FedConPE: Efficient Federated Conversational Bandits with Heterogeneous Clients

Zhuohua Li, **Maoli Liu** and **John C.S. Lui**
The Chinese University of Hong Kong
{zhli, mlliu, cslui}@cse.cuhk.edu.hk

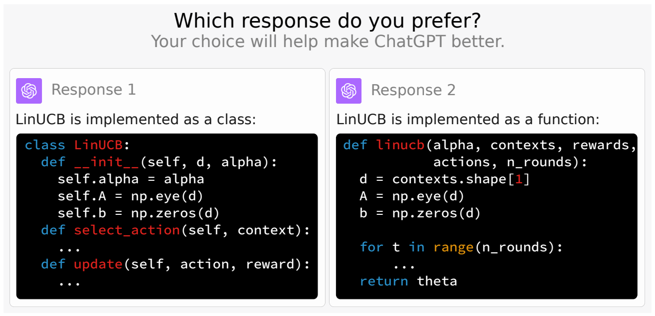
Abstract

Conversational recommender systems have emerged as a potent solution for efficiently eliciting user preferences. These systems interactively present queries associated with "key terms" to users and leverage user feedback to estimate user preferences more efficiently. Nonetheless, most existing algorithms adopt a centralized approach. In this paper, we introduce `FedConPE`, a phase elimination-based federated conversational bandit algorithm, where M agents collaboratively solve a global contextual linear bandit problem with the help of a central server while ensuring secure data management. To effectively coordinate all the clients and aggregate their collected data, `FedConPE` uses an adaptive approach to construct key terms that minimize uncertainty across all dimensions in the feature space. Furthermore, compared with existing federated linear bandit algorithms, `FedConPE` offers improved computational and communication efficiency as well as enhanced privacy protections. Our theoretical analysis shows that `FedConPE` is minimax near-optimal in terms of cumulative regret. We also establish upper bounds for communication costs and conversation frequency. Comprehensive evaluations demonstrate that `FedConPE` outperforms existing conversational bandit algorithms while using fewer conversations.

1 Introduction

In the contemporary digital era, content providers face a growing demand to serve a large number of customers with diverse geographical locations and disparate preferences around the globe. To align content with varied user interests, recommender systems are typically designed to continuously learn and adapt to users' preferences by collecting data from users' feedback. For instance, recommender systems that advertise products or news can collect users' real-time click rates and employ online learning algorithms to refine their recommendations continually. Due to the distributed nature of data and users, large-scale recommender systems usually deploy multiple servers distributed around the world. As a result, *federated learning* is leveraged to localize training on each server and foster collaboration among different servers to enhance learning efficiency while ensuring that sensitive user information is neither transmitted nor centrally stored.

One common challenge that recommender systems encounter is the "cold start" problem, meaning that recommendations may not be accurate for new users with little historical data. A viable solution to this problem is called *active learning*, or *conversational recommendation* [Christakopoulou *et al.*, 2016; Sun and Zhang, 2018; Lei *et al.*, 2020; Gao *et al.*, 2021], where the recommender system can proactively query users some questions and obtain feedback, thereby quickly accumulating data and eliciting user preference. This strategy has been widely used in many real-world applications. For example, the well-known large language model, ChatGPT, occasionally offers two responses to users, asking them to select the one they prefer. To illustrate, consider the real-world example in Figure 1, where ChatGPT is tasked with implementing a specific algorithm in Python. ChatGPT provides two implementations with different coding styles: one is object-oriented, and the other is procedural-oriented. The user is able to choose which one she prefers by clicking one of the two responses. Such interactions help ChatGPT to learn about the user's preferences, enabling it to continually refine its learning process and generate better responses for other, similar users. Note that the feedback is gathered from globally distributed users, necessitating ChatGPT to aggregate the data during the learning process.


Which response do you prefer?
Your choice will help make ChatGPT better.
Response 1
Response 2

LinUCB is implemented as a class:

```
class LinUCB:
  def __init__(self, d, alpha):
    self.alpha = alpha
    self.A = np.eye(d)
    self.b = np.zeros(d)
  def select_action(self, context):
    ...
  def update(self, action, reward):
    ...
```

LinUCB is implemented as a function:

```
def linucb(alpha, contexts, rewards,
           actions, n_rounds):
  d = contexts.shape[1]
  A = np.eye(d)
  b = np.zeros(d)

  for t in range(n_rounds):
    ...
  return theta
```

Figure 1: An example of conversational recommendation: ChatGPT sometimes gives two options for users to choose their preference.

In order to explore and understand user preferences, many recommender systems are equipped with *contextual bandit* algorithms to balance the exploration and exploitation trade-off. A contextual bandit algorithm is a sequential decision-making algorithm, where in each round, it recommends items to the user and receives feedback/rewards (e.g., whether the user clicks on the recommended item or not). The objective of the algorithm is to devise an item recommendation strategy that maximizes the user's feedback over a long period. Recently, *conversational contextual bandit* [Zhang *et al.*, 2020] has been proposed to model *active learning* or *conversational recommendation*. In this setting, besides receiving item-level feedback, the recommender occasionally prompts users with questions about *key-terms* (e.g., movie genres, news topics) of recommended items and obtains key-term level feedback from users. Intuitively, the key-term level feedback reflects the user's preference for a category of items, allowing the recommender to speed up the learning process and eliminate the need to extensively explore all the items.

While conversational recommendation is a robust solution for fast preference elicitation, the current formulation of conversational bandits still has the following problems.

- Firstly, existing works on conversational bandits mostly follow the framework by [Abbasi-Yadkori *et al.*, 2011], which is designed to deal with linear bandit problems with *infinitely many* arms (i.e., the recommended items). In practical recommender systems, however, the number of arms is typically finite.
- Secondly, existing studies about conversational bandits [Zhang *et al.*, 2020; Wang *et al.*, 2023] rely on a deterministic function $b(t)$ to control the frequency of conversations. Moreover, they fix the sequence of item recommendation and conversations, i.e., the recommender can only initiate P conversations *at once* per Q recommendations for some constant P and Q. This does not align with the best practice of conversational recommendation in reality, imposes unnecessary constraints on the recommender, and potentially diminishes the user experience.
- Thirdly, existing literature about conversational bandits solely considers the single-client scenario, neglecting the crucial need for multi-client algorithms in real-world distributed systems. How to effectively coordinate data collected across different clients and appropriately initiate conversations remains an unsolved problem.

To address the challenges mentioned above, this paper considers the federated conversational bandit problem with M clients and *finite* arm sets. We propose FedConPE (Federated Conversational Phase Elimination algorithm), where we follow the classical phase elimination algorithm [Lattimore and Szepesvári, 2020] to handle the finite arm setting and improve the regret upper bound to $\widetilde{\mathcal{O}}(\sqrt{dMT})$. Instead of deterministically pre-defining the conversation frequency, our algorithm adaptively determines whether a conversation is needed according to the collected data. Moreover, it does not mandate the order of conversations and recommendations. Although there are existing works [Huang *et al.*, 2021; Lin and Moothedath, 2023] proposed for federated linear bandit with finite arms, we highlight that integrating the conversational setting is nontrivial and has several advantages. On one hand, the existing algorithm requires all the clients to upload their active arm sets to the central server. This not only leaks sensitive information, such as clients' available choices, to the central server, but also increases the communication costs. On the other hand, the existing algorithm relies on computing the *multi-client G-optimal design* [Huang *et al.*, 2021] by the server, which is known to be computationally prohibitive; thus it is challenging to achieve real-time recommendation. In contrast, for FedConPE, with the help of conversations, each client only needs to locally compute its own *G-optimal design* on its active arm set, which can be efficiently solved by the well-known Frank-Wolfe algorithm under suitable initialization [Frank and Wolfe, 1956; Todd, 2016]. Thus, both computation and communication costs are reduced. We refer interested readers to Appendix H for a more comprehensive overview of related work. To the best of our knowledge, among all the conversational bandit literature, our algorithm is the first to consider the federated setting and the first to give an analysis on lower bounds.

In summary, our contributions are listed as follows.

- We propose FedConPE, an algorithm for federated conversational bandits with finite arm sets, which can facilitate collaboration among all the clients to improve the efficiency of recommendations and conversations.
- We exploit the property of the conversational setting to enhance the efficiency in both computation and communication compared with existing work in federated linear contextual bandits with finite arm sets.
- We show that FedConPE is nearly minimax optimal by theoretically proving the regret upper bound $\widetilde{\mathcal{O}}(\sqrt{dMT})$ and the matching lower bound $\Omega(\sqrt{dMT})$ for conversational bandits. We also prove the communication cost upper bound of $\mathcal{O}(d^2 M \log T)$, and show that conversations only happen for a small number of rounds.
- We conduct extensive experiments on both synthetic and real-world datasets to demonstrate that our algorithm achieves lower cumulative regret and uses fewer conversations than prior works.

2 Problem Formulation

2.1 Federated Conversational Bandits

We define $[M] := \{1, \ldots, M\}$ for $M \in \mathbb{N}^+$. For any real vector $\boldsymbol{x}, \boldsymbol{y}$ and positive semi-definite (PSD) matrix $\boldsymbol{V}$, $\|\boldsymbol{x}\|$ denotes the ℓ_2 norm of $\boldsymbol{x}$, $\langle \boldsymbol{x}, \boldsymbol{y} \rangle = \boldsymbol{x}^\intercal \boldsymbol{y}$ denotes the dot product of vectors, and $\|\boldsymbol{x}\|_{\boldsymbol{V}}$ denotes the Mahalanobis norm $\sqrt{\boldsymbol{x}^\intercal \boldsymbol{V} \boldsymbol{x}}$. We consider the federated conversational multi-armed bandits setting, where there are M clients and a central server. Each client i has a finite set of arms (i.e., recommended items) $\mathcal{A}_i \subset \mathbb{R}^d$ and $|\mathcal{A}_i| = K$. Note that clients are heterogeneous, meaning that different clients may have different arm sets, and we assume that arms in $\mathcal{A}_i$ span $\mathbb{R}^d$. At each time step $t \in [T]$, a client $i \in [M]$ selects an arm $\boldsymbol{a}_{i,t} \in \mathcal{A}_i$ and receives a reward $x_{i,t}$ from the environment. The reward is assumed to have a linear structure: $x_{i,t} = \boldsymbol{a}_{i,t}^\intercal \boldsymbol{\theta}^* + \eta_{i,t}$, where each arm $\boldsymbol{a}_{i,t} \in \mathbb{R}^d$ is represented as a feature vector, $\boldsymbol{\theta}^* \in \mathbb{R}^d$ is an unknown arm-level preference vector that all the clients are trying to learn, and $\eta_{i,t}$ is a noise term. Our objective is to design a learning policy to choose the arms for each time step $t = 1, \dots, T$ such that the cumulative regret, which is the difference of cumulative rewards between our policy and the best policy across all clients, is as small as possible. The cumulative regret is defined as:

$$R_M(T) = \sum_{i=1}^{M} \sum_{t=1}^{T} \left(\max_{\boldsymbol{a} \in \mathcal{A}_i} \boldsymbol{a}^\intercal \boldsymbol{\theta}^* - \boldsymbol{a}_{i,t}^\intercal \boldsymbol{\theta}^* \right). \tag{1}$$

Besides obtaining feedback by pulling arms, the central server can also occasionally query the users of each client and obtain feedback on some conversational *key terms* to accelerate the elicitation of user preferences. In particular, a "key term" refers to a keyword or topic related to a subset of arms. For example, the key term "programming language" may be related to arms such as "C/C++", "Python", "Java", etc. Let $\mathcal{K} \subset \mathbb{R}^d$ denote the finite set of key terms, with each element $\boldsymbol{k} \in \mathcal{K}$ being a fixed and known feature vector. The same as previous works [Zhang *et al.*, 2020; Wang *et al.*, 2023], the server can initiate a conversation with client i by presenting a key term $\boldsymbol{k}_{i,t} \in \mathcal{K}$, and the client's feedback is modeled as $\widetilde{x}_{i,t} = \boldsymbol{k}_{i,t}^\intercal \widetilde{\boldsymbol{\theta}}^* + \widetilde{\eta}_{i,t}$, where $\widetilde{\boldsymbol{\theta}}^*$ is an unknown key-term level preference vector and $\widetilde{\eta}_{i,t}$ is a noise term. It is important to note that in conversational bandit literature, the arm and key-term level preference vectors $\boldsymbol{\theta}^*$ and $\widetilde{\boldsymbol{\theta}}^*$ are either assumed to be very close or exactly the same. Here, we follow the formulation of [Wang *et al.*, 2023] and assume that the two preference vectors are identical.

One significant difference between our formulation and the prior conversational bandit literature is that, previous works depend on a deterministic function $b(t)$ to govern the frequency of conversations, typically defined as a linear or logarithmic function of t. This means that the recommender only periodically launches conversations without considering whether the user preference has been thoroughly estimated, potentially impacting the user experience in practice. In contrast, as illustrated in Section 3, our algorithm adaptively determines whether to initiate conversations based on the accumulated data. If the estimation of the unknown preference vector $\boldsymbol{\theta}^*$ is already sufficiently accurate, the recommender will not prompt conversations to users.

In the following, we list and explain our assumptions.

Assumption 1. We assume the feature vectors for both arms and key terms are normalized, i.e., $\|\boldsymbol{a}\| = \|\boldsymbol{k}\| = 1$ for all $\boldsymbol{a} \in \mathcal{A}_i$ and $\boldsymbol{k} \in \mathcal{K}$. We also assume the unknown preference vector is bounded, i.e., $\|\boldsymbol{\theta}^*\| \leq 1$, and $\eta_{i,t}, \widetilde{\eta}_{i,t}$ are both 1-subgaussian random variables.

Assumption 2. We assume that the elements in the key term set $\mathcal{K}$ are sufficiently rich, such that for any unit vector $\boldsymbol{v} \in \mathbb{R}^d$, there exists a key term $\boldsymbol{k} \in \mathcal{K}$ satisfying $\boldsymbol{k}^\intercal \boldsymbol{v} \geq C$, where $C \in (0, 1]$ is a universal constant that is close to 1.

Assumption 1 aligns with the conventional assumption made in most of the linear bandits literature [Abbasi-Yadkori *et al.*, 2011]. Assumption 2, serving as a technical assumption, ensures that the feature vectors of key terms are sufficiently distributed across the entire feature space, such that they are representative enough to help each client minimize uncertainty during the learning process. This assumption is mild and readily attainable. For example, it is satisfied when the key term set $\mathcal{K}$ contains an orthonormal basis in $\mathbb{R}^d$.

2.2 Communication Model

We adopt a star-shaped communication paradigm, i.e., each client can communicate with the server by uploading and downloading data with zero latency, but clients cannot directly communicate with each other. In addition, we assume that clients and the server are fully synchronized, meaning that at each time t, each client pulls and only pulls one arm. Considering that the regret definition does not take the key-term level feedback into account, without loss of generality, we assume that querying key terms does not increment the time step t, thereby allowing querying a key term and pulling an arm to happen simultaneously. To prevent ambiguity of the notation $\boldsymbol{k}_{i,t}$, we assume that at most one key term is queried per time step t for each client. Consistent with existing works [Huang *et al.*, 2021], we define the communication cost as the number of scalars (either integers or real numbers) transmitted between the server and clients.

3 Algorithms & Theoretical Analysis

3.1 Challenges

Our work is based on the classical single-client phase elimination algorithms for linear bandits (Section 22 in [Lattimore and Szepesvári, 2020]), which we refer to as `LinPE`. In this algorithm, the learner estimates the unknown preference vector $\boldsymbol{\theta}^*$ using the *optimal design* of least squares estimators. Specifically, the learner minimizes the maximum prediction variance by computing the *G-optimal design* $\pi : \mathcal{A} \to [0, 1]$, which is a distribution over the arm set $\mathcal{A}$. As shown in Lemma 1, the G-optimal design π guarantees that the prediction variance $g(\pi) \leq d$. Then the learner plays arms according to the distribution π, estimates the unknown parameter $\boldsymbol{\theta}^*$, and eliminates inferior arms based on the estimation. As a result, the regret of `LinPE` scales as $\widetilde{\mathcal{O}}(\sqrt{dT})$.

Lemma 1 (Kiefer and Wolfowitz)**.** *Assume that $\mathcal{A} \subset \mathbb{R}^d$ is compact and span$(\mathcal{A}) = \mathbb{R}^d$, let $\pi : \mathcal{A} \to [0, 1]$ be a distribution on $\mathcal{A}$ and define $\boldsymbol{V}(\pi) = \sum_{\boldsymbol{a} \in \mathcal{A}} \pi(\boldsymbol{a}) \boldsymbol{a}\boldsymbol{a}^\intercal$, then there exists a minimizer π^* of $g(\pi) = \max_{\boldsymbol{a} \in \mathcal{A}} \|\boldsymbol{a}\|_{\boldsymbol{V}(\pi)}$ such that $g(\pi^*) = d$ and $|Supp(\pi^*)| \leq d(d + 1)/2$.*

To extend `LinPE` algorithm to the federated setting, one straightforward and intuitive approach is to let each client locally run `LinPE`, and the central server aggregates data collected from them to estimate $\boldsymbol{\theta}^*$. Intuitively, with the server now possessing more data, the estimation should be more accurate. However, this simple idea *will not work* because of client heterogeneity, namely, different clients may have different arms sets. Specifically, the G-optimal design computed by one client may not contain useful information for another client, and hence the estimation made by the aggregated data is likely to be biased, meaning that it may be accurate only for a specific client, but not others. As a result, even if the central server aggregates data from all the clients, it may not help improve the regret. One can verify that this intuitive algorithm only results in a regret bound of $\widetilde{\mathcal{O}}(M\sqrt{dT})$, which can be trivially achieved by individually running the `LinPE` algorithm for each client without any communication.

3.2 Intuition of Utilizing Key Terms

As we elaborated above, the main challenge of federated conversational bandits is that aggregating data according to each client's local G-optimal design does not necessarily lead to a better estimation of $\boldsymbol{\theta}^*$ because of client heterogeneity. To mitigate this issue, the central server can leverage key terms to further explore the directions lacking information until the estimation is accurate across all directions in the feature space.

Specifically, the matrix $\boldsymbol{V}(\pi)$ in Lemma 1 (referred to as *information matrix*) contains the "information" of the feature space. Each eigenvector of $\boldsymbol{V}(\pi)$ represents a direction in the feature space, and its associated eigenvalue indicates how much information it encompasses. That is, an eigenvector associated with a larger eigenvalue points towards a direction where $\boldsymbol{V}(\pi)$ contains more information, leading to a more precise estimation in this direction. Therefore, ensuring that all the eigenvalues of $\boldsymbol{V}(\pi)$ are reasonably large guarantees that the model can make accurate predictions with high certainty throughout the entire feature space.

Based on the above intuition, our algorithm enables the server to communicate with each client, identify directions with deficient information, and accordingly select appropriate key terms for each client. Then, the server can aggregate data from all clients and minimize the uncertainty across the entire space. All of the above intuitions are rigorously formulated in our theoretical analysis (see details in Appendix C).

3.3 `FedConPE` Algorithms

The algorithms for the clients and the server are described in Algorithm 1 and Algorithm 2, respectively. In the following, we define $\mathcal{T}_{i,\boldsymbol{a}}^{\ell}$ as the set of time steps during which client i plays arm $\boldsymbol{a}$ in phase ℓ, and define $\mathcal{T}_i^{\ell} = \bigcup_{\boldsymbol{a}\in\mathcal{A}_i^{\ell}} \mathcal{T}_{i,\boldsymbol{a}}^{\ell}$. Similarly, $\widetilde{\mathcal{T}}_{i,\boldsymbol{k}}^{\ell}$ represents the set of times steps during which client i plays key term $\boldsymbol{k}$ in phase ℓ, with $\widetilde{\mathcal{T}}_i^{\ell} = \bigcup_{\boldsymbol{k}\in\mathcal{K}_i^{\ell}} \widetilde{\mathcal{T}}_{i,\boldsymbol{k}}^{\ell}$. We also denote $\lambda_{\boldsymbol{v}}$ as the eigenvalue associated with eigenvector $\boldsymbol{v}$ and define $\varepsilon_\ell = 2^{-\ell}$.

Client-side Algorithm

As shown in Algorithm 1, in each phase ℓ, each client i first computes its G-optimal design π_i^{ℓ}, a distribution among its available arms $\mathcal{A}_i^{\ell}$ (called *active* arms). We note that this can be effectively computed by the well-known Frank-Wolfe algorithm under an appropriate initialization. The main purpose of the G-optimal design is to minimize the maximum prediction variance so as to make the estimation of $\boldsymbol{\theta}^*$ as precise as possible. At line 3, by diagonalizing the information matrix $\boldsymbol{V}_i^{\ell}(\pi_i^{\ell})$, the client can check all directions (represented by eigenvectors) to see if they lack adequate certainty (Line 4), then uploads the eigenvalue-eigenvector pairs to the central server. The server subsequently returns a set of key terms $\mathcal{K}_i^{\ell}$ and the corresponding repetition times $\{n_{\boldsymbol{k}}\}_{\boldsymbol{k}\in\mathcal{K}_i^{\ell}}$ (Line 6). Then client i plays arms and key terms for the required number of times, and shares its progress by uploading the local data to the server (Line 13) so that it can benefit other clients. Note that the order of playing each arm/key term is not important. I.e., within a phase, the client can freely arrange the order as long as ensuring each arm/key term is played the required times. This flexibility affords each client more freedom to change its recommendation strategy as needed. Finally, the client downloads the estimated preference parameter $\widehat{\boldsymbol{\theta}}_\ell$, and updates its active arm set by eliminating inferior arms according to the estimation (Line 15). We emphasize that the sensitive data for each client (e.g., which arm is played and the corresponding reward) is not transmitted to the central server as the client only shares locally aggregated data ($\boldsymbol{G}_i^{\ell}$ and $\boldsymbol{W}_i^{\ell}$).

Algorithm 1: `FedConPE` Algorithm for client i

```
Input: Time horizon $T$, number of clients $M$, dimension $d$,
       number of arms $K$, arm set $\mathcal{A}_i \subseteq \mathbb{R}^d$, constant
       $C \in (0,1]$ in Assumption 2, constant $N > 0$
Initialization: Let $\ell = 1, \mathcal{A}_i^{\ell} = \mathcal{A}_i$
1  while not reaching time horizon $T$ do
2      Compute G-optimal design $\pi_i^{\ell}$ on $\mathcal{A}_i^{\ell}$, such that
           $\sum_{\boldsymbol{a}\in\mathcal{A}_i^{\ell}} \pi_i^{\ell}(\boldsymbol{a}) = 1,\ \boldsymbol{V}_i^{\ell}(\pi) = \sum_{\boldsymbol{a}\in\mathcal{A}_i^{\ell}} \pi(\boldsymbol{a})\boldsymbol{a}\boldsymbol{a}^{\top}$
           $g(\pi_i^{\ell}) = \max_{\boldsymbol{a}\in\mathcal{A}_i^{\ell}} \|\boldsymbol{a}\|^2_{\boldsymbol{V}_i^{\ell}(\pi_i^{\ell})^{-1}} = d$
3      Diagonalize $\boldsymbol{V}_i^{\ell}(\pi_i^{\ell}) = \sum_{j=1}^{d} \lambda_{\boldsymbol{v}_j} \boldsymbol{v}_j \boldsymbol{v}_j^{\top}$
4      foreach $\lambda_{\boldsymbol{v}_j} < \frac{3}{4(1-\varepsilon_\ell^2)dN}$ do
5          Upload $(\lambda_{\boldsymbol{v}_j}, \boldsymbol{v}_j)$
6      Download $\mathcal{K}_i^{\ell}$ and $\{n_{\boldsymbol{k}}\}_{\boldsymbol{k}\in\mathcal{K}_i^{\ell}}$ from server
       // The order of plays is arbitrary
7      foreach $\boldsymbol{a} \in \mathcal{A}_i^{\ell}$ do                ▷ Play arms
8          Pull $\boldsymbol{a}$ for $T_{i,\ell}(\boldsymbol{a}) = \left\lceil \frac{2d\pi_i^{\ell}(\boldsymbol{a})}{\varepsilon_\ell^2} \log \frac{2KM\log T}{\delta} \right\rceil$ times
9          Receive rewards $\{x_{i,t}\}_{t\in\mathcal{T}_{i,\boldsymbol{a}}^{\ell}}$
10     foreach $\boldsymbol{k} \in \mathcal{K}_i^{\ell}$ do                ▷ Play key terms
11         Pull $\boldsymbol{k}$ for $n_{\boldsymbol{k}}$ times
12         Receive rewards $\{\widetilde{x}_{i,t}\}_{t\in\widetilde{\mathcal{T}}_{i,\boldsymbol{k}}^{\ell}}$
13     Upload $\boldsymbol{G}_i^{\ell} = \sum_{\boldsymbol{a}\in\mathcal{A}_i^{\ell}} T_{i,\ell}(\boldsymbol{a})\boldsymbol{a}\boldsymbol{a}^{\top} + \sum_{\boldsymbol{k}\in\mathcal{K}_i^{\ell}} n_{\boldsymbol{k}}\boldsymbol{k}\boldsymbol{k}^{\top}$, and
           $\boldsymbol{W}_i^{\ell} = \sum_{t\in\mathcal{T}_i^{\ell}} \boldsymbol{a}_{i,t}x_{i,t} + \sum_{t\in\widetilde{\mathcal{T}}_i^{\ell}} \boldsymbol{k}_{i,t}\widetilde{x}_{i,t}$
14     Download $\widehat{\boldsymbol{\theta}}_\ell$ from server
15     Eliminate suboptimal arms:
           $\mathcal{A}_i^{\ell+1} = \left\{ \boldsymbol{a} \in \mathcal{A}_i^{\ell} : \max_{\boldsymbol{b}\in\mathcal{A}_i^{\ell}} \left\langle \widehat{\boldsymbol{\theta}}_\ell, \boldsymbol{b} - \boldsymbol{a} \right\rangle \le 2\sqrt{\frac{N}{M}}\varepsilon_\ell \right\}$
16     $\ell = \ell + 1$
```

Server-side Algorithm

As shown in Algorithm 2, the server's role is to find appropriate key terms in $\mathcal{K}$ that can minimize the uncertainty for each client i, and aggregate data from the clients to estimate the preference parameter. Upon receiving the eigenvalues and eigenvectors from each client, the server searches all the key terms and finds the closest one in terms of the inner product (Line 5). Then the selected key term $\boldsymbol{k}$ and the corresponding repetition times $n_{\boldsymbol{k}}$ are sent back to the client (Line 8). Finally, the server aggregates data from all the clients and estimates the unknown preference parameter via linear regression (Line 10 and Line 11). Note that estimating $\widehat{\boldsymbol{\theta}}_\ell$ requires computing matrix inverse. Without loss of generality, we assume that $\boldsymbol{G}$ is always invertible when the associated feature vectors span $\mathbb{R}^d$. When they do not span $\mathbb{R}^d$, we can always reduce the number of dimensions.

Algorithm 2: FedConPE Algorithm for server

```
Input: Time horizon $T$, number of clients $M$, dimension $d$,
       key term set $\mathcal{K} \subseteq \mathbb{R}^d$, constant $C \in (0,1]$ in
       Assumption 2, constant $N > 0$
Initialization: Let $\ell = 1, \boldsymbol{G} = \boldsymbol{0}, \boldsymbol{W} = \boldsymbol{0}$
1  while not reaching time horizon $T$ do
2      foreach $i \in [M]$ do
3          Receive $E_i = \{(\lambda_{\boldsymbol{v}_j}, \boldsymbol{v}_j)\}_j$, let $\mathcal{K}_i^\ell = \emptyset$
4          foreach $(\lambda_{\boldsymbol{v}_j}, \boldsymbol{v}_j) \in E_i$ do
5              $\boldsymbol{k} = \arg\max_{\boldsymbol{v} \in \mathcal{K}} \boldsymbol{v}^\top \boldsymbol{v}_j$
6              $\mathcal{K}_i^\ell = \mathcal{K}_i^\ell \cup (\lambda_{\boldsymbol{v}_j}, \boldsymbol{k})$
7              $n_{\boldsymbol{k}} = \left\lceil \frac{\frac{3}{2(1-\varepsilon_\ell^2)N} - 2d\lambda_{\boldsymbol{v}_j}}{C^2 \varepsilon_\ell^2} \log \frac{2KM \log T}{\delta} \right\rceil$
8          Send $\mathcal{K}_i^\ell$ and $\{n_{\boldsymbol{k}}\}_{\boldsymbol{k} \in \mathcal{K}_i^\ell}$ to client $i$
9          Receive $\boldsymbol{G}_i^\ell$ and $\boldsymbol{W}_i^\ell$
10     $\boldsymbol{G} = \sum_{p \in [\ell]} \sum_{i \in [M]} \boldsymbol{G}_i^p, \ \boldsymbol{W} = \sum_{p \in [\ell]} \sum_{i \in [M]} \boldsymbol{W}_i^p$
11     Broadcast $\widehat{\boldsymbol{\theta}}_\ell = \boldsymbol{G}^{-1}\boldsymbol{W}$ to all clients
12     $\ell = \ell + 1$
```

3.4 Theoretical Analysis

This section presents the theoretical results of FedConPE, including its cumulative regret, communication costs, and the number of conversations. The proofs of Theorem 1, 2, 3, and 4 are given in Appendix C, D, E, and F, respectively.

Theorem 1 (Regret upper bound)**.** *With probability at least* $1-\delta$*, the cumulative regret scales in* $\mathcal{O}\left(\sqrt{dMT \log \frac{KM \log T}{\delta}}\right)$.

Remark 1. When there is only one client (i.e., $M = 1$), our setting reduces to the non-federated conversational bandits and the regret scales in $\widetilde{\mathcal{O}}(\sqrt{dT})$. It improves the result of prior works [Zhang *et al.*, 2020; Wang *et al.*, 2023], which is $\widetilde{\mathcal{O}}(d\sqrt{T})$. The improvement stems from utilizing phase elimination on finite arm sets. Also, our regret bound coincides with that presented by [Huang *et al.*, 2021]. Although FedConPE relies on the conversational setting, it sidesteps the computationally prohibitive *multi-client G-optimal design*, making it easier to achieve real-time recommendations.

Theorem 2 (Regret lower bound)**.** *For any policy that chooses at most one key term at each time step, there exists an instance of the federated conversational bandits such that the policy suffers an expected regret of at least* $\Omega(\sqrt{dMT})$.

Remark 2. Theorem 1 and Theorem 2 imply that FedConPE is minimax optimal up to a logarithmic factor.

Theorem 3 (Communication cost)**.** *With probability at least* $1-\delta$*, the communication cost scales in* $\mathcal{O}(d^2 M \log T)$.

Remark 3. Compared with [Huang *et al.*, 2021], whose communication cost is $O(d^2 MK \log T)$, our result demonstrates an improvement by a factor of K. This is because FedConPE does not require each client to upload its active arm set (whose cardinality is of size $\mathcal{O}(K)$). Instead, each client independently collects data based on its own arms, and only communicates the aggregated data to the server. This not only reduces communication overhead but also enhances privacy preservation by limiting the data shared with the server.

Theorem 4 (Conversation frequency upper bound)**.** *For any client* $i \in [M]$ *and phase* $\ell \in [L]$*, let* $\beta = \lambda_{min}(V_i^\ell(\pi_i^\ell))$*. We have: (a) If* $\beta \geq \frac{3}{4(1-\varepsilon_\ell^2)dN}$*, no conversations will be initiated. (b) Otherwise, the fraction of conversations is at most* $\frac{\frac{3}{4(1-\varepsilon_\ell^2)} - dN\beta}{NC^2}$ *relative to the total phase length.*

Remark 4. Even in the worst case, where $\beta = 0$, the proportion of conversations is at most $\mathcal{O}(1/(NC^2))$. This theorem indicates that FedConPE initiates only a small number of conversations over the time horizon.

4 Performance Evaluation

In this section, we conduct extensive experiments to demonstrate the effectiveness of our algorithm. Specifically, we aim to answer the following research questions:

1. For both the single-client and multi-client settings, does our algorithm FedConPE outperform existing state-of-the-art algorithms for conversational contextual bandits?
2. How do the number of clients M and the arm set size K affect the performance of FedConPE?
3. Does our algorithm use fewer conversations in practice?

4.1 Experimental Settings

Datasets

In light of previous studies, we generate a synthetic dataset and use the following three real-world datasets. The details of data generation and preprocessing are postponed to Appendix G in the extended version of this paper [Li *et al.*, 2024].

- **MovieLens-25M [Harper and Konstan, 2015]**: A dataset from MovieLens, a movie recommendation website. It contains 25,000,095 ratings across 62,423 movies created by 162,541 users.
- **Last.fm [Cantador *et al.*, 2011]**: A dataset from an online music platform Last.fm. It contains 186,479 tag assignments, interlinking 1,892 users with 17,632 artists.
- **Yelp**[1]: A dataset from Yelp, a website where users post reviews for various businesses. It contains 6,990,280 reviews for 150,346 businesses created by 1,987,897 users.

Baseline Algorithms

We select the following algorithms from existing studies as the baselines that we will compare with.

- LinUCB [Li *et al.*, 2010; Abbasi-Yadkori *et al.*, 2011]: The standard linear contextual bandit framework designed for *infinite* arm sets. It does not consider the conversational setting so it only has arm-level feedback.

[1]https://www.yelp.com/dataset

- `ConUCB` [Zhang *et al.*, 2020]: The original algorithm proposed for conversational contextual bandits. It queries key terms when conversations are allowed and leverages conversational feedback on key terms to improve learning speed.
- `Arm-Con` [Christakopoulou *et al.*, 2016]: The algorithm that initiates conversations based on arms instead of key terms. The arm selection follows `LinUCB`.
- `ConLinUCB` [Wang *et al.*, 2023]: A series of algorithms that change the key term selection strategy. It contains 3 algorithms: `ConLinUCB-BS` computes the *barycentric spanner* of key terms as an exploration basis. `ConLinUCB-MCR` selects key terms that have the largest confidence radius. `ConLinUCB-UCB` uses a `LinUCB`-like strategy to choose key terms that have the largest upper confidence bounds.

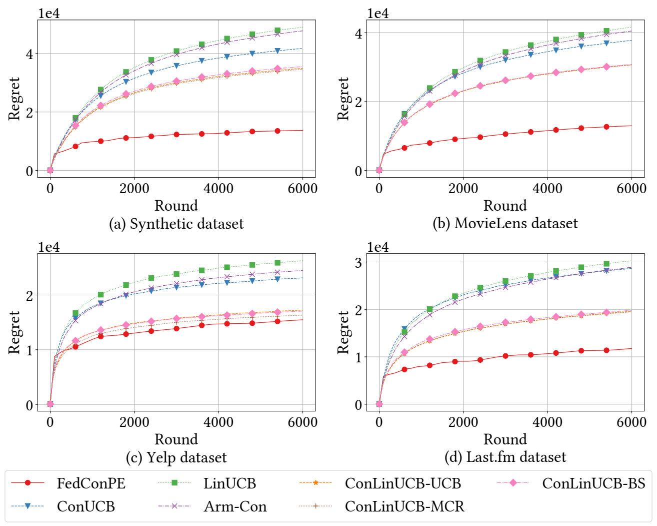


Figure 2: Cumulative regret for the single-client scenario.

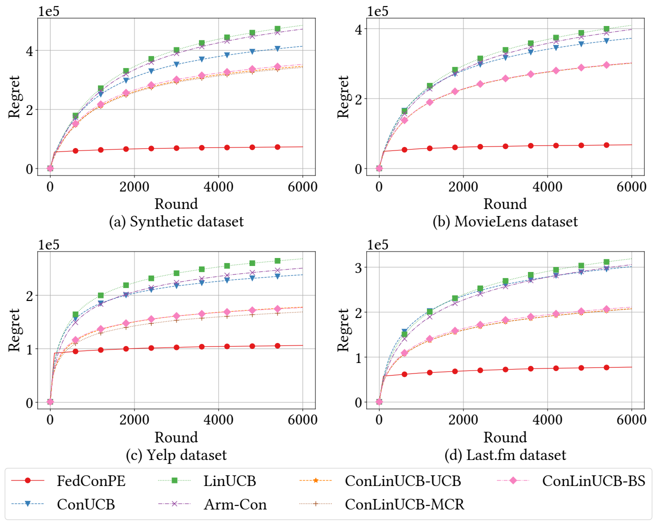


Figure 3: Cumulative regret for the multi-client scenario.

4.2 Evaluation Results

Cumulative Regret for a Single Client

First of all, we evaluate the single-client scenario and compare `FedConPE` with all the six baseline algorithms in terms of cumulative regret. We randomly select 10 users and calculate their cumulative regret over $T = 6,000$ rounds. We set the arm set size $K = 100$ and randomly select K arms from $|\mathcal{A}|$ for the client. For the baseline conversational algorithms `ConUCB`, `Arm-Con`, and `ConLinUCB`, we adopt the conversation frequency function $b(t) = 5\lfloor\log(t)\rfloor$, which adheres to their original papers. The result is shown in Figure 2, where the x-axis is the number of rounds and the y-axis is the cumulative regret. We observe similar results among all the datasets, aligning with the findings from previous studies. That is, all the algorithms exhibit sublinear regrets with respect to the number of rounds. The algorithms without querying key terms (i.e., `LinUCB` and `Arm-Con`) have the poorest performance (largest cumulative regret), while other algorithms that query key terms perform better, thereby showing the importance of conversations about key terms. Our algorithm `FedConPE` *outperforms all of them*, achieving at least 5.25% improvements among all the datasets. This performance improvement substantiates our theoretical results, showing that even under the non-federated scenario (i.e., $M = 1$), our algorithm also achieves lower regret (see discussion in Remark 1).

Cumulative Regret for Multiple Clients

Then, we evaluate the federated (i.e., multi-client) scenario. Since our work is the first to consider the federated setting in conversational bandits, when comparing with the baselines, we simply run the baseline algorithms on each client individually, without any communication with the server. We set the number of clients $M = 10$ and independently select $K = 100$ random arms for each client. Other parameters remain the same as the single-client setting. As depicted in Figure 3, `FedConPE` shows even more advantages compared with the single-client scenario, achieving at least 37.05% improvement over other algorithms. The advantage stems from the federated framework, where the central server aggregates data from each client to better estimate the unknown preference vector.

Cumulative Regret with Different Number of Clients

We further demonstrate the effect of the number of clients by varying it from 3 to 15 with other parameters fixed and compare the cumulative regret at time $T = 6,000$. For illustration purposes, we only present the result for the synthetic dataset, while the results for the real-world datasets, which exhibit similar patterns, are provided in Appendix I.1. Figure 4 depicts that, without communication, the cumulative regrets of all the baseline algorithms increase linearly with the number of clients, i.e., $\widetilde{\mathcal{O}}(dM\sqrt{T})$. In contrast, our algorithm leverages the data collected from all the clients, achieving a regret scaled in $\widetilde{\mathcal{O}}(\sqrt{dMT})$, thereby mitigating the regret increase.

Cumulative Regret with Different Arm Set Sizes

We evaluate the effect of the arm set size K. Figure 5 illustrates the simulations executed for $T = 10,000$ under different arm set sizes, ranging from 100 to 300 on the synthetic dataset. Similar results derived from the real-world datasets are also provided in Appendix I.2. One can observe that, for all the baseline algorithms, there is no substantial amplification of the cumulative regrets as the arm set size K increases. This

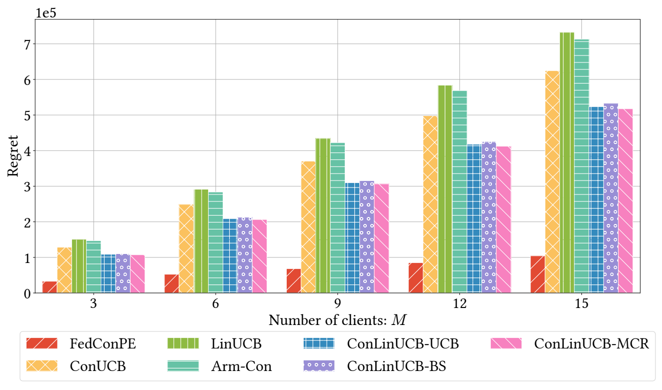


Figure 4: Cumulative regret v.s. number of clients.

result is expected because all the baseline algorithms are based on the `LinUCB` framework, which is intrinsically designed to deal with infinite arm sets. Note that `FedConPE` is also insensitive to the size K. This validates our theoretical results (Theorem 1), where the regret of our algorithm increases only logarithmically with respect to K.

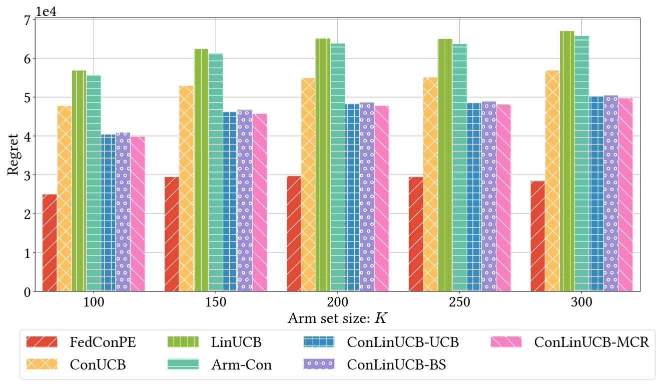


Figure 5: Cumulative regret v.s. size of arm sets.

Accuracy of Estimated Preference Vectors

We evaluate the accuracy of the estimated preference vectors by computing the average ℓ_2-distance between the estimated vector $\widehat{\boldsymbol{\theta}}_{u,t}$ and the ground truth $\boldsymbol{\theta}_u^*$ across 10 randomly selected users u. The number of clients and the arm set size are configured to 5 and 100, respectively. In Figure 6, we exhibit the average difference of all algorithms over the initial 2,000 rounds. Benefiting from the aggregated data, our algorithm `FedConPE` can estimate the preference vector more quickly and accurately than the baseline algorithms.

Number of Conversations

Finally, we measure the number of conversations (i.e., queries of key terms) initiated by each algorithm. Note that all of our baseline conversational algorithms (`ConUCB`, `Arm-Con`, and `ConLinUCB`) launch conversations in a deterministic manner, i.e., a pre-defined function $b(t)$ governs the frequency of conversations. Therefore, we directly calculate the number of conversations launched according to their algorithms and plot the results. Following the original papers, we plot

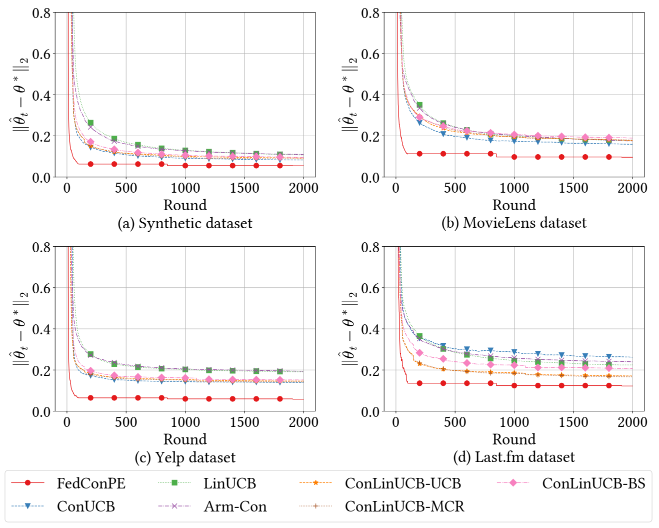


Figure 6: Accuracy of estimated preference vectors.

$b(t) = 5\lfloor\log(t)\rfloor$ and $b(t) = \lfloor t/50 \rfloor$, respectively. We note that although some baselines use a logarithmic $b(t)$ in their experiments, they require a linear $b(t)$ for their proofs. We run `FedConPE` on all the datasets and record the number of key terms selected, with 10 users, and $M = 10$, $K = 100$. As shown in Figure 7, due to the novel design of determining the need for conversations, `FedConPE` launches fewer conversations than other algorithms, offering a better user experience. It is important to note that `FedConPE` also provides enhanced flexibility concerning the order of conversations and recommendations within each phase (see details in Section 3.3).

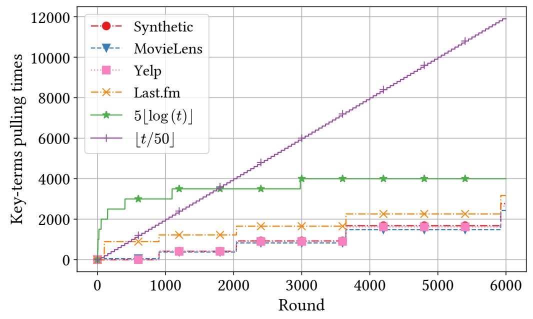


Figure 7: The pulling times of key terms.

5 Conclusion

In this paper, we introduced `FedConPE`, a phase elimination-based algorithm for federated conversational bandits with finite arm sets and heterogeneous clients. It adaptively constructs key terms that minimize uncertainty in the feature space. We proved that `FedConPE` achieves matching regret lower&upper bounds and has reduced communication cost, conversation frequency, and computational complexity. Extensive evaluations showed that our algorithm achieves a lower regret and requires fewer conversations than existing methods.

Acknowledgments

We would like to thank the anonymous reviewers for their constructive comments. The work of John C.S. Lui is supported in part by the RGC SRFS2122-4S02.

Contribution Statement

Zhuohua Li and Maoli Liu contributed equally to this work.

References

[Abbasi-Yadkori *et al.*, 2011] Yasin Abbasi-Yadkori, Dávid Pál, and Csaba Szepesvári. Improved algorithms for linear stochastic bandits. In *Proceedings of the 24th International Conference on Neural Information Processing Systems*, NIPS'11, page 2312–2320, 2011.

[Bretagnolle and Huber, 1978] J. Bretagnolle and C. Huber. Estimation des densités : Risque minimax. In C. Dellacherie, P. A. Meyer, and M. Weil, editors, *Séminaire de Probabilités XII*, pages 342–363, Berlin, Heidelberg, 1978. Springer Berlin Heidelberg.

[Cantador *et al.*, 2011] Ivan Cantador, Peter Brusilovsky, and Tsvi Kuflik. Second workshop on information heterogeneity and fusion in recommender systems (hetrec2011). In *Proceedings of the Fifth ACM Conference on Recommender Systems*, RecSys '11, page 387–388, 2011.

[Chen *et al.*, 2023] Yu-Zhen Janice Chen, Lin Yang, Xuchuang Wang, Xutong Liu, Mohammad Hajiesmaili, John CS Lui, and Don Towsley. On-demand communication for asynchronous multi-agent bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 3903–3930. PMLR, 2023.

[Christakopoulou *et al.*, 2016] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 815–824, 2016.

[Chu *et al.*, 2011] Wei Chu, Lihong Li, Lev Reyzin, and Robert Schapire. Contextual bandits with linear payoff functions. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pages 208–214, 2011.

[Dubey and Pentland, 2020] Abhimanyu Dubey and Alex Pentland. Differentially-private federated linear bandits. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, 2020.

[Frank and Wolfe, 1956] Marguerite Frank and Philip Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3(1-2):95–110, 1956.

[Gao *et al.*, 2021] Chongming Gao, Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. Advances and challenges in conversational recommender systems: A survey. *AI Open*, 2:100–126, 2021.

[Harper and Konstan, 2015] F. Maxwell Harper and Joseph A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4), dec 2015.

[He *et al.*, 2022] Jiafan He, Tianhao Wang, Yifei Min, and Quanquan Gu. A simple and provably efficient algorithm for asynchronous federated contextual linear bandits. In *Advances in Neural Information Processing Systems*, volume 35, pages 4762–4775, 2022.

[Huang *et al.*, 2021] Ruiquan Huang, Weiqiang Wu, Jing Yang, and Cong Shen. Federated linear contextual bandits. *Advances in neural information processing systems*, 34:27057–27068, 2021.

[Kiefer and Wolfowitz, 1960] J. Kiefer and J. Wolfowitz. The equivalence of two extremum problems. *Canadian Journal of Mathematics*, 12:363–366, 1960.

[Lattimore and Szepesvári, 2020] Tor Lattimore and Csaba Szepesvári. *Bandit Algorithms*. Cambridge University Press, 2020.

[Lei *et al.*, 2020] Wenqiang Lei, Xiangnan He, Maarten de Rijke, and Tat-Seng Chua. Conversational recommendation: Formulation, methods, and evaluation. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '20, page 2425–2428, 2020.

[Li and Wang, 2022] Chuanhao Li and Hongning Wang. Asynchronous upper confidence bound algorithms for federated linear bandits. In *International Conference on Artificial Intelligence and Statistics*, pages 6529–6553, 2022.

[Li *et al.*, 2010] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the 19th International Conference on World Wide Web*, WWW '10, page 661–670, 2010.

[Li *et al.*, 2024] Zhuohua Li, Maoli Liu, and John C. S. Lui. Fedconpe: Efficient federated conversational bandits with heterogeneous clients, 2024.

[Lin and Moothedath, 2023] Jiabin Lin and Shana Moothedath. Federated stochastic bandit learning with unobserved context, 2023.

[Liu *et al.*, 2022] Xutong Liu, Haoru Zhao, Tong Yu, Shuai Li, and John CS Lui. Federated online clustering of bandits. In *Uncertainty in Artificial Intelligence*, pages 1221–1231, 2022.

[Sun and Zhang, 2018] Yueming Sun and Yi Zhang. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, page 235–244, 2018.

[Todd, 2016] Michael J. Todd. *Minimum-Volume Ellipsoids*. Society for Industrial and Applied Mathematics, 2016.

[Wang *et al.*, 2020] Yuanhao Wang, Jiachen Hu, Xiaoyu Chen, and Liwei Wang. Distributed bandit learning: Near-optimal regret with efficient communication. In *8th International Conference on Learning Representations*, ICLR'20, 2020.

[Wang *et al.*, 2023] Zhiyong Wang, Xutong Liu, Shuai Li, and John C. S. Lui. Efficient explorative key-term selection strategies for conversational contextual bandits. *Pro-*

ceedings of the AAAI Conference on Artificial Intelligence, 37(8):10288–10295, Jun. 2023.

[Weyl, 1912] Hermann Weyl. Das asymptotische verteilungsgesetz der eigenwerte linearer partieller differentialgleichungen (mit einer anwendung auf die theorie der hohlraumstrahlung). *Mathematische Annalen*, 71(4):441 – 479, 1912.

[Wu *et al.*, 2021] Junda Wu, Canzhe Zhao, Tong Yu, Jingyang Li, and Shuai Li. Clustering of conversational bandits for user preference learning and elicitation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, CIKM '21, page 2129–2139, 2021.

[Xia *et al.*, 2023] Yu Xia, Junda Wu, Tong Yu, Sungchul Kim, Ryan A. Rossi, and Shuai Li. User-regulation deconfounded conversational recommender system with bandit feedback. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, page 2694–2704, 2023.

[Xie *et al.*, 2021] Zhihui Xie, Tong Yu, Canzhe Zhao, and Shuai Li. Comparison-based conversational recommender system with relative bandit feedback. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '21, page 1400–1409, 2021.

[Yang *et al.*, 2024] Hantao Yang, Xutong Liu, Zhiyong Wang, Hong Xie, John C. S. Lui, Defu Lian, and Enhong Chen. Federated contextual cascading bandits with asynchronous communication and heterogeneous users. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(18):20596–20603, Mar. 2024.

[Zhang *et al.*, 2020] Xiaoying Zhang, Hong Xie, Hang Li, and John C.S. Lui. Conversational contextual bandit: Algorithm and application. In *Proceedings of The Web Conference 2020*, WWW '20, page 662–672, 2020.

[Zhao *et al.*, 2022] Canzhe Zhao, Tong Yu, Zhihui Xie, and Shuai Li. Knowledge-aware conversational preference elicitation with bandit feedback. In *Proceedings of the ACM Web Conference 2022*, page 483–492, 2022.

[Zhou and Chowdhury, 2023] Xingyu Zhou and Sayak Ray Chowdhury. On differentially private federated linear contextual bandits, 2023.

[Zuo *et al.*, 2022] Jinhang Zuo, Songwen Hu, Tong Yu, Shuai Li, Handong Zhao, and Carlee Joe-Wong. Hierarchical conversational preference elicitation with bandit feedback. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, CIKM '22, page 2827–2836, 2022.