Bundle MCR: Towards Conversational Bundle Recommendation

Zhankui He
zhh004@eng.ucsd.edu
UC San Diego
San Diego, California, USA

Handong Zhao*
hazhao@adobe.com
Adobe Research
San Jose, California, USA

Tong Yu
tyu@adobe.com
Adobe Research
San Jose, California, USA

Sungchul Kim
sukim@adobe.com
Adobe Research
San Jose, California, USA

Fan Du
fdu@adobe.com
Adobe Research
San Jose, California, USA

Julian McAuley
jmcauley@eng.ucsd.edu
UC San Diego
San Diego, California, USA

ABSTRACT

Bundle recommender systems recommend sets of items (e.g., pants, shirt, and shoes) to users, but they often suffer from two issues: significant *interaction sparsity* and a *large output space*. In this work, we extend *multi-round conversational recommendation* (MCR) to alleviate these issues. MCR—which uses a conversational paradigm to elicit user interests by asking user preferences on tags (e.g., categories or attributes) and handling user feedback across multiple rounds—is an emerging recommendation setting to acquire user feedback and narrow down the output space, but has not been explored in the context of bundle recommendation.

In this work, we propose a novel recommendation task named *Bundle MCR*. Unlike traditional bundle recommendation (a bundle-aware user model and bundle generation), Bundle MCR studies how to encode user feedback as conversation states and how to post questions to users. Unlike existing MCR in which agents recommend individual items only, Bundle MCR handles more complicated user feedback on multiple items and related tags. To support this, we first propose a new framework to formulate Bundle MCR as Markov Decision Processes (MDPs) with multiple agents, for user modeling, consultation and feedback handling in bundle contexts. Under this framework, we propose a model architecture, called Bundle Bert (BUNT) to *(1) recommend items*, *(2) post questions* and *(3) manage conversations* based on bundle-aware conversation states. Moreover, to train BUNT effectively, we propose a two-stage training strategy. In an offline pre-training stage, BUNT is trained using multiple *cloze* tasks to mimic bundle interactions in conversations. Then in an online fine-tuning stage, BUNT agents are enhanced by user interactions. Our experiments on multiple offline datasets as well as the human evaluation show the value of extending MCR frameworks to bundle settings and the effectiveness of our BUNT design.

CCS CONCEPTS

• **Information systems** → **Personalization**.

*The corresponding author.

This work is licensed under a Creative Commons Attribution International 4.0 License.

RecSys '22, September 18–23, 2022, Seattle, WA, USA

© 2022 Copyright held by the owner/author(s).

ACM ISBN 978-1-4503-9278-5/22/09.
https://doi.org/10.1145/3523227.3546755

KEYWORDS

recommender systems, conversational recommendation, bundle recommendation

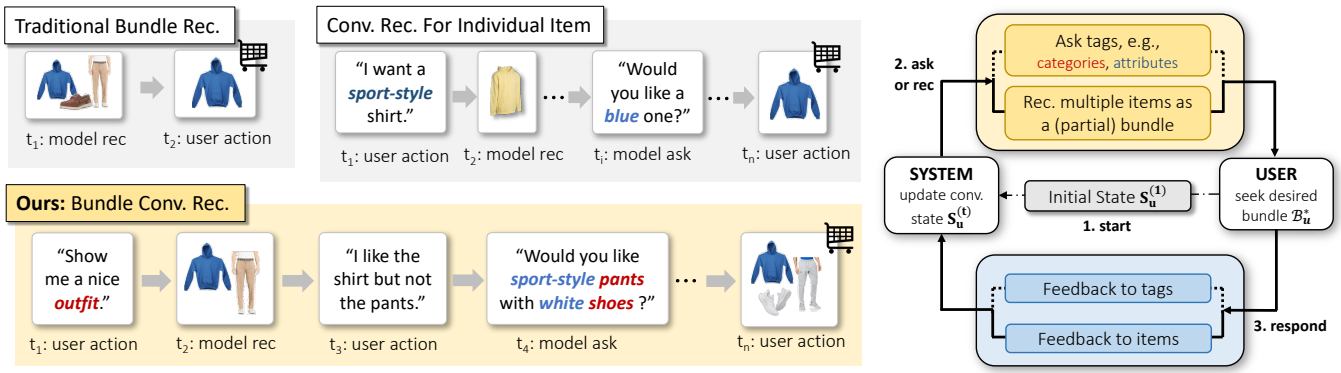
ACM Reference Format:
Zhankui He, Handong Zhao, Tong Yu, Sungchul Kim, Fan Du, and Julian McAuley. 2022. Bundle MCR: Towards Conversational Bundle Recommendation. In *Sixteenth ACM Conference on Recommender Systems (RecSys '22), September 18–23, 2022, Seattle, WA, USA.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3523227.3546755

1 INTRODUCTION

Bundle recommendation aims at recommending sets of items that can be simultaneously consumed by users [6, 12, 35] (e.g., outfits, playlists), which improves user satisfaction [8, 13]. However, bundle recommendation is inherently challenging due to (at least) two issues: (1) **Interaction sparsity**: user-bundle interactions are much sparser than user-item interactions, leading to the difficulty of modeling user preferences accurately; (2) **Output space complexity**: predicting a correct bundle (i.e., multiple items) from all item combinations is more challenging than traditional individual item recommendations.

Currently, two approaches are proposed in bundle recommendation to circumvent these issues. The first line [4, 6, 35] presents *discriminative* methods, i.e., ranking *existing* bundles which avoids the complexity issue, by treating each bundle as a generalized individual 'item'. The application scenarios of those methods are usually narrow (e.g., for pre-defined bundle sales). The second line [2, 12, 20] uses *generative* methods, i.e., generating (perhaps new) bundles, which is more flexible but still suffers from limited accuracy. In these works, bundle recommenders are *one-shot*, i.e., recommending a complete bundle with a single attempt. As the traditional bundle recommendation in Figure 1a shows, the user receives a completed bundle (shirt, shoes and pants) and reacts to this bundle (picking a shirt but ignoring others), then recommendation ends. Clearly, such one-shot setting doesn't allow the model to collect continuous user feedback and provide enhanced bundle with higher accuracy. Considering such limitations, we present a new *multi-round* and *interactive* way for bundle recommendation, i.e., allowing the user and system to "discuss" bundle composition together. Specifically, we call this *multi-round conversational bundle recommendation* task *Bundle MCR*.

The core idea of Bundle MCR is to extend one of the representative conversational recommendation mechanisms – multi-round conversational recommendation (MCR) [14, 26, 27, 53] – to bundle



(a) Example of Conversational Bundle Recommendation *vs* Others

(b) Diagram of Bundle MCR

Figure 1: *Left*: Use case comparison among traditional bundle recommendation, individual conversational recommendation and our conversational bundle recommendation. *Right*: Diagram of our proposed Bundle-Aware Multi-Round Conversational Recommendation (i.e., Bundle MCR) scenario, which extends traditional individual MCR [14, 26, 27, 53] to bundle setting.

contexts, in which the system can acquire user feedback on item tags and narrow down the output space during conversations for more accurate bundle recommendation. Although recently many conversational recommendation works [11, 14, 26, 27, 51, 53], especially MCR frameworks [14, 26, 27, 53] have proven effective to elicit user preferences for individual item recommendation, designing a new MCR framework for bundle recommendation is still non-trivial: existing MCR frameworks target recommending an individual item only (named Individual MCR); they cannot directly work for bundle settings for several reasons: (1) not considering user-bundle interactions or item-item relationships in user preference modeling; (2) recommending top-K individual items instead of multiple items as a bundle (or partial bundle); (3) handling user feedback and posting questions on tags related to an individual target item, without considering feedback or questions to different items within a bundle. We illustrate the gap between Individual MCR and Bundle MCR in Figure 1a. Individual MCR updates user feedback on tags (e.g., attributes like *sport-style*, *blue*) to narrow down the candidate item pool effectively but cannot post questions or model the feedback to multiple items (i.e., bundle-aware) directly. Instead, Bundle MCR aims to complete a bundle with the user by generating multiple items as a bundle or partial bundle, and handling questions to multiple items (e.g., *sport-style pants* and *white shoes*).

Methodologically, we formulate Bundle MCR as a Markov Decision Process (MDP) problem with multiple agents. Then, we propose a new model architecture Bundle Bert (BUNT) to conduct these functions in a unified self-attentive [23, 39, 43] architecture. Furthermore, to train BUNT efficiently, a two-stage training strategy is proposed: we pre-train BUNT with multiple *cloze* tasks, to learn the basic knowledge of how to infer correct items, tags and when to ask or recommend based on conversation contexts mimicked by offline user-bundle interactions. Then, we introduce a user simulator, create a simulated online environment, and fine-tune BUNT agents with reinforcement learning on conversational bundle interactions with users. We summarize the contributions of this work as below:

- We propose a Bundle MCR setting where users and the system complete a bundle together. To our knowledge, this is the first work that considers a conversational mechanism in bundle recommendation and also alleviates the bundle recommendation issues of *information sparsity* and *output space complexity*.
- We present an MDP framework with multiple agents for Bundle MCR. Under this framework, we propose Bundle Bert (BUNT) to conduct multiple Bundle MCR functions in a unified self-attentive architecture. We also design a two-stage (pre-training and fine-tuning) strategy for BUNT learning.
- We evaluate conversational bundle recommendations on four offline bundle datasets and conduct a human evaluation, to show the effectiveness of BUNT and the potential of conversational bundle recommendation.

2 RELATED WORK

2.1 Bundle Recommendation

Bundle (or set, basket) recommendation offers multiple items as a set to user. Traditional bundle recommendation adopts Integer Programming [31, 47, 54] or Association Analysis [9, 16] Most of them have no personalization. Some works [45, 49] apply constraint solvers. Recently, more works are learning-based and can be divided into two categories: **(1) discriminative methods:** Bundle BPR (BBPR) [35] extends BPR [37] to personalized bundle ranking (BBPR also designed a heuristic generative algorithm but it is time-consuming); DAM [6] and BGCN [3] enhance the representation of users with factorized attention networks or graph neural networks. **(2) generative methods:** An encoder-decoder framework is used in BGN [2] (RNN [10]-based) and PoG [8] (Transformer [43]-based) to generate multiple items as a personalized bundle. BGN decomposes bundle recommendation into quality/diversity via determinantal point processes. BYOB [12] treats bundle generation as a sequential decision making problem with reinforcement learning methods. In our work, the bundle recommender is for interactive

Table 1: Functionality requirements in Bundle MCR model design and comparisons with individual MCR and bundle recommendation.

Functionalities	Individual MCR	Bundle Recommendation	Bundle MCR (ours)
Bundle-Aware User Modeling	✘	[2, 4, 6, 8, 12, 35]	✔
Bundle Generation	✘	[2, 8, 12, 35]	✔
Bundle-Aware Feedback Handing	✘	✘	✔
Bundle-Aware Question Asking	✘	✘	✔
Conversation Management	[14, 26, 27, 51]	✘	✔

(conversational) settings. As Table 1 shows, existing bundle recommenders are *one-shot* so they focus on user modeling and bundle generation only. Our model (BUNT) also considers how to handle user feedback, post questions and manage conversations in a unified architecture.

2.2 Conversational Recommendation

Conversational recommender system (CRS) enables systems to converse with users actively. CRSs seek to ask questions (e.g. 'which one do you prefer') to establish user preferences efficiently or to explain recommendations. Existing CRS methods can be classified by the question spaces: **(1) Asking free text:** this method generates human-like responses in natural language [7, 22, 29]. For example, [29] collects a natural-language conversational recommendation dataset *ReDial* and builds a hierarchical RNN framework on it. KBRD [7] further incorporates knowledge-grounded information to unify recommender systems with dialog systems. **(2) Asking about items:** [11, 48] For example, [11] designs absolute (i.e., want item *A*?) or relative-question templates (i.e. item *A* or *B*?) and evaluates several question strategies such as *Greedy*, *UCB* [1] or *Thompson Sampling* [5]; **(3) Asking about tags:** the system is allowed to ask questions on user preference over different tags associated with items. For example, CRM [40] integrates conversation and recommender systems into a unified deep reinforcement learning framework to ask facets (e.g. color, branch) and recommend items. SAUR [52] proposes a *System Ask-User Respond* paradigm to ask pre-defined questions about item attributes in the right order and provide ranked lists to users. The multi-round conversational recommendation (MCR) [14, 26, 27, 53] setting also belongs to conversational recommendation setting (3).

2.3 Multi-Round Conversational Recommendation (MCR)

In our work, we focus on MCR setting, based on the following logic: (1) Completing a bundle is naturally a multi-round process, in which more user feedback to item tags is collected to make more accurate recommendations and put more items into the potential bundle. (2) MCR is arguably the most realistic setting available [14, 26, 27, 53] and widely used in recent conversational recommenders. For example, EAR [26] proposes a *Estimation-Action-Reflection* framework to ask attributes and model users' online feedback. Furthermore, SCPR [27] incorporates an item-attribute graph to provide explainable conversational recommendations. UNICORN [14] proposes a unified reinforcement learning framework based on dynamic weighted graph for MCR. To make individual MCR more realistic, MIMCR [53] allows users in MCR to select multiple choices for questions, and model user preferences with multi-interest encoders. However, existing MCR frameworks are proposed for individual item recommendation (i.e., Individual MCR). Thus the entire model architecture (e.g., FM [36]) and question strategy design is not compatible with bundle contexts. As Table 1 shows, our work uses a similar conversation management idea as exiting individual MCRs, but we design model architectures for bundle-aware user modeling, question asking, feedback handling and bundle generation.

3 BUNDLE MCR SCENARIO

We extend multi-round conversational recommendation (MCR) [14, 26, 27] to a bundle setting (i.e., Bundle MCR). Different from individual MCR, we propose a new concept *slot* for bundle MCR[1], i.e., the placeholder for a consulted item. For example, an outfit (1: shoes, 2: pants, 3: shirt) has three *slots* $\mathcal{X} = \{1, 2, 3\}$. Ideally, bundle MCR (1) determines the number of slots; (2) fills target items in the slots during conversations.

Bundle MCR is formulated as: given the set of users $\mathcal{U}$ and items $\mathcal{I}$, we collect tags corresponding to items, such as the set of attributes $\mathcal{P}$ (e.g., "dark color") and categories $\mathcal{Q}$ (e.g., "shoes"). As illustrated in Figure 1b, for a user $u \in \mathcal{U}$:

(1) Conversation starts from a state $\mathrm{S}_u^{(1)}$ which encodes user historical interactions $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$, where $\mathcal{B}_*$ represents a bundle of multiple items. Let us set conversational round $t = 1$, the system creates multiple slots as $\mathcal{X}^{(t)}$.
(2) Then, the system decides to recommend or ask, i.e., (i) recommending $|\mathcal{X}^{(t)}|$ items as a (partial) bundle to fill these proposed slots, denoting as $\mathcal{B}_u^{(t)} = \{\hat{i}_x \mid x \in \mathcal{X}^{(t)}\}$; or (ii) asking for user preference per slot on attributes $\mathcal{A}_u^{(t)} = \{\hat{a}_x \mid x \in \mathcal{X}^{(t)}\}$ and categories $C_u^{(t)} = \{\hat{c}_x \mid x \in \mathcal{X}^{(t)}\}$. Here in each slot x, $\hat{i}_x \in \mathcal{I}$, $\hat{a}_x \in \mathcal{P}$ and $\hat{c}_x \in \mathcal{Q}$.
(3) Next, user u is required to provide feedback (i.e., accept, ignore, reject) to the proposed partial bundle $\mathcal{B}_u^{(t)}$ or attributes $\mathcal{A}_u^{(t)}$ and categories $C_u^{(t)}$ per slot $x \in \mathcal{X}^{(t)}$.
(4) After that, the system updates user feedback into new state $\mathrm{S}_u^{(t+1)}$, records all the accepted items into a set, denoting as $\check{\mathcal{B}}_u$[2], and updates the slots of interest as $\mathcal{X}^{(t+1)}$ by creating new slots and removing the slots x in which user has accepted the recommended item $\hat{i}_x$.

[1] This is not the *slot* concept in dialog systems.
[2] We use the \check mark for the meaning of "being accepted by user"; similarly, we use \hat for the meaning of "being proposed to user".

After multiple rounds of step (2)-(4), the system collects rich contextual information and create bundle $\check{\mathcal{B}}_u$ for user. The conversation terminates when u is satisfied with the current bundle (i.e., $\check{\mathcal{B}}_u$ equals the target bundle $\mathcal{B}_u^*$) or this conversation reaches the maximum number of rounds T.

In Bundle MCR, we identify several interesting questions: (1) how to encode user feedback to bundle-aware state $\mathbf{S}_u^{(t+1)}$? (2) how to accurately predict bundle-aware items or tags? (3) how to effectively train models in Bundle MCR? (4) how to decide the size of slots $\mathcal{X}^{(t)}$ per round? In this work, we focus on (1)-(3), and use a simple strategy for (4), i.e., keeping the size of slots as a fixed number K. Though the slot size per round is fixed, the final bundle sizes are diverse due to different user feedback and conversation rounds. We leave more flexible slot strategies for future works.

Note that we use attribute set $\mathcal{P}$ and category set $\mathcal{Q}$ and related models for all baselines and proposed methods. But for ease of description, we only take the attribute set $\mathcal{P}$ as the example of tags in following methodology sections.

4 GENERAL FRAMEWORK

We formulate Bundle MCR as a two-step Markov Decision Process (MDPs) problem with multiple agents, since (1) the system makes two-step decisions for first recommending or asking (i.e., conversation management), then what to recommend or ask; (2) multiple agents are responsible for different decisions: an agent (using π_M) is for conversation management; a bundle agent (using π_I) decides to recommend items to compose a bundle; an attribute agent (using π_A) considers which attributes to ask. The goal of our framework is to maximize the expected cumulative rewards to learn different policy networks $\pi_M^*, \pi_I^*, \pi_A^*$. We divide a conversation round into user modeling, consultation, and feedback handling like [53], then we describe our *state*, *policy*, *action* and *transition* design under this framework in related stages.

4.1 States: Bundle-Aware User Modeling

We first introduce the shared conversation state $\mathbf{S}_u^{(t)}$ for all agents. $\mathbf{S}_u^{(t)}$ is encoded (specific encoder is introduced in Section 5) from the conversational information $\mathbb{S}_u^{(t)}$ at conversational round t, which is defined as:

$$\mathbb{S}_u^{(t)} = \big(\underbrace{\{\mathcal{B}_1, \mathcal{B}_2, \dots\}}_{\text{long-term preference}}, \underbrace{\{(\check{i}_x^{(t)}, \check{\mathcal{A}}_x^{(t)}) \mid x \in \mathcal{X}^{(\le t)}\}}_{\text{short-term contexts}}, \underbrace{\{(\mathcal{I}_x^{(t)}, \mathcal{P}_x^{(t)}) \mid x \in \mathcal{X}^{(\le t)}\}}_{\text{candidate pools}} \big). \quad (1)$$

- **Long-term preference** is represented by the set of user u's historical bundle interactions $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$.
- **Short-term contexts** collect accepted items and attributes in conversations before conversational round t. $\mathcal{X}^{(\le t)}$ is the set of slots till rounds t, i.e., $\mathcal{X}^{(\le t)} = \bigcup_{t'=1}^{t} \mathcal{X}^{(t')}$. In slot x at round t, we record the tuple $(\check{i}_x^{(t)}, \check{\mathcal{A}}_x^{(t)})$, where $\check{i}_x^{(t)}$ denotes the item id accepted by the user. If no item accepted in slot x, $\check{i}_x^{(t)}$ is set as a mask token `[MASK]`; $\check{\mathcal{A}}_x^{(t)}$ is the set of accepted attributes in slot x. For example, the initial short-term context is a K-sized set of (`[MASK]`, $\emptyset$) tuples, meaning we know nothing about accepted items or attributes.[3]
- **Candidate pools** contain item and attribute candidates per slot at round t (it is not space costly by black lists). They are initialed as completed pools $\mathcal{I}$ and $\mathcal{P}$, and updated with user u's feedback, described in Section 4.3.

Second, we introduce an additional conversation information $\bar{\mathbb{S}}_u^{(t)}$ (encoded as additional state $\bar{\mathbf{S}}_u^{(t)}$ in Section 5.1) for conversation management agent. $\bar{\mathbb{S}}_u^{(t)}$ records the result id of previous t-1 rounds as a list, such as [rec_fail, ask_fail, ...]. It is a commonly used state representation for conversation management agent in [14, 26, 27]. We follow the result id settings as [26], but apart from "rec_suc" id for successfully recommending a single item, we further introduce a "bundle_suc" id to record the result of successfully recommending the entire bundle.

4.2 Policies and Actions: Bundle-Aware Consultation

The system moves to the consultation stage after getting conversation states in user modeling stage. Now, the system makes a two-step decision: (1) whether to recommend or ask (using policy π_M); (2) what to recommend (using policy π_I) or what to ask (using policy π_A). We define these policies as:

- π_M – **conversation management:** use $\bar{\mathbf{S}}_u^{(t)}$ and $\mathbf{S}_u^{(t)}$ to predict a binary action (recommending or asking).
- π_I – **(partial) bundle generation:** if recommending, the agent uses $\mathbf{S}_u^{(t)}$ as input to generate $|\mathcal{X}^{(t)}|$ (i.e., K) items as $\mathcal{B}_u^{(t)} = \{\hat{i}_x \mid x \in \mathcal{X}^{(t)}\}$, where $\hat{i}_x$ is the action corresponding to slot x and the actions space is $\mathcal{I}_x^{(t)}$.
- π_A – **attributes consultation:** if asking, the agent uses $\mathbf{S}_u^{(t)}$ as input to generate $|\mathcal{X}^{(t)}|$ (i.e., K) attributes as $\mathcal{A}_u^{(t)} = \{\hat{a}_x \mid x \in \mathcal{X}^{(t)}\}$, where $\hat{a}_x$ is the action corresponding to slot x and the actions space is $\mathcal{P}_x^{(t)}$.

4.3 Transitions: Bundle-Aware Feedback Handling

The system handles user feedback in a transition step. The user u will react to the proposed K items or attributes with acceptance, rejection or ignoring. Generally, in our transition step, "acceptance" is mainly used to update short-term contexts, "rejection" is used to update candidate pools and we change nothing when getting "ignoring".

- **Update $\mathbb{S}_u^{(t+1)}$:** long-term preference is fixed, we update the short-term contexts and candidate pools as follows: **(1) Feedback to items:** for each consulted item $\hat{i}_x$, (i) all item candidates pools $\mathcal{I}_{x'}^{(t+1)}$ where $x' \in \mathcal{X}^{(t)}$ delete $\hat{i}_x$ because it has been recommended; (ii) if $\hat{i}_x$ is accepted, short-term contexts in slot x will assign $\hat{i}_x$ to $\check{i}_x^{(t)}$. **(2) Feedback to attributes:** for each consulted attribute $\hat{a}_x$, (i) different from

[3] We can record rejected items or attributes as well, but we omit them since they are currently not effective empirically in our experiments.

consulted items, only attribute pool $\mathcal{P}_x^{(t+1)}$ removes $\hat{a}_x$ because user has different preference on attributes in different slots (e.g., *white* shirt but *black* pants); (ii) if $\hat{a}_x$ is accepted, $\check{\mathcal{A}}_x^{(t+1)}$ in short-term context is updated by $\check{\mathcal{A}}_x^{(t)} \cup \{\hat{a}_x\}$; (iii) if $\hat{a}_x$ is explicitly rejected, it only happens when user strongly dislikes this attribute. So $\hat{a}_x$ will be removed from all attributes candidates pools, and items associated with $\hat{a}_x$ will be removed from all item candidate pools as well.

- **Update $\bar{\mathbb{S}}_u^{(t+1)}$:** it is updated by appending a new result id for round t, result in a t-sized list.
- **Update slots $\mathcal{X}^{(t+1)}$:** as Section 3 described, if items accepted, we remove the corresponded slots from $\mathcal{X}^{(t)}$, and create new slots to keep the size as K. For a new slot x', the short-term contexts is ([MASK], $\emptyset$), and candidate pools are the union sets of previous candidate pools to excluded items or attributes that user strongly dislikes.

4.4 Rewards: Two-Level Reward Definitions

We define two-level rewards for these multiple agents. **(1) Low-level rewards** are for π_I and π_A, i.e., to make item recommendations and question posting more accurate online. At round t, for each slot x the reward $r_x^I = 1$ if π_I hits target item, otherwise 0. Reward r_x^A for π_A is similar. **(2) High-level rewards** are for the conversation management agent π_M reflecting the quality of a whole conversation. The reward r^M is 0 unless the conversation ends, where we calculate r^M using one of the final bundle metrics (e.g., F1 score, accuracy).

5 MODEL ARCHITECTURE

Under this framework, we propose a unified model, Bundle BERT (BUNT). In this section, we first describe the architecture of BUNT, then we describe how to train BUNT with offline pre-training and online fine-tuning.

BUNT is an encoder-decoder framework with multi-type inputs and multi-type outputs to handle user modeling, consultation, and feedback handling. The encoder-decoder framework is commonly used in traditional bundle recommendation tasks [2, 8, 20]. We use a self-attentive architecture for three reasons: (1) Self-attentive models have already been proven as an effective representation encoder and accurate decoder in recommendation tasks [8, 19, 23, 28, 39]; (2) RNN [10]-based model inputs have to be "ordered", while self-attentive model discards unnecessary order information to reflect the unordered property in bundles; (3) A self-attentive model can be naturally used in *cloze* tasks (e.g., BERT [15]), which is suitable for predicting unknown items or attributes in slots.

5.1 BUNT for Bundle-Aware User Modeling

5.1.1 Long-Term Preference Representation. We encode user historical interactions $\{\mathcal{B}_1, \mathcal{B}_2, \dots\}$ as user long-term preferences $\mathbf{E}_u$ using hierarchical transformer (TRM) [43] encoders:

$$\mathbf{E}_u = \mathtt{TRM}_{\mathtt{bundle}}(\{\mathbf{B}_1, \mathbf{B}_2, \dots\}), \text{ where } \mathbf{B}_n = \mathtt{AVG}(\mathtt{TRM}_{\mathtt{item}}(\mathcal{B}_n)), \quad n = 1, 2, \dots. \tag{2}$$

$\mathtt{TRM}_{\mathtt{bundle}}$ is a transformer encoder over the set of bundle-level representations $\{\mathbf{B}_1, \mathbf{B}_2, \dots\}$, the output $\mathbf{E}_u \in \mathbb{R}^{N_u \times d}$ represents user long-term preferences, N_u is the number of historical bundles, and d is the hidden size of the $\mathtt{TRM}_{\mathtt{bundle}}$ model. The bundle representation $\mathbf{B}_n \in \mathbb{R}^{1\times d}$ is also extracted by a transformer encoder, namely $\mathtt{TRM}_{\mathtt{item}}$, over the set of item embeddings in this bundle, then the set of output embeddings from $\mathtt{TRM}_{\mathtt{item}}$ is aggregated by average pooling AVG as $\mathbf{B}_n$. Our two-level transformers contain no positional embeddings since the input representations are unordered.

5.1.2 Short-Term Contexts Representation. We describe how to represent short-term contexts $\{(\check{i}_x^{(t)}, \check{\mathcal{A}}_x^{(t)}) | x \in \mathcal{X}^{(\le t)}\}$. We feed the contexts into a special embedding layer EMB, then obtain two sets of embeddings for items, attributes:

$$\mathbf{E}_{I,u}^{(t)},\ \mathbf{E}_{A,u}^{(t)} = \mathtt{EMB}(\{(\check{i}_x^{(t)}, \check{\mathcal{A}}_x^{(t)}) | x \in \mathcal{X}^{(\le t)}\}), \tag{3}$$

where $\mathbf{E}_{*,u}^{(t)} \in \mathbb{R}^{|\mathcal{X}^{(\le t)}|\times d}$ denotes item (I) and attribute (A) embeddings. For items, we retrieve embeddings of the accepted item ids (or [MASK] id). For attributes, we retrieve embeddings corresponding to the accepted attribute ids (or [PAD] id) in $\check{\mathcal{A}}_x^{(t)}$ for $x \in \mathcal{X}^{(t)}$, then apply average pooling AVG on embeddings to obtain $\mathbf{E}_{A,u}^{(t)} \in \mathbb{R}^{|\mathcal{X}^{(\le t)}|\times d}$.

5.1.3 Long- and Short-Term Representation Fusion. We feed user long-term preferences $\mathbf{E}_u$ and short-term contexts $\mathbf{E}_{*,u}^{(t)}$ into an L-layer transformer. For notation simplicity[4], we denote $\mathbf{E}_{I,u}^{(t)}$ as $\mathbf{O}^0$ and get the fused representation:

$$\mathbf{O}^l = \mathtt{TRM}_l(\widetilde{\mathbf{O}}^{l-1}, \mathbf{E}_u),\ \widetilde{\mathbf{O}}^{l-1} = \mathtt{LN}\left(\mathbf{O}^{l-1} \oplus \mathbf{E}_{A,u}^{(t)}\mathbf{W}^{l-1}\right), \quad \text{where } l = 1, \dots, L, \tag{4}$$

where $\mathtt{TRM}_l$ is the l^{th} transformer layer with cross attention [43], $\mathbf{W}^{l-1} \in \mathbb{R}^{d\times d}$ is a learnable projection matrix at layer l-1 for attribute representation. $\oplus$ is element-wise addition and LN denotes LayerNorm [42] for training stabilization. We incorporate the attribute feature $\mathbf{E}_{A,u}^{(t)}$ before each transformer layer in order to incorporate multi-resolution levels, which is effective in transformer-based recommender models [30]. Thus for the output representation $\mathbf{O}^L \in \mathbb{R}^{|\mathcal{X}^{(\le t)}|\times d}$, each row $\mathbf{O}_x^L$ ($x \in \mathcal{X}^{(\le t)}$) contains contextual information from slots in conversation contexts. We treat $\mathbf{O}^L$ and candidate pools $\mathcal{I}_x^{(t)}, \mathcal{P}_x^{(t)}$ for all slots $x \in \mathcal{X}^{(\le t)}$ as the encoded state $\mathbf{S}_u^{(t)}$. Moreover, for the additional conversation records $\bar{\mathbb{S}}_u^{(t)}$ introduced in Section 4.1, we encode it as a vector $\bar{\mathbf{S}}_u^{(t)}$ by using result id embeddings and average pooling.

5.2 BUNT for Bundle-Aware Consultation

For consultation step, we feed the encoded state into multiple policy networks to get outputs for each slot $x \in \mathcal{X}^{(t)}$:

$$\begin{cases} P_M(a \mid \bar{\mathbf{S}}_u^{(t)}, \mathbf{O}_x^L) = \beta \cdot \pi'_M(a \mid \bar{\mathbf{S}}_u^{(t)}) + (1-\beta)\cdot \pi''_M(a \mid \mathbf{O}_x^L), \\ \qquad\qquad \text{where } a \in \{0, 1\}, \text{ (Conv. Management)} \\ P_I(a \mid \mathbf{O}_x^L) = \pi_I(a \mid \mathbf{O}_x^L), \quad \text{where } a \in \mathcal{I}_x^{(t)}, \text{ (Bundle Generation)} \\ P_A(a \mid \mathbf{O}_x^L) = \pi_A(a \mid \mathbf{O}_x^L), \quad \text{where } a \in \mathcal{P}_x^{(t)}. \text{ (Attribute Consultation)} \end{cases} \tag{5}$$

[4] It should be exactly represented as $\mathbf{O}_{I,u}^{(t,0)}$; we omit some notations for simplicity of the decoder description below.

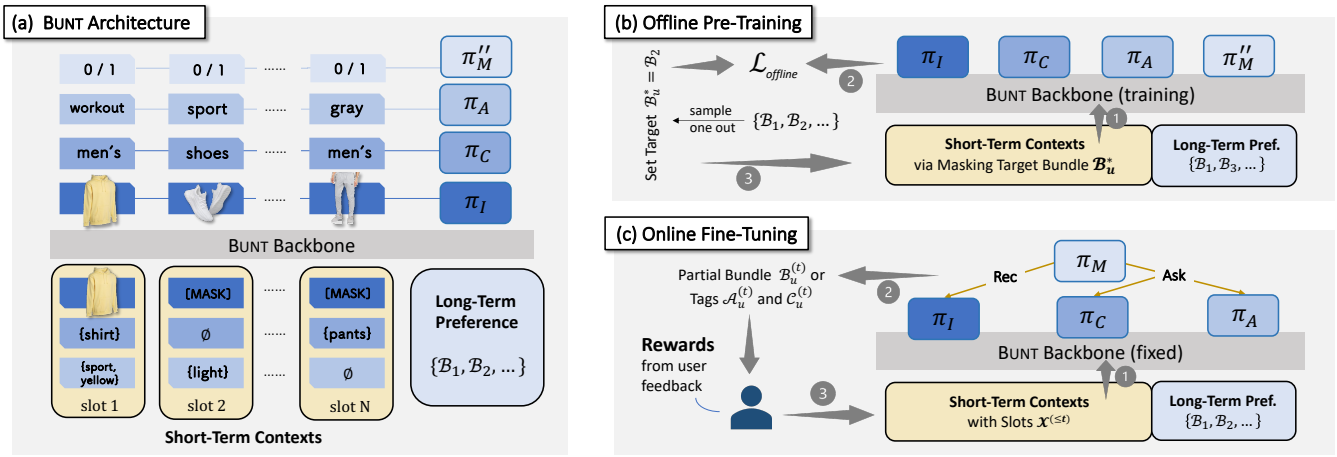


Figure 2: (a) BUNT architecture illustration. Bunt is a Bert-like model which encodes long-term preference and short-term contexts to infer masked items, categories and attributes per slot $x \in \mathcal{X}^{(t)}$. In this example, $\mathcal{X}^{(t)} = \{2, N\}$ because the related items are still unknown (i.e.,with [MASK]), and $\mathcal{X}^{(\leq t)} = \{1, \ldots, N\}$. We define long-term preference and short-term contexts in Section 4.1. (b) BUNT offline pre-training diagram, where ① denotes user modeling, ② mimics the consultation step, ③ mimics the feedback handling step, but instead of updating the conversation state at the next round, offline training simply re-masks the target bundle to generate the next masked bundle as BUNT inputs. (c) BUNT online training diagram, where ① is user modeling, ② is the consultation step to generate partial bundle $\mathcal{B}_u^{(t)}$ or attributes $\mathcal{A}_u^{(t)}$ and categories $C_u^{(t)}$, ③ is the feedback handling step to update short-term contexts. We describe steps ①-③ in Sections 4 and 5, where we keep π_A and omit the similar policy π_C, for ease of description.

P_* represents the probability. Policy network π_M is linearly combined by two sub models π'_M and π''_M for state $\bar{\mathbf{S}}_u^{(t)}$ and $\mathbf{O}_x^L$ respectively, β is a gating weight[5]. π'_M, π''_M, π_I and π_A are MLP [18] models with ReLU [34] activation and softmax layer. We use π_I or π_A to infer the masked items or attributes in slot x. In inference stage, we take the actions with the highest probability to decide recommending or asking, to construct the consulted (partial) bundle $\mathcal{B}_u^{(t)}$ or questions on attributes $\mathcal{A}_u^{(t)}$. Compared with other individual-item MCR models, the contextual information stored in different slots matters in bundle recommendation, so it is natural to share the state encoded from different slots for both recommendation and question predictions in a unified self-attentive architecture.

5.3 Offline Pre-Training

Due to the large action spaces of items and attributes, it is difficult to directly train agents from scratch. Thus, we first pre-train the BUNT model on collected offline user-bundle interactions. The core idea of pre-training is to mimic model inputs and outputs in the process of Bundle MCR, which can be treated as multiple *cloze* (i.e., "fill the slot") tasks given a few accepted items and attributes to infer the masked items and attributes.

5.3.1 Multi-Task Loss. BUNT offline training is based on a multi-task loss for recommendation and question asking simultaneously, i.e., $\mathcal{L}_{offline} = \mathcal{L}_{rec} + \lambda \mathcal{L}_{ask}$, where λ is a trade-off hyper-parameter to balance the importance of these two losses in offline pre-training. We treat item prediction as a multi-class classification task for masked slots $\mathcal{X}^{(t)}$:

$$\mathcal{L}_{rec} = -\sum_{x \in \mathcal{X}^{(t)}} \sum_{i \in \mathcal{I}_x^{(t)}} y_i \log P_I(i \mid \mathbf{O}_x^L), \tag{6}$$

where y_i is the binary label (0 or 1) for item i. Meanwhile, attribute predictions are formulated as multi-label classification tasks. We use a weighted cross-entropy loss function considering the imbalance of labels to prevent the model from only predicting popular attributes. The loss function of attribute predictions is:

$$\mathcal{L}_{ask} = -\sum_{x \in \mathcal{X}^{(t)}} \sum_{a \in \mathcal{P}_x^{(t)}} w_a \cdot y_a \log P_A(a \mid \mathbf{O}_x^L), \tag{7}$$

Algorithm 1 BUNT Offline Pre-Training

Input: historical user bundle interactions $\mathcal{D}$, masking ratio ρ, BUNT (including π''_M, π_I, π_A) parameters Θ, slot size K;
Output: BUNT parameters Θ after pre-training;

1: **while** not meet training termination criterion **do**
2: Sample a user $u \in \mathcal{U}$, get historical bundles $\{\mathcal{B}_1, \ldots, \mathcal{B}_{N_u}\}$ from $\mathcal{D}$; sample a historical bundle as target bundle, e.g., $\mathcal{B}_n$;
3: Get $\mathbf{E}_u \leftarrow$ Section 5.1.1 with input $\{\mathcal{B}_1, \ldots, \mathcal{B}_{N_u}\} \setminus \{\mathcal{B}_n\}$;
4: Sample l items in $\mathcal{B}_n$ as $\mathcal{B}_n^l$; ▷ Mimic partial bundle. W.l.o.g., assume $|B_n| > K$
5: Sample $k \in [1, K]$, then mask k items in B_n^l, set the masked positions as slots $\mathcal{X}$;
6: Retrieve attributes for all the items in $\mathcal{B}_n^l$ and mask attributes with probability ρ; ▷ Mimic short-term contexts
7: Predict the distributions of masked items, attributes and conversation management in slot $x \in \mathcal{X}$ via Section 5.2;
8: Compute loss $\mathcal{L}_{offline}$ with Equations (6) to (8); update Θ using gradient-related optimizer (e.g., [25]).

[5] β is predicted by an MLP model with sigmoid function and using concatenated $\bar{S}_u^{(t)}$, $\mathbf{O}_x^L$ as input.

where w_a is a balance weight of attribute a following [24], and note that multiple y_a can be 1 for multi-label classification. Furthermore, we pre-train part of conversational manager, i.e., π''_M, to decide whether to recommend or ask:

$$\mathcal{L}_{conv} = -\sum_{x \in \mathcal{X}^{(t)}} \mathbb{I}(l_x \neq -1) \cdot \log \pi''_M(l_x \mid \mathbf{O}^L_x). \tag{8}$$

For slot x, as long as item agent π_I hits the target item, l_x is set as 1; otherwise, if the attribute agent hits the target, l_x is 0. l_x is set as -1 when no agents make successful predictions. We denote $\mathcal{L}_{ask} = \mathcal{L}_{cate} + \mathcal{L}_{attr} + \mathcal{L}_{conv}$.

5.3.2 Training Details. Figure 2b illustrates BUNT offline training. We pre-train BUNT on offline user-bundle interactions, to obtain the basic knowledge to predict the following items or attributes given historical bundle interactions and conversational information. The training details are in Algorithm 1.

5.4 Online Fine-Tuning

Figure 2c shows the online-training diagram, where we fine-tune BUNT agents during interactions with (real or simulated) users. Our core idea is fixing BUNT backbone parameters, fine-tune agents π_I, π_A and π_M in a Bundle MCR environment to update related parameters and improve the accuracy after interacting with users. The online fine-tuning details are in Algorithm 2. We omit the details of RL value networks like [27].

Algorithm 2 Online BUNT Fine-Tuning

Input: trainable BUNT parameters Θ_I, Θ_A and Θ_M for three networks π_I, π_A and π_M, empty buffer $\mathbf{M}_M$, $\mathbf{M}_I$ and $\mathbf{M}_A$;
Output: BUNT policy networks parameters Θ_I, Θ_A and Θ_M;

for episode $e = 1, 2, \ldots$ **do**
  Sample a user u, get target bundle $\mathcal{B}^*_u$; initialize $\check{\mathcal{B}}_u \leftarrow \emptyset$ for recording all the accepted items;
  for conversation round $t = 1, 2, \ldots, T$ **do**
    Get conversation states $\mathbf{S}^{(t)}_u$ and $\bar{\mathbf{S}}^{(t)}_u$ via Section 5.1; get slots $\mathcal{X}^{(t)}$ via Section 4.3; ▷ 1. user modeling
    Sample action a_M from $\{0, 1\}$ using π_M via Section 5.2 ; ▷ 2. consultation
    if $a_M == 1$ **then**
      Use $\mathbf{O}^L$ from $\mathbf{S}^{(t)}_u$ to generate a partial bundle $\mathcal{B}^{(t)}_u$ using π_I via Section 5.2; ▷ 2.1 recommending
      Update conversation states $\mathbf{S}^{(t+1)}_u$ and $\bar{\mathbf{S}}^{(t+1)}_u$ via Sections 4.3 and 5.1; get $\tilde{\mathbf{O}}^L$ from $\mathbf{S}^{(t+1)}_u$; ▷ 3. feedback handling
      Add $\{(\mathbf{O}^L_x, \tilde{\mathbf{O}}^L_x, \hat{\imath}_x, r^I_x) \mid x \in \mathcal{X}^{(t)}\}$ to $\mathbf{M}_I$, calculating r^I_x via Section 4.4; ▷ i.e., (state, next_state, action, reward)
      Add accepted items into $\check{\mathcal{B}}_u$;
    else if $a_M == 0$ **then**
      Use $\mathbf{O}^L$ from $\mathbf{S}^{(t)}_u$ to generate questions on attributes $\mathcal{A}^{(t)}_u$ using π_A via Section 5.2; ▷ 2.1 asking
      Update conversation states $\mathbf{S}^{(t+1)}_u$ and $\bar{\mathbf{S}}^{(t+1)}_u$ via Sections 4.3 and 5.1; get $\tilde{\mathbf{O}}^L$ from $\mathbf{S}^{(t+1)}_u$; ▷ 3. feedback handling
      Add $\{(\mathbf{O}^L_x, \tilde{\mathbf{O}}^L_x, \hat{a}_x, r^A_x) \mid x \in \mathcal{X}^{(t)}\}$ to $\mathbf{M}_A$, calculating r^A_x via Section 4.4; ▷ i.e., (state, next_state, action, reward)
      Add $((\mathbf{O}^L, \bar{\mathbf{S}}^{(t)}_u), (\tilde{\mathbf{O}}^L, \bar{\mathbf{S}}^{(t+1)}_u), a_M, r^M)$ to $\mathbf{M}_M$, calculating r^M via Section 4.4; ▷ i.e., (state, next_state, action, reward)
      if $\check{\mathcal{B}}_u = \mathcal{B}^*_u$ or $t = T$ **then**
        Current conversation terminates;
  if $\mathbf{M}_k$ ($k = \{M, I, A\}$) meets pre-defined buffer training criterion (e.g., buffer size) **then**
    Update Θ_k using $\mathbf{M}_k$ with RL methods (e.g., DQN [33], PPO [38]); Then reset $\mathbf{M}_k$; ▷ policy learning

6 EXPERIMENTS

6.1 Evaluation Protocol and Metrics

Following [12, 23, 27], we conduct a *leave-one-out* data split (i.e. for each user, randomly select N-1 bundles for offline training, the last bundles for online training, validation and testing respectively in a ratio of 6:2:2). We choose the multi-label precision, recall, F1, and accuracy, defined in [50] to measure the quality of the generated bundle.

6.2 Datasets

We extend four datasets (see statistics in Table 2) for Bundle MCR.[6] **(1) Steam**: This dataset collects user interactions with game bundles in [35] from the Steam[7] platform. We use item *tags* as *attributes* in Bundle MCR and item *genres* as *categories* and discard users with fewer than two bundles according to our evaluation protocol. **(2) MovieLens**: This dataset is a benchmark dataset [17] for collaborative filtering tasks. We use the ML-10M version by treating movies rated with the same timestamps (second-level granularity) as a bundle. We treat provided *genres* as *categories*, *tags* as *attributes* in Bundle MCR. **(3) Clothing**: This dataset is collected in [32] from Amazon[8] e-commerce platform; we use the subcategory *clothing*. We treat co-purchased items as a bundle by timestamp. We use item *categories* in the metadata as *categories* in Bundle MCR, and *style* in item reviews (*style* is a dictionary of the product metadata, e.g., "format" is "hardcover", we use "hardcover") as *attributes*. For MovieLens and Clothing, bundles are grouped by timestamp thus noisy. To improve data quality, we filter out users and items that appear no more than three times. **(4) iFashion** is an outfit dataset with user interactions [8]. Similar to [44], we pre-process iFashion as a 10-core dataset to ensure data quality. We use *categories* features from iFashion metadata, and tokenize the *title* as *attributes* in Bundle MCR.

[6] We cannot use other bundle datasets such as *Youshu* or *NetEase* because they do not provide item attributes or categories information.
[7] https://store.steampowered.com
[8] https://www.amazon.com

6.3 Baselines

We introduce three groups of recommendation baselines to evaluate Bundle MCR and BUNT (we call our full proposed BUNT in Section 5 as BUNT-Learn). More technical details of baselines can be found in Section 2.

6.3.1 Traditional bundle recommenders. **Freq** uses the most frequent bundle as the predicted bundle. **BBPR [35]:** considering the infeasible time cost of cold bundle generation in BBPR, we use BBPR to rank existing bundles. **BGN [2]** adopts an encoder-decoder [41] architecture to encode user historical interactions and generate a sequence of items as a bundle. We use the top-1 bundle in BGN generated bundle list as the result. **PoG [8]** is a transformer-based [43] encoder-decoder model to generate personalized outfits. We use it

for general bundle recommendation. **BYOB [12]** is the most recent bundle generator using reinforcement learning methods.

6.3.2 Adopted individual recommenders for Bundle MCR. **FM-All** is an FM [36] variant used in MCR frameworks [26, 27], "All" means this model in Bundle MCR only recommends top-K items per round without asking any questions. **FM-Learn** follows the item predictions in FM-All, but use other pre-trained agents in BUNT for conversation management and question posting. **EAR [26]** and **SCPR [27]** are popular Individual MCR frameworks based on FM. We keep the core ideas of estimation-action-reflection in our EAR and asking attributes by path reasoning in our SCPR, and change the names to EAR* and SCPR* because some implemented components are changed for adapting into Bundle MCR. We do not use recent UNICORN [14] or MIMCR [53], because the unified action space in UNICORN is incompatible with Bundle MCR to generate multiple items or attributes per round; main contributions of MIMCR are based on the multiple choice questions setting, which is incompatible with Bundle MCR.

6.3.3 Simple bundle recommenders for Bundle MCR. **BUNT-One-Shot** uses BUNT in traditional bundle recommendation following the inference of PoG [8]. **{BYOB, BGN, BUNT}-All** models are simple bundle recommender implementations in Bundle MCR, only recommending top-K items per round without asking any questions.

6.4 Experimental Setup

6.4.1 Training Details. Our training phases are two-stage[9]: (1) in offline pre-training, we follow Algorithm 1 to implement and train our BUNT model with `PyTorch`. The number of transformer layers and heads are searched from {1,2,4}, $d = 32$, $K = 2$, $\lambda = 0.1$ and masking ratio $\rho = 0.5$. We use an Adam [25] optimizer with initial learning rate 1e-3 for all datasets with batch size 32. The maximum bundle size is set as 20. (2) In online fine-tuning, we implement Algorithm 2 using `OpenAI Stable-Baselines` RL training code. We reuse Proximal Policy Optimization [38] (PPO) in `Stable-Baselines`[10] to train four agents (π_M, π_I, π_C, π_A) jointly (π_C is category policy, similar to π_A) using Adam optimizer with lr=1e-3. Other hyperparameters follow the default settings in `Stable-Baselines`. We re-run all experiments three times with different random seeds and report the average performance and related standard errors.

6.4.2 User Simulator Setup. Due to the difficulty and cost of interacting with real users, we mainly evaluate our frameworks with user simulators, similar to previous works [14, 26, 27, 53]. We simulate a user with a target bundle $\mathcal{B}^*$ in mind, which is sampled from our online dataset. To mimic real user behavior, the user simulator *accepts* system-provided items which agree with target bundle $\mathcal{B}^*$; *accepts* categories and attributes that agree with the potential target items in the current slot.[11] The user simulator only explicitly *rejects* categories or attributes that are not associated with any items in $\mathcal{B}^*$. In other cases, the user simulator *ignores* items, categories and attributes provided by system. The user simulator is able to terminate conversations when all items in $\mathcal{B}^*$ have been recommended. Otherwise, the system ends conversations after $t = T$ conversation rounds. We set the maximum conversation rounds T to 10 in our experiments.

6.5 Main Performance of Bundle MCR and BUNT-Learn

Table 3 and Figure 3 show the main performance of our proposed framework and model architecture compared with other conversational recommendation baselines. We make the following observations:

6.5.1 BUNT Backbone Performance. Though we propose BUNT for the Bundle MCR task, we first show BUNT is competitive in traditional *one-shot* bundle recommendation. Figure 3a shows BUNT outperforms classic bundle recommenders (i.e., BBPR) markedly, and is comparable to or sometimes better than recent bundle generators (e.g., BGN, PoG). In this regard, BUNT backbone is shown to learn basic "bundle recommendation" knowledge much like other models.

6.5.2 Effectiveness of Bundle MCR. We show the effectiveness of Bundle MCR by comparing model (e) and (i) in Table 3. For example, the accuracy on MovieLens data is improved from 0.061 to 0.181. This indicates even given the same backbone model, introducing a conversational mechanism (Bundle MCR) can collect immediate feedback and improve recommendation performance. Also, we observe the relative improvement on the other three datasets is higher than on Steam. For example, the relative improvement in accuracy is 61.56% on Steam compared to 196.72% on MovieLens. This shows challenging datasets (e.g., sparser, larger item space) can gain more benefit from Bundle MCR, since it allows users to provide feedback during conversations.

6.5.3 Effectiveness of BUNT-Learn. We adopted several Individual MCR recommenders (a)-(d) in Table 3 into bundle settings, in which the backbone model (FM [36]) recommends top-K items without considering bundle contexts. Compared with these individual MCR recommenders, *BUNT-Learn* achieves the best performance. For example, compared with model (b), where we only replace BUNT backbone with FM, ours improves accuracy from 0.239 to 0.727 on Steam. This shows that directly applying existing individual MCR recommenders in Bundle MCR is not optimal, and also shows the benefits of our BUNT design. Moreover, compared with bundle recommenders only recommending items (models (f)-(h)) ours introduces *question asking* and improves recommendation performances consistently, except for the recall score in iFashion. This is because we should replace recommending with asking, so recall may drop given fewer recommendation rounds (F1-Score is improved still).

6.5.4 Accuracy Curve with Conversation Rounds. The cumulative accuracy curves in Figure 3b show BUNT-All achieves the best results in beginning conversation rounds, then is outperformed by BUNT-Learn. This is because BUNT-Learn requires several rounds to ask questions and elicit preferences. Thus, BUNT-Learn in late rounds can recommend more accurately and surpasses baselines.

[9] More details of metric definitons, data processing, BUNT implementation and human evaluation setup are in https://github.com/AaronHeee/Bundle-MCR.
[10] https://stable-baselines3.readthedocs.io
[11] Given a slot x, initially all items in $\mathcal{B}^*$ are potential items, but some items are removed with the acceptance of items, categories and attributes in slot x.

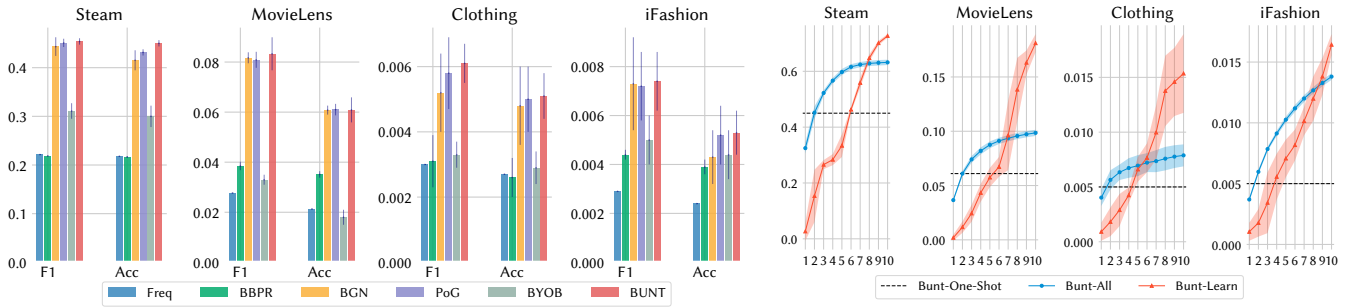
Table 2: Data Statistics, where # denotes quantity number, *U* denotes user, *I* denotes item, *B* denotes bundle, *C* denotes category and *A* denotes attributes. *B/U* represents the number of bundles per user, *B size* represents the average number of items per bundle.

Dataset	#U	#I	#B	#C	#A	#Inter	B/U	B Size	#Offline	#Online	#Valid	#Test
Steam	13,260	2,819	229	21	327	261,241	2.95	5.76	13,260	7,956	2,652	2,652
MovieLens	46,322	5,899	851,361	19	190	3,997,583	27.81	3.11	46,322	27,793	9,264	9,265
Clothing	19,065	25,408	79,610	668	4,027	285,391	5.03	3.17	19,065	11,439	3,813	3,813
iFashion	340,762	68,921	5,593,387	61	4,264	21,552,716	16.41	3.79	340,762	204,457	68,152	68,153

Table 3: BUNT and other individual conversational recommendation methods that are adopted for bundle settings. The best is bold.

		Steam				MovieLens			
Group	Method	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Individual Rec. Model	(a) FM-All	.149±.001	.611±.004	.239±.001	.138±.001	.019±.001	.087±.002	.031±.001	.017±.001
	(b) FM-Learn	.269±.019	.664±.018	.382±.016	.239±.001	.038±.002	.096±.008	.055±.005	.031±.002
	(c) EAR*	.186±.034	.592±.025	.282±.041	.166±.031	.036±.003	.099±.009	.053±.005	.029±.003
	(d) SCPR*	.173±.009	.544±.043	.262±.008	.151±.006	.044±.012	.110±.007	.063±.009	.032±.006
Bundle Rec. Model	(e) BUNT-One-Shot	.456±.006	.452±.007	.454±.007	.450±.006	.075±.007	.093±.006	.083±.007	.061±.005
	(f) BYOB-All	.328±.046	.799±.023	.463±.047	.323±.047	.020±.001	.113±.007	.034±.002	.018±.001
	(g) BGN-All	.568±.019	.919±.007	.702±.013	.567±.019	.073±.005	.216±.006	.109±.006	.070±.006
	(h) BUNT-All	.633±.012	.927±.002	.752±.008	.632±.012	.100±.004	.289±.004	.149±.004	.096±.003
	(i) BUNT-Learn	**.737±.003**	**.928±.015**	**.822±.006**	**.727±.006**	**.251±.015**	**.302±.013**	**.275±.013**	**.181±.008**

		Clothing				iFashion			
Group	Method	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Individual Rec. Model	(a) FM-All	.003±.001	.013±.001	.005±.001	.003±.001	.006±.001	.026±.001	.010±.001	.005±.001
	(b) FM-Learn	.006±.001	.010±.003	.008±.002	.004±.001	.008±.003	028±.002	.012±.002	.006±.002
	(c) EAR*	.011±.003	.022±.002	.014±.003	.008±.002	.017±.003	.026±.001	.020±.002	.010±.001
	(d) SCPR*	.013±.006	.028±.004	.018±.003	.009±.005	.014±.005	.032±.003	.019±.004	.010±.002
Bundle Rec. Model	(e) BUNT-One-Shot	.006±.001	.005±.001	.005±.001	.005±.001	.008±.002	.007±.001	.007±.001	.005±.002
	(f) BYOB-All	.002±.001	.010±.001	.003±.001	.002±.001	.005±.001	.023±.001	.008±.001	.004±.001
	(g) BGN-All	.009±.001	.023±.002	.013±.001	.008±.001	.011±.001	.032±.002	.016±.002	.010±.001
	(h) BUNT-All	.008±.001	.023±.002	.012±.002	.008±.001	.014±.001	**.043±.001**	.021±.001	.014±.001
	(i) BUNT-Learn	**.019±.003**	**.026±.008**	**.021±.005**	**.015±.004**	**.020±.001**	.035±.003	**.025±.001**	**.017±.001**



(a) BUNT performance compared with other bundle recommenders in *one-shot* bundle recommendation.

(b) Cumulative accuracy curve, i.e., accuracy after t rounds, $t \in [1, 10]$.

Figure 3: BUNT performance in *one-shot* setting, and cumulative accuracy curves.

Table 4: Ablation Studies (F1-score) to evaluate model architecture, fine-tuning (FT) and pre-training (PT).

Ablation	Steam	MovieLens
(a) Bunt-Learn	**.822±.006**	**.275±.013**
(b) w/o Long-term Pref.	.701±.080	.148±.010
(c) w/o Short-term Tags	.787±.013	.165±.012
(d) w/o Short-term Items	.330±.011	.084±.009
(e) replace Bunt π_I with FM	.382±.016	.032±.006

Ablation	Steam	MovieLens
(a) Bunt-Learn	**.822±.006**	**.275±.013**
(f) w/o FT π_M	.765±.002	.210±.002
(g) w/o FT π_I	.817±.003	.268±.007
(h) w/o FT $\pi_{\{A,C\}}$	.811±.001	.257±.003
(i) w/o FT All	.753±.008	.206±.008

Ablation	Steam	MovieLens
(a) Bunt-Learn	**.822±.006**	**.275±.013**
(j) w/o PT π_M	.807±.022	.258±.003
(k) w/o PT π_I	.056±.002	.008±.001
(l) w/o PT $\pi_{\{A,C\}}$	.815±.018	.171±.002
(m) w/o PT All	.003±.001	.001±.001

For example, on MovieLens, Bunt-Learn outperforms the baselines after $t = 6$.

6.5.5 Bunt-Learn Component Analysis. Compared with (a), (b)-(d) show the effectiveness of long-term preference and short-term context encoding; (e) indicates the importance of using bundle-aware models; (f)-(i) show the benefit of online fine-tuning, which helps π_M most because conversation management is hard to mimic in offline datasets, and π_M with only a binary action space is easier for online learning than other policies; (j)-(m) show pre-training is necessary, especially for item policy, because bundle generation is challenging to directly learn from online interactions with RL. This also indicates the proposed multiple cloze pre-training tasks are suitable for training Bundle MCR effectively.

6.6 Human Evaluation on Conversation Trajectories

Considering the cost of deploying real interactive Bundle MCRs, similar to [21, 46], we conduct human evaluation by letting real users rate the generated conversation trajectories from Section 6.5. From Steam and MovieLens datasets, we sample 1000 (in total) pairs of conversation trajectories from <Bunt-Learn, SCPR*> or <Bunt-Learn, FM-Learn> (SCPR* and FM-Learn are the best baselines using individual item recommenders). Each pair of conversation trajectories is posted to collect 5 answers from MTurk[12] workers, who are required to measure the subjective quality by browsing the conversations and selecting the best model from the given pair. We use the answers from high-quality workers who spend more than 30 seconds and the `LifeTimeAcceptanceRate` is 100%, and count the majority votes per pair. Lastly, we collected 388 valid results: <Bunt-Learn, SCPR*> votes are 121:88, and <Bunt-Learn, FM-Learn> votes are 110:69. This result shows the superiority of Bunt-Learn. Interestingly, the performance gap in human evaluation is not as large as results in simulators (e.g., on Steam, Bunt-Learn accuracy is 3x as FM-Learn).

7 CONCLUSION AND FUTURE WORK

In this work, we first extend existing Multi-Round Conversational Recommendation (MCR) settings to bundle recommendation scenarios, which we formulate as an MDP problem with multiple agents. Then, we propose a model architecture, Bunt, to handle bundle contexts in conversations. Lastly, to let Bunt learn bundle knowledge from offline datasets and an online environment, we propose a two-stage training strategy to train our Bunt model with multiple *cloze* tasks and multi-agent reinforcement learning respectively. We show the effectiveness of our model and training strategy on four offline bundle datasets and human evaluation. Since ours is the first work to consider conversation mechanisms in bundle recommendation, many research directions can be explored in the future. In Bunt, our question spaces are about categories and attributes, so how to use *free text* in bundle conversational recommendation is still an open question. Meanwhile, how to explicitly incorporate item relationships (e.g., substitutes, complements) in conversational bundle recommendation should be another interesting and challenging task. Moreover, since individual items can be treated as a special bundle, it is interesting to unify existing individual conversational recommenders into conversational bundle recommendation, i.e., augmenting conversational agents' abilities without extra cost.

[12]https://requester.mturk.com/

ACKNOWLEDGMENTS

We thank the reviewers for their insightful comments, and thank Yisong Miao for precious discussions on this project.

REFERENCES

[1] Peter Auer. 2002. Using Confidence Bounds for Exploitation-Exploration Tradeoffs. *J. Mach. Learn. Res.* 3 (2002), 397–422.
[2] Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. 2019. Personalized Bundle List Recommendation. *The World Wide Web Conference* (2019).
[3] Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. 2020. Bundle recommendation with graph convolutional networks. In *Proceedings of the 43rd international ACM SIGIR conference on Research and development in Information Retrieval*. 1673–1676.
[4] Jianxin Chang, Chen Gao, Xiangnan He, Yong Li, and Depeng Jin. 2020. Bundle Recommendation with Graph Convolutional Networks. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).
[5] Olivier Chapelle and Lihong Li. 2011. An Empirical Evaluation of Thompson Sampling. In *NIPS*.
[6] Liang Chen, Yang Liu, Xiangnan He, Lianli Gao, and Zibin Zheng. 2019. Matching User with Item Set: Collaborative Bundle Recommendation with Deep Attention Network. In *IJCAI*.
[7] Qibin Chen, Junyang Lin, Yichang Zhang, Ming Ding, Yukuo Cen, Hongxia Yang, and Jie Tang. 2019. Towards Knowledge-Based Recommender Dialog System. *ArXiv* abs/1908.05391 (2019).
[8] Wen Chen, Pipei Huang, Jiaming Xu, Xin Ze Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. 2019. POG: Personalized Outfit Generation for Fashion Recommendation at Alibaba iFashion. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).
[9] Y. Chen, K. Tang, Ren-Jie Shen, and Ya-Han Hu. 2005. Market basket analysis in a multiple store environment. *Decis. Support Syst.* 40 (2005), 339–354.
[10] Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *SSST@EMNLP*.
[11] Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards Conversational Recommender Systems. *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (2016).
[12] Qilin Deng, Kai Wang, Minghao Zhao, Runze Wu, Yu Ding, Zhene Zou, Yue Shang, Jianrong Tao, and Changjie Fan. 2021. Build Your Own Bundle - A Neural

Combinatorial Optimization Method. *Proceedings of the 29th ACM International Conference on Multimedia* (2021).

[13] Qilin Deng, Kai Wang, M. Zhao, Zhene Zou, Runze Wu, Jianrong Tao, Changjie Fan, and Liang Chen. 2020. Personalized Bundle Recommendation in Online Games. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* (2020).

[14] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified Conversational Recommendation Policy Learning via Graph-based Reinforcement Learning. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *ArXiv* abs/1810.04805 (2019).

[16] R. Garfinkel, R. Gopal, Arvind K. Tripathi, and Fang Yin. 2006. Design of a shopbot and recommender system for bundle purchases. *Decis. Support Syst.* 42 (2006), 1974–1986.

[17] F. Maxwell Harper and Joseph A. Konstan. 2015. The MovieLens Datasets: History and Context. *ACM Trans. Interact. Intell. Syst.* 5 (2015), 19:1–19:19.

[18] Simon Haykin. 1994. *Neural networks: a comprehensive foundation*. Prentice Hall PTR.

[19] Zhankui He, Handong Zhao, Zhe Lin, Zhaowen Wang, Ajinkya Kale, and Julian McAuley. 2021. Locker: Locally Constrained Self-Attentive Sequential Recommendation. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*. 3088–3092.

[20] Haoji Hu and Xiangnan He. 2019. Sets2Sets: Learning from Sequential Sets with Neural Networks. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2019).

[21] D. Jannach and Ahtsham Manzoor. 2020. End-to-End Learning for Conversational Recommendation: A Long Way to Go?. In *IntRS@RecSys*.

[22] Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul A. Crook, Y-Lan Boureau, and J. Weston. 2019. Recommendation as a Communication Game: Self-Supervised Bot-Play for Goal-oriented Dialogue. In *EMNLP/IJCNLP*.

[23] Wang-Cheng Kang and Julian McAuley. 2018. Self-Attentive Sequential Recommendation. *2018 IEEE International Conference on Data Mining (ICDM)* (2018), 197–206.

[24] Gary King and Langche Zeng. 2001. Logistic Regression in Rare Events Data. *Political Analysis* 9 (2001), 137 – 163.

[25] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *CoRR* abs/1412.6980 (2015).

[26] 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. *Proceedings of the 13th International Conference on Web Search and Data Mining* (2020).

[27] Wenqiang Lei, Gangyi Zhang, Xiangnan He, Yisong Miao, Xiang Wang, Liang Chen, and Tat-Seng Chua. 2020. Interactive Path Reasoning on Graph for Conversational Recommendation. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (2020).

[28] Lei Li, Yongfeng Zhang, and Li Chen. 2021. Personalized Transformer for Explainable Recommendation. In *ACL/IJCNLP*.

[29] Raymond Li, S. Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and C. Pal. 2018. Towards Deep Conversational Recommendations. *ArXiv* abs/1812.07617 (2018).

[30] Shihao Li, Dekun Yang, and Bufeng Zhang. 2020. MRIF: Multi-resolution Interest Fusion for Recommendation. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval* (2020).

[31] A. Marchetti-Spaccamela and C. Vercellis. 1995. Stochastic on-line knapsack problems. *Mathematical Programming* 68 (1995), 73–104.

[32] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-Based Recommendations on Styles and Substitutes. *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015).

[33] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin A. Riedmiller, Andreas Fidjeland, Georg Ostrovski, Stig Petersen, Charlie Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. 2015. Human-level control through deep reinforcement learning. *Nature* 518 (2015), 529–533.

[34] Vinod Nair and Geoffrey E. Hinton. 2010. Rectified Linear Units Improve Restricted Boltzmann Machines. In *ICML*.

[35] Apurva Pathak, Kshitiz Gupta, and Julian McAuley. 2017. Generating and Personalizing Bundle Recommendations on Steam. *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2017).

[36] Steffen Rendle. 2010. Factorization Machines. *2010 IEEE International Conference on Data Mining* (2010), 995–1000.

[37] Steffen Rendle, Christoph Freudenthaler, Zeno Gantner, and Lars Schmidt-Thieme. 2009. BPR: Bayesian Personalized Ranking from Implicit Feedback. In *UAI*.

[38] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal Policy Optimization Algorithms. *ArXiv* abs/1707.06347 (2017).

[39] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. 2019. BERT4Rec: Sequential Recommendation with Bidirectional Encoder Representations from Transformer. *Proceedings of the 28th ACM International Conference on Information and Knowledge Management* (2019).

[40] Yueming Sun and Yi Zhang. 2018. Conversational Recommender System. *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018).

[41] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *NIPS*.

[42] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[43] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc.

[44] Xiang Wang, Tinglin Huang, Dingxian Wang, Yancheng Yuan, Zhenguang Liu, Xiangnan He, and Tat-Seng Chua. 2021. Learning Intents behind Interactions with Knowledge Graph for Recommendation. *Proceedings of the Web Conference 2021* (2021).

[45] Agung Toto Wibowo, Advaith Siddharthan, Judith Masthoff, and Chenghua Lin. 2018. Incorporating Constraints into Matrix Factorization for Clothes Package Recommendation. *Proceedings of the 26th Conference on User Modeling, Adaptation and Personalization* (2018).

[46] Yikun Xian, Handong Zhao, Tak Yeon Lee, Sungchul Kim, Ryan A. Rossi, Zuohui Fu, Gerard de Melo, and S. Muthukrishnan. 2021. EXACTA: Explainable Column Annotation. *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining* (2021).

[47] M. Xie, L. Lakshmanan, and P. Wood. 2010. Breaking out of the box of recommendations: from items to packages. In *RecSys '10*.

[48] Zhihui Xie, Tong Yu, Canzhe Zhao, and Shuai Li. 2021. Comparison-based Conversational Recommender System with Relative Bandit Feedback. *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2021).

[49] Markus Zanker, Markus Aschinger, and Markus Jessenitschnig. 2010. Constraint-based personalised configuring of product and service bundles. *International Journal of Mass Customisation* 3, 4 (2010), 407–425.

[50] Min-Ling Zhang and Zhi-Hua Zhou. 2014. A Review on Multi-Label Learning Algorithms. *IEEE Transactions on Knowledge and Data Engineering* 26 (2014), 1819–1837.

[51] Xiaoying Zhang, Hong Xie, Hang Li, and John C.S. Lui. 2020. Conversational Contextual Bandit: Algorithm and Application. *Proceedings of The Web Conference 2020* (2020).

[52] Yongfeng Zhang, X. Chen, Qingyao Ai, Liu Yang, and W. Croft. 2018. Towards Conversational Search and Recommendation: System Ask, User Respond. *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018).

[53] Yiming Zhang, Lingfei Wu, Qi Shen, Yitong Pang, Zhihua Wei, Fangli Xu, Bo Long, and Jian Pei. 2022. Multiple Choice Questions based Multi-Interest Policy Learning for Conversational Recommendation. *Proceedings of the ACM Web Conference 2022* (2022).

[54] Tao Zhu, Patrick Harrington, Junjun Li, and Lei Tang. 2014. Bundle recommendation in ecommerce. In *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. 657–666.