Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-start Users

SHIJUN LI, University of Science and Technology of China, China
WENQIANG LEI, National University of Singapore, Republic of Singapore
QINGYUN WU, University of Virginia, United States
XIANGNAN HE, University of Science and Technology of China, China
PENG JIANG, Kuaishou Inc., China
TAT-SENG CHUA, National University of Singapore, Republic of Singapore

Static recommendation methods like collaborative filtering suffer from the inherent limitation of performing real-time personalization for cold-start users. Online recommendation, e.g., multi-armed bandit approach, addresses this limitation by interactively exploring user preference online and pursuing the exploration-exploitation (EE) trade-off. However, existing bandit-based methods model recommendation actions homogeneously. Specifically, they only consider the *items* as the arms, being incapable of handling the *item attributes*, which naturally provide interpretable information of user's current demands and can effectively filter out undesired items. In this work, we consider the conversational recommendation for cold-start users, where a system can both ask the attributes from and recommend items to a user interactively. This important scenario was studied in a recent work [54]. However, it employs a hand-crafted function to decide when to ask attributes or make recommendations. Such separate modeling of attributes and items makes the effectiveness of the system highly rely on the choice of the hand-crafted function, thus introducing fragility to the system. To address this limitation, we seamlessly unify attributes and items in the same arm space and achieve their EE trade-offs automatically using the framework of Thompson Sampling. Our *Conversational Thompson Sampling* (ConTS) model holistically solves all questions in conversational recommendation by choosing the arm with the maximal reward to play. Extensive experiments on three benchmark datasets show that ConTS outperforms the state-of-the-art methods *Conversational UCB* (ConUCB) [54] and *Estimation—Action—Reflection* model [27] in both metrics of success rate and average number of conversation turns.

CCS Concepts: • **Computing methodologies** → **Artificial intelligence**; **Machine learning algorithms**; • **Human-centered computing** → *Collaborative and social computing design and evaluation methods;*

Additional Key Words and Phrases: Conversational recommendation, interactive recommendation, recommender system, dialogue system

S. Li and W. Lei contributed equally to this research.

This work is supported by the National Natural Science Foundation of China (Grant No. 61972372) and the National Key Research and Development Program of China (Grant No. 2020AAA0106000).

Authors' addresses: S. Li and X. He, University of Scinece and Technology of China, 443 Huangshan Road, He Fei, China, 230027; emails: lishijun@mail.ustc.edu.cn, hexn@ustc.edu.cn; W. Lei (corresponding author) and T.-S. Chua, National University of Singapore, 13 Computing Drive, National University of Singapore, Singapore, Republic of Singapore, 117417; emails: wenqianglei@gmail.com, dcscts@nus.edu.sg; Q. Wu, University of Virginia, Computer Science 85 Engeers.s way, Charlottesville, VA, United States, 22903; email: qw2ky@virginia.edu; P. Jiang, Kuaishou Inc., 6 West Shangdi Road, Haidian District, Beijing, China; email: jp2006@139.com.

1046-8188/2021/08-ART40 $15.00
https://doi.org/10.1145/3446427

ACM Reference format:
Shijun Li, Wenqiang Lei, Qingyun Wu, Xiangnan He, Peng Jiang, and Tat-Seng Chua. 2021. Seamlessly Unifying Attributes and Items: Conversational Recommendation for Cold-start Users. *ACM Trans. Inf. Syst.* 39, 4, Article 40 (August 2021), 29 pages.
https://doi.org/10.1145/3446427

1 INTRODUCTION

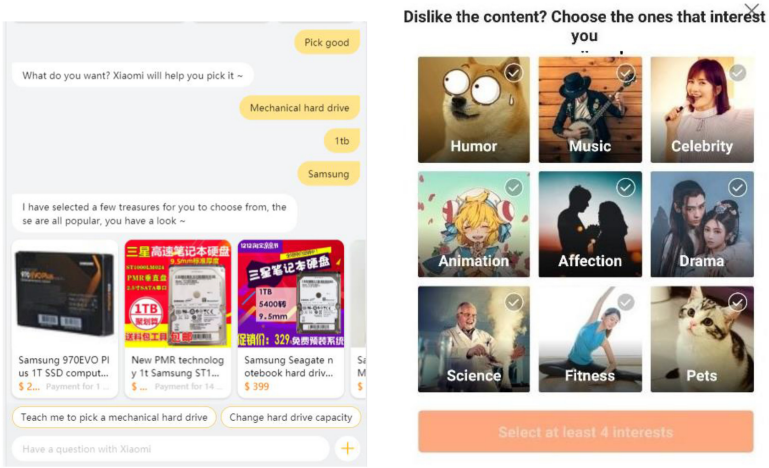
Recommendation system plays an increasingly important role in the current era of information explosion. In the past decades, most recommendation research has focused on advancing modeling historical user behaviours, proposing various methods like traditional collaborative filtering [20, 39], content- and context-aware filtering [7, 38], to the recently prevalent neural network methods [9, 19] and graph-based methods [18, 48]. We call such methods *static recommendation*, as they learn static user preference from past interaction history, which inevitably requires an adequate amount of past behaviour histories for each user. However, the dependency of historical data makes such methods suffer from the cold-start scenario where new users come with no past historical data.

Cold-start users typically interact with the system very shortly but expect to receive high-quality recommendations. Moreover, their preferences might dynamically change over time, making the recommendation even harder. The recent emerging application and technology of ***conversational recommendation system*** **(CRS)** bring a promising solution to the cold-start problem. A CRS is envisioned as an interactive system that can ask the user preference toward item attributes before making the recommendations. With preferred attributes known, the item pool can be significantly reduced [44] and the candidate items can be better scored [27], leading to more desired recommendations. Owing to this great potential, conversational recommendation becomes an emerging topic that attracts extensive research attention recently [8, 10, 11, 35, 42, 44, 52, 54, 55].

On the industry side, we have also witnessed an increasing trend on exploring the technology of CRS in various scenarios. A classical scenario is the customer service chatbot in E-commerce, which can help users find their interested products by dialogues. Figure 1(a) shows the interface of the chatbot in Taobao:[1] when a user comes into the system, the bot can ask questions to elicit user preferences such as asking the user preferred attribute (e.g., *brand* and *categories*); then the bot makes recommendations according to the user's responses.

In addition to the chatbot, many scenarios that interact with users to facilitate information seeking can be abstracted as the problem of CRS. Taking Kuaishou app[2] as an example, when the user browses videos, the app will pop up a box to ask the user's favorite attributes of the videos (see Figure 1(b)). This can be formalized as a conversational recommendation process [27, 43, 45, 49, 55], which is characterized as form-based conversational recommendation system in Reference [22]. Popping up attribute box can be formalized as "asking what attributes a user likes" in the conversational recommendation while displaying videos can be formalized as making recommendation actions. Such mechanism has demonstrated to be effective hence been widely adopted in many applications like YouTube,[3] Spotify,[4] Steam.[5]

[1]The largest Chinese E-commerce platform: https://consumerservice.taobao.com/.
[2]A famous Chinese video-sharing platform: https://www.kuaishou.com/.
[3]A famous online video-sharing platform: https://www.youtube.com/.
[4]An international media services provider: https://www.spotify.com/us/.
[5]The largest game platform: https://store.steampowered.com/.



(a) Taobao (b) Kuaishou

Fig. 1. The chatbot interface in Taobao (a) and the pop-up box of attribute asking in Kuaishou (b).

As an emerging topic, conversational recommendation has been explored under various settings with different focuses [8, 10, 11, 35, 42, 44, 52, 54, 55]. Focusing on cold-start users, the first attempt is made by Reference [11], which identified the key of cold-start conversational recommendation as pursuing the **exploration-exploitation (EE)** trade-off. It addresses the problem with the multi-arm bandit approach [29], modeling items as the arms to achieve the EE trade-off. However, this method is inefficient by soliciting user preference only of the level of items, since the number of items can be million or even billion in real-world applications. To address this issue, a recent work [54] integrates the attribute asking into the system. It models attributes and items as two different types of arms, using a manually defined function to determine the timing of attribute asking (e.g., asking an attribute per three turns). We argue that a simple heuristic function is insufficient to find the best timing of attribute asking, which depends on many ad hoc factors like the number of current turn and user's feedback. The strategy of attribute asking is critical for the usability and effectiveness of a CRS, which aims to identify the desired item(s) with the fewest turns, to avoid bothering the user with too many questions. However, it has not been seriously considered in conversational recommendation with EE trade-off.

In this work, we explore the central theme of conversational recommendation for cold-start users. We highlight two desired properties for such a CRS: (1) deciding the strategy of asking and recommending in an intelligent way, and (2) keeping the balance between exploiting known preferences and exploring new interests to achieve successful recommendations. To this end, we propose the ***Conversational Thompson Sampling* (ConTS)** method, modeling attribute asking, and item recommending in the unified framework of Thompson Sampling. The key idea is to model attributes and items as undifferentiated arms in the same arm space, where the reward estimation for each arm is jointly decided by the user's representation, candidate items, and candidate attributes to capture the mutual promotion of attributes and items. Through this way, we seamlessly unify attributes and items in conversational recommendation, leading to two main advantages of ConTS: (1) it addresses *conversation policy questions* in CRS proposed in Reference [27]—what items to recommend, what attributes to ask, and whether to ask or recommend in a turn—as the single problem of arm choosing, which is optimized for maximized rewards, and (2) it inherits the sampling and updating mechanism of contextual Thompson Sampling [1], being able to achieve the EE balance naturally.

The main contributions of this article are as follows:

- We study the new task of conversational recommendation for cold-start users with attribute preference. By modeling attributes and items as indiscriminate arms in the same space, we propose a holistic solution named ConTS to solve the three conversation policy questions in CRS in an end-to-end manner. Meanwhile, we apply contextual Thompson Sampling to conversational recommendation to keep an EE balance in the cold-start scenario.
- We conduct experiments on two existing datasets from Yelp and LastFM, and contribute a new dataset for cold-start conversational recommendation evaluation from Kuaishou (video-click records). Extensive experiments show that our ConTS outperforms the state-of-the-art CRS methods ***Conversational UCB* (ConUCB)** [54] and ***Estimation—Action—Reflection* (EAR)** [27] in both metrics of success rate and average turn for cold-start users. Further analysis shows the importance of exploration and the effectiveness of ConTS's action policy. All codes and data will be released to facilitate the research community studying CRS.[6]

2 RELATED WORKS

In the field of recommendation system, many static recommendation methods have achieved good performance. For example, matrix factorization [25], factorization machines [38], NeuralFM [17], and DeepFM [16] make use of historical user-item interaction records to estimate users' static preference based on collaborative filtering hypothesis. Recently, some graph-based recommendation models [46, 48] attracts much attention for the ability of graph to describe complex relations between different agents. However, most of these works suffer from intrinsic problems for cold-start users as they need to be trained on a large offline dataset to make recommendation based on historical logs.

This limitation motivates research efforts in online recommendation in Reference [30]. Among them, muti-arm bandit [3, 13, 29, 33, 47, 50, 51] is a prominent type of approach. Bandit methods aim to maximize the accumulated reward for playing arms during a period of time by solving the exploit-explore problem. The most popular bandit algorithms can be divided into two categories: Thompson Sampling [1, 14, 15, 31, 33] and UCB [3, 26, 29]. Different from Thompson Sampling, which samples from a changeable distribution, UCB keeps an upper confidence bound to keep EE balance. Some research works [6, 34, 40] have shown that Thompson Sampling has a better performance than UCB both theoretically and empirically. However, bandit algorithms only work in small arm pools, which usually requires a separate pre-processing on the candidate pool [6, 29]. It is because the large item pool makes it hard to efficiently hit the user-preferred item duration exploration.

Conversational recommendation systems (CRS) [8, 10, 11, 35, 42, 44, 52, 54, 55] provide a promising solution. To enhance the performance of recommendation system with a large item pool, conversational recommendation ask questions to directly acquire the user's preference. Such information can help a CRS to adjust its recommendation strategy effectively and enhance its performance. Extensive research efforts have been recently devoted to explore this topic spreading various task formulations. Jannach et al. [22] classify CRS into two categories: NLP-based and form-based models. CRSs based on natural language [23, 24, 28, 37] are dynamic in terms of the possible dialogue flow. Works in this direction focus on how to understand users' preferences and intentions from their utterances in various forms and generate fluent responses to deliver natural and effective dialogues [23, 24, 28, 37]. In form-based CRS [5, 12, 21, 32], users typically follow pre-defined dialogue paths and interact with the application by filling forms through pre-defined

[6] https://github.com/xiwenchao/conTS-TOIS-2021.

options. These models are based on forms (e.g., buttons, radio buttons) and structured text for the output (e.g., in the form of a list of options). The example given in Section 1 of the Kuaishou app also belongs to this category. It requires the user to choose his favorite attributes listed in the box, which is an option pre-defined by the system. However, most of the models are designed for existing users instead of considering cold-start users with few historical data. A typical example is the EAR [27], which is trained on existing user's interaction records to estimate the preference of these users, thus fail to handle new users without historical information.

There are pioneer works studying conversational recommendation problems for cold-start users. They basically leverage bandit methods to achieve EE balance. Christakopoulou et al. [11] systematically compare several bandit methods (i.e., Thompson Sampling, UCB, Greedy, Max Item Trait, and Min Item Trait) and find that Thompson Sampling achieves the best performance. However, the resultant CRS is only able to recommend items to users but not able to ask question about the attributes. We argue that it is more efficient to explore the item space with the information of item attributes. In many real-world applications, the number of items could be very large-scale, e.g., there are over billions of products (videos) in Alibaba (YouTube). Soliciting user preference in the level of *item attribute* is far more efficient than that at the level of *item*; thus, if we know what attributes are preferred by the user now, then we can safely filter out the items that do not contain the preferred attributes and facilitate efficient product searching. For example, if a user specifies that he wants to watch *Science Fiction* movie, we can exclude other categories of movies for the consideration.

A very recent work by Zhang et al. [54] further considers attribute asking in the framework of UCB [3, 26, 29] for conversational recommendation, which helps to keep EE balance when recommending for cold-start users. It models the attributes and items separately as two set of arms in different arm spaces and decides whether to ask or recommend in a totally handcrafted way. Specifically, the authors define a frequency function $b(t) = 5 * \lfloor \log(t) \rfloor$, where t denotes the current conversation turn. ConUCB asks attributes only if $b(t) - b(t-1) > 0$. However, we argue such handcrafted way is not robust. In contrast, we model attributes and items as undifferentiated arms in the same arm space, and nicely fit the arm choosing in the framework of contextual Thompson Sampling. As such, all the three *conversation policy questions* (e.g., what item to recommend, what attribute to ask, and whether to ask attribute or to recommend) are seamlessly modeled as the single problem of arm choosing. This results in a holistic model that is more robust and efficient than ConUCB.

3 PRELIMINARY

This article studies conversational recommendation for cold-start users. We choose the **multi-round conversational recommendation (MCR)** scenario [27] as the problem setting, because it is more reflective of real-world scenario [27]. It is worth to mention that the MCR is a type form-based [22] setting, which means users interact with the application by choosing from pre-defined options instead of dynamically generated natural languages. In this section, we first introduce the problem setting, and then sketch Thompson Sampling, since our method is inspired from it. For the ease of understanding, the complex mathematics are skipped. We refer the reader to References [1, 14] for more details. We give an intuitive illustration about how Thompson Sampling works in Section 7 (Supplementary).

3.1 Problem Setting

In MCR, the system can perform attribute asking and item recommending multiple times until the recommendation is accepted or a maximum number of turns is reached. The objective is to achieve successful recommendations with the fewest conversation turns for each cold-start user

Table 1. A Summary of Main Notations Used in the Article

Notation	Implication
$u, \mathcal{U}$	User and collection of all users
$v, \mathcal{V}$	Item and collection of all items
$p, \mathcal{P}$	Attribute and collection of all attributes
$\mathbf{u}, \mathbf{v}, \mathbf{p}$	Embedding of user, item and attribute
$a, \mathbf{x}_a$	Arm and embedding of arm a in Bandit algorithms
r_a	Reward gotten from the user on arm a
$\mathcal{A}$	Collection of all arms
$\mathbf{B}_u, \boldsymbol{f}_u, \boldsymbol{\mu}_u, l$	Parameters for user u in contextual Thompson Sampling
$\mathcal{P}_u$	Currently known user u's preferred attributes
$\mathcal{P}_v$	Item v' attributes
d	Dimension of embedding of user, item and attribute
k	The number of recommended items in one turn
T	The max turn of a conversation in a CRS
$\mathcal{V}_k$	Top k items in the candidate item list
$\mathbf{u}_{init}$	Initialization of user embedding
$\mathbf{u}_{orig}$	User embedding in original contextual Thompson Sampling
r'_a	The de-biased reward in ConTS
$I_{training}, I_{testing}, I_{validation}$	The set for training, testing and validation

u who has no past historical interaction records. Let $\mathcal{V}$ denotes all the items and $\mathcal{P}$ denotes all item attributes. Each item v is described by a set of categorical attributes $\mathcal{P}_v$, e.g., the attributes of restaurants may include location, category, price level, and so on. In each turn t ($t = 1, 2, \ldots, T$, where T is the maximum number of turns) in a session, the system maintains two candidate pools, one for items $\mathcal{V}_t \subseteq \mathcal{V}$ and another for attributes $\mathcal{P}_t \subseteq \mathcal{P}$, from which we select attribute(s) to ask and top-k items to recommend, respectively. Note that if it exceeds the maximum conversation turn T, we assume that the user will quit the conversation due to limited patience. In the initial turn ($t = 1$), the candidate pools contain all the items and attributes, which are gradually reduced with the proceeding of the conversation process.

With $\mathcal{V}_t$ and $\mathcal{P}_t$ established in the beginning of turn t, the system takes an action to interact with the user—either asking for attribute(s) or recommending k items. Then the user will give feedback to the system, which can either be (1) like the attribute(s), (2) dislike the attribute(s),[7] (3) accept the recommendations, or (4) reject the recommendations. Only if the user accepts the recommendations, the session ends successfully; in other cases, the session continues and the system needs to adjust the candidate pools, and possibly, the action policy. In terms of the policy for making actions, there are three questions to consider: (1) what attributes to ask, (2) what items to recommend, and (3) whether to ask or recommend in a turn. Instead of addressing each question with a separate component [27, 44], our method seamlessly addresses them in a holistic framework.

3.2 Thompson Sampling

As our method is inspired by Thompson Sampling, we here briefly introduce it. Thompson Sampling [1, 14, 15, 31, 33] is a typical class of bandit algorithms [41], which balance the exploration and exploitation during online learning. Here, the exploitation means taking actions according to the model's current estimation of the user's interest, and exploration means exploring a user's

[7]More comprehensive than Reference [27], here we explicitly model user negative feedback on attributes.

unknown preference to pursue a possibly higher reward in the future. Differently from bandit algorithms that use the **upper confidence bound (UCB)** [4, 29] to model the uncertainty of the learner's estimation on a user's preference, Thompson Sampling balances EE by sampling from a posterior distribution of the reward estimation, which is shown to be a more effective and robust method in many situations according to some previous research works [6, 40].

In the typical setting of contextual bandit problem, there is an arm pool $\mathcal{A}$ and a set of contexts $\mathcal{X}$. In an interaction turn, the model needs to select an arm $a \subseteq \mathcal{A}$ to play given a context $\mathbf{x} \subseteq \mathcal{X}$, and then gets a reward r_a from the users accordingly. The goal of the arm choosing is to maximize the accumulated reward given the limited interaction turns. In the interactive recommendation scenario, an arm a can represent an item and the context $\mathbf{x}$ can represent the accessible information for the recommendation decision-making, such as the arm (i.e., item) embeddings. In a turn, the recommender chooses an item to display to the user (i.e., play an arm) and gets user feedback as the reward, either by accepting (positive reward) or rejecting (negative reward). In the linear contextual bandit setting, the reward r_a for an arm a under context $\mathbf{x}_a$ is assumed to be generated from an (unknown) distribution with mean $\mathbf{u}^\mathsf{T}\mathbf{x}_a$, where $\mathbf{u} \in \mathbf{R}^d$ is a fixed but unknown parameter. In the recommendation scenario, this unknown variable $\mathbf{u}$ can be considered as an embedding that reflects a users' preference.

In Thompson sampling, with a prior distribution $P(\mathbf{u})$ on the unknown parameter $\mathbf{u}$ and a collection of observed triples $\mathcal{D}_t = \{(\mathbf{x}_1; a_1; r_{a(1)}), (\mathbf{x}_2; a_2; r_{a(2)}), \ldots (\mathbf{x}_{t-1}; a_{t-1}; r_{a(t-1)})\}$ in the current session at time t, a posterior estimation $P(\mathbf{u})$ of the user's preference can be derived according to the Bayes rule:

$$P(\mathbf{u}|\mathcal{D}_t) \propto P(\mathcal{D}_t|\mathbf{u})P(\mathbf{u}), \tag{1}$$

where $P(\mathcal{D}_t|\mathbf{u})$ is the likelihood function.

Thompson sampling samples $\tilde{\mathbf{u}}$ from $P(\mathbf{u}|\mathcal{D}_t)$ to get instantiated parameters in the turn, based on which, the reward for each arm a is calculated as $\tilde{r}_a = \tilde{\mathbf{u}}^\mathsf{T}\mathbf{x}_a$. An arm is chosen to play according to the reward calculated from the sampled parameter $\arg\max_a \tilde{\mathbf{u}}^\mathsf{T}\mathbf{x}_a$. After getting the user's feedback, we have a new triple added into $\mathcal{D}_t$. In the process, sampling and updating are the key mechanisms to ensure the EE balance. We next use an example to illustrate it.

A commonly used likelihood function $P(\mathcal{D}_t|\mathbf{u})$ and prior $P(\mathbf{u})$ are Gaussian likelihood function and Gaussian prior [1], respectively. Contextual Thompson Sampling [1] naturally assumes the embedding for user u also follows a Gaussian distribution, which can be written as $\mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$. The mean $\boldsymbol{\mu}_u$ denotes the estimated expectation of user embedding $\mathbf{u}$ and the covariance matrix $l^2\mathbf{B}_u^{-1}$ denotes the uncertainty about the embedding (note that $\mathbf{B}_u$ is the inverse covariance matrix), and l is a hyper-parameter to control the uncertainty. After the user gives the feedback (reward) r_a on arm a, the model updates the parameters as

$$\mathbf{B}_u = \mathbf{B}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^T, \tag{2}$$

$$\boldsymbol{f}_u = \boldsymbol{f}_u + r_a * \mathbf{x}_{a(t)}, \tag{3}$$

$$\boldsymbol{\mu}_u = \mathbf{B}_u^{-1}\boldsymbol{f}_u, \tag{4}$$

where $\mathbf{x}_{a(t)}$ is the embedding of the chosen arm $a(t)$ at turn t, $\mathbf{B}_u$ is initialized as the identity matrix and $\boldsymbol{f}_u$ as a zero vector. Then contextual Thompson Sampling will sample from the posterior distribution $\mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$ to get an estimation of user embedding $\mathbf{u}$ under uncertainty.

It will keep updating $\mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$ during the interaction process. On the one hand, this posterior update incorporates past experience into the reward estimation; on the other hand, it reflects the model's uncertainty on arms with different context. For example, if the user gives a negative feedback to an arm a, then contextual Thompson Sampling tends to score lower for a and other arms of the similar embedding with a by updating the mean $\boldsymbol{\mu}_u$. And whenever the user

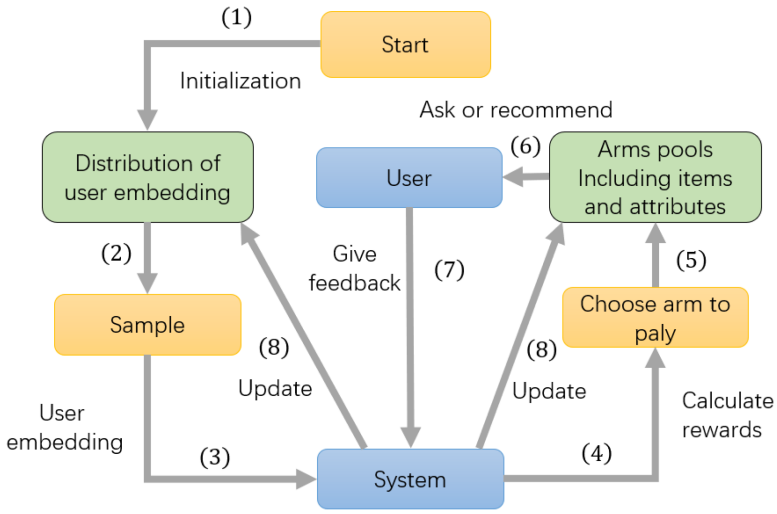


Fig. 2. The example of ConTS' actions during a whole conversational recommendation process for cold-start users. Orange blocks represent operations, green blocks represent parameters, and blue blocks resemble agents in the scenario.

gives feedback to an arm, contextual Thompson Sampling will update $\mathbf{B}_u$ to discourage further exploration of this arm, because we have already known the user's preference on it. More details about the strategy of update is given in Section 7.

4 METHOD

As discussed in Section 1, the key challenges in conversational recommendation for cold-start scenario lie on two aspects: (1) the strategy to decide whether to ask or recommend and (2) the balance of EE. To tackle the two challenges, we propose to unify the attributes and items seamlessly into the one arm space, resulting in our ConTS model. Specifically, it comprises three parts, namely, (1) Initialization and Sampling, (2) Arm Choosing, and (3) Updating. We will detail them after a model overview in this section.

4.1 Overview

Figure 2 shows the workflow of our ConTS in a conversation session. We overview the model by going through the major components (the numbers in text description correspond to the numbers in Figure 2).

Initialization and Sampling: If it is the first turn in the session, then we need to initialize the distribution of user embedding by averaging the pre-trained embedding of existing users (steps (1)~(2)). Then, the conversation cycle starts from sampling an instantiated embedding for the user from the currently estimated distribution (step (3)). The sampling step exactly follows the contextual Thompson Sampling [1], leveraging its advantage to achieve EE balance.

Arm Choosing: ConTS decides an action (asking or recommending) by choosing an arm with the goal to maximize the reward (steps (4)~(5)). Here, we make a key contribution to unify the attributes $p \subseteq \mathcal{P}$ and items $v \subseteq \mathcal{V}$ as the undifferentiated arms in the same space (the whole arm pool is $\mathcal{A} = \mathcal{P} \cup \mathcal{V}$). Our ConTS only needs to choose arms based on a unified reward estimation function that captures the mutual promotion of attributes and items. As such, the *conversation policy questions*—what items to recommend, what attributes to ask, and whether to ask or

recommend in a turn—are reduced to a single question of arm choosing in ConTS. If the chosen arm is an item, then a CRS recommends the top k items; and if the arm is an attribute, a CRS asks the attribute (or several top attributes, depending on the problem setting, cf. Section 5.1) (step (6)).

We argue this is the key difference between our ConTS and ConUCB [54]. In ConUCB, the items and attributes are separately modelled in two different arm spaces. The problems of *what item to recommend* and *what attribute to ask* are modeled independently where each requires specific efforts and the question of *whether to ask or recommend in a turn* is addressed by an ad hoc rule (e.g., asking attributes each five turns).

Updating: After playing an arm (recommend item or ask attribute), the system will receive feedback from the user (step (7)). It first updates the current arm pool according to the feedback (e.g., removing items not containing the user's preferred attributes, see Section 4.4). ConTS then updates the distribution of user embedding based on our unified reward estimation function and the user's feedback (step (8)). We will detail it in Section 4.4.

4.2 Initialization and Sampling

Initialization with offline FM. As the standard step in contextual Thompson Sampling, we need to get the arm embedding. We train an offline **Factorization Machine (FM)** model following EAR [27] on records of existing users to simultaneously get the embeddings of all attributes, items and existing users. Specifically, we train the FM by **Bayesian Personalized Ranking (BPR)** [39], aiming to make it rank user's preferred items and attributes higher than the others. We do mutitask training, jointly training on the two task of item prediction and attribute prediction. Specifically, we first train the model for item prediction. After it converges, we continue to optimize the FM for attribute prediction. The embeddings of all attributes and items are in the same embedding space, which will be treated as arm embeddings later in Section 4.3.

Online posterior sampling. ConTS follows the contextual Thompson Sampling described in Section 3.2, by assuming the embedding for a new user u follows a multidimensional Gaussian distribution $\mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$. Conventionally, $\mathbf{B}_u$ and $\boldsymbol{\mu}_u$ are initialized with identity matrix and zero vector separately. However, we hypothesize that the historical action of existing users can help to estimate the preference of new users. Without any previous information, it is reasonable to assume that the preference of new users to be the average of exsing users'. Thus, we initialize $\boldsymbol{\mu}_u$ as the average embedding of existing users while following the convention to initialize $\mathbf{B}_u$ as identity matrix. Specifically, if $\mathcal{U}^{old}$ denotes the collection of all embeddings of existing users, then,

$$\mathbf{u}_{init} = \frac{1}{N}\sum_{i=1}^{N}\mathbf{u}_i, \mathbf{u}_i \in \mathcal{U}^{old}. \tag{5}$$

Correspondingly, the intermediate variable $\boldsymbol{f}_u$ is also initialized with $\mathbf{u}_{init}$, since $\boldsymbol{f}_u$ is updated by Equation (14) (i.e., $\boldsymbol{\mu}_u = \mathbf{B}_u^{-1}\boldsymbol{f}_u$, where $\mathbf{B}_u$ is initialized by identity matrix).

After initialization, ConTS starts a MCR session to interact with the user. At the beginning of each turn $t = 1, 2, \ldots, T$ in a MCR session, ConTS samples from $\mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$ to get user embedding $\tilde{\mathbf{u}}$.[8] The sampling is the key step to achieve EE balance. On the one hand, ConTS uses the mean $\boldsymbol{\mu}_u$ to control the expectation of the sampling result to exploit user's currently known preference. On the other hand, ConTS uses covariance $l^2\mathbf{B}_u^{-1}$ to model the uncertainty about the estimated user preference, which decides how ConTS explore user's latent unknown preference.

[8]Note that both $\boldsymbol{\mu}_u$ and $l^2\mathbf{B}_u^{-1}$ are changed in each turn according to user feedback (see Section 4.4).

4.3 Arm Choosing

Once obtaining user embedding by posterior sampling, the agent needs to take an action, either by asking an attribute or recommending items. This is much more complex than existing contextual Thompson Sampling methods where the action is simply to choose an item to recommend. In the MCR scenario, a CRS needs to consider more questions: (1) what attributes to ask, (2) what items to recommend, and (3) whether to ask or recommend in a turn. To address those problems, ConTS adopts a simple but efficient strategy to model all the items and attributes as undifferentiated arms in the same arm space. As such, the aforementioned three problems in the MCR scenario reduced to the problem of arm choosing, which can be nicely solved by the *maximum rewards* approach in contextual Thompson Sampling. Specially, we assume the expected reward of arm a of user u is generated by

$$r(a, u, \mathcal{P}_u) \sim \mathcal{N}\left(\mathbf{u}^\top \mathbf{x}_a + \sum_{p_i \in \mathcal{P}_u} {\mathbf{x}_a}^T \mathbf{p}_i, l^2\right), \tag{6}$$

where l is the standard derivation of the Gaussian noise in the reward observation. $\mathbf{u}$ and $\mathbf{x}_a$ represent the embedding of the user and arm, and $\mathcal{P}_u$ denotes user's currently known preferred attributes obtained from the conversation by asking attributes. It explicitly takes the attributes confirmed by the user during a conversation as a strong indicator for user preference. The first part $\mathbf{u}^\top \mathbf{x}_a$ models the general preference of user u to arm a, and the second part $\sum_{p_i \in \mathcal{P}_u} {\mathbf{x}_a}^T \mathbf{p}_i$ models the affinity between arm a and the user's preferred attributes $\mathcal{P}_u$. This formula is different from the linear reward generation assumption in contextual Thompson Sampling in the sense that it explicitly capture the mutual promotion of items and attributes. We intentionally design this formula to exactly follow EAR [27]. This is because we want to establish a fair comparison between our ConTS and EAR to demonstrate the effectiveness of EE balance of the Thompson Sampling framework in the conversational recommendation for cold-start users (see Section 5).

Based on Equation (6), the system will choose the arm with the maximal reward to play. That is to say, if the chosen arm represents an attribute, the model will ask the user's preference about the attribute (or several top attributes). However, if the chosen arm is an item, then the model will directly recommend the top k items with max rewards to the user. The arm-choosing policy seamlessly unifies the asking attributes and recommending items together in an holistic framework, while ConUCB [54] separately models attributes and items, and decides whether to ask or recommend based on a hand-crafted heuristics.

4.4 Updating

After asking a question or recommending items, the agent will receive feedback from the user and the feedback will be used to update our model. First, ConTS renews the arm pool in the current conversation. If the user gives a negative feedback to either attribute(s) or items, then we remove them from the current arm pool $\mathcal{A}_u$. If he likes the attribute, then it is added into the user's preference attribute pool $\mathcal{P}_u$ and the agent filters out items without the preferred attribute from candidate pool. And if the user accepts the recommended item, the session successfully ended according to the MCR scenario.

Different from the typical update methods in contextual Thompson Sampling described in Section 3.2 in Equation (12), ConTS is adapted to conversational recommendation scenario for cold-start users by considering two additional factors: $\mathbf{u}_{init}$ for user embedding initialization and $\sum_{p_i \in \mathcal{P}_u}$ for reward estimation. In fact, Equation (6) can be re-written as

ALGORITHM 1: Conversational Thompson Sampling (ConTS)

Input: user u, the set of all attributes $\mathcal{P}$, the set of all items $\mathcal{V}$, the number of items to recommend k, maximum turn number T, hyper parameter l;
Output: Recommendation result: success or fail;

Initialization: for a new user u, do the following initialization:
$\mathcal{A}_u = \mathcal{P} \cup \mathcal{V}$; $\mathbf{u}_{init} = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{u}_i | u_i \in \mathcal{U}^{old}]$; $\boldsymbol{\mu}_u = \boldsymbol{f}_u = \mathbf{u}_{init}$; $\mathbf{B}_u = \mathbf{I}_d$; $\mathcal{P}_u = \emptyset$;
for Conversation turn $t = 1, 2, 3 \ldots T$ **do**
  Sample $\tilde{\mathbf{u}}$ from the distribution: $\tilde{\mathbf{u}} \sim \mathcal{N}(\boldsymbol{\mu}_u, l^2\mathbf{B}_u^{-1})$
  Play arm $a(t) = argmax_{a \subset \mathcal{A}_u}\ \tilde{\mathbf{u}}^{\mathsf{T}}\mathbf{x}_a + \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_a{}^{T}\mathbf{p}_i$
  if $a(t) \in \mathcal{P}$ **then**
    ask u's preference on $a(t)$
    get u's feedback r_a
    if u accepts $a(t)$ **then**
      $\mathcal{P}_u = \mathcal{P}_u \cup a(t)$
      $\mathcal{A}_u = \{a \,|\, a \in (\mathcal{P} \setminus \mathcal{P}_u)\ or\ (a \in \mathcal{V}\ \&\ a(t) \in \mathcal{P}_u)\}$
    else: $\mathcal{A}_u = \mathcal{A}_u \setminus a(t)$
  else $a(t) \subset \mathcal{V}$
    recommend top k items $\mathcal{V}_k$ from ranked item pool $\mathcal{V}_{rank}$
    get u's feedback r
    if u accepts $\mathcal{V}_k$ **then**
      Recommendation succeeds, Quit;
    else User rejects $\mathcal{V}_k$
      Update: $\mathcal{A}_u = \mathcal{A}_u \setminus \mathcal{V}_k$
  Update Bandit parameters in sequence by:
    $\mathbf{B}_u = \mathbf{B}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^{T}$
    $r'_a = r_a - \mathbf{x}_{a(t)}{}^{T}(\mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i)$
    $\boldsymbol{f}_u = \boldsymbol{f}_u + r'_a * \mathbf{x}_{a(t)}$
    $\boldsymbol{\mu}_u = \mathbf{B}_u^{-1}\boldsymbol{f}_u$
  if $t = T$ **then**
    Recommendation fails, Quit;

$$r(a, u, \mathcal{P}_u) \sim \mathcal{N}\left(\mathbf{u}^{\mathsf{T}}\mathbf{x}_{a(t)} + \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_{a(t)}{}^{T}\mathbf{p}_i, l^2\right) = \mathcal{N}\left(\mathbf{x}_{a(t)}{}^{T}\left(\mathbf{u}_{orig} + \mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i\right), l^2\right), \tag{7}$$

where $\mathbf{u}_{orig}$ denotes the user embedding exactly in the original contextual Thompson Sampling [1], in which $\mathbf{u}_{orig}$ is initialized with zero vector, and $r(a, u, \mathcal{P}_u)$ is our observed reward according to user's feedback. During the whole process, $\mathbf{u}_{orig}$ is the only variable to be updated while $\mathbf{u}_{init}$ and $\sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i$ are given, which can be regarded as the bias used in predicting the real reward from the original user embedding. According to Equation (7), we have $\mathbf{x}_{a(t)}{}^{T}\mathbf{u}_{orig} = \mathbb{E}[r(a, u, \mathcal{P}_u)] - \mathbf{x}_{a(t)}{}^{T}(\mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i)$. Intuitively speaking, the reward generated by the user embedding u_{orig} needs to be de-biased from the observed reward r_a by removing the effect of the two bias terms. This intuition can also be reflected in the derivation of the user embedding's posterior distribution as shown in Equation (8). To simplify the notations, we define $r'_a = r_a - \mathbf{x}_{a(t)}{}^{T}(\mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i)$, $\Sigma = l^2\mathbf{B}_u^{-1}$, and simplify $\mathbf{u}_{orig}$ as $\mathbf{u}_o$.

$$
\begin{aligned}
P(\mathbf{u}_o|(\mathbf{x},a,r_a)) &\propto P((\mathbf{x},a,r_a)|\mathbf{u}_o)P(\mathbf{u}_o) \qquad (8)\\
&= \mathcal{N}\left(\mathbf{u}_o^\top \mathbf{x}_a + \mathbf{u}_{init}^\top \mathbf{x}_a + \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_a{}^T \mathbf{p}_i, l^2\right)\mathcal{N}(\boldsymbol{\mu}_u, \Sigma)\\
&= \frac{\det(\Sigma)^{-1/2}}{\sqrt{(2\pi)^{d+1}\sigma^2}} \exp\left(-\frac{(r_a - \mathbf{u}_o^\top \mathbf{x}_a - \mathbf{u}_{init}^\top \mathbf{x}_a - \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_a{}^T \mathbf{p}_i)^2}{2l^2}\right)\exp\left(-\frac{(\mathbf{u}_o - \boldsymbol{\mu}_u)^\top \Sigma^{-1}(\mathbf{u}_o - \boldsymbol{\mu}_u)}{2}\right)\\
&= \frac{\det(\Sigma)^{-1/2}}{\sqrt{(2\pi)^{d+1}\sigma^2}} \exp\left(-\frac{(r'_a - \mathbf{u}_o^\top \mathbf{x}_a)^2}{2l^2}\right)\exp\left(-\frac{(\mathbf{u}_o - \boldsymbol{\mu}_u)^\top \Sigma^{-1}(\mathbf{u}_o - \boldsymbol{\mu}_u)}{2}\right)\\
&= \frac{\det(\Sigma)^{-1/2}}{\sqrt{(2\pi)^{d+1}\sigma^2}} \exp\left(\frac{\mathbf{u}_o^\top(\Sigma^{-1} + \frac{1}{l^2}\mathbf{x}_a\mathbf{x}_a^\top) - 2\mathbf{u}_o^\top(\Sigma^{-1}\boldsymbol{\mu}_u + \frac{1}{l^2}\mathbf{x}_a r'_a) + \boldsymbol{\mu}_u^\top\Sigma^{-1}\boldsymbol{\mu}_u + \frac{r'^2_a}{l^2}}{-2}\right).
\end{aligned}
$$

Since Gaussian distribution has conjugate prior, the posterior $P(\mathbf{u}_o|(\mathbf{x},a,r_a))$ still follows the Gaussian distribution, i.e., it can be written as $P(\mathbf{u}_o|(\mathbf{x},a,r_a)) = \mathcal{N}(\mu_{o,\text{post}}, l^2\Sigma_{o,\text{post}})$, in which $\mathbf{u}_{o,\text{post}}$ and $l^2\Sigma_{o,\text{post}}$ are the posterior mean and covariance matrix, respectively. According to Equation (8), it is easy to obtain

$$\Sigma^{-1}_{o,\text{post}} = \Sigma^{-1} + \frac{1}{l^2}\mathbf{x}_a\mathbf{x}_a^\top = \frac{1}{l^2}(\mathbf{B}_u + \mathbf{x}_a\mathbf{x}_a^\top), \qquad (9)$$

$$\mathbf{u}_{o,\text{post}} = \Sigma_{o,\text{post}}\left(\Sigma^{-1}\boldsymbol{\mu} + \frac{1}{l^2}\mathbf{x}_a r'_a\right) = \frac{1}{l^2}\Sigma_{o,\text{post}}(\mathbf{B}_u\boldsymbol{\mu}_u + \mathbf{x}_a r'_a) = \frac{1}{l^2}\Sigma_{o,\text{post}}(\mathbf{f}_u + \mathbf{x}_a r'_a). \qquad (10)$$

According to the derived posterior mean and covariance matrix in Equations (9) and (10), ConTS updates parameters of the distribution for the user's embedding as follows:

$$\mathbf{B}_u = \mathbf{B}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^T, \qquad (11)$$

$$r'_a = r_a - \mathbf{x}_{a(t)}{}^T\left(\mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i\right), \qquad (12)$$

$$\boldsymbol{f}_u = \boldsymbol{f}_u + r'_a * \mathbf{x}_{a(t)}, \qquad (13)$$

$$\boldsymbol{\mu}_u = \mathbf{B}_u^{-1}\boldsymbol{f}_u. \qquad (14)$$

4.5 Complexity Analysis

Last, we briefly analyse the complexity of ConTS. If $n = |\mathcal{V}_t|$ denotes the length of current candidate item pool, then $k = |\mathcal{P}_t|$ denotes the length of the current candidate attribute pool, and d denotes the length of arm's embedding, the complexity of ConTS is $O(2 * d * (n + k) + d^2)$. The first part $O(2 * d * (n + k)$ is the complexity of ranking all items and attributes (usually $n >> k$), which is general in most of traditional statistic recommendation models such as matrix factorization [25] and factorization machines [38]. The second part $O(d^2)$ is the complexity of calculating the inverse of matrix of user embedding, which is a typical part of complexity for contextual Thompson Sampling algorithms [1, 29]. Note that the pre-trail computational complexity of matrix inverse can be reduced to $O(d^2)$ according to Reference [29]. Since ConTS is based on contextual Thompson Sampling, the detailed analysis of complexity of ConTS is the same as contextual contextual Thompson Sampling. Therefore, we directly give the result as readers can easily find the detailed analysis on the classic literature about contextual Thompson Sampling [1, 29]. To sum up, our ConTS does not involve any higher-order complexities compared to standard static recommendation methods as well as Contextual Thompson Sampling methods, being practicable in real applications.

5 EXPERIMENTS

In this section, we conduct experiments to compare ConTS with representative conversational recommendation systems and the three variants of ConTS, each with one key component dropped in the original design. We do not only examine the overall performances, but also analyze the key mechanisms designed in ConTS e.g., keeping EE balance, seamlessly unifying items and attributes, and so on. Moreover, we design experiments to discuss ConTS's advantages over ConUCB and compare the performance of Thompson Sampling and UCB in our setting. Furthermore, we explore the performance of all methods on smaller max conversation turns. Last, we conduct experiments to study the impact of different cold-start degrees and do a case study for comparison.

5.1 Datasets

We perform experiments on three real-world datasets, which vary significantly in domains and settings. The settings include enumerated questions on Yelp, binary questions on LastFM and multi-attribute questions on Kuaishou (details are shown in the following). We compare ConTS with other methods on the three different settings, aiming to evaluate the model's generalization in various conversational recommendation scenarios.

- **Yelp:**[9] This is a dateset for business recommendation. We use the version processed by Reference [27], which contains a two-level taxonomy on the attributes in the original Yelp data.[10] For example, the original attributes "wine," "beer," and "whiskey" all belong to the parent attribute "alcohol." There are 29 such parent attributes over the 590 original attributes. The agent will choose from these 29 attributes to ask during a conversation. Meanwhile, all the child attributes will be displayed to the user for himself to choose the preferred one. Similar to EAR [27], this setting is denoted as *enumerated question*. This step is to avoid overly lengthy conversation caused by a large attribute space.
- **LastFM:**[10] This dataset is for music artist recommendation. We also follow exactly the pre-processing by Reference [27]. Different from the setting in Yelp dataset, this dataset is designed for binary questions. This means the that the system chooses one of the 33 attributes and ask the user to indicate the binary feedback (like or dislike) on the asked attribute.
- **Kuaishou:**[11] We construct a new dataset built on the video-click records of cold-start users in Kuaishou app. As shown in Figure 1(b), the app will show an interface to the user and ask for his preference on multiple attributes instead of a single one. Following this scenario, each time when the agent decides to ask for preference on attributes, our ConTS chooses the top 12 attributes according the estimated rewards. We still ask the user to give binary feedback to each attribute. It is also worth mentioning that we follow Reference [27] to filter out low-frequency data, including the users with less than 10 view records and the attributes with less than 5 occurrences. This is to make sure we can simulate more than 10 conversations for each user such that our evaluation is more robust.

5.1.1 Dataset Partition. The statistics of the three datasets are summarized in Table 2. For each dataset, we partition it into a training set, a testing set and a validation set. As discussed in Section 4.3, the training set is used to train the embedding of existing users, items and attributes offline. And the testing set is utilized for the conversation simulation of cold-start users which is detailed in Section 5.2.2. The validation set is used for parameter tuning.

[9] https://drive.google.com/open?id=13WcWe9JthbiSjcGeCWSTB6Ir7O6riiYy.
[10] https://www.yelp.com/dataset/.
[11] https://github.com/xiwenchao/Kuaishou-data/.

Table 2. Statistics of Datasets after Preprocessing

Dataset	users	items	interaction records	attributes
Yelp	27,675	70,311	1,345,606	590
LastFM	1,801	7,432	76,693	33
Kuaishou	6,861	18,612	100,699	665

Table 3. Statistics of the Training Set, Testing Set, and Validation Set

Dataset	users			interaction records		
	Training	Testing	Validation	Training	Testing	Validation
Yelp	21,309	3,205	3,161	941,928	201,845	201,833
LastFM	1,260	279	262	53,636	11,532	11,525
Kuaishou	4,537	1,175	1,149	70,492	15,110	14,097

ALGORITHM 2: Partition of Datasets

Input: collection of all users U, collection of all user-item interaction records I, collection of all interaction records for any user I_u where $u \in U$;

Output: training set $I_{training}$, testing set $I_{testing}$, validation set $I_{validation}$;

Initialization: the training, testing and validation set $I_{training} = \emptyset$, $I_{testing} = \emptyset$, $I_{validation} = \emptyset$;
loop Randomly sample a user u from U:
  if $|I_{training}| \leq 70\% * |I|$ ($|\ |$ denotes the size of the set) **then**
    $I_{training} = I_{training} \cup I_u$
    continue
  if $|I_{testing}| \leq 15\% * |I|$ **then**
    $I_{testing} = I_{testing} \cup I_u$
    continue
  $I_{validation} = I \setminus (I_{testing} \cup I_{training})$
  End sampling, quit

To simulate the cold-start setting, the training set, testing sets and validation set are user-disjoint. This means that, for any user in one set, we cannot find any corresponding user-item interaction records in the other two sets. As such, we treat the users in the training set as existing users while the users in validation and testing set as new users as their interaction records do not appear in the training set. Note that the partition is based on the proportion of interaction records. Specifically, we use Algorithm 2 to generate such partition so that the user-item interaction records in training set take up the **70%** of the all records in the whole dataset while the testing and validation set each take up **15%**, respectively. We choose this partition strategy to make our offline training of FM model to be on par with Reference [27], where the FM model is trained using **70%** of the user-item interaction records. The statistics of the training set, testing set and validation set of the three datasets are given in Table 3.

5.2 Experimental Setting

5.2.1 Baselines. While many good conversational recommendation systems such as [2, 8, 10, 35, 36, 42, 44, 52, 53] have been proposed recently, most of them cannot tackle the cold-start scenarios

in MCR, hence being incomparable with our ConTS. We try our best to select following representative models as our baselines:

- **Abs Greedy [11]:** This work trains an offline FM model based on existing users' records. Then it initializes the embedding for the new user with the average of existing users' embeddings. Each time it will recommend the top k items ranked by FM and update the FM according to the rejected item list. But it does not consider any attribute information and recommend without asking any questions. Christakopoulou et al. [11] have shown that Abs-Greedy has a better performance than classic Thompson Sampling, so we do not take classic Thompson Sampling as a baseline here.
- **EAR [27]:** EAR is a state-of-the-art model on MCR scenario for warm-start users. It trains a policy network by reinforcement learning to decide whether to ask questions or recommend items. The input of the network is an vector which encodes the entropy information of each attribute, user's preference on each attribute, conversation history as well as length of the current candidate list. The output is a probability distribution over all actions, which includes all attributes and a dedicated action for recommendation. It uses an pretrained offline FM model to decide which items to recommend and employs a policy network trained by policy gradient to decide conversational policy. To adapt EAR to our setting, we use the average of existing users' embedding as the embedding of the new user and use the existing user set for offline training. The new user set is used as testing. The rest of the configurations are exactly the same as the original EAR implements.
- **ConUCB [54]:** Conversational UCB is a recently proposed method to apply bandit to conversational recommendation scenario. It models the attributes and items as different arms and choose them separately by different ranking score. As for the attribute choice policy, the authors put forward four methods in the article to choose the attribute to ask: Random, Maximal Confidence, Maximal Confidence Reduction and Advanced Strategy. If $n = |\mathcal{V}_t|$ denotes the length of current candidate item pool, then $k = |\mathcal{P}_t|$ denotes the length of the current candidate attribute pool, and d denotes the length of item's embedding, Maximal Confidence Reduction is an $O(n^2 + 2 * d^2)$ algorithm and Advanced Strategy is an $O(2 * n * d * k + 2 * d^2)$ algorithm, while our ConTS is $O(2 * d * (n + k) + d^2)$ (note that $n >> (d + k)$ in our setting). We choose the second method "Maximal Confidence" proposed in the work. The third and the fourth ones in the article are infeasible in our setting, because a large arm pool in our corpus will result in a gigantic computing complexity. Moreover, all the parameters are tuned following the methods in original paper on validation set. The optimal performance can be achieved when $\lambda = 0.8$ and $\tilde{\lambda} = 1$. We also explore different policy functions in ConUCB and report the best result, which is discussed in details in Section 5.4.

To validate the key component of our ConTS design, we also compare ConTS with the following variants, each with one component ablated:

- **ConTS-$\mathbf{u}_{init}$:** As discussed in Section 4.2, ConTS initializes the mean of new user's preference distribution with the average of existing user embeddings, which is denoted as $\mathbf{u}_{init}$. We probe its impact on ConTS by removing it from the model, where we follow Agrawal and Goyal [1] to initialize the mean with a zero vector.
- **ConTS-$\mathcal{P}_u$:** ConTS models user's currently known preferred attributes $\sum_{p_i \in \mathcal{P}_u}$ when calculating the respected reward for each arm. This is to capture the mutual promotion of items and attributes, since they are in the same space. We design experiments to discuss its effect on ConTS by removing $\mathcal{P}_u$ from the reward estimation function (Equation (6) in Section 4.3). As such, the reward estimation exactly follows the contextual Thompson Sampling [1]. The resultant system is denoted as ConTS-$\mathcal{P}_u$.

- **ConTS-exp:** ConTS inherits the sampling mechanism from contextual Thompson Sampling to explore the user's latent unknown preferences. We conduct experiments to investigate the effectiveness of such exploration in ConTS. Specifically, we remove the sampling step and just take the mean of the distribution as user embedding in each turn (we also keep the parameter $\mathbf{B}_u$ as identity matrix), aiming to learn the performance of ConTS without exploration.

5.2.2 User Simulator. As conversational recommendation is an interactive process, a CRS needs to be trained and evaluated by interacting with users. However, it is expensive and less reproducible to directly recruit human beings to interact with the CRSs. Therefore, we follow the approach of Lei et al. [27], Sun and Zhang [44] to build a user simulator to simulate a user's feedback based on the user's historical user-item interaction records. Specifically, if there is an user-item record (u, v), then we treat v as the user's preferred item and its corresponding attributes $\mathcal{P}_v$ as his preferred attributes in this session. We simulate a conversation session for each interaction record and the user will give positive feedback only to item v and attributes $\mathcal{P}_v$ in current session. We also follow Reference [27] to set the max turn T to 15 in our primary experiments, but we will also discuss the model performances when setting it to other numbers in Section 5.6. If the user accept the items recommended by CRS, then the conversation will successfully end. If the recommendation is still not successful when achieving the number of max turns, then the simulated user will terminate the session due to the lack of patience. Note that we ask enumerated questions on Yelp. If every user rejects a parent attribute, then ConTS will naturally update the algorithm by all the child attributes together. Last, it is worth mentioning that we initialize the distribution of embedding for a new user in the first conversation with him. After the first conversation finishes, his parameters are kept and continue to be updated in following conversations.

5.2.3 Evaluation Metrics. As what has been done in Reference [11], we use metrics in conversational recommendation for evaluation. Following Christakopoulou et al. [11], Lei et al. [27], we use two metrics to measure the performance of each model. First, we use the success rate (SR@t) [44] to measure the ratio of successful conversation, i.e., successfully make recommendation by turn t.[12] It can be formulated as $SR@t = \frac{\#successful\ conversations\ by\ turn\ t}{\#conversations}$. We also report the average turns (AT) needed to end the session. Note that if a conversation session has not been end at turn T, we simply set it to be T. Higher SR@t represents more successful turns and smaller AT means more efficient conversation. We use one-sample paired t-test to judge the statistical significance.

5.2.4 Implementation Details. The working process includes offline and online stages. In the offline stage, we use the records of all existing users (training set) to train the users, items and attributes embedding using the FM proposed in Reference [27]. The training objective is to rank the user's preferred items and attributes higher using the FM proposed in Reference [27]. It is optimized using SGD with a regularization strength of 0.001.

In the online stage, we run the ConTS on the testing set. Specifically, we let our ConTS interact with the new user simulator described in Section 5.2.2. The dialogue sessions are simulated using user–item historical records simulated in the testing set. The parameters of ConTS are tuned on the simulated conversations from validation set. The values of user's feedback in ConTS is searched in the range of $[-8, 8]$, and we apply grid search to find the optimal settings. We find that the absolute value of positive feedback should be larger while the absolute value of negative feedback should be smaller for better performance. Specifically, the optimal parameters are set as follows: $r_{fail\_rec} = -0.15, r_{fail\_ask} = -0.03, r_{suc\_ask} = 5, r_{suc\_rec} = 5$; besides, the hyperparameters are set as $l = 0.01$

[12]This means the success rate with at most t turns.

and $k = 10$. Following EAR [27], the dimension d is set to 64 for embeddings for items, attributes and users.

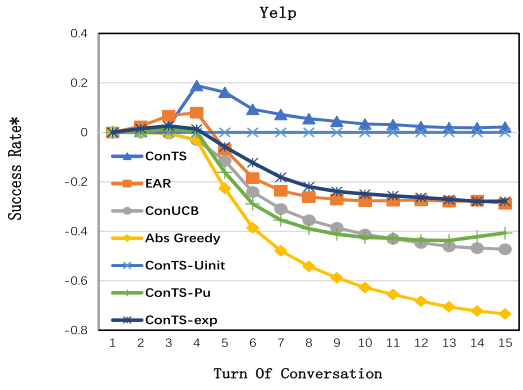
5.3 Experiment Results

Figure 3 and Table 4 report the overall experiment of results. Specifically, Figure 3 compares the recommendation Success Rate (SR)@t at each turn ($t = 1$ to 15) of different models. We set ConTS -$\mathbf{u}_{init}$ (i.e., ConTS without $\mathbf{u}_{init}$) as $y = 0$ on Yelp, ConUCB as $y = 0$ on LastFM and EAR as $y = 0$ on Kuaishou dataset for a better view. From Figure 3, we can see that ConTS achieves a higher success rate in nearly all turns than the other models with the exception of the first several turns on Yelp Dataset. We find it is because the policy is a bit random at the very beginning of each conversation session, hence the performances are not very indicative. This is especially severe on the datasets with a larger item pool like Yelp. Table 4 reports the performance of all models on **average turn (AT)** and final success rate SR@15 (we set the max turn to be 15 here). It is clear to see that ConTS significantly outperforms all the other models on both (SR)@15 and AT. We notice that nearly all the mentioned models perform much worse on LastFM and Kuaishou than on Yelp. The main reason is that ConTS asks enumerated questions on Yelp (cf. Section 5.1) where the user chooses the child-level attributes, which is finer grained (a numerated question asks more than 20 original attributes on average in a turn). This makes the candidate items shrink much faster than in LastFM and Kuaishou corpus, which adopts binary question setting. This phenomenon has also been discussed by Lei et al. [27] in detailed.

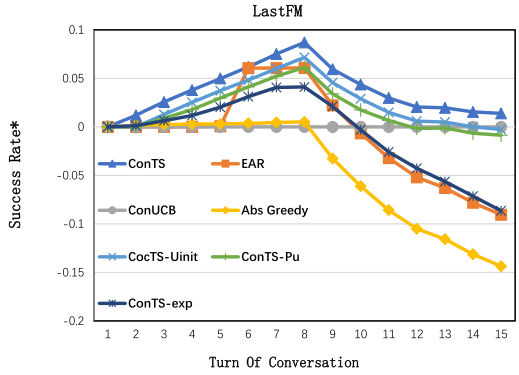
The experiments also demonstrate the effectiveness of asking for attribute preference. We can see that Abs-Greedy, the only model unable to ask attributes, almost has the worst success rate on all turns. We argue that asking attributes significantly help with this scenario on two points: (1) eliminating unqualified candidate items, i.e., the items that do not contains all the user preferred attributes, and (2) helping better estimating user preference, i.e., estimating arm's reward. While the first point is obvious, the second point can be validated by comparing ConTS and ConTS-$\mathcal{P}_u$. The only difference between ConTS and ConTS-$\mathcal{P}_u$ is that ConTS models user's known preferred attributes $\sum_{p_i \in \mathcal{P}_u}$ into calculation of arms' reward. From Figure 3 and Table 4, we can clearly see ConTS-$\mathcal{P}_u$ significantly underperforms ConTS. The results suggests that calculating rewards with $\sum_{p_i \in \mathcal{P}_u}$ contributes significantly to estimating user's preference in ConTS.

By comparing EAR with ConTS, we can see the importance of exploration for cold-start users. EAR is the state-of-the-art model for warm start users in the MCR scenario [27]. It uses the same function to estimate the user preference (Equation (6) in Section 4.3). The major difference between ConTS and EAR is that ConTS customizes contextual Thompson Sampling to make exploration-exploitation balance while EAR uses reinforcement learning to only exploit the learned user's preference in the training stage. The lack of the exploration mechanism leads to the inferior performance compared with our ConTS. The importance of exploration can further be demonstrated by comparing ConTS with ConTS-exp, which is the variant of our ConTS by only removing the exploration mechanism in contextual Thompson Sampling. From Figure 3 and Table 4, we can observe that ConS-exp has worse performance than that of ConTS.

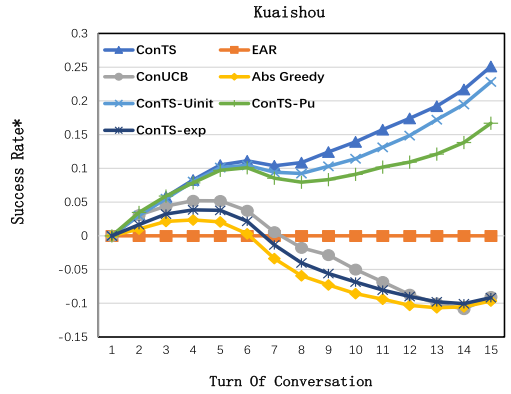
We have also studied the impact of initialization. ConTS initializes the distribution of the new user's embedding by setting the initial mean as the average of the embeddings of existing users (i.e., $\mathbf{u}_{init}$ in Equation(5)). We argue this step is also useful. The superior performance of ConTS to ConTS-$\mathbf{u}_{init}$ validates this (cf. Figure 3 and Table 4). Interestingly, we also notice the success rate curve on Yelp in Figure 3 of ConTS-$\mathbf{u}_{init}$ gradually approaches to the curve of ConTS as the number of conversation turns increases. It is due to the updates. As the conversation evolves, the CRS gets more information of the user preference from his feedback. Accordingly, the model gets less benefits from the initialization. This trend is not obvious on LastFM and Kuaishou, we believe



(a)



(b)



(c)

Fig. 3. Success Rate* (SR@t) of compared methods at different conversation turns on Yelp, LastFM, and Kuaishou. Success Rate* denotes the difference of SR between each method and the system represented by the line of $y = 0$. We set ConTS-$\mathbf{u}_{init}$ as $y = 0$ on Yelp, ConUCB as $y = 0$ on LastFM, and EAR as $y = 0$ on Kuaishou dataset to facilitate visualization.

Table 4. Results of the Difference on SR@15 and AT of the Compared Methods for Yelp, LastFM, and Kuaishou

	Yelp		LastFM		Kuaishou	
	SR@15	**AT**	**SR@15**	**AT**	**SR@15**	**AT**
Abs Greedy	0.340	14.0115	0.0104	14.9203	0.1395	14.0573
ConUCB	0.262	13.0117	0.1540	14.2279	0.1457	13.7136
EAR	0.446	11.5825	0.0640	14.3278	0.2362	13.4753
ConTS-$\mathbf{u}_{init}$	0.734	8.6195	0.1514	13.9767	0.4644	12.0534
ConTS-$\mathcal{P}_u$	0.328	12.8060	0.1505	14.0210	0.4031	12.2949
ConTS-exp	0.491	10.9975	0.0610	14.5912	0.1446	13.7879
ConTS	**0.756***	**8.5808***	**0.1628***	**13.9162***	**0.4872***	**11.8703***

Bold scores denote the best in each column, while the underlined scores denote the best baseline. Statistic significant testing is performed ($p < 0.01$).

that is due to the intrinsically different question settings. The binary question adopted in the LastFM and Kuaishou dataset helps less than the enumerated question in decreasing the candidates pool, making the candidate pool much larger. As a result, the updates are less efficient.

5.4 Discussion: Unifying Attributes and Items

The key contribution in our ConTS is the holistic modeling: we seamlessly unify the attributes and items in the same arm space, naturally fitting them into the framework of contextual Thompson Sampling. The model only needs to calculate reward for each arm using a unified function (Equation (6)). This reduces the *conversation policy questions* (e.g., deciding what attributes to ask, what items to recommend and whether to ask attributes or making recommendations) as a single problem of arm choosing. This also helps to capture the mutual promotion between items and attributes by conside ring both in Equation (6). In this section, we discuss such issues by investigating ConUCB deeply, which also takes the items and attributes as arms in the framework of bandit algorithm but models them separately. We report the experimental results on Yelp dataset, upon which the original ConUCB [54] is built, as they are more representative. Conclusions in this section also apply to the rest two datasets.

Specifically, in the original paper [54] of ConUCB, the model decides when to ask attributes based on the heuristics that a system only asks attributes if $b(t) - b(t-1) > 0$, where $b(t)$ is a function of the turn number in conversation. However, different function may work better in different datasets or application scenario. This makes the model be less robust as it is hard to choose a universal rule for all cases. For example, the author tried several different functions to control the policy, such as $5 * \log \lfloor t \rfloor$, $10 * \log \lfloor t \rfloor$,$15 * \log \lfloor t \rfloor$, $10 * \log \left\lfloor \frac{t}{50} \right\rfloor$, and so on. The paper compares the model's performance on those different functions, but it does not give an compelling strategy about how to choose it according to different situations. In the original paper [54], the authors make a discussion that the model may perform better on Yelp dataset if we choose the function that encourages asking more questions. However, to make sure we can make successful recommendation in limited conversation turns, we should always find out the balanced point between asking questions and recommending items. If we follow the way in ConUCB and luckily seek out a suitable function with good performance on the current dataset from countless candidate functions, then we can never make sure it will work well next time when we face a different recommendation situation. To investigate the robustness of the function, we compare the performance of ConUCB by choosing different functions on Yelp. Table 5 reports the results. We can see that, the performance fluctuates hugely with regard to different function. Different from ConUCB, our ConTS is

Table 5. SR@15 and AT of ConUCB with Respect to Different Functions to Decide *Whether to Ask Attributes* on Yelp Is Statistically Significant ($p < 0.01$)

Policy Function $b(t)$	**SR@15**	**AT**
$5 * \lfloor \log(t) \rfloor$	0.117	13.8065
$5 * \log(t)$	0.119	13.7725
$10 * \log(t)$	0.262	13.0117
$15 * \log(t)$	0.162	12.8927

Table 6. Comparing Maximal Confidence and Modified FM as Attribute Choosing Policy in ConUCB on Yelp

	Maximal Confidence		**Modified FM**	
Policy $b(t)$	**SR@15**	**AT**	**SR@15**	**AT**
$5 * \lfloor \log(t) \rfloor$	0.117	13.8060	0.125	13.7007
$5 * \log(t)$	0.119	13.7725	0.532	10.3027
$15 * \log(t)$	0.197	13.0117	0.598	9.7536

Statistically significant testing is performed ($p < 0.01$).

totally data driven—it learns conversation policies totally based on the model, which is estimated based on user feedback, being more intelligent and portable to new application scenarios.

We also investigate the mutual promotion of the items and attributes. In ConUCB, the model uses two separate methods to choose the the items and attributes where the two are totally different. We probe the mutual promotion by replacing the attribute choosing function (i.e., the *Maximal Confidence*) used in ConUCB with our reward estimation function[13] (Equation (6)), which is a modified FM. Namely,

$$\mathbb{E}[r(a, u, \mathcal{P}_u)] = \mathbf{u}^{\mathsf{T}} \mathbf{x}_a + \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_a{}^T \mathbf{p}_i. \tag{15}$$

Table 6 shows that (Equation (14)) helps to improve the performance of ConUCB by a large degree. This demonstrates the effectiveness of the mutual promotion between attributes and items, thus further validating our idea of unifying the modeling of attributes and items in the conversational recommendation scenario.

5.5 Discussion: Thompson Sampling vs. UCB

Despite the difference between separately or jointly modeling item and attributes, ConTS and ConUCB also differ in the bandit algorithm framework they adopt: ConTS utilizes Thompson Sampling while ConUCB utilizes UCB. To further discuss the effect of our strategy of unifying attributes and items modeling, we design another experiment to remove the effects of different bandit algorithms. To do that, we adapt LinUCB [29] to our setting by replacing the sampling step with an upper confidence bound (all the rest modules in ConTS remain the same). The reward for each arm in the adapted LinUCB (denoted as Seamless-UCB) is calculated as follows:

$$\mathbb{E}[r(a, u, \mathcal{P}_u)] = \mathbf{u}^T \mathbf{x}_\mathbf{a} + \sum_{p_i \in \mathcal{P}_u} \mathbf{x}_\mathbf{a}^T \mathbf{p}_i + \alpha \sqrt{\mathbf{x}_a{}^T \mathbf{A}_u^{-1} \mathbf{x}_a}, \tag{16}$$

[13]It is not portable to experiment with the item choosing function as it requires complex mathematics for adaptation. Hence, we only experiment with the attribute choosing function in ConUCB.

Table 7. Comparison between ConTS, ConUCB and Seamless-UCB

	Yelp		LastFM		Kuaishou	
	SR@15	**AT**	**SR@15**	**AT**	**SR@15**	**AT**
ConUCB	0.2620	13.0117	0.1540	14.2279	0.1457	13.7136
Seamless-UCB	0.7394	8.7785	0.1593	14.0126	0.4136	12.7126
ConTS	**0.7558***	**8.5808***	**0.1628***	**13.9162***	**0.4872***	**12.1033***

Bold scores are the best in each column. Statistically significant testing is performed ($p < 0.01$).

where $\alpha\sqrt{\mathbf{x}_a{}^T\mathbf{A}_u^{-1}\mathbf{x}_a}$ denotes the upper confidence bound and $\mathbf{u}$ and $\mathbf{A}_u$ are updated as

$$\mathbf{A}_u = \mathbf{A}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^T, \tag{17}$$

$$r'_a = r_a - \mathbf{x}_{a(t)}{}^T\left(\mathbf{u}_{init} + \sum_{p_i \in \mathcal{P}_u} \mathbf{p}_i\right), \tag{18}$$

$$\mathbf{b}_u = \mathbf{b}_u + r'_a * \mathbf{x}_a, \tag{19}$$

$$\mathbf{u} = \mathbf{A}_u^{-1}\mathbf{b}_u. \tag{20}$$

From Table 7, we can see that the Seamless-UCB, which uses our unifying item and attribute strategy in the UCB framework, still outperforms ConUCB. This also demonstrates the advantage of our seamless modeling strategy.

Interestingly, from the comparison of Seamless-UCB and ConTS, we can see that Thompson Sampling has a better performance in our setting. This is inline with previous research [34, 40], which reports Thompson Sampling performs better on many real-world scenarios and also reaches a smaller theoretical regret bound. This interesting discovery indicates that Thompson Sampling might be more powerful than UCB in conversational recommendation scenario. However, we leave more discussions to the future works as the comparison between Thompson Sampling and UCB is still an open question [41].

5.6 Discussion: Effect of Max Conversation Turn

As mentioned in Section 5.2.3, we set the max conversation turn to 15 in the above experiments. However, some real-world scenarios may acquire smaller max conversation turn due to the less patience of users. In this section, we conduct experiments to explore the situation with a much tighter cap on utterance. Specifically, we compare the performance of all mentioned methods by setting two smaller max conversation turn (**7** and **10**) on Yelp, LastFM and Kuaishou. Here, we report the results when max conversation turn is **7** instead of **5**. The reason is that several models, such as EAR, ConUCB, Abs Greedy, have a terrible performance on the first five turns (in Figure 3). It is hard to get meaningful interpretations. Therefore, we report the results in turn 7, which can better reveal the difference between these compared methods.

Tables 8 and 9 report the results when the max turn set to **7** and **10** separately. It is clear to see that ConTS still significantly outperforms all other models on both SR@(7) (or SR@10) and AT. This demonstrates that ConTS has a better performance even when given a much smaller max conversation turn.

Apart from it, it is interesting to see the gap of performance between ConTS and other methods becomes larger when the max conversation turn is bigger. We attribute it to two reasons: First, since we make recommendation for cold-start users, all the policies are somehow random at the beginning of each conversation session, which makes the difference not obvious when the max conversation turn is small; Second, ConTS is able to better capture user's preference in each

Table 8. Results of Difference on SR@7 and AT of Compared Methods when the Max Conversation Turn Is **7**

	Yelp		LastFM		Kuaishou	
	SR@7	**AT**	**SR@7**	**AT**	**SR@7**	**AT**
Abs Greedy	0.1798	6.6899	0.0046	6.983	0.0637	6.7793
ConUCB	0.1692	6.5184	0.0024	6.9876	0.1029	6.6072
EAR	0.2425	6.2305	0.0606	6.9215	0.0973	6.8606
ConTS-$\mathbf{u}_{init}$	0.4784	5.9671	0.0600	6.8282	0.1915	6.3848
ConTS-$\mathcal{P}_u$	0.1237	6.5986	0.0522	6.8612	0.1828	6.3893
ConTS-exp	0.2967	6.2155	0.0406	6.8979	0.0838	6.8746
ConTS	**0.5508***	**5.3995***	**0.0752***	**6.7512***	**0.2012***	**6.3610***

Bold scores denote the best in each column, while the underlined scores denote the best baseline. Statistically significant testing is performed ($p < 0.01$).

Table 9. Results of Difference on SR@10 and AT of Compared Methods when the Max Conversation Turn Is **10**

	Yelp		LastFM		Kuaishou	
	SR@10	**AT**	**SR@10**	**AT**	**SR@10**	**AT**
Abs Greedy	0.2569	9.1584	0.0065	9.9649	0.0918	9.5315
ConUCB	0.2145	8.7965	0.0275	9.9228	0.1276	9.2240
EAR	0.3502	8.2744	0.0619	9.6966	0.1775	9.3712
ConTS-$\mathbf{u}_{init}$	0.6269	7.1572	0.0964	9.5764	0.2918	8.6071
ConTS-$\mathcal{P}_u$	0.2023	9.0030	0.0849	9.6435	0.2687	8.6696
ConTS-exp	0.3773	8.1632	0.0645	9.7279	0.1089	9.3899
ConTS	**0.6609***	**6.4655***	**0.1112***	**9.4549***	**0.3168***	**8.5216***

Bold scores denote the best in each column, while the underlined scores denote the best baseline. Statistically significant testing is performed ($p < 0.01$).

conversation turn than the other methods, thus making the accumulated advantage larger as the number of conversation turn increases.

5.7 Discussion: Effects of Cold-start Degrees

In the above discussions, we have explored the cold-start cases where historical user-item interaction records are totally unavailable for the new users. That is the typical scenario that the Thompson Sampling is originally designed for [1, 14, 15, 31, 33]. However, in real applications, another common cold-start case is that limited historical records are available. There comes a natural question: what is the performance of ConTS in such case?

To have more idea of this question, we test the performance of ConTS in different cold-start degrees. Specifically, we randomly delete the user-item interaction records with the ratio of θ for each testing user and add such deleted records into the training and validation sets.[14] Therefore, θ can be used to quantify the "cold-start degree." Then, we use the new training data to train the offline FM model discussed in Section 4.3 and directly obtain the initial embeddings for testing users for the online updating.[15] To make the discussion more interesting, we compare the performance

[14]We follow 7:1.5 when distributing such records.

[15]Comparing to the case where no historical interactions for the testing user, and we average the embedding for all existing users as the embedding for the testing user.

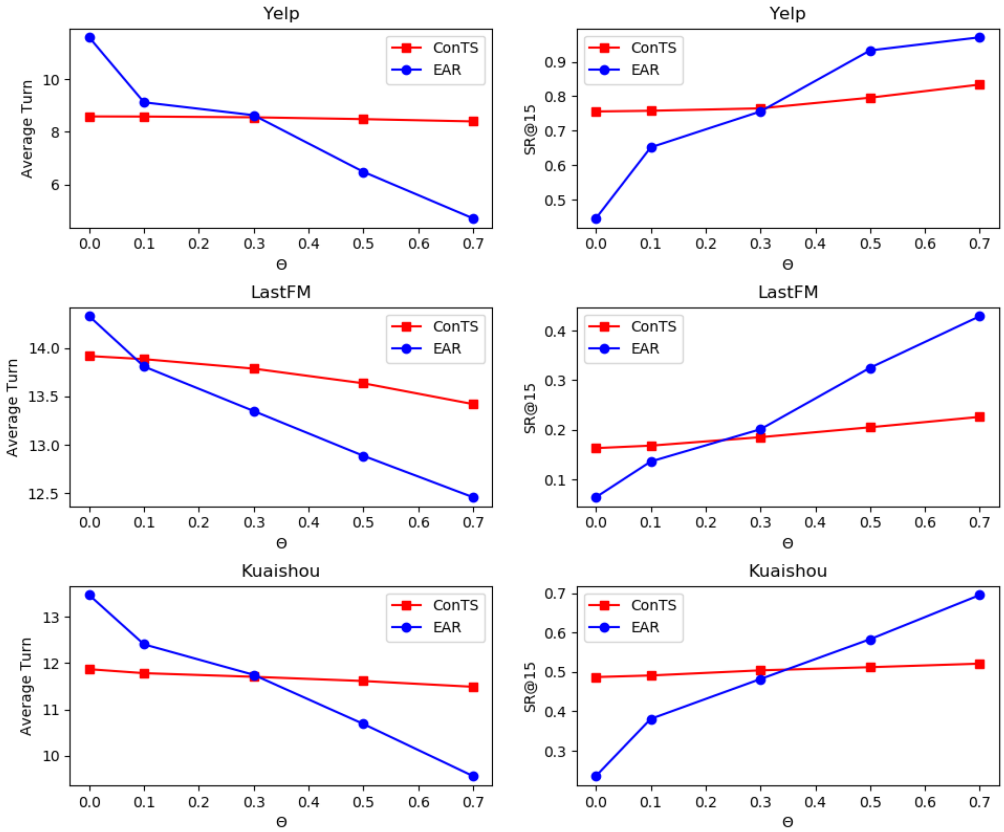


Fig. 4. SR@15 and AT of EAR and ConTS on Yelp, LastFM and Kuaishou with different cold-start degrees. θ quantifies the "cold-start degree."

of EAR under such different cold-start degrees.[16] For EAR, we follow Reference [27] to train the FM model on the training set and train the policy network on the validation set.

The experimental results are demonstrated in Figure 4. The trend is consistent on all three datasets. When there are very few existing historical records (e.g., $\theta < 0.1$), ConTS consistently gets better performances on both Average Turn and SR@15. As the existing records become more, the performance of EAR increases much sharper than ConTS and eventually outperforms ConTS by a large margin. This suggests that in the real application, it is better to apply ConTS to the users when there are very few historical records. When the records become more, it would be more helpful to shift to warm start solutions like EAR. However, to the best of our knowledge, few research has conduct conclusive discussions on the comparison between the bandit-based and warm-start recommendation systems in terms of the degree of cold start. We hope our discussions can be a start to trigger more explorations in the future.

[16]Note that EAR and ConTS use the same method (i.e., Equation (6)) to rank items. The difference is that ConTS uses Thompson Sampling to achieve EE balance for cold-start interaction strategy while EAR employs policy network trained by reinforcement learning for warm-start interaction strategy.

5.8 Case Study

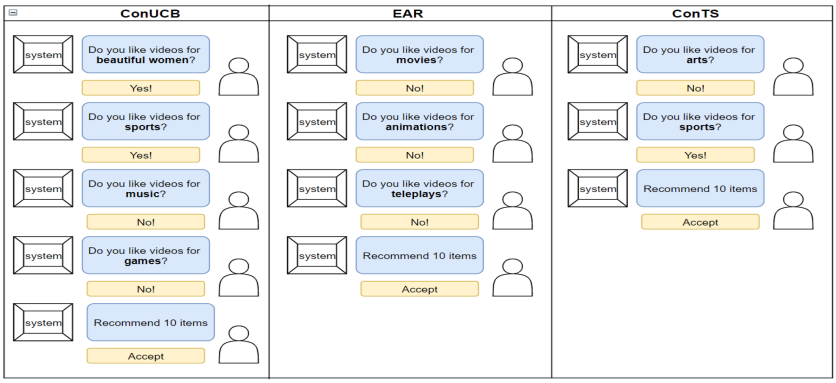
In this section, we perform three case studies (in Figure 5) on real interaction records of users in Kuaishou dataset for one-attribute questions by two baselines ConUCB, EAR and our proposed model ConTS. In case (a), the user wants a video of a famous beautiful female tennis player. The video has two attributes: *beautiful women* and *sports*. As mentioned in Section 5.4, ConUCB implements a handcrafted way to decide whether to ask or recommend. From Figure 5(a), we can see that ConUCB keeps asking attributes for the first four turns in the conversation. However, we notice that the system has already known all the user's preferred attributes after the first two turns. It is clearly the perfect time to recommend items as asking more questions will not bring additional benefits. This reveals less flexibility of ConUCB in making conversation policies. EAR decides the conversation policy questions based on historical interaction records of existing users by a pre-trained FM model a policy network. As a result, it tends to ask some popular attributes to new users according to training data, failing to explore new user's interest. For example, in the conversation in Figure 5(a), EAR insists asking attributes with similar themes (such as "movies," "animations," and "teleplays"). Although these attributes are popular among existing users, the new user may not be interested in them at all. In contrast, ConTS can explore new user's unknown interest. In the conversation, since the user gives negative feedback to attribute "arts," ConTS learns to explore his interest on some different attributes, like "sports." And once it captures user's preference on an attribute, it directly recommends items aiming to successfully end the conversation within short conversation turns.

Apart from the first case where ConTS get better performances than the baselines, we also give two more "bad" cases where ConTS is not the best. In Figure 5(b), only ConUCB successfully completes recommendation within the first five conversation turns. The user in this case prefers to watch a video about the development of Buddhism in China. It has the attribute *religion*. All the three methods have difficulties in inferring the user's interest at the beginning and the user gives negative feedback to the first questions from all the three methods. However, ConUCB keeps asking questions in the first four turns as its rule-based policy function force it to do so, while EAR and ConTS have less patience and turn to recommending items when the user gives negative feedback. This indicates that ConUCB may have better performances when the user's interest is hard to infer, because we can employ a rule-based policy function to force it to ask more questions before making recommendations. This observation is in line with the results in Table 4: ConUCB gets better relative performance on LastFM, which is supposed to be the most difficult dataset (cf. Table 4 where all models consistently get worst performances in LastFM dataset). Although it is possible for EAR and ConTS to learn to patiently ask more questions before making recommendations, the process is less tractable the policies are trained automatically.

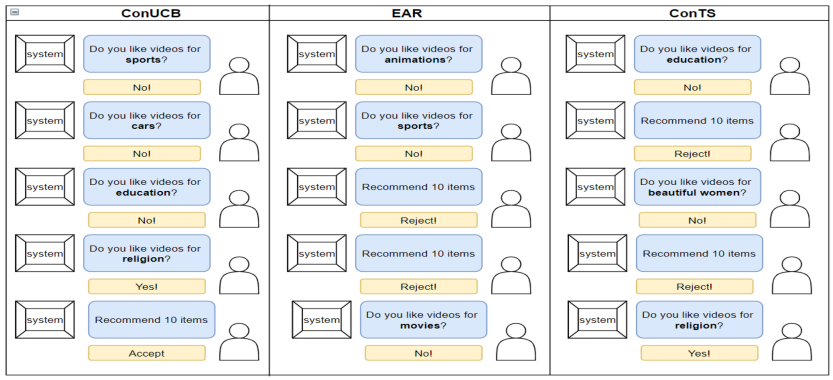
The last case (Figure 5(c)) demonstrates a user who likes a video introducing a recently released movie. It has the attribute *movie*. As introduced in the first paragraph of this section, movie is a quite popular attribute among the existing users. Since EAR tends to ask popular attributes to new users, it quickly hits the user's preferred attribute and then successfully recommend items within three conversation turns. In contrast, ConTS and EAR tend to further explore user's interest on some attributes that are not very popular, such as "cars" or "history." This indicates that if the new user has similar preference with the average of existing users', EAR can be a more efficient method.

6 CONCLUSION AND FUTURE WORKS

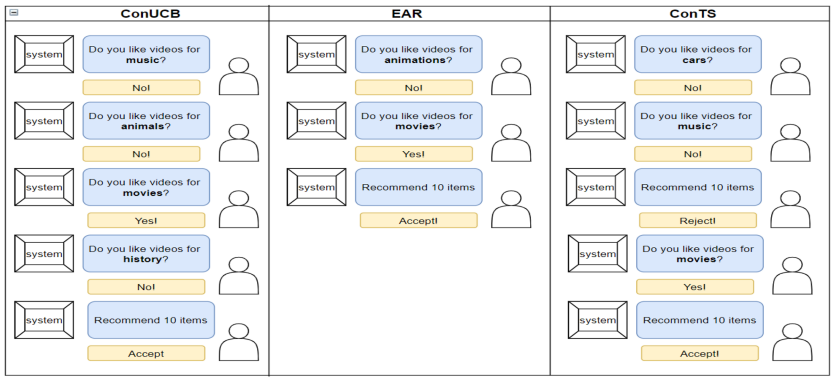
In summary, this article focuses on the cold-start problems in conversational recommendation. We customized contextual Thompson Sampling, a classic bandit algorithm, to conversation scenario,



(a) The user likes a video about a famous beautiful female tennis player.



(b) The user likes a video about the development of Buddhism in China.



(c) The user likes a video about a recently released movie.

Fig. 5. Samples of conversation of ConUCB(left), EAR(middle), and ConTS(right) of three users.

resulting in our ConTS model. ConTS makes the key contribution to seamlessly unify items and attributes as undifferentiated arms in one space. As such, our ConTS addresses the three *conversation policy questions*—what attributes to ask, what items to recommend, whether to ask attributes or recommend items in a turn—as a single problem of arm choosing based on a unified reward

estimation function. With this simple strategy, ConTS nicely fits the conversational recommendation problem in the framework of contextual Thompson Sampling, achieving good EE balance and intelligent dialogue policy within a holistic model.

We designed a series of experiments on Yelp, LastFM and Kuaishou datasets and empirically validated the effectiveness of our proposed ConTS on various settings. First, the model outperforms various strong baselines (e.g., Abs Greedy, EAR, and ConUCB) both on success rate in nearly all turns and average turn for successful recommendation. Second, we validated important components, i.e., initialization, reward estimation and exploration mechanism, by ablating each of them from our ConTS. The results demonstrate that each of the components do help to improve ConTS's performance. Then, we discussed ConUCB in details as it also uses the bandit algorithm to model item and attributes together but handle them separately. We found that, separately modeling items and attributes cannot perform stably and hard to capture the mutual promotion of both attributes and items. This further demonstrates the advantages of our strategy of seamlessly unifying items and attributes in the same space. Furthermore, we conducted experiments by setting a smaller max conversation turn and found that our ConTS still performs well with a tighter cap on conversation. We also did a case study to compare the performance of different methods.

There are still works to do in the future to explore conversational recommendation in cold-start scenario. First, we can make improvement on bandit algorithms by exploring different reward estimation functions and exploring more update strategies. Second, we can also consider more complex problem settings. For example, we can consider how to handle "don't know" of "don't care" responses. Other types of response are also possible, for example "I'd prefer X to Y," and some systems would be expected to handle this (especially if they presented in natural language). Flexibly handling this sort of response would be a very practical advance. Third, we can extend the current framework into neural fashion, leveraging on the powerful modeling capability of neural network to model more complex patterns of user preferences.

7 SUPPLEMENTARY

In this Supplement, we will try to illustrate how contextual Thompson Sampling help to keep EE balance in an intuitive way. As shown in Algorithm 1, we use a multidimensional Gaussian distribution to describe each user's preference on arms. So the mean $\boldsymbol{\mu}_u$ and covariance $l^2\mathbf{B}_u^{-1}$ determine the characters of the distribution. Each time when the agent plays an arm $a(t)$ with embedding $\mathbf{x}_{a(t)}$ and gets the feedback r, it will first update the covariance $l^2\mathbf{B}_u^{-1}$ by the formula:

$$\mathbf{B}_u = \mathbf{B}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^T. \tag{21}$$

We focus on the diagonal elements of $\mathbf{B}_u$ that have the biggest impact on the sampling result. We denote these elements as a vector $\boldsymbol{\lambda}_\mathrm{B}$, so they will be updated like this:

$$\boldsymbol{\lambda}_B = \boldsymbol{\lambda}_B + \mathbf{x}_{a(t)}^2. \tag{22}$$

Apparently, the diagonal elements of matrix $\mathbf{B}_u$ will increase after taking each action (asking attribute or recommending item). Since the covariance matrix is $l^2\mathbf{B}_u^{-1}$, the diagonal elements of it will decrease instead. This means our uncertainty on estimating the user's preference on the arms also reduces. Next time we do sampling, the result will be closer to the mean value $\boldsymbol{\mu}_u$ for those played arms because of the smaller covariance value due to the decreased values in covariance matrix. It tells the system to do less exploration on those arms since we have already know the user's preference on them.

As for the mean $\boldsymbol{\mu}_u$, we update it in this way:

$$\mathbf{B}_u = \mathbf{B}_u + \mathbf{x}_{a(t)}\mathbf{x}_{a(t)}{}^T, \tag{23}$$

$$\boldsymbol{f}_u = \boldsymbol{f}_u + r'_a * \mathbf{x}_{a(t)}, \tag{24}$$

$$\boldsymbol{\mu}_u = \mathbf{B}_u^{-1}\boldsymbol{f}_u. \tag{25}$$

Consider the formula separately. First, we fix $\mathbf{B}_u$ and investigate the impact of $\boldsymbol{f}_u$. In that case, after getting the feedback of arm $a(t)$ and updating $\boldsymbol{\mu}_u$, the agent will calculate the reward of the played arm $a(t)$ next time as follows. The expectation of the new estimated reward is

$$\begin{aligned}
\mathbb{E}[reward_{new}] &= \mathbf{x}_{a(t)}{}^T\boldsymbol{\mu}_{new} \\
&= \mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}\boldsymbol{f}_{new} \\
&= \mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}(\boldsymbol{f} + \mathbf{x}_{a(t)} * r'_a) \\
&= \mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}\boldsymbol{f}_{old} + \mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}\mathbf{x}_{a(t)} * r'_a \\
&= \mathbb{E}[reward_{old}] + \mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}\mathbf{x}_{a(t)} * r'_a.
\end{aligned} \tag{26}$$

Since $\mathbf{B}_u^{-1}$ is a positive semidefinite matrix, $\mathbf{x}_{a(t)}{}^T\mathbf{B}_u^{-1}\mathbf{x}_{a(t)}$ is non-negative. It is easy to see the change of the expectation of new reward is decided by de-biased user's feedback r'_a. For example, if a user gives a positive feedback on an arm $a(t)$, next time the algorithm tends to score higher for $a(t)$ as well as the arms with similar embeddings with $a(t)$. And if the feedback is negative the score will be relatively smaller. This strategy helps the algorithm to do exploitation by remembering past experience and taking actions accordingly. As for the second part $\mathbf{B}_u^{-1}$ in the update formula, we take it as a regularization term to confine the updating of $\boldsymbol{\mu}_u$ (cause the decrease of diagonal elements of $\mathbf{B}_u^{-1}$).

We also need to mention that since we regard all attributes and items as equivalent arms, we can estimate the expected rewards of asking attribute or recommending items by updating the parameters in the same way, thus enabling the algorithm to choose the action intelligently according to different situations. The model will learn to identify the benefit of taking different actions and make the best choice accordingly.

REFERENCES

[1] Shipra Agrawal and Navin Goyal. 2013. Thompson sampling for contextual bandits with linear payoffs. In *Proceedings of the ICML*. 127–135.

[2] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. 2019. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the SIGIR*. 475–484.

[3] Peter Auer, Nicolo Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* 47, 2–3 (2002), 235–256.

[4] Peter Auer, Nicolò Cesa-Bianchi, and Paul Fischer. 2002. Finite-time analysis of the multiarmed bandit problem. *Mach. Learn.* 47, 2-3 (May 2002), 235–256.

[5] Olga Averjanova, Francesco Ricci, and Quang Nhat Nguyen. 2008. Map-based interaction with a conversational mobile recommender system. In *Proceedings of the UBICOMM*. IEEE, 212–218.

[6] Olivier Chapelle and Lihong Li. 2011. An empirical evaluation of thompson sampling. In *Proceedings of the NIPS*. 2249–2257.

[7] Jingyuan Chen, Hanwang Zhang, Xiangnan He, Liqiang Nie, Wei Liu, and Tat-Seng Chua. 2017. Attentive collaborative filtering: Multimedia recommendation with item- and component-level attention. In *Proceedings of the SIGIR*. 335–344.

[8] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards knowledge-based recommender dialog system. In *Proceedings of the EMNLP-IJCNLP*. 1803–1813.

[9] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. 2016. Wide & deep learning for recommender systems. In *Proceedings of the DLRS@RecSys*. 7–10.

[10] Konstantina Christakopoulou, Alex Beutel, Rui Li, Sagar Jain, and Ed H. Chi. 2018. Q&R: A two-stage approach toward interactive recommendation. In *Proceedings of the SIGKDD*. 139–148.

[11] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the KDD*. 815–824.
[12] Linus W. Dietz, Saadi Myftija, and Wolfgang Wörndl. 2019. Designing a conversational travel recommender system based on data-driven destination characterization. In *Proceedings of the RecSys*. 17–21.
[13] Claudio Gentile, Shuai Li, Purushottam Kar, Alexandros Karatzoglou, Evans Etrue, and Giovanni Zappella. 2016. On context-dependent clustering of bandits. In *Proceedings of the ICML*. 1253–1262.
[14] Thore Graepel, Joaquin Quinonero Candela, Thomas Borchert, and Ralf Herbrich. 2010. Web-scale Bayesian click-through rate prediction for sponsored search advertising in Microsoft's Bing search engine. In *Proceedings of the ICML*.
[15] Ole-Christoffer Granmo. 2010. Solving two-armed Bernoulli bandit problems using a Bayesian learning automaton. *Int. J. Intell. Comput. Cybernet.* 3, 2 (2010), 207–234.
[16] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: A factorization-machine based neural network for CTR prediction. In *Proceedings of the IJCAI*.
[17] Xiangnan He and Tat-Seng Chua. 2017. Neural factorization machines for sparse predictive analytics. In *Proceedings of the SIGIR*. 355–364.
[18] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. LightGCN: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the SIGIR*.
[19] Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *Proceedings of the WWW*. 173–182.
[20] Xiangnan He, Hanwang Zhang, Min-Yen Kan, and Tat-Seng Chua. 2016. Fast matrix factorization for online recommendation with implicit feedback. In *Proceedings of the SIGIR*. 549–558.
[21] Zeng-Wei Hong, Rui-Tang Huang, Kai-Yi Chin, Chia-Chi Yen, and Jim-Min Lin. 2010. An interactive agent system for supporting knowledge-based recommendation: A case study on an e-novel recommender system. In *Proceedings of the ICUIMC*. 1–8.
[22] Dietmar Jannach, Ahtsham Manzoor, Wanling Cai, and Li Chen. 2021. A survey on conversational recommender systems. *ACM Computing Surveys (CSUR)* 54, 5 (2021), 1–36.
[23] Shaojie Jiang, Pengjie Ren, Christof Monz, and Maarten de Rijke. 2019. Improving neural response diversity with frequency-aware cross-entropy loss. In *Proceedings of the WWW*. ACM, 2879–2885.
[24] Xisen Jin, Wenqiang Lei, Zhaochun Ren, Hongshen Chen, Shangsong Liang, Yihong Zhao, and Dawei Yin. 2018. Explicit state tracking with semi-supervisionfor neural dialogue generation. In *Proceedings of the CIKM*. ACM, 1403–1412.
[25] Yehuda Koren, Robert M. Bell, and Chris Volinsky. 2009. Matrix factorization techniques for recommender systems. *IEEE Comput.* 42, 8 (2009), 30–37.
[26] Tze Leung Lai and Herbert Robbins. 1985. Asymptotically efficient adaptive allocation rules. *Adv. Appl. Math.* 6, 1 (1985), 4–22.
[27] Wenqiang Lei, Xiangnan He, Yisong Miao, Qingyun Wu, Richang Hong, Min-Yen Kan, and Tat-Seng Chua. 2020. Estimation–action–reflection: Towards deep interaction between conversational and recommender systems. In *Proceedings of the WSDM*.
[28] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the ACL*. 1437–1447.
[29] Lihong Li, Wei Chu, John Langford, and Robert E. Schapire. 2010. A contextual-bandit approach to personalized news article recommendation. In *Proceedings of the WWW*. 661–670.
[30] Seth Siyuan Li and Elena Karahanna. 2015. Online recommendation systems in a B2C E-commerce context: A review and future directions. *J. Assoc. Info. Syst.* 16, 2 (2015), 72.
[31] Benedict C. May and David S. Leslie. 2011. Simulation studies in optimistic Bayesian sampling in contextual-bandit problems. *Statistics Group, Department of Mathematics, University of Bristol* 11, 02 (2011).
[32] Kevin McCarthy, James Reilly, Lorraine McGinty, and Barry Smyth. 2004. On the dynamic generation of compound critiques in conversational recommender systems. In *Proceedings of the AH*. Springer, 176–184.
[33] Kevin P. Murphy. 2012. *Machine Learning: A Probabilistic Perspective*. MIT Press.
[34] Ian Osband and Benjamin Van Roy. 2017. On optimistic versus randomized exploration in reinforcement learning. Retrieved from https://arXiv:1706.04241.
[35] Bilih Priyogi. 2019. Preference elicitation strategy for conversational recommender system. In *Proceedings of the WSDM*. ACM, 824–825.
[36] Filip Radlinski, Krisztian Balog, Bill Byrne, and Karthik Krishnamoorthi. 2019. Coached conversational preference elicitation: A case study in understanding movie preferences. In *Proceedings of the SIGdial*. 353–360.
[37] Pengjie Ren, Zhumin Chen, Christof Monz, Jun Ma, and Maarten de Rijke. 2020. Thinking globally, acting locally: Distantly supervised global-to-local knowledge selection for background based conversation. In *Proceedings of the AAAI*.

[38] Steffen Rendle. 2010. Factorization machines. In *Proceedings of the ICDM*. 995–1000.
[39] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian personalized ranking from implicit feedback. In *Proceedings of the UAI*.
[40] Daniel Russo and Benjamin Van Roy. 2014. Learning to optimize via posterior sampling. *Math. Oper. Res.* 39, 4 (2014), 1221–1243.
[41] Daniel J. Russo, Benjamin Van Roy, Abbas Kazerouni, Ian Osband, Zheng Wen, et al. 2018. A tutorial on thompson sampling. *Found. Trends Mach. Learn.* 11, 1 (2018), 1–96.
[42] Nicola Sardella, Claudio Biancalana, Alessandro Micarelli, and Giuseppe Sansonetti. 2019. An approach to conversational recommendation of restaurants. In *Proceedings of the ICHCI*. Springer, 123–130.
[43] Hideo Shimazu. 2002. ExpertClerk: A conversational case-based reasoning tool fordeveloping salesclerk agents in e-commerce webshops. *Artific. Intell. Rev.* 18, 3–4 (2002), 223–244.
[44] Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *SIGIR*. 235–244.
[45] Cynthia A. Thompson, Mehmet H. Goker, and Pat Langley. 2004. A personalized system for conversational recommendations. *J. Artific. Intell. Res.* 21 (2004), 393–428.
[46] Rianne van den Berg, Thomas N. Kipf, and Max Welling. 2017. Graph convolutional matrix completion. In *Proceedings of the KDD*.
[47] Huazheng Wang, Qingyun Wu, and Hongning Wang. 2017. Factorization bandits for interactive recommendation.. In *Proceedings of the AAAI*. 2695–2702.
[48] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the SIGIR*. 165–174.
[49] Bifan Wei, Jun Liu, Qinghua Zheng, Wei Zhang, Chenchen Wang, and Bei Wu. 2015. DF-Miner: Domain-specific facet mining by leveraging the hyperlink structure of wikipedia. *Knowl.-Based Syst.* 77 (2015), 80–91.
[50] Qingyun Wu, Naveen Iyer, and Hongning Wang. 2018. Learning contextual bandits in a non-stationary environment. In *Proceedings of the SIGIR*. 495–504.
[51] Qingyun Wu, Huazheng Wang, Quanquan Gu, and Hongning Wang. 2016. Contextual bandits in a collaborative environment. In *Proceedings of the SIGIR*. 529–538.
[52] Tong Yu, Yilin Shen, and Hongxia Jin. 2019. An visual dialog augmented interactive recommender system. In *Proceedings of the SIGKDD*. ACM, 157–165.
[53] Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. 2020. Generating clarifying questions for information retrieval. In *Proceedings of the WWW*.
[54] Xiaoying Zhang, Hong Xie, Hang Li, and John Lui. 2020. Conversational contextual bandit: Algorithm and application. In *Proceedings of the WWW*.
[55] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W. Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the CIKM*. 177–186.

Received May 2020; revised October 2020; accepted December 2020