LatentCRS: Latent Intent-enhanced Conversational Recommendation System with Large Langauge Models

Anonymous ACL submission

Abstract

Conversational Recommender Systems (CRS) leverage interactive dialogues to deliver personalized recommendations, with large language models (LLMs) enhancing their natural language understanding and response generation. However, LLMs struggle to utilize non-verbal user-item interactions from user behavior, as they primarily process textual information and lack explicit mechanisms to interpret these interactions. This gap is critical since such behavioral data is essential for accurate recommendations. To bridge this gap, we propose a Latent Intent-enhanced Conversational Recommendation System with Large Language Models (LATENTCRS) that integrates LLMs with recommendation models through latent user intents. Specifically, LATENTCRS employs a variational expectation-maximization framework: a recommendation model infers the intent distribution from collaborative data, which then guides the refinement of behavioral and textual information to generate recommendations. Crucially, our approach avoids costly LLM fine-tuning, ensuring computational efficiency. Extensive experiments demonstrate that LATENTCRS consistently outperforms state-of-the-art baselines in both single-turn and multi-turn recommendation scenarios.

1 Introduction

Conversational recommender systems (CRS) engage users in dynamic, interactive dialogues to provide personalized recommendations (Sun and Zhang, 2018; Zhang et al., 2018), which enable users to actively express their interests through natural language. Large language models (LLMs) have highlighted their remarkable capabilities in dialogue (Kojima et al., 2022; Team et al., 2024), presenting new opportunities for capturing users' evolving interests through conversation. By enabling human-like interactions and improving natural language understanding, LLMs significantly enhance the dialogue quality and contextual awareness (Yang and Chen, 2024; Gao et al., 2023).

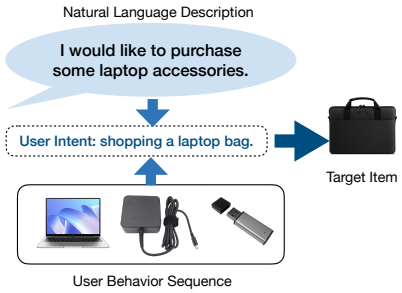

Figure 1: The User Intent from Behavior Sequence and Natural Language Descriptions.

Unlike standard dialogue tasks, which depend on natural language processing ability, CRS also demand strong recommendation capabilities. However, LLMs have yet to reach comparable performance with dedicated recommendation models (Liu et al., 2023; Zhang et al., 2024b; Kim et al., 2024). A fundamental limitation is that accurate recommendations depend significantly on collaborative information, namely the latent patterns mined from interactions between users and items. These patterns reveal behavioral correlations extending beyond purely semantic relationships. Due to the inherent mismatch between natural language modeling and user behavior modeling, LLMs struggle to detect such collaborative information. For instance, consumers who purchase diapers on Friday often also buy beer (Padmanabhan and Tuzhilin, 1999). This behavior is effectively captured by recommendation models but may confuse LLMs, as these items lack semantic similarity. Given the complementary strengths of LLMs and recommendation models, this raises the question: *How can we effectively integrate LLMs with recommendation models to leverage their respective advantages?*

Existing approaches for integrating LLMs with recommendation models often oversimplify user in-

teractions, limiting their effectiveness. Prior work either reduces user dialogue to binary feedback signals (likes/dislikes) to align the two models (Christakopoulou et al., 2016, 2018; Hu et al., 2022) or employs LLMs as agents that mechanically trigger recommendation tools through binary decisions (Gao et al., 2023; Huang et al., 2023). While these methods bridge the gap between modalities, they discard rich contextual information from dialogues and weaken user engagement. More recent work projects collaborative signals into the LLM's embedding space and fine-tunes fine-tuning LLMs on behavioral data (Yang et al., 2024; Yang and Chen, 2024). While this narrows the gap, the fine-tuning stage is computationally expensive because of the large parameter count of LLMs.

To overcome these limitations, we propose a new paradigm that unifies user behavior and natural language through their shared latent intent. As shown in Figure 1, a user's intent (needing a laptop bag) may manifest semantically ("I need laptop accessories") and behaviorally (purchasing a laptop and accessories but no bag). This dual manifestation enables our model to capture complementary signals from both natural language and user behavior. Crucially, because this intent space operates independently of LLM, our approach achieves effective integration without costly LLM tuning, significantly reducing computational overhead.

Specifically, we propose the Latent Intent-enhanced Conversational Recommendation System with Large Language Models (LATENTCRS). Since user intent is inherently latent and unobservable, we represent it as a learnable mixture of behavioral anchors derived from clustering user embeddings generated by a pretrained recommendation model. To connect the recommendation model with the LLM via this intent representation, we adopt a variational expectation–maximization framework composed of two modules: an inference module (IM) and a generative module (GM). The IM uses the pretrained recommendation model to infer a distribution over the intent space from user behavior data, capturing collaborative signals. The GM then merges this inferred distribution with the user behavior encodings by the recommendation model and the dialogue context encoding by the LLM to generate recommendations that combine both collaborative and conversational information. Experiments on single-turn and multi-turn scenarios show that LATENTCRS outperforms state-of-the-art baselines. Ablation studies verify the importance of modeling user intent. And efficiency analyses reveal LATENTCRS's substantially reduced computational overhead, which is a crucial advantage for practical deployment.

2 Related Works

Conversational recommender systems (CRS) have evolved from pipeline architectures that treat dialogue management and recommendation as separate tasks to multi-turn settings using reinforcement learning (Sun and Zhang, 2018; Bi et al., 2019; Zhang et al., 2020; Lei et al., 2020; Li et al., 2018). These early methods limited user feedback to simple signals, such as like or dislike, which reduced the richness of the collected preferences. With the rise of large language models (LLMs), new designs have emerged. Some treat an LLM as an agent and the recommendation engine as an external tool (Gao et al., 2023; Huang et al., 2023), while others use the ReAct framework to combine planning and action within a single LLM-driven pipeline (Yao et al., 2023; He et al., 2024). Although these integrated approaches allow more flexible interaction, they still ignore collaborative information from recommender systems. We address these limitations by introducing a lightweight method that uses latent user intent as a bridge between collaborative signals and natural language. Another set of related works are intent modeling for recommendation. These methods usually extract latent interests from user behavior (Li et al., 2019, 2021; Chen et al., 2022; Sun et al., 2024). These approaches excel at deriving intent from interaction data, but they do not link user behavior with language expressions in conversation. In contrast, our work uses user intent not only to summarize past actions but also to guide natural language prompts, unifying behavior-driven and language-driven signals within a single framework. More details about the related works are shown in Appendix A.

3 Problem Definition and Preliminaries

3.1 Task Formulation

A conversational recommender system (CRS) comprises two main components, user behavior modeling and interactive conversation. The behavior modeling component defines a set of users $\mathcal{U}$ and items $\mathcal{V}$, together with historical interaction data $\mathcal{D}$. To capture temporal changes in user intent, we group $\mathcal{D}$ by user in chronological order. Each user $u \in \mathcal{U}$ is associated with an interaction se-

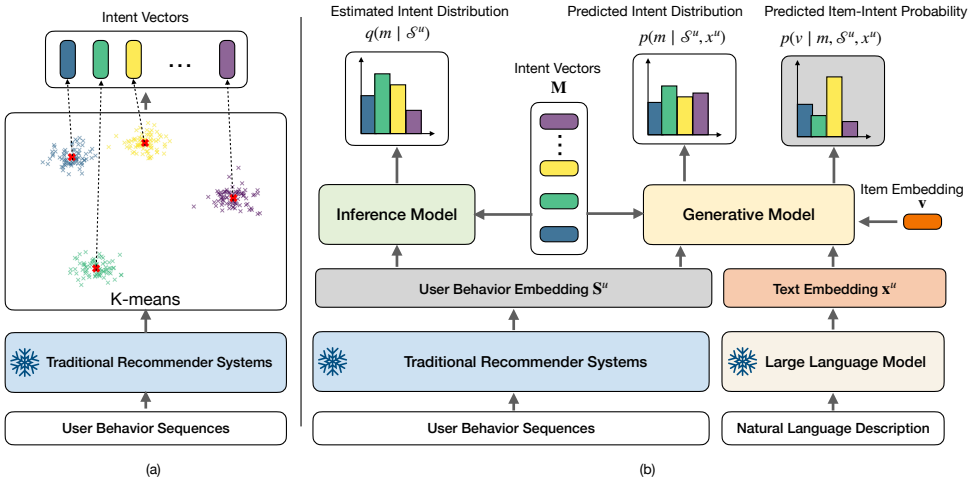

Figure 2: Overall Structure of LATENTCRS: (a) Representation of Intent in Latent Space; (b) Structure of the Inference and Generative Models.

quence $\mathcal{S}^u = [s_1^u, s_2^u, \ldots, s_{|\mathcal{S}^u|}^u]$, where s_t^u denotes the item interacted with at time step t, and $|\mathcal{S}^u|$ is the length of this sequence. The interactive conversation component models the dialogue between the CRS and the user. For a user u, the conversation is denoted $C^u = \{(x_j^u, r_j^u)\}_{j=1}^N$, where x_j^u is the user input at turn j expressing intent in natural language, and r_j^u is the system response. We define the aggregated intent descriptor x^u by concatenating all user inputs x_j^u across the turns.

CRS use both the interaction history and the conversational context to refine item recommendations. We formulate this as a maximum likelihood estimation problem, which aim to learn model parameters θ that maximize the probability of observed interactions. Formally, it is represented as follows:

$$\max_{\theta}; \prod_{(u,v)\in\mathcal{D}} p\big(v \mid \mathcal{S}^u, x^u; \theta\big) \tag{1}$$

Here, θ denotes the parameters of the conversational recommendation model. For simplicity, we omit θ in the subsequent formulas.

3.2 Latent Intent Modeling in CRS

As discussed in Section 1, we utilize user intent as a bridge to incorporate recommendation models with LLMs. We assume that the distribution of latent intent can be represented as the distribution on archor vector $M = \{m_i\}_{i=1}^K$, where K represents the total number of anchor intent vector and m_i represents the ith anchor intent. And the user intents are represented by both the natural language descriptions from the conversation and the collaborative information derived from the user behavior patterns. Incorporating the latent intent representation, the probability objectives of CRS can be rewritten as follows:

$$p(v \mid \mathcal{S}^u, x^u) = \mathbb{E}_{(M)}\Big(p(v \mid M, \mathcal{S}^u, x^u) p(M \mid \mathcal{S}^u, x^u)\Big) \tag{2}$$

Here, $\mathcal{S}^u$ denotes the sequence of items interacted with by user u, while x^u represents the natural language description of the user's intent. v refers to an item from the set of all items $\mathcal{V}$.

4 Method

The overall architecture of LATENTCRS is illustrated in Figure 2. We begin by introducing the method for representing latent intent space by constructing anchor vectors. Next, we describe the variational Expectation-Maximization (EM) framework, including its components and interactively training strategy. Finally, we introduce three variants of our framework to demonstrate its adaptability under different LLM usage strategies.

4.1 Intent Representation

The foundation of LATENTCRS lies in effectively representing user intent. Directly optimizing the probability in Equation (2) is impractical because user intent is unobserved, creating a circular dependency: the model cannot be optimized without the intent distribution M, while accurate estimation of M itself depends on having a well-trained model. To break this circular dependency, we adopt a two-stage approach, as shown in Figure 2(a). First, we leverage a pre-trained recommendation model

and utilize its user embedding as the user behavior sequence embeddings $\mathbf{S}^u$, which encode both collaborative filtering signals and historical interaction patterns. These embeddings serve as observable proxies for the underlying user intents. We denote the embedding of variable x in bold as $\mathbf{X}$ and we adopt this type of notation throughout the following sections. To transform these continuous embeddings into a discrete and processable intent space, we apply K-Means clustering:

$$\mathbf{M} = \text{K-Means}\big(\{\mathbf{S}^u\}_{u\in\mathcal{U}}\big) \tag{3}$$

The resulting cluster centroids $\mathbf{M} = \{\mathbf{m}_i\}_{i=1}^K$, where each vector $\mathbf{m}_i \in \mathbb{R}^{d_u}$ serve as anchor vectors that define our latent intent space. The cluster centroids capture meaningful patterns and provide a computationally efficient discrete approximation of the continuous intent space. By establishing this anchor-based representation, we create a foundation for subsequent model optimization.

4.2 Recommendation with Intent

Based on the intent representation above, we optimize the CRS objective in Equation (1) within a variational EM framework. Direct maximization of the log-likelihood requires summing over all latent intents, which is unavailable as there is no available data in the latent intents space. The EM algorithm addresses this by breaking the problem into two tractable steps. In the intent reference step (E-step), we get the input from a pre-trained recommendation model to estimate the posterior of each intent anchor. In the recommendation learning step (M-step), we get information from both the recommendation model and the LLM to maximize the expected log-likelihood under the estimated posterior. Here, we maximize the Evidence Lower Bound (ELBO) rather than the true log-likelihood. Formally, we define the ELBO as:

$$\begin{aligned}\text{ELBO}^{uv} = &\sum_{j=1}^{K} q\big(m_j \mid \mathcal{S}^u\big) \\ &\Big(\log p\big(v \mid m_j, \mathcal{S}^u, x^u\big) \\ &+ \log p\big(m_j \mid \mathcal{S}^u, x^u\big) - \log q\big(m_j \mid \mathcal{S}^u\big)\Big)\end{aligned} \tag{4}$$

Here, K is the number of intents. The ELBO is equal to the original log-likelihood $\log p\big(v \mid \mathcal{S}^u, x^u\big)$ only when $q(m_j \mid \mathcal{S}^u) = p\big(m_j \mid v, \mathcal{S}^u, x^u\big)$. Maximizing this bound yields stable parameter updates and preserves the convergence guarantees of the EM algorithm. The following sections describe in detail the intent inference step, the recommendation learning step, and our overall optimization procedure.

4.2.1 Recommendation Learning Step

In the recommendation learning step (M-Step), we optimize the ELBO with the estimated intent distribution from the intent learning step. From Equation (4), the recommendation learning step should predict $p(m_j \mid \mathcal{S}^u, x^u)$ and $p(v \mid m_j, \mathcal{S}^u, x^u))$. We first introduce how to get the collaborative information $\mathcal{S}^u$ from recommendation models and dialogue context information x^u from LLM, then we introduce how to predict $p(m_j \mid \mathcal{S}^u, x^u)$ and $p(v \mid m_j, \mathcal{S}^u, x^u))$. Finally, we introduce the actual optimization function.

For $\mathcal{S}^u$, we utilize the user representation embedding from the same recommendation model in Section 4.1. For x^u, we follow recent studies that obtain embeddings from LLMs (Jiang et al., 2024), using a simple prompt *"This sentence: '*sentence*' means in one word:"* to extract a semantic embedding $\mathbf{x}^u$. We then estimate $p(m_j \mid \mathcal{S}^u, x^u)$ with a neural network $f(\mathbf{m}_j, \mathbf{S}^u, \mathbf{x}^u)$:

$$\begin{aligned}p(m_j \mid \mathcal{S}^u, x^u) &= f(\mathbf{m}_j, \mathbf{S}^u, \mathbf{x}^u) \\ &= \text{softmax}\big(\text{FFN}([\mathbf{S}^u; \mathbf{x}^u])(W_m\mathbf{m}_j)^T\big)\end{aligned} \tag{5}$$

where $\text{FFN}(\cdot)$ is a feed-forward network, $[\,;\,]$ denotes concatenation, and W_m is a learnable parameter matrix. We also include a linear projection layer to align the dimensionality of $\mathbf{S}^u$ and $\mathbf{x}^u$, though this layer is omitted in the equation for brevity.

Intuitively, $p(v \mid m_j, \mathcal{S}^u, x^u)$ needs to be calculated by softmax, but it becomes computationally expensive when the number of items is large. Thus, instead of directly modeling $p(v \mid m_j, \mathcal{S}^u, x^u)$, we use Information Noise Contrastive Estimation (infoNCE) (van den Oord et al., 2018) to estimate $\frac{p(v|m_j,\mathcal{S}^u,x^u)}{p(v)}$. We use a neural network $g(\mathbf{v}, \mathbf{m}_j, \mathbf{S}^u, \mathbf{x}^u)$ to estimate it

$$\begin{aligned}g(\mathbf{v},&\mathbf{m}_j, \mathbf{S}^u, \mathbf{x}^u) = \\ &(W_j\mathbf{m}_j)\big(W_v\mathbf{v}\big)^T \cdot \text{FFN}([\mathbf{S}^u; \mathbf{x}^u])(W_m\mathbf{m}_j)^T\end{aligned} \tag{6}$$

Here, $\mathbf{v}, \mathbf{m}_j, \mathbf{S}^u, \mathbf{x}$ represent the embeddings of $v, m_j, \mathcal{S}^u, x^u$, respectively, and W_j, W_v, W_m are model parameters. For efficient training, we share W_m with $f(\mathbf{m}_j, \mathbf{S}^u, \mathbf{x}^u)$ in Equation (5). As n increases, the infoNCE loss converge to the same solution as the maximum likelihood estimation (Ma

and Collins, 2018). Moreover, as the ultimate goal of the recommendation learning step is to make accurate recommendations, we add a recommendation auxiliary objective which optimizes the maximization of the likelihood of $p(v \mid \mathcal{S}^u, x^u)$.

For each real data point (u, v), we randomly sample n elements from $p(v)$ to form a negative set $\mathcal{V}_{neg}^{uv}$. The training loss is the combination of ELBO and recommendation auxiliary loss. To display more clearly, we define the following:

$$Z(v,m_j,s^u,x^u) = \exp(g(\mathbf{v},\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)) + \sum_{v_{neg}\in\mathcal{V}_{neg}^{uv}} \exp\big(g(\mathbf{v}_{neg},\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)\big)$$

$$\mathcal{L}_{infoNCE} = -\sum_{(u,v)\in\mathcal{D}}\sum_{j=1}^{K}\Big(q(m_j \mid \mathcal{S}^u) \log\Big(\frac{\exp\big(g(\mathbf{v},\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)\big)}{Z(v,m_j,s^u,x^u)}\Big)\Big) \tag{7}$$

$$\mathcal{L}_{\text{rec}}^M = \sum_{u\in\mathcal{U}}\Big(\log h(\mathbf{v},\mathbf{S}^u,\mathbf{x}^u) - \log\big(h(\mathbf{v},\mathbf{S}^u,\mathbf{x}^u) + \sum_{v^-\in\mathcal{V}_{neg}} h(\mathbf{v}^-,\mathbf{S}^u,\mathbf{x}^u)\big)\Big) \tag{8}$$

Here, $h(\mathbf{v},\mathbf{S}^u,\mathbf{x}^u) = \sum_{j=1}^{k}\big(f(\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)g(\mathbf{v}, \mathbf{m}_j,\mathbf{S}^u,x^u)\big)$. $f(\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)$ are defined in Equation (5) and $g(\mathbf{v},\mathbf{m}_j,\mathbf{S}^u,x^u)$ in Equation (6). Consequently, the training loss is:

$$\mathcal{L}^M = \mathcal{L}_{infoNCE} + \lambda\text{KL}\big(\hat{q}(m \mid \mathcal{S}^u) \,\|\, p(m_j \mid \mathcal{S}^u,x^u)\big) + \alpha^M\mathcal{L}_{\text{rec}}^M \tag{9}$$

Here, $\hat{q}(m \mid \mathcal{S}^u)$ is the intent distribution estimated by the model trained in intent inference step, and $p(m_j \mid \mathcal{S}^u, x^u)$ is given by Equation (5). α^M and λ are weight hyperparameters. Unlike the standard evidence lower bound (ELBO), we introduce λ to better regulate the training process.

4.2.2 Intent Inference Step

The goal of the intent inference step (E Step) is to estimate the intent distribution based on a recommendation model. The core in this step is to calculate the probability $q(m \mid \mathcal{S}^u)$. Similar to the recommendation learning step, we first obtain the representation embedding $\mathbf{S}^u$ from a recommendation model, then apply a neural network $l(\mathbf{m},\mathbf{s}^u)$ to generate the estimates. Formally,

$$l(\mathbf{m},\mathbf{s}^u) = \text{softmax}\Big(\frac{W_q\mathbf{S}^u\big(W_k\mathbf{m}_j\big)^T}{\sqrt{d_m}}\Big) \tag{10}$$

where W_q and W_k are learnable parameters, d_m is the dimension of the intent embedding. The intent inference step aims to maximize the Evidence Lower Bound (ELBO). This is equivalent to minimizing the following loss function:

$$\mathcal{L}_{ELBO}^E = \sum_{(u,v)\in\mathcal{D}} \text{KL}\Big(q(m \mid \mathcal{S}^u) \,\|\, p(m_j \mid \mathcal{S}^u,x^u)p(v \mid m_j,\mathcal{S}^u,x^u)\Big) \tag{11}$$

where $\text{KL}(\cdot)$ is the KL divergence, $q(m \mid \mathcal{S}^u)$, $p(m_j \mid \mathcal{S}^u, x^u)$ and $p(v \mid m_j, \mathcal{S}^u, x^u)$ come from Equations (10) (5) and (6), respectively.

Moreover, we also utilize an auxiliary recommendation objective, under the assumption that more accurate intent distributions yield better recommendation performance. Formally, we represent the recommendation score under certain intent distribution and the training objective as:

$$\mathbf{r}^u = \sum_{j=1}^{K} q(m_j \mid \mathcal{S}^u)\,\mathbf{m}_j \tag{12}$$

$$\mathcal{L}_{\text{rec}}^E = \sum_{u\in\mathcal{U}}\Big(\log\exp(\mathbf{r}^u\,W_e\,\mathbf{v}) - \log\big(\exp(\mathbf{r}^u\,W_e\,\mathbf{v}) + \sum_{v^-\in\mathcal{V}_{neg}}\exp(\mathbf{r}^u\,W_e\,\mathbf{v}^-)\big)\Big) \tag{13}$$

Here, $\mathbf{m}_j$ denoting the embedding of anchor intent m_j, $\mathbf{v}$ is the embedding of an item v, and $\mathcal{V}_{neg}$ is a set of negative items sampled according to the procedure from Section 4.2.1. The training loss is:

$$\mathcal{L}^E = \mathcal{L}_{\text{ELBO}}^E + \alpha^E\,\mathcal{L}_{\text{rec}}^E \tag{14}$$

where α^E is a hyperparameter that balances the ELBO term and the recommendation performance term. Since the intent distribution estimates are unreliable at the start of training, we only use $\mathcal{L}_{\text{rec}}^E$ to train until performance stabilizes.

4.2.3 Overall Optimization

As shown in Algorithm 1, we first train $l(\mathbf{m},\mathbf{s}^u)$ in Equation (10) only with $\mathcal{L}_{rec}^E$. Next, we train $f(\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)$ and $g(\mathbf{v},\mathbf{m}_j,\mathbf{S}^u,\mathbf{x}^u)$ in Equation (5) and (6) using the intent distribution estimated by $l(\mathbf{m},\mathbf{s}^u)$. This two-step initialization makes sure both modules yield stable outputs. Finally, we alternate between the Intent Inference Step (E-Step) and the Recommendation Learning Step (M-Step) until the model converges.

Algorithm 1: Optimization Steps

Input: A user set $\mathcal{U}$, an item set $\mathcal{V}$, a sequence of items $\mathcal{S}^u$ for each user u, and a natural language description of each user's interests x^u.
Output: $p(v \mid \mathcal{S}^u, x^u)$ for each $v \in \mathcal{V}$
while *not converged* **do**
  Train the inference model q using $\mathcal{L}^E_{rec}$ in Equation (13).
while *not converged* **do**
  Estimate $\hat{q}(m \mid \mathcal{S}^u)$ with the inference model;
  Fix the inference model and update the prior model using $\mathcal{L}^M$ in Equation (9).
while *not converged* **do**
  $\odot$ **Intent Inference Step (E Step)**;
  Estimate $\hat{p}(m_j \mid \mathcal{S}^u, x^u)$ and $\hat{p}(v \mid m_j, \mathcal{S}^u, x^u))$ with the prior model;
  Fix the generative model and update the inference model using $\mathcal{L}^E$ in Equation (14);
  $\odot$ **Recommendation Learning Step (M Step)**;
  Estimate $\hat{q}(m \mid \mathcal{S}^u)$ using the inference model;
  Fix the inference model and update the generative model using $\mathcal{L}^M$ in Equation (9).
Predict $p(v \mid \mathcal{S}^u, x^u)$ with the generative model by Equation (2).

4.3 Multi-Turn Conversation and Variants

In this section, we describe how to apply the trained model in a multi-turn interaction setting. We introduce three variants based on different strategies for leveraging the large language model: Basic usage (LATENTCRS$_B$), Filtering by user constraints (LATENTCRS$_F$), and Re-ranking recommendations (LATENTCRS$_V$). For LATENTCRS$_B$, the LLM conducts the dialogue but does not participate in any post-processing of the recommendation outputs. For LATENTCRS$_F$, the LLM extracts strict requirements mentioned by the user such as a minimum storage capacity for a smartphone. And we filter the candidate recommendations to ensure they all satisfy these requirements. For LATENTCRS$_V$, the LLM re-orders the initial set of recommendations, allowing it to resolve intents that the user expresses in a vague.

5 Experimental Setups

5.1 Evaluation Settings

Collecting actual user responses from real-world users is prohibitively expensive. Following prior work, we instead adopt a user simulation strategy that leverages LLMs to emulate user behavior (Zhang et al., 2024a; Yoon et al., 2024). Specifically, we provide Gemini-1.5-pro (Team et al., 2024) with role-playing prompts to generate realistic user interactions. The details are shown in Appendix D. We evaluate our system under two settings: **One-Turn Recommendation** and **Multi-Turn Recommendation** (Jin et al., 2023; Huang et al., 2023). In the One-Turn setting, users express their preferences in a single natural language query, and the Conversational Recommender System (CRS) must generate recommendations based solely on this input. In contrast, the Multi-Turn setting allows for iterative dialogue, where users refine their preferences through successive interactions until they receive satisfactory recommendations.

Table 1: Dataset Information.

	Movielens-1M	Video Games	CDs and Vinyl
Number of users	6034	15582	104544
Number of items	3522	7233	76616
Actions	575272	122179	1259069
Avg. behavior length	95.34	7.84	12.04
Sparsity	97.30%	99.89%	99.98%

5.2 Datasets

We conduct experiments on three distinct datasets (Hou et al., 2024): Movielens-1M (movielens)[1], Amazon VideoGames (VideoGames), and Amazon CDs (CDs) (Hou et al., 2024). Each dataset contains user-item interaction history and item metadata, and we adhere to their respective licenses and privacy policies. The key statistics are summarized in Table 1. Due to budget constraints, we also follow previous works by sampling 500 instances from the test set for evaluation the LLM based methods and the multi-turn setting, which are represented as Dataset Name$_{sample}$. More details are shown in Appendix B.

5.3 Baselines

To evaluate the effectiveness of LATENTCRS, we compare it with two categories of methods: traditional conversational recommendation models, and LLM-based conversational methods. We restrict our comparison to models that do not require modifications to the parameters of the LLMs. For recommendation methods, we compare **CRM** (Sun and Zhang, 2018), **UNICORN** (Deng et al., 2021), and **CRIF** (Hu et al., 2022). For LLM-based methods, we compare with **Llama 3.1-8B** (Dubey et al., 2024), **Gemini 1.5 pro** (Team et al., 2024), **GPT-4** (Achiam et al., 2023), **Chat-Rec** (Gao et al., 2023), and **InteRecAgent** (Huang et al., 2023). More details about these baselines are shown in Appendix C.

[1] https://grouplens.org/datasets/movielens/1m/

Table 2: Performance comparison on full datasets under one turn settings.

	Movielens-1M				VideoGames				CDs			
	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20
CRM	0.1165	0.0690	0.3273	0.1279	0.0024	0.0014	0.0090	0.0032	0.0089	0.0055	0.0297	0.0111
UNICORN	0.1215	0.0734	0.3086	0.1274	0.1138	0.0753	0.2317	0.1099	**0.0839**	**0.0518**	**0.1911**	**0.0833**
CRIF	0.0784	0.0489	0.2363	0.0926	0.0427	0.0267	0.1058	0.0443	0.0157	0.0100	0.0442	0.0180
LatentCRS$_B$	**0.1437**	**0.0921**	**0.3305**	**0.1449**	**0.1378**	**0.0964**	**0.2598**	**0.1310**	0.0485	0.0326	0.1083	0.0494

Table 3: Performance comparison on sampled datasets under one turn settings.

	Movielens$_{sample}$				VideoGames$_{sample}$				CDs$_{sample}$			
	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20
Llama 3.1-8B	0.054	0.0396	0.092	0.0507	0.020	0.0143	0.034	0.0182	0.012	0.0074	0.016	0.0085
Gemini 1.5 pro	0.068	0.0414	0.096	0.0604	0.034	0.0216	0.062	0.0409	0.018	0.0136	0.088	0.0433
GPT4	0.064	0.0421	0.102	0.0622	0.032	0.0204	0.058	0.0400	0.028	0.0204	0.104	0.0476
Chat-Rec	0.086	0.0546	0.150	0.0950	0.038	0.0268	0.080	0.0630	0.016	0.0111	0.108	**0.0787**
InteRecAgent	0.006	0.0043	0.022	0.0087	0.014	0.0086	0.024	0.0118	0.018	0.0129	0.030	0.0163
LatentCRS$_B$	0.142	0.0879	**0.340**	0.1426	0.140	0.1008	**0.250**	0.1326	0.040	0.0270	0.106	0.0454
LatentCRS$_F$	**0.180**	**0.1209**	0.318	**0.1604**	0.146	0.1072	0.230	0.1347	**0.058**	**0.0375**	**0.126**	0.0573
LatentCRS$_V$	0.160	0.1139	0.320	0.1582	**0.180**	**0.1386**	0.246	**0.1592**	0.048	0.0302	0.108	0.0489

Table 4: Performance comparison on sampled datasets under multi turn settings.

	Movielens$_{sample}$			VideoGames$_{sample}$			CDs$_{sample}$		
	S@3	S@5	AT	S@3	S@5	AT	S@3	S@5	AT
CRM	0.413	-	-	0.013	-	-	0.042	-	-
UNICORN	0.248	0.378	4.174	0.184	0.260	4.362	0.174	0.232	4.406
CRIF	0.266	0.448	4.064	0.126	0.216	4.578	0.130	0.208	4.610
Llama 3.1-8B	0.188	0.540	3.624	0.208	0.276	4.668	0.00	0.258	4.832
Gemini 1.5 pro	0.224	0.580	3.484	0.248	0.384	4.108	0.170	0.498	4.412
GPT4	0.246	0.602	3.472	0.236	0.398	4.246	0.184	0.548	4.254
Chat-Rec	0.212	**0.686**	3.618	0.224	**0.724**	3.760	0.146	**0.558**	**4.202**
InteRecAgent	0.070	0.230	4.754	0.156	0.292	4.492	0.102	0.230	4.754
LatentCRS$_B$	0.388	0.482	3.760	0.264	0.336	4.078	0.100	0.142	4.642
LatentCRS$_F$	0.472	0.562	3.424	0.286	0.356	4.042	**0.190**	0.258	4.236
LatentCRS$_V$	**0.532**	0.650	**3.298**	**0.378**	0.426	**3.706**	0.182	0.220	4.422

5.4 Metrics

As the experiments are divided into one-turn conversational recommendation and multi-turn conversational recommendation, we utilize different metrics for different settings. For one-turn conversational recommendation, we utilize the widely used Recall and Normalized Discounted Cumulative Gain (NDCG) metrics. In the multi-turn conversational recommendation setting, following (Gao et al., 2023; Huang et al., 2023), we use the Average Success Rate (S) and Average Turns (AT). S@K represents the success rate of recommending the target item within K turns, while AT@K indicates the average number of turns required to achieve a successful recommendation.

5.5 Implementation Details

As shown in the methods, LatentCRS utilize a recommendation model and LLM. we utilize Llama-3.1-8B-Instruct (Dubey et al., 2024) as our backbone LLM, and ICLRec (Chen et al., 2022) as recommendation model. Moreover, we utilize Gemini-1.5-pro for the user simulator to validate the results. We report the mean results of 3 runs with different random seed. All code is available on Anonymous GitHub[2]. And more details about the implementation are shown in Appendix E.

6 Experimental Results

6.1 One-Turn Recommendation

We only evaluate them on sampled datasets due to the high computational costs of LatentCRS$_F$ and LatentCRS$_V$. The results are presented in Table 2 and Table 3. Our findings include: (1) LatentCRS consistently achieves superior performance across most scenarios, demonstrating the effectiveness of combining collaborative information with dialogue context. LatentCRS consistently outperforms other baselines in most scenarios, underscoring the effectiveness of integrating collaborative information with natural language user interests. Furthermore, leveraging LLMs to either analyze user responses (LatentCRS$_F$) or validate and rerank recommendations (LatentCRS$_V$) enhances personalization and yields additional performance gains. (2) Current LLM-based CRS approaches underperform traditional methods in one-turn recommendation settings. Even when employing state-of-the-art models (GPT-4, Gemini 1.5 Pro) and incorporating historical interactions as prompts, these methods fail to achieve competitive results. This performance gap underscores the disconnect between LLMs' knowledge and the collaborative signals derived from user behavior, a critical challenge our work addresses through user intent modeling and integration.

[2] https://anonymous.4open.science/r/LatentCRS-FF05

Table 5: Ablation Study.

	Movielens-1M				VideoGames				CDs			
	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20	Recall@5	NDCG@5	Recall@20	NDCG@20
LatentCRS$_B$	0.1468	0.0918	0.3333	0.1446	0.1378	0.0964	0.2598	0.1310	0.0485	0.0326	0.1083	0.0494
w/o Inference Model	0.0232	0.0147	0.0678	0.0271	0.0309	0.0204	0.0595	0.0283	0.0240	0.0150	0.0667	0.0269
direct KL Align	0.0847	0.0518	0.2256	0.0911	0.0728	0.0472	0.1740	0.0756	0.0258	0.0165	0.0689	0.0285
w/o Recloss	0.0698	0.0755	0.2073	0.0805	0.1006	0.0700	0.2092	0.1098	0.0241	0.0132	0.0650	0.0264

Table 6: Token usage of LLM across different methods.

	Movielens		Games		CDs	
	Input	Output	Input	Output	Input	Output
Llama 3.1-8B	119.51	148.20	127.20	199.83	124.26	211.57
Gemini 1.5 pro	120.49	140.56	112.43	188.75	126.89	223.91
GPT4	118.95	152.81	106.85	181.36	129.88	235.46
LatentCRS$_B$	85.50	1.00	75.44	1.00	91.90	1.00
LatentCRS$_F$	317.50	35.59	781.44	50.76	2086.89	70.70
LatentCRS$_V$	810.26	98.63	2117.60	96.72	2178.55	111.98

6.2 Multi-Turn Recommendation

We further evaluate LatentCRS in multi-turn conversational settings, with results presented in Table 4. Since the CRM method (Sun and Zhang, 2018) queries item features category and the datasets have a limited number of categories, the maximum number of dialogue turns in CRM is inherently constrained. From the data, we observe: (1) LatentCRS consistently outperforms other baselines in most scenarios, particularly in achieving higher success rates within three interaction turns. This capability allows LatentCRS to accurately infer user intent within a limited number of interactions and improve the user experience. (2) Response analysis (LatentCRS$_F$) outperforms the base generative model, and reranking (LatentCRS$_V$) yields further improvement. This indicates that deeper integration of LLMs enhances performance under LatentCRS. However, increased LLM utilization also leads to higher costs. More details are shown in Section 6.3.

6.3 Efficiency Analysis

Due to the high operational costs of LLMs, efficiency is a critical concern for LLM-based CRS. Given that LLM costs dominate other system components, we approximate total expenses by analyzing input and output token counts (see Table 6). Key findings include: (1) All the variants of LatentCRS utilize significantly fewer tokens compared to existing LLM-based CRS models. LatentCRS$_B$ only needs to generate 1 new token, reducing computational load by hundreds of operations due to autoregressive decoding. This dramatic cost reduction makes it particularly suitable for large-scale deployment. (2) We observe a trade-off between cost and performance in LatentCRS. While expanded LLM utilization improves accuracy, but also increases cost. The three variants of LatentCRS offer flexible choices to balance these priorities based on application needs.

6.4 Ablation Study

To evaluate the contribution of each component in LatentCRS, we conduct an ablation study under a one-turn recommendation setting. The results are presented in Table 5. We examine four ablated variants: (1) w/o Inference Model: The generative model is trained using only the auxiliary recommendation loss, without incorporating user intent. (2) Direct KL Alignment: The inference and generative models are jointly trained with a KL divergence constraint on the intent distribution. (3) w/o Recloss: The auxiliary recommendation loss is removed. The performance degradation observed in all ablated variants demonstrates the importance of each component. The significant performance drop without the inference model highlights the crucial role of our intent inference mechanism, as directly combining recommendation models with LLMs fails without proper user intent modeling. More analyses are in Appendix G.

7 Conclusion

In this paper, we investigate the collaboration between recommendation models and large language models (LLMs). We identify user intent as a key bridge to effectively close the gap between these systems. Since user intent is inherently unobservable, we propose LatentCRS, a novel approach that addresses this challenge by representing user intent through anchor vectors and optimizing CRS tasks with user intent using an EM framework. A notable advantage of LatentCRS is its high efficiency, as it eliminates the need to integrate LLMs into the training process. In future work, we plan to explore additional real-world scenarios and further examine the impact of different user dialogue styles on recommendation performance.

8 Limitations

While LATENTCRS demonstrates significant improvements over existing methods and provide an efficient way for LLM-based CRS, it also exhibits limitations in cold-start Scenarios. For new users with limited interaction history, inferring accurate intent distributions becomes challenging, potentially degrading recommendation quality until sufficient behavioral data is accumulated. Moreover, intent inference relies heavily on a pretrained recommendation model to extract user behavior patterns. Any errors or biases in this model may propagate to the intent distribution estimation, affecting recommendation accuracy. Recognizing these limitations, future work could explore combining latent intents with content-based or demographic features to mitigate cold-start issues and developing uncertainty-aware models or adaptive clustering techniques to improve the robustness of intent inference.

9 Ethical Considerations

Our research utilizes three publicly available datasets, MovieLens-1M, Amazon VideoGames, and Amazon CDs, all of which have been rigorously anonymized by their original providers through irreversible hashing of user identifiers. We strictly comply with each dataset's license. Our study neither infers nor utilizes any personally identifiable information. For LLM integration, we employ LLama 3.1 and Gemini 1.5, both allow academic usage under their policies.

While our framework represents an advancement in conversational recommendation systems (CRS), we acknowledge several potential risks that warrant consideration: (1) The pretrained recommendation model may inadvertently propagate historical biases present in the training data; and (2) Although designed specifically for CRS applications, the system remains potentially vulnerable to adversarial prompts that could compromise security or generate misleading outputs. To mitigate these risks, we recommend implementing additional safeguards, including bias auditing protocols and robust input validation mechanisms.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, and 1 others. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Keqin Bao, Jizhi Zhang, Xinyu Lin, Yang Zhang, Wenjie Wang, and Fuli Feng. 2024. Large language models for recommendation: Past, present, and future. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2993–2996.

Keping Bi, Qingyao Ai, Yongfeng Zhang, and W Bruce Croft. 2019. Conversational product search based on negative feedback. In *Proceedings of the 28th acm international conference on information and knowledge management*, pages 359–368.

Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable multi-interest framework for recommendation. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2942–2951.

Yongjun Chen, Zhiwei Liu, Jia Li, Julian McAuley, and Caiming Xiong. 2022. Intent contrastive learning for sequential recommendation. In *Proceedings of the ACM Web Conference 2022*, pages 2172–2182.

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 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 139–148.

Konstantina Christakopoulou, Filip Radlinski, and Katja Hofmann. 2016. Towards conversational recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining*, pages 815–824.

Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. 2021. Unified conversational recommendation policy learning via graph-based reinforcement learning. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1431–1441.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Siamak Farshidi, Kiyan Rezaee, Sara Mazaheri, Amir Hossein Rahimi, Ali Dadashzadeh, Morteza Ziabakhsh, Sadegh Eskandari, and Slinger Jansen. 2024. Understanding user intent modeling for conversational recommender systems: a systematic literature review. *User Modeling and User-Adapted Interaction*, pages 1–64.

Yunfan Gao, Tao Sheng, Youlin Xiang, Yun Xiong, Haofen Wang, and Jiawei Zhang. 2023. Chat-rec: Towards interactive and explainable llms-augmented recommender system. *arXiv preprint arXiv:2303.14524*.

Zhankui He, Zhouhang Xie, Rahul Jha, Harald Steck, Dawen Liang, Yesu Feng, Bodhisattwa Prasad Majumder, Nathan Kallus, and Julian McAuley. 2023. Large language models as zero-shot conversational recommenders. In *Proceedings of the 32nd ACM international conference on information and knowledge management*, pages 720–730.

Zhankui He, Zhouhang Xie, Harald Steck, Dawen Liang, Rahul Jha, Nathan Kallus, and Julian McAuley. 2024. Reindex-then-adapt: Improving large language models for conversational recommendation. *arXiv preprint arXiv:2405.12119*.

Yupeng Hou, Jiacheng Li, Zhankui He, An Yan, Xiusi Chen, and Julian McAuley. 2024. Bridging language and items for retrieval and recommendation. *arXiv preprint arXiv:2403.03952*.

Chenhao Hu, Shuhua Huang, Yansen Zhang, and Yubao Liu. 2022. Learning to infer user implicit preference in conversational recommendation. In *Proceedings of the 45th International ACM SIGIR conference on research and development in information retrieval*, pages 256–266.

Xu Huang, Jianxun Lian, Yuxuan Lei, Jing Yao, Defu Lian, and Xing Xie. 2023. Recommender ai agent: Integrating large language models for interactive recommendations. *arXiv preprint arXiv:2308.16505*.

Ting Jiang, Shaohan Huang, Zhongzhi Luan, Deqing Wang, and Fuzhen Zhuang. 2024. Scaling sentence embeddings with large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*.

Jiarui Jin, Xianyu Chen, Fanghua Ye, Mengyue Yang, Yue Feng, Weinan Zhang, Yong Yu, and Jun Wang. 2023. Lending interaction wings to recommender systems with conversational agents. *Advances in Neural Information Processing Systems*, 36:27951–27979.

Wang-Cheng Kang and Julian McAuley. 2018. Self-attentive sequential recommendation. In *2018 IEEE international conference on data mining (ICDM)*, pages 197–206. IEEE.

Sara Kemper, Justin Cui, Kai Dicarlantonio, Kathy Lin, Danjie Tang, Anton Korikov, and Scott Sanner. 2024. Retrieval-augmented conversational recommendation with prompt-based semi-structured natural language state tracking. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2786–2790.

Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. 2024. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 1395–1406.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2022. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213.

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 13th International Conference on Web Search and Data Mining*, page 304–312, New York, NY, USA. Association for Computing Machinery.

Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-interest network with dynamic routing for recommendation at tmall. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 2615–2623.

Haoyang Li, Xin Wang, Ziwei Zhang, Jianxin Ma, Peng Cui, and Wenwu Zhu. 2021. Intention-aware sequential recommendation with structured intent transition. *IEEE Transactions on Knowledge and Data Engineering*, 34(11):5403–5414.

Raymond Li, Samira Ebrahimi Kahou, Hannes Schulz, Vincent Michalski, Laurent Charlin, and Chris Pal. 2018. Towards deep conversational recommendations. *Advances in neural information processing systems*, 31.

Junling Liu, Chao Liu, Peilin Zhou, Renjie Lv, Kang Zhou, and Yan Zhang. 2023. Is chatgpt a good recommender? a preliminary study. *arXiv preprint arXiv:2304.10149*.

Zhiwei Liu, Xiaohan Li, Ziwei Fan, Stephen Guo, Kannan Achan, and S Yu Philip. 2020. Basket recommendation with multi-intent translation graph neural network. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 728–737. IEEE.

Zhuang Ma and Michael Collins. 2018. Noise contrastive estimation and negative sampling for conditional models: Consistency and statistical efficiency. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3698–3707.

Balaji Padmanabhan and Alexander Tuzhilin. 1999. Unexpectedness as a measure of interestingness in knowledge discovery. *Decision Support Systems*, 27(3):303–318.

Zhiqiang Pan, Fei Cai, Yanxiang Ling, and Maarten de Rijke. 2020. An intent-guided collaborative machine for session-based recommendation. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*, pages 1833–1836.

Mathieu Ravaut, Hao Zhang, Lu Xu, Aixin Sun, and Yong Liu. 2024. Parameter-efficient conversational recommender system as a language processing task. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–165.

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. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450.

Yueming Sun and Yi Zhang. 2018. Conversational recommender system. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 235–244. ACM.

Zhu Sun, Hongyang Liu, Xinghua Qu, Kaidong Feng, Yan Wang, and Yew Soon Ong. 2024. Large language models for intent-driven session recommendations. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 324–334.

Md Mehrab Tanjim, Congzhe Su, Ethan Benjamin, Diane Hu, Liangjie Hong, and Julian McAuley. 2020. Attentive sequential models of latent intent for next item recommendation. In *Proceedings of The Web Conference 2020*, pages 2528–2534.

Gemini Team, Petko Georgiev, Ving Ian Lei, Ryan Burnell, Libin Bai, Anmol Gulati, Garrett Tanzer, Damien Vincent, Zhufeng Pan, Shibo Wang, and 1 others. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Aäron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *CoRR*, abs/1807.03748.

Yunjia Xi, Weiwen Liu, Jianghao Lin, Bo Chen, Ruiming Tang, Weinan Zhang, and Yong Yu. 2024. Memocrs: Memory-enhanced sequential conversational recommender systems with large language models. In *Proceedings of the 33rd ACM International Conference on Information and Knowledge Management*, pages 2585–2595.

Li Yang, Anushya Subbiah, Hardik Patel, Judith Yue Li, Yanwei Song, Reza Mirghaderi, and Vikram Aggarwal. 2024. Item-language model for conversational recommendation. *arXiv preprint arXiv:2406.02844*.

Ting Yang and Li Chen. 2024. Unleashing the retrieval potential of large language models in conversational recommender systems. In *Proceedings of the 18th ACM Conference on Recommender Systems*, pages 43–52.

Jing Yao, Zhicheng Dou, Ruobing Xie, Yanxiong Lu, Zhiping Wang, and Ji-Rong Wen. 2021. User: A unified information search and recommendation model based on integrated behavior sequence. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 2373–2382.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R. Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net.

Se-eun Yoon, Zhankui He, Jessica Echterhoff, and Julian McAuley. 2024. Evaluating large language models as generative user simulators for conversational recommendation. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 1490–1504.

Erhan Zhang, Xingzhu Wang, Peiyuan Gong, Yankai Lin, and Jiaxin Mao. 2024a. Usimagent: Large language models for simulating search users. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 2687–2692.

Junjie Zhang, Yupeng Hou, Ruobing Xie, Wenqi Sun, Julian McAuley, Wayne Xin Zhao, Leyu Lin, and Ji-Rong Wen. 2024b. Agentcf: Collaborative learning with autonomous language agents for recommender systems. In *Proceedings of the ACM on Web Conference 2024*, pages 3679–3689.

Xiaoying Zhang, Hong Xie, Hang Li, and John CS Lui. 2020. Conversational contextual bandit: Algorithm and application. In *Proceedings of the web conference 2020*, pages 662–672.

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 27th acm international conference on information and knowledge management*, pages 177–186.

A Related Work

A.1 Conversational Recommender Systems

Conversational recommender systems (CRS) aim to provide accurate recommendations through interactions with users. Early approaches usually treated the recommendation and conversation components as two separate modules (Sun and Zhang, 2018; Bi et al., 2019; Zhang et al., 2020). Initial work focused on how to interact with users by determining which attributes or items to query and by extracting information from user responses for more accurate recommendations (Christakopoulou et al., 2016, 2018). Subsequent research extended these ideas to multi-turn conversational settings, often using reinforcement learning methods (Lei et al., 2020; Li et al., 2018; Zhang et al., 2018; Hu et al., 2022). Although these methods were successful, they restricted user responses to binary signals (like or dislike), which limited the information that users could provide. The growing use of LLMs has introduced a new paradigm to interact with users, which inspired the design of LLM-based CRS (Jin et al., 2023; Gao et al., 2023; He et al., 2023; Huang et al., 2023; Bao et al., 2024). Some methods treat an LLM as an agent and the recommendation model as a tool (Gao et al., 2023; Huang et al., 2023; Kemper et al., 2024). Although this approach allows more expressive user responses, it still treats conversation and recommendation separately and only merges them through a binary flag to indicate when a recommendation should be made.

Other methods integrate recommendation and conversation in a single framework. Because LLMs have strong text processing and reasoning abilities, some systems rely solely on LLMs to perform both tasks. For instance, certain approaches follow the ReAct framework (Yao et al., 2023), where the LLM is instructed to plan and reason, carry out specific actions, and adjust if mistakes are detected (Gao et al., 2023; He et al., 2023, 2024; Xi et al., 2024). Further approaches incorporate additional training of LLMs using interaction data (Yang et al., 2024; Yang and Chen, 2024; Ravaut et al., 2024). However, these strategies ignore the collaborative knowledge found in recommender systems, and they can be expensive due to frequent LLM calls or extensive fine-tuning. Moreover, their performance can be uncertain because many studies only compare them with baselines in cold-start situations (Jin et al., 2023; Yang and Chen, 2024). In contrast, we propose a more efficient approach that uses user intent as a bridge. We incorporate collaborative information from recommendation models while jointly learning representations of natural language and user behavior.

A.2 Intent Modeling in Recommendation

Recently, many studies have examined user intent to improve recommendation accuracy (Cen et al., 2020; Liu et al., 2020; Pan et al., 2020; Tanjim et al., 2020; Yao et al., 2021). Most approaches identify similar user intent. For example, MIND (Li et al., 2019) introduces a multi-interest extractor layer based on a capsule routing mechanism to deal with multiple user intents. ISRec (Li et al., 2021) builds an intention graph to uncover each user's underlying intentions and employs a message-passing mechanism to predict future intentions. ICLRec (Chen et al., 2022) leverages an expectation-maximization framework to integrate learned intents into sequential recommendation models via contrastive self-supervised learning. Furthermore, PO4ISR (Sun et al., 2024) incorporates intent modeling with LLMs for session recommendation, allowing these models to discern diverse user intents at a semantic level by iteratively refining and adjusting prompts. However, current intent modeling methods mostly concentrate on discovering similar user intents from behavioral data (Farshidi et al., 2024). In contrast, we find that user intent can also act as a link between user behavior and interests expressed in natural language for CRS.

B Dataset Processing

Since not all items in the Amazon Games and Amazon CDs datasets have associated metadata, we preprocess the raw data to exclude items lacking metadata. Additionally, we focus on the "5-core" data, where each user and item has at least five interactions. This is because cold-start items, which lack sufficient collaborative information, are not the focus of our work. We follow the data splitting methodology outlined in (Kang and McAuley, 2018; Sun et al., 2019; Chen et al., 2022). Specifically, we use the last item in each user's sequence as the test set, the second-to-last item as the validation set, and all other items as the training set. Moreover, we employ a widely adopted data augmentation method to generate additional training samples (Chen et al., 2022). Specifically, we ran-

domly sample sliding windows from user behavior sequences to create new synthetic data. To ensure a fair comparison, all methods including both our proposed approach and the baselines utilize the same augmented training set.

C Baselines

- Traditional Conversational Recommendation Models: **CRM** (Sun and Zhang, 2018) feeds the belief tracker results to an FM-based recommendation method to integrate the conversational and recommender components. **UNICORN** (Deng et al., 2021) formulates the conversational recommendation problem as unified policy learning task. **CRIF** (Hu et al., 2022) utilizes four-phase process consisting of offline representation learning, tracking, decision-making, and inference.

- LLM-based Conversational Recommendation Methods: **Llama 3.1-8B** (Dubey et al., 2024), **Gemini 1.5 pro** (Team et al., 2024), **GPT-4** (Achiam et al., 2023): They are representative LLMs, including both open-source and closed-source models. **Chat-Rec** (Gao et al., 2023) converts user profiles and historical interactions into prompts to innovatively augment LLMs to build CRS. **InteRecAgent** (Huang et al., 2023) employs LLMs as the core processing unit and recommender models as tools.

D User Simulator

The prompt to instruct LLM to play as a user is illustrated as follows:

Prompt for Candidate Explanations Generation:
You are a user chatting with a recommender for movies recommendation in turn.
Your history is {user_behavior_sequence}.
Your target items: {target_item}.
Here is the information about {target_item_features}.
You must follow the instructions below during chat.
State your intention to guide the recommender towards the target item. If the recommender recommends {target_item}, you should accept. If the recommender recommends other items, you should refuse them and provide the information about Toy Story. You should never directly tell the target item title. If the recommender asks for your preference, you should provide the information about {target_item}.
You could provide your history. You should never directly tell the target item title. Your output is only allowed to be the words from the user you act. If you think the conversation comes to an ending, output a ⟨END⟩.
You should never directly tell the target item.
Only use the provided information about the target. Never give many details about the target items at one time. Less than 3 conditions is better.
Now lets start, you first, act as a user.

E Implementation Details

E.1 Implementation

As previously discussed, we utilize an LLM to act as a user simulator, specifically employing Gemini-1.5-pro for this purpose. Within our framework, we use Llama-3.1-8B-Instruct (Dubey et al., 2024) as the incorporated LLMs. For the recommendation component, we implement the ICLRec (Chen et al., 2022) model. All optimization steps are carried out using PyTorch on 1 x NVIDIA RTX A6000 GPU. Following (Gao et al., 2023), we limit the maximum number of interaction turns to 5 and present at most 5 items in each dialogue turn. In our implementation, we make a conditional independence assumption, where the variable v is conditionally independent of the observed data $\mathcal{S}^u, x^u$ given the latent variable M. This can be formally represented

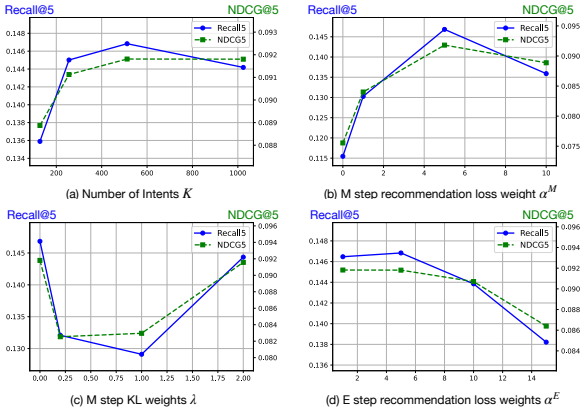

Figure 3: Analysis of the influence of hyperparameters.

as:

$$p(v \mid \mathcal{S}^u, x^u) = \mathbb{E}_{(M)}\, p(v \mid M) p(M \mid \mathcal{S}^u, x^u)$$

Moreover, due to the limited amount of training data, we employ a simple data augmentation method by randomly segmenting user behavior sequences to generate new data instances. This approach helps to artificially expand the dataset and improve model generalization by simulating diverse interaction scenarios. Additionally, we employ a caching scheme to enhance training efficiency. Specifically, we pre-embed the natural language descriptions and store them, thereby offloading the computationally expensive LLM calculations.

E.2 Hyper-Parameters Sensitivity

In this section, we present experimental results to examine the sensitivity of LATENTCRS to key hyperparameters. We evaluate the performance differences with respect to the number of intents K and the weight hyperparameters α^M, λ in Equation (9), and α^E in Equation (14). The results are illustrated in Figure 3. Due to space constraints, we only report the results obtained on the Movielens dataset. From the analysis, we observe that the optimal selection of hyperparameters plays a crucial role in achieving superior performance.

F Ai Assistants Usage

We employ DeepSeek and ChatGPT solely for grammar correction and language polishing. These tools are not used to generate original content or ideas. All AI-assisted revisions undergo careful review to ensure they preserve the intended meaning and maintain the integrity of our original work.

G Ablation Study

To evaluate the effectiveness of each component of LATENTCRS, we conduct an ablation study on the full version of three datasets under one-turn recommendation setting, and the results are presented in Table 5. (1) w/o Inference Model: In this scenario, we train the generative model solely using the auxiliary recommendation loss. The significant performance gap underscores the difficulty of training models directly with the latent intent space. This result also highlights the importance of incorporating collaborative information from recommendation models into conversational recommender systems, which is the core motivation behind our approach. (2) Direct KL Alignment: Here, we simultaneously train the inference model and the generative model, applying an additional Kullback-Leibler divergence constraint on the intent distribution. The performance decline suggests that simply aligning the intent distributions does not lead to the correct distribution, emphasizing the effectiveness of the EM training framework. (3) w/o Recloss: In this scenario, we remove the auxiliary recommendation loss. The degraded performance demonstrates the importance of the recommendation loss in guiding the alternating EM training steps.