldsymbol{b}_{i_t,t-1} + \tilde{\boldsymbol{x}}_k\tilde{r}_{k,t}$;
7:         $q(t) -= 1$;
8: else
9:     $\boldsymbol{M}_{i_t,t} = \boldsymbol{M}_{i_t,t-1}, \boldsymbol{b}_{i_t,t} = \boldsymbol{b}_{i_t,t-1}$;
```

where $\lambda > 0$ is the regularization parameter. The closed-form solution for this optimization problem is given by: $\hat{\boldsymbol{\theta}}_{\overline{V}_t,t-1} = \overline{\boldsymbol{M}}^{-1}_{\overline{V}_t,t-1}\overline{\boldsymbol{b}}_{\overline{V}_t,t-1}$, where $\overline{\boldsymbol{M}}_{\overline{V}_t,t-1} = \lambda\boldsymbol{I} + \sum_{\substack{s\in[t-1]\\ i_s\in\overline{V}_t}} \boldsymbol{x}_{a_s}\boldsymbol{x}_{a_s}^\top$, $\overline{\boldsymbol{b}}_{\overline{V}_t,t-1} = \sum_{\substack{s\in[t-1]\\ i_s\in\overline{V}_t}} r_{a_s}\boldsymbol{x}_{a_s}$. This estimation is utilized in the upper confidence bound (UCB) strategy for recommending an arm, as shown in Line 9:

$$a_t = \text{argmax}_{a\in\mathcal{A}_t}\left\{\min\left\{1, \hat{R}_{a,t} + C_{a,t}\right\}\right\}, \tag{5}$$

where $\hat{R}_{a,t} = \boldsymbol{x}_a^\top\hat{\boldsymbol{\theta}}_{\overline{V}_t,t-1}$ is the estimated reward for arm a at time t, and $C_{a,t}$ is the confidence radius for arm a at round t, which accounts for the uncertainty in the estimation:

$$C_{a,t} = \beta\|\boldsymbol{x}_a\|_{\overline{\boldsymbol{M}}^{-1}_{\overline{V}_t,t-1}} + \epsilon^* \sum_{\substack{s\in[t-1]\\ i_s\in\overline{V}_t}} \left|\boldsymbol{x}_a^\top\overline{\boldsymbol{M}}^{-1}_{\overline{V}_t,t-1}\boldsymbol{x}_{a_s}\right|, \tag{6}$$

where $\beta = \sqrt{\lambda} + \sqrt{2\log(T) + d\log(1 + \frac{b_{i_t}(T)+T}{\lambda d}})$, and ϵ^* is the pre-misspecified parameter, accounting for the additional uncertainty due to deviations from linearity. The construction of $C_{a,t}$ is designed from our theoretical analysis, which will be presented in Section V.

Update User Preference Statistics: Upon receiving feedback r_t from user i_t regarding arm a_t, the agent updates the preference statistics for user i_t in Lines 10 and 11, while the statistics for other users remain unaltered. Specifically, the agent refines the estimate of the preference vector $\boldsymbol{\theta}_{i_t}$ by solving the following regularized least squares problem:

$$\hat{\boldsymbol{\theta}}_{i_t,t} = \mathop{\arg\min}_{\boldsymbol{\theta}\in\mathbb{R}^d} \sum_{\substack{s\in[t]\\ i_s=i_t}} (r_s - \boldsymbol{x}_{a_s}^\top\boldsymbol{\theta})^2 + \lambda\|\boldsymbol{\theta}\|_2^2, \tag{7}$$

which yields the closed-form solution: $\hat{\boldsymbol{\theta}}_{i_t,t} = (\lambda\boldsymbol{I} + \boldsymbol{M}_{i_t,t})^{-1}\boldsymbol{b}_{i_t,t}$, where $\boldsymbol{M}_{i_t,t}$ and $\boldsymbol{b}_{i_t,t}$ are defined as: $\boldsymbol{M}_{i_t,t} = \sum_{\substack{s\in[t]\\ i_s=i_t}} \boldsymbol{x}_{a_s}\boldsymbol{x}_{a_s}^\top$, $\boldsymbol{b}_{i_t,t} = \sum_{\substack{s\in[t]\\ i_s=i_t}} r_{a_s}\boldsymbol{x}_{a_s}$.

Conversation on Key-terms: The agent engages with users at the key-term level to refine its understanding of their preferences, as outlined in Algorithm 2. At each round t, the agent assesses the possibility of initiating conversations based on the function $b_{i_t}(t)$ for the current user i_t. Should the conditions allow, the agent solicits the user's input on $q(t)$ key-terms. This input is then utilized to adjust the system's parameters, following a predefined selection strategy $\boldsymbol{\pi}$. The strategy $\boldsymbol{\pi}$ bifurcates into two distinct approaches. The first approach aims to maximize the confidence radius with uncertainty considered, as defined by:

$$k \in \arg\max_{k \in \mathcal{K}} \beta \left\| \tilde{\boldsymbol{x}}_k \right\|_{\boldsymbol{M}_{i_t,t}^{-1}} + \epsilon^* \sum_{j=1}^{q(t)} \left| \boldsymbol{x}_k^\top \boldsymbol{M}_{i_t,t}^{-1} \boldsymbol{x}_{k_j} \right| , \quad (8)$$

where $k_j \in \mathcal{K}$ represents the key-term selected between the number of conversation. The rationale behind this approach is that a key-term with a larger confidence radius suggests that the recommendation system has yet to thoroughly probe the user's preferences concerning the items associated with that key-term. Consequently, such key-terms are prime candidates for further exploration. The second strategic option also takes into account the information already exploited, targeting key-terms associated with less explored areas:

$$k \in \arg\max_{k \in \mathcal{K}_t} \tilde{\boldsymbol{x}}_k^\top \hat{\boldsymbol{\theta}}_{i_t,t} + \beta \left\| \tilde{\boldsymbol{x}}_k \right\|_{\boldsymbol{M}_t^{-1}} + \epsilon^* \sum_{j=1}^{q(t)} \left| \boldsymbol{x}_k^\top \boldsymbol{M}_{i_t,t}^{-1} \boldsymbol{x}_{k_j} \right| , \quad (9)$$

where the first term $\tilde{\boldsymbol{x}}_k^\top \hat{\boldsymbol{\theta}}_{i_t,t}$ represents the estimated reward for key-term k, and the later two terms accounts for the uncertainty in the estimate, thus encouraging a balance between exploiting known preferences and exploring new potential interests.

Update the Graph G_t: Subsequently, in Line 12, the agent assesses the persisting similarity between user i_t and other connected users based on the newly estimated $\hat{\boldsymbol{\theta}}_{i_t,t}$. For each user $\ell \in \mathcal{U}$ linked to user i_t by an edge $(i_t, \ell) \in E_{t-1}$, the agent examines if the discrepancy between their estimated preference vectors $\hat{\boldsymbol{\theta}}_{\ell,t}$ and $\hat{\boldsymbol{\theta}}_{i_t,t}$ exceeds a certain threshold. The threshold is carefully designed, taking both estimation uncertainty in a linear model and deviations from linearity into consideration. As delineated in the proof analysis in the supplementary material, employing this threshold ensures that, with high probability, edges within the same *ground-truth clusters* remain intact, while those connecting users who are not ζ-close are pruned. This process, with the filtering mechanism in Line 7, allows the algorithm to effectively harness the collaborative information of similar users and eschew the data of those who are dissimilar. The refined graph $G_t = (\mathcal{U}, E_t)$ is carried forward to the next round.

Algorithm 3: Robust Set-Based Clustering of Misspecified Bandits Algorithm (RSCLUMB).

```
1: Input: Deletion parameter $\alpha_1, \alpha_2 > 0$,
   $f(T) = \sqrt{\frac{1+\ln(1+b_i(T)+T)}{1+b_i(T)+T}}$, $\lambda, \beta, \epsilon^* > 0$.
2: Initialization:
 • $\boldsymbol{M}_{i,0} = 0_{d\times d}, \boldsymbol{b}_{i,0} = 0_{d\times 1}, T_{i,0} = 0, \forall i \in \mathcal{U}$;
 • Initialize the set of cluster indexes by $J = \{1\}$ and the
   single cluster $\boldsymbol{S}_1$ by $\boldsymbol{M}_1 = 0_{d\times d}$, $\boldsymbol{b}_1 = 0_{d\times 1}$, $T_1 = 0$,
   $C_1 = \mathcal{U}$, $j(i) = 1$, $\forall i$, s=1;
3: for all $t = 1, 2, \ldots, T$ do
4:   if $t = 2^s - 1$ then
5:     Mark every user unchecked for each cluster;
6:     For each cluster $V_j$, compute $\tilde{T}_{V_j} = T_{V_j}$,
       $\hat{\boldsymbol{\theta}}_{V_j} = (\lambda \boldsymbol{I} + \boldsymbol{M}_{V_j})^{-1} \boldsymbol{b}_{V_j}$, $\tilde{\boldsymbol{\theta}}_{V_j} = \frac{\sum_{i \in V_j} \hat{\boldsymbol{\theta}}_i}{[V_j]}$; s=s+1;
7:     Receive the user $i_t$ and the decision set $\mathcal{D}_t$;
8:     Select key-terms to by Conversation (Algorithm 2)
       based on the specified key-term selection strategy $\boldsymbol{\pi}$;
9:     Determine the cluster index $j = j(i_t)$;
10:      Recommend item $a_t$ with the largest UCB index as
         shown in (5);
11:      Received the feedback $r_t$;
12:      Update the information:
           $\boldsymbol{M}_{i_t,t} = \boldsymbol{M}_{i_t,t-1} + \boldsymbol{x}_{a_t} \boldsymbol{x}_{a_t}^{\mathrm{T}}, \boldsymbol{b}_{i_t,t} = \boldsymbol{b}_{i_t,t-1} + r_t \boldsymbol{x}_{a_t}$,
           $T_{i_t,t} = T_{i_t,t-1} + 1, \hat{\boldsymbol{\theta}}_{i_t,t} = (\lambda \boldsymbol{I} + \boldsymbol{M}_{i_t,t})^{-1} \boldsymbol{b}_{i_t,t}$
           $\boldsymbol{M}_{V_j,t} = \boldsymbol{M}_{V_j,t-1} + \boldsymbol{x}_{a_t} \boldsymbol{x}_{a_t}^{\mathrm{T}}, \boldsymbol{b}_{V_j,t} = \boldsymbol{b}_{V_j,t-1} + r_t \boldsymbol{x}_t$,
           $T_{V_j,t} = T_{V_j,t-1} + 1, \hat{\boldsymbol{\theta}}_{V_j,t} = (\lambda \boldsymbol{I} + \boldsymbol{M}_{V_j,t})^{-1} \boldsymbol{b}_{V_j,t}$,
           $\tilde{\boldsymbol{\theta}}_{V_j,t} = \frac{\sum_{i \in V_j} \hat{\boldsymbol{\theta}}_i, t}{[V_j]}$
13:      if $i_t$ is unchecked then
14:        Run Split (Algorithm 4);
15:        Mark user $i_t$ has been checked
16:        Run Merge (Algorithm 5);
```

B. *Set-Based RSCLUMB Algorithm Design*

We then present the "Robust Set-based Clustering of Misspecified Bandits Algorithm" (RSCLUMB, see in Algorithm 3). RSCLUMB diverges from RCLUMB, by adopting a set-based clustering structure. Furthermore, while RCLUMB is limited to partitioning clusters, RSCLUMB is designed to dynamically adjust its clustering by both splitting and merging sets. Specifically, the agent will excise a user from its current set (cluster) upon detecting a discrepancy between the user's behavior and the set's profile. Conversely, should the agent ascertain that two clusters exhibit sufficiently similar estimated preferences, it will amalgamate them into a single set. For an in-depth exploration of the relationship between the graph and set structures within this context, the reader is referred to [15].

Initialization and Phase Structure: RSCLUMB commences by forming a universal set $\boldsymbol{S}_1$ that encapsulates all users. This set is dynamically refined throughout the learning process. The algorithm operates in distinct phases (see Algorithm 3 Line 3), with each phase s spanning 2^s rounds. At the onset of a phase, all users are labeled as "unchecked." As users are active on the recommendation system, they transition to "checked" status. Once all its constituent users are inspected, a cluster achieves a "checked" status, signifying its validity for that phase. This stratagem ensures that each phase upholds a certain accuracy standard, allowing the agent to concentrate on refining clusters that have not yet reached this benchmark. Subsequently, akin to RCLUMB, for the received user $i_t \in \mathcal{U}$ Algorithm 2 is executed for conducting conversation interactions.

Algorithm 4: Split.

```
1: if $\|\hat{\boldsymbol{\theta}}_{i_t,t} - \tilde{\boldsymbol{\theta}}_{V_j,t}\| > \alpha_1(f(T_{i_t,t}) + f(T_{V_j,t})) + \alpha_2\epsilon^*$
   then
2:     Split user $i_t$ from cluster $V_j$ and form a new cluster $V_j'$
       $\boldsymbol{M}_{V_j,t} = \boldsymbol{M}_{V_j,t} - \boldsymbol{M}_{i_t,t}, \boldsymbol{b}_{V_j} = \boldsymbol{b}_{V_j} - \boldsymbol{b}_{i_t,t},$
       $T_{V_j,t} = T_{V_j,t} - T_{i_t,t}, C_{j,t} = C_{j,t} - \{i_t\},$
       $\boldsymbol{M}_{V_j',t} = \boldsymbol{M}_{i_t,t}, \boldsymbol{b}_{V_j',t} = \boldsymbol{b}_{i_t,t},$
       $T_{V_j',t} = T_{i_t,t}, C_{j',t} = \{i_t\};$
```

Algorithm 5: Merge.

```
1: for any two checked clusters $V_{j_1}, V_{j_2}$ satisfying
       $\left\|\tilde{\boldsymbol{\theta}}_{j_1} - \tilde{\boldsymbol{\theta}}_{j_2}\right\| < \frac{\alpha_1}{2}(f(T_{V_{j_1}}) + f(T_{V_{j_2}})) + \frac{\alpha_2}{2}\epsilon^*$
   do
2:     Merge them:
       $\boldsymbol{M}_{V_{j_1}} = \boldsymbol{M}_{j_1} + \boldsymbol{M}_{j_2}, \boldsymbol{b}_{V_{j_1}} = \boldsymbol{b}_{V_{j_1}} + \boldsymbol{b}_{V_{j_2}},$
       $T_{V_{j_1}} = T_{V_{j_1}} + T_{V_{j_2}}, C_{V_{j_1}} = C_{V_{j_1}} \cup C_{V_{j_2}};$
3:     Set $j(i) = j_1, \forall i \in j_2$, delete $V_{j_2}$;
```

Cluster Maintenance and User Interaction: Similar to RCLUMB, RSCLUMB employs two distinct vectors for each cluster V_j: the recommendation vector $\hat{\boldsymbol{\theta}}_{V_j}$ and the cluster integrity vector $\tilde{\boldsymbol{\theta}}_{V_j}$. The former, akin to $\hat{\boldsymbol{\theta}}_{\overline{V}_j}$ in RCLUMB, is utilized for generating recommendations. The latter represents the centroid of user preference estimates within the cluster and is instrumental in assessing the need for cluster modification. At each time step t within phase s, upon the arrival of user i_t with their corresponding item set $\mathcal{D}_t$ (where t indexes total time steps), RSCLUMB ascertains the user's cluster and proceeds with a cluster-based recommendation. Post-recommendation, the algorithm updates its information (Algorithm 3 Line 12) and evaluates the potential for cluster modification (Algorithm 3 Lines 13-17). A cluster is deemed "good" if all user estimates are proximal to its integrity vector. Users are considered "consistent" with a cluster if their preference estimate aligns closely with the cluster's integrity vector. Conversely, "inconsistent" users are candidates for cluster separation. Similarly, clusters with closely aligned integrity vectors may be merged.

Robust Clustering Mechanism: RSCLUMB maintains two sets of estimated cluster vectors: i) cluster-level estimation with integrated user information, which is for recommendations (Line 12 and Line 10 in Algorithm 3); ii) the average of estimated user vectors, which is used for robust clustering (Line 2 in Algorithm 4 and Line 2 in Algorithm 5). This dual-vector approach is a departure from previous set-based CB methodologies [15], which relied solely on the former for both recommendations and clustering. Such a singular approach is prone to clustering inaccuracies in the presence of model misspecifications, potentially leading to non-trivial regret bounds in the OLCMU.

V. Theoretical Analysis

In this section, we theoretically analyze the performance of our proposed RCLUMB and RSCLUMB algorithms by giving an upper bound of the expected regret defined in (2). Due to the space limitation, we only show the main results, and a sketched proof. Detailed proofs and other technical lemmas can be found in the appendix of the supplementary material. Consistent with previous CB literature [12], [13], [14], we operate under the following assumptions, with ϵ^* equal to ϵ_*.

Assumption V.1 (Gap between different clusters): The gap between any two preference vectors for different *ground-truth clusters* is at least an *unknown* positive constant γ

$$\left\|\boldsymbol{\theta}^j - \boldsymbol{\theta}^{j'}\right\|_2 \geq \gamma > 0\,, \forall j, j' \in [m]\,, j \neq j'\,.$$

Assumption V.2 (Uniform arrival of users): At each round t, a user i_t comes uniformly at random from $\mathcal{U}$ with probability $1/u$, independent of the past rounds.

Assumption V.3 (Item regularity): The feature vector $\boldsymbol{x}_a$ of each arm $a \in \mathcal{A}_t$ at round t is drawn independently from a fixed but unknown distribution ρ over $\{\boldsymbol{x} \in \mathbb{R}^d : \|\boldsymbol{x}\|_2 \leq 1\}$, where $\mathbb{E}_{\boldsymbol{x}\sim\rho}[\boldsymbol{x}\boldsymbol{x}^\top]$ is full rank with minimal eigenvalue $\lambda_x > 0$. Additionally, for any fixed unit vector $\boldsymbol{\theta} \in \mathbb{R}^d$, $(\boldsymbol{\theta}^\top \boldsymbol{x})^2$ has sub-Gaussian tail with variance upper bounded by σ^2.

Definition V.1 (Minimum separable gap γ_1): The minimum separable gap constant γ_1 of the OLCMU problem instance is the minimum gap over the gaps among users that are greater than ζ. Formally, for $\forall i, \ell \in \mathcal{U}$,

$$\gamma_1 = \min\{\|\boldsymbol{\theta}_i - \boldsymbol{\theta}_\ell\|_2 : \|\boldsymbol{\theta}_i - \boldsymbol{\theta}_\ell\|_2 > \zeta\}, \min\emptyset = \infty.$$

Remark 1: Our approach relaxes the assumption in earlier studies, particularly those concerning the upper bound variance σ^2. We start by introducing two key definitions. The first pertains to the minimum separable gap constant γ_1 for the OLCMU problem instance, and the second relates to the number of "hard-to-cluster users" $\tilde{u}$. The coefficient in this context is detailed in the proof of Lemma V.1 in the supplementary material. In the OLCMU setting, the term $\gamma_1 - \zeta$ measures the clustering difficulty of the problem instance. A larger γ_1 generally implies easier clustering. The subtraction of ζ represents the increased difficulty arising from model misspecifications. When there are no misspecifications, that is, $\zeta = 0$, γ_1 equals γ, as specified in Assumption V.1. This equivalence aligns our results with those found in prior research [13], [14].

Definition V.2 (number of "hard-to-cluster users" $\tilde{u}$): The number of "hard-to-cluster users" $\tilde{u}$ in $\mathcal{U}$ is the number of users in the *ground-truth clusters* which are ζ-close to at least one another *ground-truth cluster* in V, i.e.,

$$\tilde{u} = \sum_{j\in[m]} |V_j| \times \mathbb{I}\{\exists j' \in [m], j' \neq j : \left\|\boldsymbol{\theta}^{j'} - \boldsymbol{\theta}^j\right\|_2 \leq \zeta\}\,,$$

where $\mathbb{I}\{\cdot\}$ denotes the indicator function of the argument, $|V_j|$ denotes the number of users contained in V_j.

Remark 2: $\tilde{u}$ represents the number of "hard to cluster" users from different clusters with separation gaps less than ζ. These users are prone to being incorrectly merged into a single cluster, leading to potential errors in the clustering process. This occurs because if two clusters are within ζ of each other, the uncertainty from model misspecifications might cause the algorithm to mistakenly combine them, resulting in clustering challenges for users in these close proximity clusters.

Denoting $\tilde{\lambda}_x \triangleq \int_0^{\lambda_x}(1-e^{-\frac{(\lambda_x-x)^2}{2\sigma^2}})^C dx$, the following lemma establishes the time rounds required for the algorithms to achieve a "good partition."

Lemma V.1: Employing the meticulously devised clustering methods, it is guaranteed that after

$$t \geq O\left(u\left(\frac{d}{\tilde{\lambda}_x(\gamma_1-\zeta)^2}+\frac{1}{\tilde{\lambda}_x^2}\right)\log\frac{1}{\delta}\right)$$

rounds, with a high probability, both RCLUMB and RSCLUMB will have obtained a "good partition".

Assuming the conditions outlined in Section III hold true, the subsequent theorem delineates an upper bound for the expected cumulative regret incurred by RCLUMB and RSCLUMB.

Theorem V.1 (Regret upper bound): The expected regret of the RCLUMB and RSCLUMB algorithms for T rounds satisfies

$$R(T) \leq O(\epsilon_* T\sqrt{md\log T} + d\sqrt{mT}\log T)\,. \qquad (10)$$

Remark 3: The derived upper bound is composed of four main components. The first term indicates the number of rounds needed to gather enough data for accurate user clustering, crucial for successful outcomes in later rounds. The second term measures the regret from misclustering users who are closely related (ζ-close) due to inevitable model inaccuracies. The third term reflects errors in estimating user preferences resulting from model deviations. The final term, a standard component in bandit clustering, assumes perfect linearity, as discussed in prior studies [14], [15]. Notably, both the second and third terms increase linearly with T, aligning with the linear trends noted in earlier research [17], [31]. These trends underscore that linear terms are typical in scenarios with misspecified linear bandits, caused by unforeseen deviations. Let's explore how these parameters influence the overall regret bound.

- When $\gamma_1-\zeta$ is substantial, indicating that the separation between clusters not "ζ-close" is significantly larger than the minimum discernible gap ζ, the first term of the regret is minimized. This is because the algorithm can more readily detect dissimilarities between clusters. In the OLCMU context, $\gamma_1-\zeta$ plays a role analogous to γ in classical CB models.
- If the count of "hard-to-cluster" users, denoted by $\tilde{u}$, is minimal, suggesting that few *ground-truth clusters* are "ζ-close", the likelihood of the algorithm *misclustering* distinct *ground-truth clusters* is reduced.
- Finally, if the model deviation level ϵ_* is small, meaning the user models are close to linear models and the misspecification will not affect the estimations much.

The following theorem gives a regret lower bound of the OLCMU problem (The proof is in the supplementary material).

Theorem V.2 (Regret lower bound): For the OLCMU problem, there exists at least one problem instance where the regret $R(T)$ for any algorithm is bounded from below as follows: $R(T) \geq \Omega(\epsilon_* T\sqrt{d})$.

Remark 4: The upper bounds in Theorem V.1 asymptotically align with this lower bound with respect to T, accounting for logarithmic factors and a constant factor of $\sqrt{m}$ (where m is generally small in real-world applications), illustrating the robustness of our theoretical findings. We speculate that the gap for the m factor might stem from the stringent assumption of known cluster structures used to establish this lower bound, leaving room to explore a potentially tighter lower bound in future research. Moreover, we also compare our results with several special cases, as our work is pioneering in addressing the OLCMU problem. First, for $m=1$ (implying $\tilde{u}=0$), our framework simplifies to the misspecified linear bandits (MLB) problem, where all users share a common preference vector, achieving a regret bound of $O(\epsilon_* T\sqrt{d\log T}+d\sqrt{T}\log T)$, which aligns with the best current bound for MLB [31]. Second, when $\epsilon_*=0$, our framework corresponds to the online clustering bandits (CB) problem with perfectly linear user models, achieving a bound of $O(d\sqrt{mT}\log T)$, matching the best existing bound for CB [14], [15]. These comparisons affirm the precision of the regret bounds for our proposed RCLUMB and RSCLUMB algorithms.

Before we give the sketched proof for Theorem V.1, we define the following "good partition" for ease of interpretation.

Definition V.3 (Good partition): We say that RCLUMB or RSCLUMB does a "good partition" at t, if the cluster $\overline{V}_t$ assigned to user i_t is a ζ-good cluster, and it contains all the users in the same *ground-truth cluster* as i_t, i.e.,

$$V_{j(i_t)} \subseteq \overline{V}_t\,, \|\boldsymbol{\theta}_{i_t}-\boldsymbol{\theta}_\ell\|_2 \leq \zeta, \forall \ell \in \overline{V}_t\,. \qquad (11)$$

Remark 5: One can notice that when the algorithm does a "good partition" at t, $\overline{V}_t$ will contain all the users in the same *ground-truth cluster* as i_t and may only contain some other ζ-close users with respect to i_t, which means the information gathered associated with $\overline{V}_t$ can be used to infer user i_t's preference with high accuracy. Also, it is obvious that under a "good partition", if $\overline{V}_t \in V$, then $\overline{V}_t = V_{j(i_t)}$ by definition.

Next, we give a sketched proof for Theorem V.1. For the sake of our analysis, we posit a linear relationship of the form $b_i(t) = b_i \cdot t$, where b_i is a constant residing within the open interval (0, 1) and more basic strategy in (8) is discussed here. We focus on RCLUMB for this specific analysis, with RSCLUMB based on a similar conceptual framework. Comprehensive proofs are in the supplementary material.

Sketch for Theorem V.1: The proof is structured into two primary segments. Initially, we establish the existence of a sufficient time frame, denoted as T_0, within which RCLUMB is capable of obtaining a "good partition." The regret incurred during the initial T_0 rounds can be straightforwardly capped by T_0, given that $r_t \in [0,1]$ for all t. Subsequently, the more intricate aspect of the proof involves formulating an upper bound for the regret attributable to the *misclustering* of ζ-close users, even after a "good partition" has been achieved. As shown in Lemma V.1, we can prove that after $t \geq O(u(\frac{d}{\tilde{\lambda}_x(\gamma_1-\zeta)^2}+\frac{1}{\tilde{\lambda}_x^2})\log T)$, for any user $i \in \mathcal{U}$, the gap between the estimated $\hat{\boldsymbol{\theta}}_{i,t}$ and the ground-truth $\boldsymbol{\theta}^{j(i)}$ is less than $\frac{\gamma_1}{4}$ with high probability. With this, we can get the following results. For any two users i and ℓ, if their gap is greater than ζ, it will trigger the deletion of the edge (i,ℓ) (Line 12 of Algorithm 1). On the other hand, when the deletion condition of the edge (i,ℓ) is satisfied, then $\|\boldsymbol{\theta}^{j(i)}-\boldsymbol{\theta}^{j(\ell)}\|_2>0$, which means user i and ℓ belong to different *ground-truth clusters* by Assumption V.1. Therefore, all those users in the same *ground-truth cluster* as i_t can be partitioned into the connected component $\tilde{V}_t$, and users in $\tilde{V}_t$ will be directly connected with i_t if and only if they are ζ-close to i_t. By the extraction method of $\overline{V}_t$ from $\tilde{V}_t$ and the definition of "good partition", we can ensure that RCLUMB will keep a "good partition" afterward. After obtaining a "good partition", at round t, if: i) $\overline{V}_t \in V$, meaning that the cluster assigned for user i_t is the same as her *ground-truth cluster*, i.e., $\overline{V}_t = V_{j(i_t)}$, the instantaneous regret R_t can be bounded by $2C_{a_t,t}+2\epsilon_*$. ii) $\overline{V}_t \notin V$, which means that the algorithm has *misclustered* user i_t, i.e., $\overline{V}_t \neq V_{j(i_t)}$, but all the users in $\overline{V}_t$ are ζ-close to i_t (by definition of "good partition"), then R_t is added by the extra term $\epsilon_*\sqrt{2d}/\tilde{\lambda}_x^{1.5}$ due to the *misclustering*, which causes the algorithm to use the information of i_t's ζ-close users in $\overline{V}_t$ lying in different *ground-truth clusters* from i_t to estimate $\boldsymbol{\theta}_{i_t}$. The expected number of occurrences of case (ii) is bounded by $\frac{\tilde{u}}{u}T$ according to Assumption V.2, Definition V.2 and Definition V.3. The result then follows by bounding the expected summation of the bounds for R_t, which is similar to the proof for CB [14], but with a more subtle analysis due to the time-varying clustering structure kept by our proposed algorithms. □

VI. Simulations

To evaluate the adaptability of our proposed algorithms, we designed two distinct simulation setups: one that excludes conversational feedback to verify the algorithms' applicability in the effectiveness of misspecified user clustering, and another that incorporates various types of conversational feedback for comparative analysis of conversational interactions.

A. Baseline Comparisons

1) User Clustering Efficacy Comparisons: To evaluate the effectiveness of our proposed algorithms in clustering users with similar preferences under misspecified models, we conduct a comparative analysis against a suite of established benchmark algorithms. Our comparison includes the following baselines:

- *LinUCB:* A method uses a single estimated preference vector shared across all users [5].
- *LinUCB-Ind:* An approach maintains separate estimated preference vectors for each user based on LinUCB [5].
- *RLinUCB* and *RLinUCB-Ind:* Two refined versions of LinUCB targeted at misspecification models in [31], which we denote as RLinUCB and RLinUCB-Ind, respectively.
- *CLUB:* Introduced by [12], this algorithm employs a graph-based approach to cluster multiple users.
- *SCLUB:* An improved set-based clustering algorithm for bandits with multiple users [15].

2) Conversational Interaction Comparisons: To provide a comprehensive comparison within the context of CRSs and to highlight the capability of our algorithm with potentially misspecified conversational feedback, we include the following baseline algorithms that also facilitate direct user interactions:

- *Arm-Con:* A conversational bandit algorithm that engages users in discussions about arms without considering key-terms, using LinUCB for arm selection [21].
- *ConUCB:* A fundamental conversational bandit algorithm that chooses a key-term to minimize the estimation error when conversational interaction is possible [9].
- *ConLinUCB:* A suite of algorithms with varying key-term selection strategies [11], including: *ConLinUCB-BS*, which selects key-terms based on the barycentric spanner for exploration, and *ConLinUCB-MCR*, which utilizes historical key-term selection data to identify terms with the largest confidence radius.

Regarding the strategy $\boldsymbol{\pi}$ to select key terms in conversation, for both RCLUMB and RSCLUMB, we employ (8) during the first half of the user's turns to encourage as much exploration as possible. After accumulating a certain amount of feedback information, we adopt (9) to balance the existing information with potential exploration.

B. Simulation Settings

To gauge user satisfaction in cases where feedback may be misspecified, we employ the metric of average rewards over time t for randomly selected users, averaging the results across ten independent runs [14]. Moreover, for assessing the recommendation efficacy of RCLUMB and RSCLUMB, we rely on cumulative regret (defined in (2)), which is a widely recognized measure in bandit literature [6], [7], [32]. These evaluations are carried out on computers that are configured with an Intel(R) Xeon(R) Gold 6240 C CPU @ 2.60 GHz and an AMD Ryzen 7 4800H with Radeon Graphics @ 2.90 GHz.

C. Simulation Without Conversational Feedback

1) Data Generation: We consider a setting with $u=1,000$ users and $m=10$ clusters, where each cluster contains 100 users. The preference and feature vectors' dimension is set as $d=50$. The preference and feature vectors are drawn in d dimension with each entry a standard Gaussian variable and then normalized to vectors with $\|.\|_2=1$ [15]. We fix an arm set with $|\mathcal{A}|=1000$ items, at each round t, 20 items are randomly selected to compose a set $\mathcal{A}_t$ for the agent to choose from. We construct a matrix $\boldsymbol{\epsilon} \in \mathbb{R}^{1,000\times 1,000}$ in which each element $\boldsymbol{\epsilon}(i,j)$ is drawn uniformly from the range $(-0.2,0.2)$ to represent the misspecification according to [17]. At round t, for user i_t and the item a_t selected, $\boldsymbol{\epsilon}(i_t,a_t)$ will be added to the feedback as the misspecification, which corresponds to the $\boldsymbol{\epsilon}_{a_t}^{i_t,t}$ defined in (1).

2) Performance Insights: The averaged reward results depicted in Fig. 3 underscore the superior performance of our algorithms compared to established baselines. Specifically, RLinUCB outshines LinUCB, and RLinUCB-Ind outperforms both LinUCB-Ind and CLUB. SCLUB's distinct cluster separation mechanism confers an early advantage by accelerating cluster

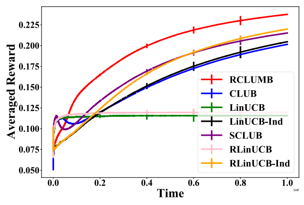


Fig. 3. Averaged reward without conversation feedback.

differentiation. However, its performance is eventually eclipsed by RLinUCB-Ind, which more adeptly accounts for model misspecification. Our proposed RCLUMB outperforms CLUB by 21.9%, LinUCB by 194.8%, LinUCB-Ind by 20.1%, SCLUB by 12.0%, RLinUCB by 185.2%, and RLinUCB-Ind by 10.6%. In general, RSCLUMB performs slightly better than RCLUMB, which is consistent with previous CB works [14].

Attentive readers may observe an initial performance drop in Fig. 3, attributed to the *misclustering* of ζ-close users described in Section V. In our analysis, each user's unique item misspecifications yield varied feedback, even within the same cluster. Initially, this leads algorithms like RCLUMB, SCLUB, and CLUB to mistakenly cluster users based on potentially inaccurate feedback. For example, users with a preference for romantic movies might be grouped with action movie fans, further complicating the clustering. However, as more recommendation data accumulates, these inaccuracies lessen, enhancing the clustering accuracy. Over time, these algorithms refine their user clusters, affecting performance rates; SCLUB initially leads due to its cluster merging strategy. Nevertheless, as algorithms prune non-representative edges, a temporary dip in average rewards occurs until all misleading connections are eliminated. RCLUMB's conservative approach to information sharing softens this performance dip. Initially, RSCLUMB and SCLUB can separate clusters more quickly, benefiting early performance, but ultimately RCLUMB surpasses RSCLUMB, and SCLUB lags due to its inadequate handling of misspecifications. In contrast, RCLUMB's design to counteract these issues shows a significant performance advantage.

We further assess the robustness of RSCLUMB, RCLUMB, RLinUCB-Ind, SCLUB, and CLUB under different levels of model misspecification. We examine the performance of RCLUMB and RSCLUMB by varying the actual misspecification value ϵ_*, testing at 0.05, 0.1, 0.2, 0.3, and 0.4, in scenarios where the degree of misspecification is both known and unknown to the agent. In cases where the misspecification level is known, we align the pre-set misspecification parameter ϵ^* with the actual level and compare our algorithms against other baselines with relatively better performance. In the unknown misspecification scenario, we fix ϵ^* at 0.2 and benchmark our algorithms solely against RLinUCB-Ind, as it is the only baseline with a pre-specified parameter ϵ^*. The outcomes are depicted in Fig. 5, where we plot the final cumulative regret for each algorithm across the various misspecification levels. As anticipated, all algorithms exhibit degraded performance as misspecification increases. Notably, our algorithms consistently outperform the baselines. Moreover, the regret observed in the unknown misspecification cases is marginally higher than that in the known cases. These empirical findings confirm our algorithm's capability to effectively manage unknown levels of misspecification. Furthermore, our algorithm can exhibit similar learning curves across different misspecification levels ϵ^* (please refer to the supplementary material).

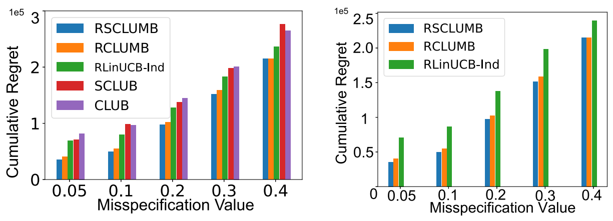
D. Simulation With Conversational Feedback

1) Data Generation: The generation of synthetic data adheres to a process similar to the above described. Our emphasis, therefore, is on elaborating the process of generating key-terms at the conversational level. This approach is following [9], [11], [45], but with potential key-terms subject to misspecification in our adaptation. Specifically, we independently sample each user preference vector $\boldsymbol{\theta}_i$ and each arm feature vector $\boldsymbol{x}_a$ from a standard normal distribution $\mathcal{N}(-1, 1)$, followed by normalization to ensure consistency in scale. The construction of the weight matrix $\boldsymbol{W} \triangleq [w_{a,k}]$ proceeds as follows: For each key-term k, we randomly select an integer n_k from the set $\{1, 2, \ldots, 5\}$, which determines the number of arms associated with key-term k. A corresponding subset of arms $\mathcal{A}_k$ is randomly chosen to represent the arms linked to key-term k. For each arm a associated with n_a key-terms, denoted by $\mathcal{K}_a$, we assign uniform weights such that $w_{a,k} = \frac{1}{n_a}$ for all $k \in \mathcal{K}_a$. This ensures that the influence of each arm is equitably distributed across its associated key-terms. The feature vector for each key-term k is computed as $\tilde{\boldsymbol{x}}_k = \sum_{a \in \mathcal{A}} \frac{w_{a,k}}{\sum_{a' \in \mathcal{A}} w_{a',k}} \boldsymbol{x}_a$, encapsulating the aggregate characteristics of the arms related to key-term k. As our experiments consistently employ the method for constructing conversational key terms as described in [9], [11], [45], we do not include comparisons with approaches like [10] that incorporate additional information sources for conversational feedback. Finally, the feedback at the key-term level is synthesized in alignment with (3), capturing user's response to the key-term based on their preferences and the noise under misspecification level $\epsilon_* = \epsilon^* = 0.25$.

2) Performance Insights: Similar to scenarios without conversational interaction, we calculate the average regret for each algorithm across all users. However, to highlight the effectiveness of the clustering concept among real user groups, we do not artificially specify the number of clusters m, but rather naturally generate user feature preference vectors randomly. As shown in Fig. 7(a), first, all other algorithms outperform LinUCB-Ind, demonstrating the advantage of conversational interaction. ArmCon, which only follows LinUCB-Ind in selecting arms, does not show a significant difference from LinUCB-Ind in large-scale user scenarios. Within our proposed OLCMU framework, the graph-based RCLUMB and the set-based RSCLUMB demonstrate significant advantages over traditional conversational algorithms ConUCB, ConLinUCB-BS, and ConLinUCB-MCR. Specifically, their regrets decrease by 91.5% and 95.3%, respectively, compared to the best-performing ConLinUCB-MCR, showing their effectiveness in dealing with potentially misspecified users. Regarding the impact of exploratory conversational functions and the poolsize of the arm sets, we will provide

DataSet	# of users	# of items
Yelp	1,987,897	62,423
Movie-Lens	162,541	17,632
Last.fm	1,892	150,346

Fig. 4. Statistics of real-world recommendation datasets.



(a) Known actual ϵ_* (b) Unknown actual ϵ_*

Fig. 5. Regret under different misspecification values ϵ^*.

a detailed discussion based on real recommendation system datasets in the following sections.

VII. Experiments

We now turn our attention to evaluating the performance of our algorithm on real-world recommendation datasets. Comprehensive experiments have been designed to assess the effectiveness of our proposed algorithms and to answer the following research questions:

- *RQ1:* How effectively can our algorithms learn and utilize user preferences and relationships to deliver personalized online recommendations with misspecified users?
- *RQ2:* For real-world recommendation datasets, do our algorithms outperform existing state-of-the-art algorithms for better recommendation quality under the CRS?
- *RQ3:* How do the misspecification settings, the various conversation functions, and the arm set ploosize affect the performance of RCLUMB and RSCLUMB?

Fig. 4 provides detailed information about the publicly available real-world datasets used in our article: 1) *Movielens* dataset [46] is sourced from the MovieLens[2] movie recommendation service; 2) *Yelp* dataset [47] is collected from Yelp,[3] a platform where users post reviews for a wide range of businesses, including restaurants and stores; 3) *Last.fm* dataset [48] is gathered from Last.fm,[4] an online music platform. To assess user experience in situations where feedback may be misspecified, we calculate a metric based on the aggregate ratings from the above dataset. Similar to Section VI, we divide the analysis into two parts: one with conversational feedback and one without, and discuss each separately.

[2][Online]. Available: https://movielens.org/
[3][Online]. Available: https://www.yelp.com/dataset
[4][Online]. Available: https://www.last.fm

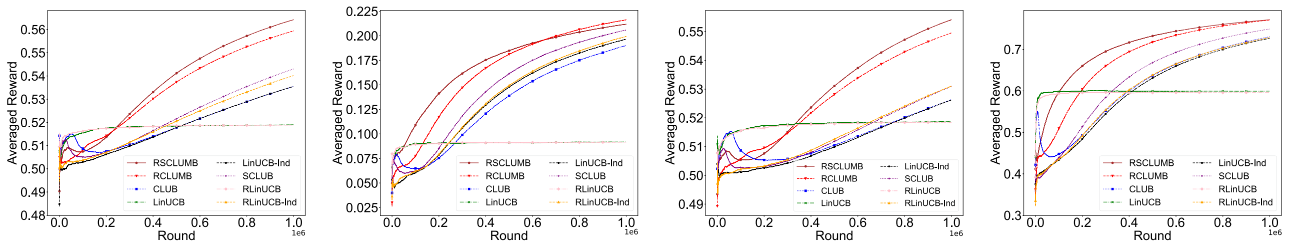
A. Assessment in the Absence of Conversational Feedback

In this section, our investigation is centered on scenarios devoid of conversational feedback, characterized by the condition $b_i(t) \equiv 0, \forall i \in \mathcal{U}$. This setting allows us to isolate and directly assess the performance of RSCLUMU and RCLUMU on clustering users under misspecification models within the context of real-world recommendation datasets. By excluding conversational feedback, we can address *RQ1* more clearly.

1) Dataset Preprocessing: For each dataset, we have two cases due to the different methods for generating feedback. For case 1, we extract 1,000 items with the most ratings and 1,000 users who rate most; then we construct a binary matrix $\boldsymbol{H}^{1,000\times 1,000}$ based on the user rating: if the user rating is greater than 3, the feedback is 1; otherwise, the feedback is 0. Then we use this binary matrix to generate the preference and feature vectors by singular-value decomposition (SVD). Similar to the synthetic experiment, we construct a matrix $\boldsymbol{\epsilon} \in \mathbb{R}^{1,000\times 1,000}$ in which each element is drawn uniformly from the range $(-0.2, 0.2)$. For case 2, we extract 1,100 users who rate the most and 1000 items with the most ratings. We construct a binary feedback matrix $\boldsymbol{H}^{1,100\times 1,000}$ based on the same rule as case 1. Then we select the first 100 rows $\boldsymbol{H}_1^{100\times 1,000}$ to generate the feature vectors by SVD. The remaining 1,000 rows $\boldsymbol{F}^{1,000\times 1,000}$ is used as the feedback matrix, meaning user i will receive $\boldsymbol{F}(i,j)$ as feedback while choosing item j. In both cases, at time t, we randomly select 20 items for the algorithms to choose from. In case 1, the feedback is computed by the preference and feature vector with misspecification $\epsilon_* = 0.2$, in case 2, the feedback is from the feedback matrix.

2) Performance Analysis: Similar to what is shown in Fig. 3, the early-stage performance dip observed in both the Yelp and Movielens datasets, followed by subsequent improvement, represents a trade-off between the effects of misspecification and the gradual accumulation of interaction data. Specifically, the Yelp dataset's results are illustrated in Fig. 6(a) and (b). In the first case, where the misspecification level ϵ_* is known, RCLUMB outperforms CLUB by 45.1%, SCLUB by 53.4%, LinUCB-One by 170.1%, LinUCB-Ind by 46.2%, RLinUCB by 171.0%, and RLinUCB-Ind by 21.5%. In the second case of an unknown misspecification level, with ϵ^* set to 0.2 as per our synthetic dataset, RCLUMB still surpasses other algorithms. Here, RCLUMB's gains over CLUB are 13.9%, SCLUB 5.1%, LinUCB-One 135.6%, LinUCB-Ind 10.1%, RLinUCB 138.6%, and RLinUCB-Ind 8.5%. The robust clustering mechanism of RCLUMB is pivotal in its outperformance, highlighting the importance of addressing misspecification and implementing a strategic clustering framework.

Mirroring the Yelp dataset's configuration, the Movielens dataset results, presented in Fig. 6(c) and (d), show that in the first case, RCLUMB's improvement margins over CLUB are 58.8%, SCLUB 92.1%, LinUCB-One 107.7%, LinUCB-Ind 61.5%, RLinUCB 109.5%, and RLinUCB-Ind 21.3%. In the second case, the improvements are 5.5% over CLUB, 2.9% over SCLUB, 28.5% over LinUCB-One, 6.1% over LinUCB-Ind, 29.3% over RLinUCB, and 5.8% over RLinUCB-Ind. The Movielens dataset corroborates the Yelp dataset's findings, reinforcing the efficacy of the RCLUMB algorithm.



(a) Yelp with known $\epsilon_* = 0.2$ (b) Yelp with unknown ϵ_* (c) Movielens with known $\epsilon_* = 0.2$ (d) Movielens with unknown ϵ_*

Fig. 6. Comparison of RCLUMB and RSCLUMB with baseline algorithms, excluding conversational feedback. Figs (a) and (b) present results from the Yelp dataset, while (c) and (d) feature the Movielens dataset. Experiments involve 1,000 users, 10 clusters, and a 50-dimensional feature space, averaged over 10 random trials. Error bars represent the standard deviation divided by $\sqrt{10}$ for consistent and reliable interpretation. For the setting of the unknown misspecification level, $\epsilon^* = 0.2$.

Additionally, the differing performances of both algorithms under known and unknown true misspecification levels in the Yelp and Movielens datasets address RQ3 regarding the impact of misspecification settings. Moreover, both the graph-based RCLUMB and the set-based RSCLUMB algorithms theoretically show similar regret upper bounds. However, in practical evaluations, the set-based RSCLUMB outperforms due to its more efficient clustering method. Unlike the graph-based RCLUMB, which forms clusters based on connected components and requires cutting all connections between dissimilar users to separate them, the set-based RSCLUMB quickly splits dissimilar users into different clusters, resulting in faster and more effective clustering.

B. Assessment in the Presence of Conversational Feedback

1) Dataset Preprocessing: Key-terms are derived from the MovieLens, Yelp, and Last.fm datasets, indicating movie genres, business categories, or tag IDs, respectively. We select the top 2,000 arms ($|\mathcal{A}| = 2,000$) with the highest frequency of user-assigned tags and the 500 most active users ($N_u = 500$) in terms of tag assignment. For each arm, a maximum of 20 tags associated with the most arms are retained as key-terms, forming the key-term set $\mathcal{K}$, with sizes of 2, 726, 5, 585, and 805 for Last.FM, MovieLens, and Yelp datasets respectively. The weights of all key-terms related to an arm are equal. Following [9], key-term feature vectors are computed as $\tilde{\boldsymbol{x}}_k = \sum_{a\in\mathcal{A}} \frac{w_{a,k}}{\sum_{a'\in\mathcal{A}} w_{a',k}} \boldsymbol{x}_a$. For user feedback, with a probability of 0.5, a misspecification of 0.25 is added, and with a probability of 0.5, it remains unaltered, addressing both known and unknown misspecification levels of two cases in Section VI. Algorithms RCLUMB and RSCLUMB apply a consistent misspecification value ϵ^* of 0.25.

2) Performance Analysis: The results of the average cumulative regret for all algorithms across three real-world recommendation system datasets are depicted in Fig. 7. Notably, even without a mechanism for direct conversation on user preferences, our algorithms significantly outperform existing online user clustering methods, which do not incorporate CRS architecture. For fair comparison and visual clarity, including these methods is deemed unnecessary. It is evident across all datasets that our proposed RCLUMB and RSCLUMB algorithms demonstrate the best performance, highlighting the robustness of our algorithms even in scenarios where both user arms and feedback

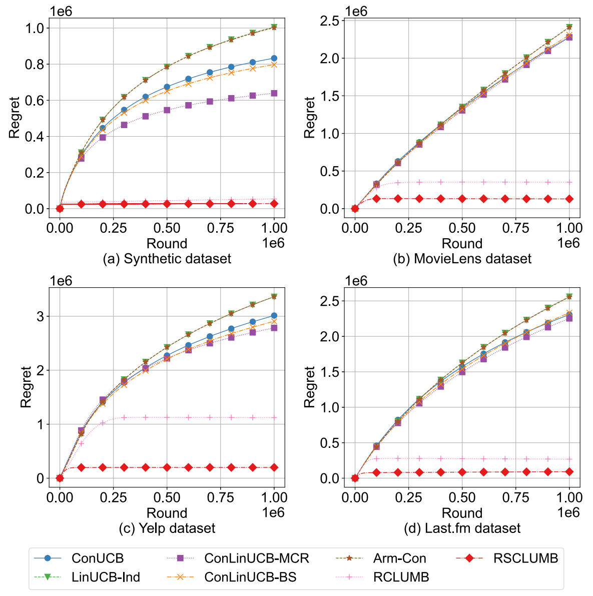


Fig. 7. Cumulative regret in synthetic and real-world datasets with potentially misspecified conversational feedback.

might be misspecified. Taking the Yelp dataset, which includes the largest user base, as an example, compared to the traditionally best-performing conversational algorithm ConLinUCB-MCR, RCLUMB, and RSCLUMB reduced the cumulative regret by 59.7% and 92.8%, respectively. Furthermore, RSCLUMB is 1.4 times more time-consuming than RCLUMB. In terms of performance and time efficiency, our two designed algorithms offer different options. The above analysis on real-world recommendation datasets thus addresses *RQ2* effectively under CRSs.

Next, we assess the impact of the various conversation functions and arm set poolsize as outlined in *RQ3*. To examine the effect of conversation frequency on all algorithms, given the random arrival of users, we apply a uniform conversational function across all algorithms, defined as $\{5\lfloor\log(t)\rfloor, 50\lfloor\log(t)\rfloor, 200\lfloor\log(t)\rfloor, 300\lfloor\log(t)\rfloor, 400\lfloor\log(t)\rfloor\}$ respectively. The cumulative ratings of items recommended to users arriving randomly by round $T = 100,000$ are evaluated, with results averaged over five random trials. A higher $b(t)$ value enables more extensive engagement in conversations.

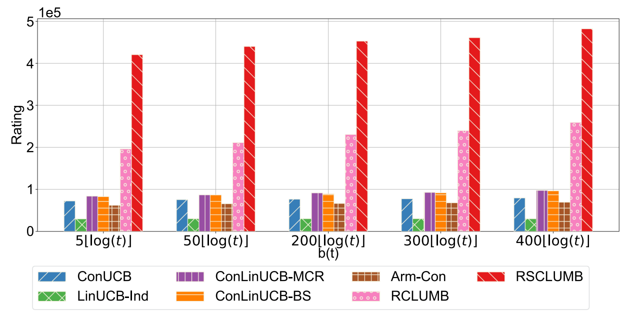


Fig. 8. Ratings with varied conversational functions.

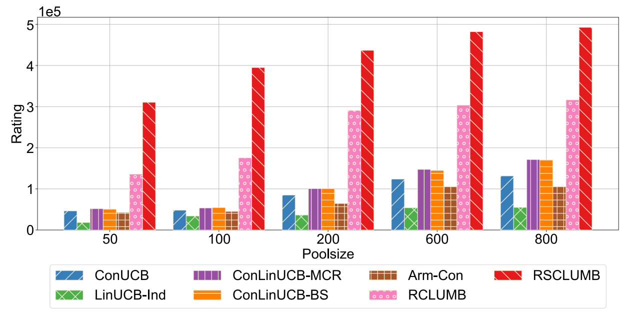


Fig. 9. User rating performance across different arm poolsizes.

The findings in Fig. 8 for the largest dataset Yelp, indicate that as $b(t)$ increases, there is a notable improvement in ratings, affirming the positive influence of more frequent conversations. In every tested scenario, RCLUMB and RSCLUMB surpass ConUCB, Arm-Con, ConLinUCB-MCR, and ConLinUCB-BS in performance, demonstrating their robustness across various conversation frequency functions $b(t)$. This also attests to the effectiveness of their key-term selection strategy π.

We then evaluate the impact of the arm set poolsize on algorithm effectiveness by adjusting the size of $|\mathcal{A}_t|$ to include $\{50, 100, 200, 600, 800\}$ options within the Yelp dataset, which has the largest user base. An increase in $|\mathcal{A}_t|$ generally poses a greater challenge in identifying the optimal arm. However, as demonstrated in Fig. 9, our proposed algorithm, RCLUCB-WCU, exhibits increasingly significant advantages as $|\mathcal{A}_t|$ expands. Although the growth in arm size complicates the selection process, it also offers a broader range of potentially high-reward options, leading to an increase in cumulative ratings with the expansion of the arm poolsize. This observation aligns with intuitive expectations in real-world recommendation systems, where, for example, a music platform with the most extensive copyright portfolio can more likely attract users.

VIII. Conclusion

In this article, we have addressed the challenges posed by the extensive user base and the inherent complexity of accurately discerning user preferences, which often result in persistent recommendation inaccuracies. To mitigate these issues, we introduced two novel algorithms, RCLUMB and RSCLUMB, designed to robustly cluster users with potentially misspecified preferences. These algorithms aim to expedite the recommendation process and enhance the quality of recommendations, even in the face of potential misspecifications in direct user interactions. We establish regret bounds for our algorithms under less restrictive assumptions than previous works. These bounds are shown to be asymptotically optimal in terms of the time horizon T, up to logarithmic factors, and they align with the best-known results in several special cases. Through extensive testing on both synthetic and real-world datasets, we have demonstrated the superior performance of our algorithms. They exhibit remarkable robustness to errors in preference modeling, maintaining accurate clustering and recommendation quality where traditional methods may falter.

References

[1] Q. Li, C. Zhao, T. Yu, J. Wu, and S. Li, "Clustering of conversational bandits with posterior sampling for user preference learning and elicitation," *User Model. User-Adapted Interaction*, vol. 33, pp. 1065–1112, 2023.

[2] S. Li, W. Lei, Q. Wu, X. He, P. Jiang, and T.-S. Chua, "Seamlessly unifying attributes and items: Conversational recommendation for cold-start users," *ACM Trans. Inf. Syst.*, vol. 39, no. 4, pp. 1–29, 2021.

[3] C. Gao, W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Advances and challenges in conversational recommender systems: A survey," *AI Open*, vol. 2, pp. 100–126, 2021.

[4] Z. Li, M. Liu, and J. Lui, "Fedconpe: Efficient federated conversational bandits with heterogeneous clients," 2024, *arXiv:2405.02881*.

[5] L. Li, W. Chu, J. Langford, and R. E. Schapire, "A contextual-bandit approach to personalized news article recommendation," in *Proc. 19th Int. Conf. World Wide Web*, 2010, pp. 661–670.

[6] W. Chu, L. Li, L. Reyzin, and R. Schapire, "Contextual bandits with linear payoff functions," in *Proc. 14th Int. Conf. Artif. Intell. Statist.. JMLR Workshop Conf. Proc.*, 2011, pp. 208–214.

[7] Y. Abbasi-Yadkori, D. Pál, and C. Szepesvári, "Improved algorithms for linear stochastic bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2011, pp. 2312–2320.

[8] N. Hariri, B. Mobasher, and R. Burke, "Context adaptation in interactive recommender systems," in *Proc. 8th ACM Conf. Recommender Syst.*, 2014, pp. 41–48.

[9] X. Zhang, H. Xie, H. Li, and J. C. S. Lui, "Conversational contextual bandit: Algorithm and application," in *Proc. Web Conf.*, 2020, pp. 662–672.

[10] Z. Xie, T. Yu, C. Zhao, and S. Li, "Comparison-based conversational recommender system with relative bandit feedback," in *Proc. 44th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2021, pp. 1400–1409.

[11] Z. Wang, X. Liu, S. Li, and J. C. S. Lui, "Efficient explorative key-term selection strategies for conversational contextual bandits," in *Proc. AAAI Conf. Artif. Intell.*, 2023, pp. 10288–10295.

[12] C. Gentile, S. Li, and G. Zappella, "Online clustering of bandits," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2014, pp. 757–765.

[13] X. Liu, H. Zhao, T. Yu, S. Li, and J. C. S. Lui, "Federated online clustering of bandits," in *Proc. 38th Conf. Uncertainty Artif. Intell.*, 2022, pp. 1221–1231.

[14] S. Li and S. Zhang, "Online clustering of contextual cascading bandits," in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 3554–3561.

[15] S. Li, W. Chen, S. Li, and K.-S. Leung, "Improved algorithm on online clustering of bandits," in *Proc. 28th Int. Joint Conf. Artif. Intell.*, 2019, pp. 2923–2929.

[16] J. Hainmueller and C. Hazlett, "Kernel regularized least squares: Reducing misspecification bias with a flexible and interpretable machine learning approach," *Political Anal.*, vol. 22, no. 2, pp. 143–168, 2014.

[17] A. Ghosh, S. R. Chowdhury, and A. Gopalan, "Misspecified linear bandits," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3761–3767.

[18] S. Li, A. Karatzoglou, and C. Gentile, "Collaborative filtering bandits," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 539–548.

[19] Z. Wang, J. Xie, X. Liu, S. Li, and J. Lui, "Online clustering of bandits with misspecified user models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2024, vol. 36, pp. 3785–3818.

[20] Q. Wu, H. Wang, Q. Gu, and H. Wang, "Contextual bandits in a collaborative environment," in *Proc. 39th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2016, pp. 529–538.

[21] K. Christakopoulou, F. Radlinski, and K. Hofmann, "Towards conversational recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2016, pp. 815–824.
[22] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft, "Towards conversational search and recommendation: System ask, user respond," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 177–186.
[23] K. Christakopoulou, A. Beutel, R. Li, S. Jain, and E. H. Chi, "Q&R: A two-stage approach toward interactive recommendation," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 139–148.
[24] J. Wu, C. Zhao, T. Yu, J. Li, and S. Li, "Clustering of conversational bandits for user preference learning and elicitation," in *Proc. 30th ACM Int. Conf. Inf. Knowl. Manage.*, 2021, pp. 2129–2139.
[25] C. Zhao, T. Yu, Z. Xie, and S. Li, "Knowledge-aware conversational preference elicitation with bandit feedback," in *Proc. ACM Web Conf.* 2022, 2022, pp. 483–492.
[26] Z. Chu, H. Wang, Y. Xiao, B. Long, and L. Wu, "Meta policy learning for cold-start conversational recommendation," in *Proc. 16th ACM Int. Conf. Web Search Data Mining*, 2023, pp. 222–230.
[27] Q. Tu, S. Gao, Y. Li, J. Cui, B. Wang, and R. Yan, "Conversational recommendation via hierarchical information modeling," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 2201–2205.
[28] X. Wang, K. Zhou, J.-R. Wen, and W. X. Zhao, "Towards unified conversational recommender systems via knowledge-enhanced prompt learning," in *Proc. 28th ACM SIGKDD Conf. Knowl. Discov. Data Mining*, 2022, pp. 1929–1937.
[29] G. Zhang, "User-centric conversational recommendation: Adapting the need of user with large language models," in *Proc. 17th ACM Conf. Recommender Syst.*, 2023, pp. 1349–1354.
[30] X. Zhang et al., "Price does matter! modeling price and interest preferences in session-based recommendation," in *Proc. 45th Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2022, pp. 1684–1693.
[31] T. Lattimore, C. Szepesvari, and G. Weisz, "Learning with good feature representations in bandits and in rl with a generative model," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 5662–5670.
[32] T. Lattimore and C. Szepesvári, *Bandit Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2020.
[33] A. Pacchiano et al., "Model selection in contextual stochastic bandit problems," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 10328–10337.
[34] D. J. Foster, C. Gentile, M. Mohri, and J. Zimmert, "Adapting to misspecification in contextual bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2020, pp. 11478–11489.
[35] A. A. Deshmukh, U. Dogan, and C. Scott, "Multi-task learning for contextual bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4848–4856.
[36] L. Cella, K. Lounici, G. Pacreau, and M. Pontil, "Multi-task representation learning with stochastic linear bandits," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2023, pp. 4822–4847.
[37] Z. Wang, C. Zhang, and K. Chaudhuri, "Thompson sampling for robust transfer in multi-task bandits," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2022, pp. 23363–23416.
[38] Z. Wang, C. Zhang, M. K. Singh, L. Riek, and K. Chaudhuri, "Multitask bandit learning through heterogeneous feedback aggregation," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2021, pp. 1531–1539.
[39] R. Wan, L. Ge, and R. Song, "Towards scalable and robust structured bandits: A meta-learning framework," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2023, pp. 1144–1173.
[40] J. Hong, B. Kveton, M. Zaheer, and M. Ghavamzadeh, "Hierarchical Bayesian bandits," in *Proc. Int. Conf. Artif. Intell. Statist.*, PMLR, 2022, pp. 7724–7741.
[41] L. Cella, A. Lazaric, and M. Pontil, "Meta-learning with stochastic linear bandits," in *Proc. Int. Conf. Mach. Learn.*, PMLR, 2020, pp. 1360–1370.
[42] C. Shi and C. Shen, "Federated multi-armed bandits," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 9603–9611.
[43] R. Huang, W. Wu, J. Yang, and C. Shen, "Federated linear contextual bandits," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 27057–27068.
[44] R. Wan, L. Ge, and R. Song, "Metadata-based multi-task bandits with Bayesian hierarchical models," in *Proc. Adv. Neural Inf. Process. Syst.*, 2021, pp. 29655–29668.
[45] W. Lei, X. He, M. de Rijke, and T.-S. Chua, "Conversational recommendation: Formulation, methods, and evaluation," in *Proc. 43rd Int. ACM SIGIR Conf. Res. Develop. Inf. Retrieval*, 2020, pp. 2425–2428.
[46] F. M. Harper and J. A. Konstan, "The MovieLens datasets: History and context," *ACM Trans. Interactive Intell. Syst.*, vol. 5, no. 4, pp. 1–19, 2015.
[47] S. Rayana and L. Akoglu, "Collective opinion spam detection: Bridging review networks and metadata," in *Proc. 21th ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2015, pp. 985–994.
[48] I. Cantador, P. Brusilovsky, and T. Kuflik, "Second workshop on information heterogeneity and fusion in recommender systems (HetRec2011)," in *Proc. 5th ACM Conf. Recommender Syst.*, 2011, pp. 387–388.

Xiangxiang Dai (Graduate Student Member, IEEE) received the BE degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology (HUST), Wuhan, China, in 2023. He is currently working toward the PhD degree with the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2023, advised by Prof. John C. S. Lui. His research interests include reinforcement learning, and bandit theory and their algorithm design for various applications, such as web recommendation systems, video analytics, and computer networks.

Zhiyong Wang received the bachelor's degree from the School of Electronic Information and Communications, Huazhong University of Science and Technology, in 2021. He is currently working toward the PhD degree with the Department of Computer Science and Engineering, Chinese University of Hong Kong, advised by Prof. John C. S. Lui. His research interests include machine learning, reinforcement learning theory, online learning, and bandits.

Jize Xie received the bachelor's degree from the Department of Electrical Engineering, Shanghai Jiao Tong University, in 2019. He is currently working toward the PhD degree with the Department of Industrial Engineering and Decision Analytics, The Hong Kong University of Science and Technology, Hong Kong SAR, from 2023. His research interests include machine learning, operations research, and bandits.

Xutong Liu received the bachelor's degree from the School of Computer Science and Technology, University of Science and Technology of China, in 2017, and the PhD degree from the Department of Computer Science and Engineering, The Chinese University of Hong Kong, in 2022, advised by Prof. John C. S. Lui. He is currently a post-doctoral research fellow with the Department of Computer Science and Engineering, The Chinese University of Hong Kong. His research interests include reinforcement learning, online learning, network science, combinatorial optimization, and stochastic modeling.

John C.S. Lui (Fellow, IEEE) received the PhD degree in computer science from UCLA. He is currently the Choh-Ming Li chair professor with the Department of Computer Science & Engineering (CSE), Chinese University of Hong Kong (CUHK). After his graduation, he joined the IBM Laboratory and participated in research and development projects on file systems and parallel I/O architectures. He later joined the CSE Department, CUHK. His current research interests are in online learning algorithms and applications (e.g., multi-armed bandits, reinforcement learning), machine learning on network sciences and networking systems, large-scale data analytics, network/system security, network economics, large-scale storage systems, and performance evaluation theory. He has served with the IEEE Fellow Review Committees. He is an elected member of the IFIP WG 7.3, fellow of ACM, senior research fellow of the Croucher Foundation, fellow of the Hong Kong Academy of Engineering Sciences (HKAES).