Diffusion Generative Recommendation with Continuous Tokens

Haohao Qu
haohao.qu@connect.polyu.hk
The Hong Kong Polytechnic
University
Hong Kong

Shanru Lin
lllam32316@gmail.com
City University of Hong Kong
Hong Kong

Yujuan Ding*
dingyujuan385@gmail.com
The Hong Kong Polytechnic
University
Hong Kong

Yiqi Wang
yiq@nudt.edu.cn
National University of Defense
Technology
China

Wenqi Fan
wenqifan03@gmail.com
The Hong Kong Polytechnic
University
Hong Kong

Abstract

Recent advances in generative artificial intelligence, particularly large language models (LLMs), have opened new opportunities for enhancing recommender systems (RecSys). Most existing LLM-based RecSys approaches operate in a discrete space, using vector-quantized tokenizers to align with the inherent discrete nature of language models. However, these quantization methods often result in lossy tokenization and suboptimal learning, primarily due to inaccurate gradient propagation caused by the non-differentiable `argmin` operation in standard vector quantization. Inspired by the emerging trend of embracing continuous tokens in language models, we propose **ContRec**, a novel framework that seamlessly integrates continuous tokens into LLM-based RecSys. Specifically, ContRec consists of two key modules: a σ-VAE Tokenizer, which encodes users/items with continuous tokens; and a Dispersive Diffusion module, which captures implicit user preference. The tokenizer is trained with a continuous Variational Auto-Encoder (VAE) objective, where three effective techniques are adopted to avoid representation collapse. By conditioning on the previously generated tokens of the LLM backbone during user modeling, the Dispersive Diffusion module performs a conditional diffusion process with a novel Dispersive Loss, enabling high-quality user preference generation through next-token diffusion. Finally, ContRec leverages both the textual reasoning output from the LLM and the latent representations produced by the diffusion model for Top-K item retrieval, thereby delivering comprehensive recommendation results. Extensive experiments on four datasets demonstrate that ContRec consistently outperforms both traditional and state-of-the-art LLM-based recommender systems. Our results highlight the potential of continuous tokenization and generative modeling for advancing the next generation of recommender systems.

CCS Concepts

• **Information systems → Web mining**.

---

*Corresponding author.

Keywords

Recommender Systems, Large Language Models, Diffusion Models.

Resource Availability:

The source code of this paper has been made publicly available at https://doi.org/10.5281/zenodo.17285302.

1 Introduction

Web mining is a pivotal research area that focuses on extracting valuable knowledge and patterns from vast and heterogeneous web data. Within this domain, recommender systems (RecSys) serve as crucial tools to tackle information overload and enrich user experiences on various web applications such as e-commerce, job search, and social media platforms [9, 10, 46]. The emerging focus on Generative Artificial Intelligence (GenAI), especially Large Language Models (**LLMs**), has spurred a growing interest in **Generative Recommendation**, driven by the desire to leverage their exceptional generalization capabilities and adeptness at in-context learning [45, 58]. These distinct abilities empower LLM-based RecSys to generalize more effectively across new tasks and domains.

Most existing LLM-based RecSys utilize LLM as the backbone, therefore inevitably inherit its core architecture design centered on discrete token representations [18, 57]. For example, basic settings for item/user representation in RecSys include numeric indexing (e.g., "*item_2025*") and textual title indexing (e.g., "*Apple iPhone 16, 1TB, White*") [1, 13, 21, 50]. However, these approaches become inefficient when dealing with large-scale real-world recommender systems, in which the sizes of user and item sets typically reach millions. To address the token set overload issue, Vector-Quantized Variational Autoencoder (VQ-VAE) [34, 42] has been applied to construct a discrete vocabulary (also referred to as a "codebook", such as "$\langle a-3\rangle\langle b-24\rangle\langle c-58\rangle\langle d-12\rangle$") and train a neural tokenizer on users and items. This strategy has been widely employed in a series of LLM-based RecSys, including TIGER [36], UTGRec [59], and OneRec [5], among others. These quantization-based solutions may be practical for encapsulating rich item-side information into
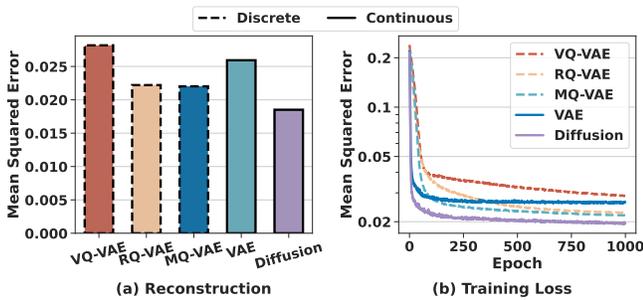
Figure 1: Comparison of five representative deep generative models, namely VQ-VAE [42], RQ-VAE [36], MQ-VAE [34], VAE [27], and Diffusion [19], in reconstructing item embeddings on the Beauty dataset. The standard diffusion model with continuous embeddings demonstrates superior reconstruction performance and loss convergence compared to the vanilla VAE and its discrete counterparts.

a sequence of tokens that align with the discrete nature of language models, but they still encounter two critical limitations. First, their reconstruction quality and generation performance are constrained by lossy tokenization, as some information will inevitably be compressed and omitted during the quantization process [4, 6, 41]. Second, due to the non-differentiable argmin in typical vector quantization, a "straight-through" trick [2] is normally used, directly copying the gradient from the decoder input to the encoder output. Such a surrogate gradient can be inaccurate, potentially impeding both the achievement of high compression ratios and the effective learning of the latent space in quantization-based methods [12, 23].

Such limitations of discrete learning-based models have garnered attention in language tasks, especially for multimodal modeling (e.g., image, video, and protein) [48, 49]. For example, Fan et al. [6] reveal that quantization-based visual models exhibit poor performance improvements when scaling up model size, compared to models operating on continuous tokens. Furthermore, there have been increasing efforts to introduce *continuous tokens* for learning better representations [28]. For example, Chen et al. [4] introduce an efficient continuous image tokenizer for language models, which achieves a high compression ratio while enriching the semantic content of the latent space for image reconstruction. These advancements in language models inspire the exploration of continuous tokens as a new tokenization paradigm for LLM-based generative recommendation. To achieve the goal, a few LLM-based recommendation methods have attempted to include continuous tokens. For example, CoLLM [55] and LlaRA [29] propose directly projecting collaborative item representations learned from traditional recommendation methods into the token space of language models. However, these methods only continualize the input part of the model by directly employing simple projection layers, e.g., a multi-layer perceptron (MLP), for user/item tokenization, while their output part inherits the LLM backbone's generator, which is learned for in-vocabulary discrete tokens. Such discrete-continuous gaps in one model might degrade its unification [41] and result in suboptimal recommendation performance.

To enable the seamless incorporation of continuous tokens into LLM-based generative recommendation, we propose leveraging the powerful Diffusion Model to continualize user preference modeling and generation, alongside the tokenization of users and items. It is

widely recognized that user preference modeling is the core technical part leading to effective recommendation, while this task is highly challenging because user preferences are inherently complex and differences between individual users can be exceptionally subtle. Diffusion Models are particularly promising for addressing this challenge, as they have demonstrated remarkable success in handling continuous tokens across various domains. For instance, DEEM [30] demonstrates that diffusion models can serve as the "eyes" of LLMs for image perception by encoding and decoding images as continuous tokens, while recent advances in protein language models demonstrate diffusion models' capability in constructing continuous protein structure embeddings [14, 47]. To validate this insight in recommendation scenarios, we conduct a preliminary item embedding reconstruction experiment on the Amazon-Beauty dataset. As shown in Figure 1, our results are consistent with previous findings in the literature, showing that utilizing diffusion models with continuous representations delivers higher reconstruction accuracy and a smoother loss decline than the discrete counterparts. Detailed information is provided in Section 4.3.3 and Appendix C.

In this paper, we present a novel LLM-based recommendation framework, **ContRec**, utilizing **Cont**inuous tokens to enhance the capabilities of LLM-based **Rec**Sys. With an LLM being the recommender backbone for user modeling with hybrid input, ContRec consists of two core continuous token learning components: a VAE module to learn continuous tokenization for users/items via the reconstruction of their associated features (e.g., IDs, texts, and images) and a novel Disperse Diffusion module to learn continuous-distributed user preference representations based on the LLM-based user modeling. Additionally, we design a hybrid retrieval mechanism that integrates the explicit LLM-output prediction text and the implicit diffusion-enhanced user preference representation to output a ranking list of items for precise recommendations.

Our major contributions are summarized as follows:

- We propose ContRec, a new framework that harnesses continuous tokens within LLM-based generative recommender systems, unlocking the advantages of non-quantized tokenization for users/items and enabling the generation of high-quality recommendations. By moving beyond the limitations of quantization, this solution paves the way for transformative advances in generative recommendation, offering a powerful and complementary perspective to existing discrete-based methods.
- We introduce two critical modules: a novel Dispersive Diffusion module and a robust σ-VAE tokenizer. The coefficient σ is designed to safeguard against the collapse of continuous representations, ensuring rich and expressive latent spaces. The Dispersive Diffusion module is developed to learn implicit user preference representations through contrastive self-supervised learning without the need for explicit negative-positive pairs.
- We conduct abundant experiments, including recommendation performance evaluation, ablation study, optimization robustness analysis, and hyperparameter tuning, to rigorously validate the effectiveness of the proposed method. The results demonstrate that ContRec outperforms the state-of-the-art baselines, notably surpassing the quantization-based method TIGER, with average improvements of 11.76% on HR@10 and 10.11% on NDCG@10 across four datasets.

2 Preliminary

Generative Recommendations. This part recaps the basic notions of generative recommendations, as established by the pioneering LLM framework. Consider $\mathcal{U} = \{u_1, u_2, ..., u_n\}$ and $\mathcal{V} = \{v_1, v_2, ..., v_m\}$ as the sets of users and items, respectively, where n signifies the user count, and m indicates the number of items in the dataset. Among these, the item set associated with the historical interactions of user u_i is symbolized by $\mathcal{V}_{(u_i)}$. Generally speaking, recommender systems aim to grasp user preferences through analyzing interactions (e.g., clicks and reviews) between users $\mathcal{U}$ and items $\mathcal{V}$. As a widely used solution, sequential recommendation techniques are developed to predict the future user preference from historical user-item interactions. Correspondingly, we reformulate the sequential recommendation into a language model paradigm. Given tokens $\mathcal{T}_i$ for each user u_i and $\mathcal{T}_j$ for each item v_j, we can construct the integrated input for the LLM-based recommender with certain textual prompts $\mathcal{P}$ and obtain the output as:

$$\mathcal{Y}_i = \text{LLM}(\mathcal{P}(\mathcal{T}_i, \{\mathcal{T}_j | v_j \in \mathcal{V}_{(u_i)}\})). \tag{1}$$

The LLM output $\mathcal{Y}_i$ can take various forms, such as item titles, special tokens, or latent vectors, depending on the model design.

User and Item Tokenization. In conventional recommendation techniques, a user u_i and an item v_j are typically represented by D-dimensional latent vector representations, denoted as $\mathbf{p}_i$ and $\mathbf{q}_j \in \mathbb{R}^D$, respectively. These representations can be learned with various methods, e.g., graph-based models [8, 9], whose effectiveness has been extensively demonstrated over the past decades. In contrast, LLM-based recommender systems often require discrete representations for indexing users and items, in order to align with the discrete nature of language data present in their training corpora. Beyond trivial indexing methods, the most prevalent tokenization approach for LLM-based RecSys is quantization [34, 36, 42], which involves mapping user or item features (e.g., IDs, titles, categories, and descriptions) into a set of discrete tokens. A vector-quantized tokenizer is typically trained via a reconstruction process, which can be formulated as follows:

$$\mu = \text{Encoder}(\mathbf{x}), \tag{2}$$

$$s = \text{argmin}_s \|\mu - \mathbf{c}_s\|^2, \tag{3}$$

$$\hat{\mathbf{x}} = \text{Decoder}(\mathbf{c}_s), \tag{4}$$

where $\mathbf{x}$ denotes the input features, $\mathbf{c}_s$ is the s_{th} token in a predefined discrete token space $C \in \mathbb{R}^{S \times D_c}$, and $\hat{\mathbf{x}}$ is the predicted result for the reconstruction objective. Through this process, users u_i and items v_j can be tokenized using their corresponding codeword s and the codeword embedding $\mathbf{c}_s$, denoted as $\mathcal{T}_i$ and $\mathcal{T}_j$ in the context of LLM-based RecSys.

Diffusion Models. We provide a brief overview of diffusion models, focusing on the representative formulation known as Denoising Diffusion Probabilistic Models (DDPM) [19]. In the *forward process*, DDPM gradually adds Gaussian noise to the original input $\mathbf{x}^0$ according to a variance schedule β_t:

$$f(\mathbf{x}^{1:T}|\mathbf{x}^0) = \prod_{t=1}^{T} f(\mathbf{x}^t|\mathbf{x}^{t-1}), \tag{5}$$

$$f(\mathbf{x}^t|\mathbf{x}^{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}^{t-1}, \beta_t\mathbf{I}). \tag{6}$$

Let $\alpha_t = 1 - \beta_t$, $\overline{\alpha} = \prod_{\tau=1}^{t} \alpha_\tau$, $\epsilon \in \mathcal{N}(\mathbf{0}, \mathbf{I})$, we can express the noisy sample at step t as $\mathbf{x}^t = \sqrt{\overline{\alpha}_t}\mathbf{x}^0 + \sqrt{1-\overline{\alpha}_t}\epsilon$. In the *reverse process*, DDPM jointly models the target variable $\mathbf{x}^0$ along with a sequence of latent variables $\mathbf{x}^1, ..., \mathbf{x}^T$ as a Markov chain with Gaussian transitions:

$$g(\mathbf{x}^{0:T}) = g(\mathbf{x}^T) \prod_{t=1}^{T} g(\mathbf{x}^{t-1}|\mathbf{x}^t), \tag{7}$$

$$g(\mathbf{x}^{t-1}|\mathbf{x}^t) = \mathcal{N}(\mathbf{x}^{t-1}; \eta(\mathbf{x}^t, t), \xi(\mathbf{x}^t, t)), \tag{8}$$

where the initial state $\mathbf{x}^T$ is sampled from standard Gaussian noise $\mathbf{x}^T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. Here, $\eta(\mathbf{x}^t, t)$ denotes a time-dependent mean of the Gaussian distribution used to sample the previous timestep's data (often initialized as untrained constants), and $\xi(\mathbf{x}^t, t)$ determines the variance (or covariance) of the Gaussian distribution used in the reverse denoising process. Given the remarkable success of diffusion models in handling continuous data such as images and protein structures, an increasing number of studies have explored integrating diffusion models with LLMs [41, 47, 48].

3 Methodology

3.1 Overview of ContRec

In order to leverage continuous tokens into an LLM-based recommender system to enhance the model's capability for user, item, and implicit user preference representation, we propose a novel framework: ContRec. As illustrated in Figure 2, our ContRec consists of two core innovative modules, the σ-*VAE Tokenizer* and *Disperse Diffusion*. Due to space constraints, we present only the core formulas of the key components of the proposed method in this section; supplementary information and detailed derivations are provided in Appendix B.

3.2 σ-VAE Tokenizer

The proposed tokenizer is built upon the VAE framework [27, 39], which encodes input data into a latent space and subsequently decodes it back to the original space. Maintaining a larger variance in the latent representations improves the robustness of the VAE tokenizer to exposure bias between tokens during inference, while standard VAEs often suffer from channel variance collapse, adversely affecting autoregressive modeling [37]. To address this issue, we introduce three simple yet effective techniques to enhance our Tokenizer: (i) a masking operation based on the Bernoulli distribution to enhance robustness, (ii) a K-way encoder with parallel encoding channels to allow implicit encoding [34], and (iii) a Gaussian kernel applied to each channel to prevent variance collapse [41]. The reconstruction pass is computed as:

$$\mu_k = \text{Enc}^k(\text{Mask}(\mathbf{x}, \rho)), \tag{9}$$

$$\mathbf{z}_k = \mu_k + \sigma_k \odot \epsilon, \text{ where } \epsilon \sim \mathcal{N}(0, 1), \ \sigma_k \sim \mathcal{N}(0, \Gamma), \tag{10}$$

$$\hat{\mathbf{x}} = \text{Dec}(\text{Concat}\{\mathbf{z}_k\}^K), \tag{11}$$

where σ_k is a scalar coefficient sampled from a Gaussian distribution $\mathcal{N}(0, \Gamma)$, with Γ representing the standard deviation computed from the first batch of training samples. At its core, the σ kernel prevents variance collapse by enforcing a fixed variance in the latent space. In addition, the notations used in the formulas are specified as follows.
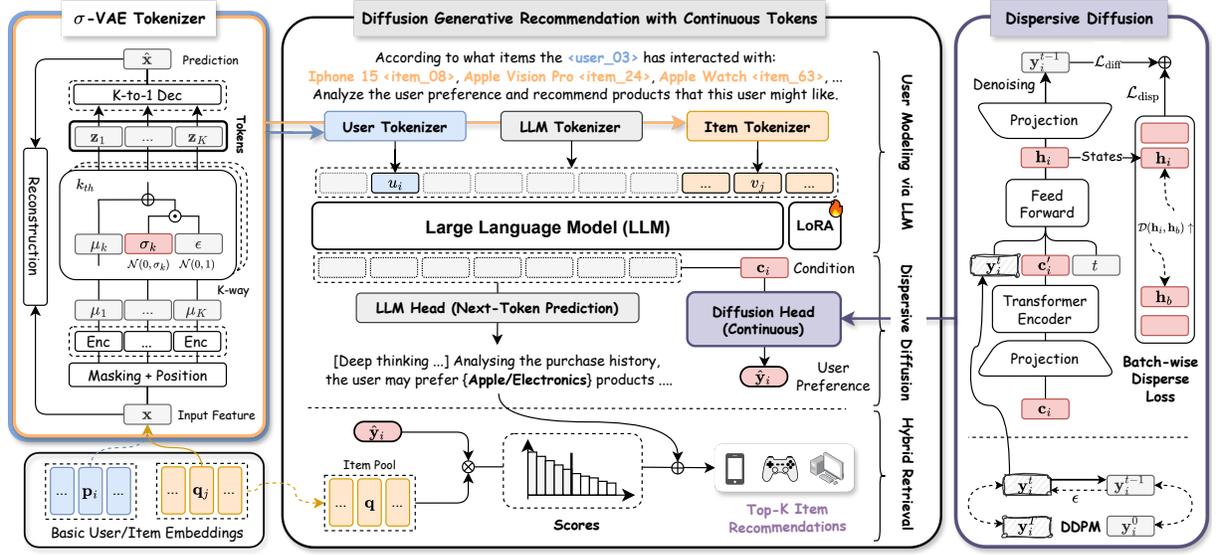
Figure 2: Overview of the proposed ContRec. ContRec represents users&items as latent vector representations using a not-quantized tokenizer and leverages the exceptional continuous-valued generation capability of diffusion models to operate within continuous spaces and generate implicit user preferences conditioned on the reasoning content of LLMs.

First, the input feature ($\mathbf{x}$) is basic user/item embeddings in various forms. For example, for an item, it can be the embedding derived from its textual description or a collaborative embedding learned by graph-based models. Specifically, we set $\mathbf{x} = \mathbf{q}_j$ for item v_j and $\mathbf{x} = \mathbf{p}_i$ for user u_i. Second, Mask$(\cdot, \rho)$ refers to an element-wise masking strategy following a Bernoulli distribution with a masking ratio of ρ. This operation generates a boolean vector of the same size as the collaborative embeddings, where each element is set to "True" with probability ρ and "False" with probability $(1 - \rho)$. Enc^k denotes the k_{th} sub-encoder, while Dec represents a K-to-1 decoder. Both components can be implemented as three-layer Multi-Layer Perceptron (MLP) models. Moreover, μ_k is the latent state encoded by the k_{th} sub-encoder, while $\mathbf{z}_k$ is the corresponding continuous-distributed token. In the end, each item v_j or user u_i (coped as $\mathbf{x}$ in the Tokenizer) can be tokenized as $\{\mathbf{z}_1, \ldots, \mathbf{z}_K\}$, which are fed into the decoder to achieve the prediction value $\hat{\mathbf{x}}_j$ and trained by minimizing the reconstruction loss:

$$\mathcal{L}_{\text{vae}} = \|\hat{\mathbf{x}} - \mathbf{x}\|_2^2 + \frac{\beta}{K} \sum_{k=1}^{K} \|\mu_k\|_2^2. \tag{12}$$

The first term measures the reconstruction error, and the second term is to constrain the adherence to the prior distribution [17]. This constraint limits the capacity of $\mathbf{z}_k$, and encourages the model to learn the most efficient representation of the data. The hyper-parameter β controls the trade-off between reconstruction quality and learning efficiency.

3.3 User Modeling via Large Language Model

The goal of user modeling is to capture the preference of a certain user (u_j) with an adequate representation. In contrast to the majority of existing methods, which use LLMs to predict recommendation results directly, our LLM conducts user modeling by decoding user preferences in natural language and outputs the implicit user preference representation as a byproduct. As shown in Figure 2, the LLM backbone summarizes the user preference (e.g., brands and categories) based on the given purchase history. The input $\mathcal{X}_i$ of our LLM backbone can be formed by selecting a prompt template $\mathcal{P}$ and introducing the corresponding titles and ID tokens for user u_i, as well as his/her historical interacted item set $\mathcal{V}_{(u_i)}$ as follows:

$$\mathcal{X}_i := \mathcal{P}(\mathcal{T}_i, \{\mathcal{T}_j | v_j \in \mathcal{V}_{(u_i)}\}), \tag{13}$$

where $\mathcal{T}_i$ and $\mathcal{T}_j$ are the tokens used to represent user u_i and item v_j, respectively. In our cases, $\mathcal{T}_j$, for example, includes the title (represented by discrete and in-vocabulary tokens) of item v_j and its embeddings $\{\mathbf{z}_j^1, \ldots, \mathbf{z}_j^K\}$ (derived from the previous σ-VAE tokenizer module), which incorporates both discrete semantic information and continuous collaborative knowledge. Notably, two special tokens, $\langle z\_start\rangle$ and $\langle z\_end\rangle$, are employed to denote the beginning and end of the continuous token sequence, respectively. Please refer to Appendix B for detailed prompt templates. Following the definition of generative recommendation shown in Eq (1), the LLM backbone models the user preference with its hybrid descriptions, with discrete and continuous tokens, by a next-token prediction format: $\mathcal{Y}_i = \text{LLM4Rec}(\mathcal{X}_i)$. The learning objective is to minimize the following autoregressive cross-entropy loss:

$$\mathcal{L}_{\text{llm}} = -\sum_{l=1}^{L} \log P(\mathbf{y}_i^l | \mathbf{y}_i^{<l}, \mathcal{X}_i), \tag{14}$$

where L is the maximum length of generated content, and $\mathbf{y}_i^l$ is the lth token of the preference reasoning texts for user u_i. The LLM outputs $\mathcal{Y}_i$ may be intermediate features to generate a final recommendation (see details in Section 3.5.2). Meanwhile, they can also be directly decoded into natural language to illustrate the reasoning process of the LLM and provide justifications for recommendation results, thereby enhancing user trust.

3.4 Dispersive Diffusion

Denoising diffusion models have recently demonstrated robust capabilities in modeling complex data distributions and learning continuous representation spaces for various modalities, including images [28], protein structures [47], and videos [41]. Transferring this success to LLM-based recommender systems is a promising direction to help with the delicate user preference modeling [19, 31]. To investigate the feasibility of applying diffusion models to LLM-based recommender systems, we conduct a mathematical analysis that proves the integration of continuous representations in the LLM backbone [28]. Specifically, language models formulate the generation recommendation problem as "next token prediction" in an autoregressive manner as follows:

$$P(\mathbf{y}_i^1, \ldots, \mathbf{y}_i^L) = \prod_{l=1}^{L} P(\mathbf{y}_i^l | \mathbf{y}_i^1, \ldots, \mathbf{y}_i^{l-1}, \mathcal{X}_i). \quad (15)$$

A common solution is to apply the generated tokens $\{\mathbf{y}_i^1, \ldots, \mathbf{y}_i^L\}$ of the LLM backbone as the representation to describe user preference. However, such a description is found to be a discrete distribution; therefore, it is less effective in modeling the complexity and subtlety of user preferences. To this end, we introduce a diffusion model conditioned on the output of the LLM backbone to further learn a continuous-distributed user preference representation space. Specifically, we aggregate LLM-output tokens for all previous steps to obtain the conditioning vector $\mathbf{c}_i := f_{\mathbf{c}}(\mathbf{y}_i^1, \ldots, \mathbf{y}_i^L)$. We define the continuous-learning model as $P(\mathbf{y}_i^{L+1} | \mathbf{c}_i)$, which can also be seen as the additional-token probability of the LLM. We propose parameterizing this model using diffusion models, in which the gradient can be backpropagated to $\mathbf{c}$ and further to the LLM backbone, enabling end-to-end optimization for the entire recommendation model.

Specifically, we employ a conditional Denoising Diffusion Probabilistic Model (DDPM) with Classifier-free Guidance [20]. The forward process follows the standard DDPM formulation as presented in Eq. (6), while the reverse process recovers the latent representation of the target item from a standard Gaussian distribution, conditioned on $\mathbf{c}_i$. Consider the denoising transition is intractable, the reverse process chooses to learn a model to estimate the noise ϵ given the noisy token $\mathbf{y}_i^t$ at the tth step. The closed-form objective of diffusion is to minimize the following loss:

$$\mathcal{L}_{\text{diff}} = \mathbb{E}_{(\mathbf{y}_i, \mathbf{c}_i, t)} \|\mathcal{F}_\theta(\mathbf{y}_i^t, \mathbf{c}_i, t) - \mathbf{y}_i\|_2^2, \quad (16)$$

$$= \mathbb{E}_{(\mathbf{y}_i, \mathbf{c}_i, t)} \|\epsilon - \epsilon_\theta(\mathbf{y}_i^t, \mathbf{c}_i, t)\|_2^2, \quad (17)$$

where $t \in \{1, \ldots, T\}$ is the sampled timestep, each corresponding to a time embedding $\mathbf{e}_t$; and θ denotes the learnable parameters of the denoising network. $\mathbf{y}_i$ is the target representation, which corresponds to the embedding of the ground-truth item, namely the target item to be recommended for the target user u_i. $\hat{\mathbf{y}}_i^t$ is the predicted representation during the diffusion process at t^{th} step. As illustrated in Figure 2, the denoising network comprises a Transformer encoder layer to encode the conditioning information, a feed-forward layer to fuse features, and two projection layers for dimensionality mapping.

The training objective typically consists of a regression term focused on reconstruction (e.g., denoising), but lacks an explicit regularization term on the representations learned for generation [53]. Contrastive learning provides a conceptually simple yet effective framework for learning representations from sample pairs in recommendation tasks [3, 51]. It encourages attraction between item pairs interacted with by the same user ("positive pairs") and repulsion between those that are not ("negative pairs"). Mathematically, the typical contrastive loss in recommendation can be formulated as:

$$\mathcal{L}_{\text{contrast}} = -\log \frac{\exp(-\mathcal{D}(\hat{\mathbf{y}}_i, \mathbf{y}_i)/\iota)}{\sum_m \exp(-\mathcal{D}(\hat{\mathbf{y}}_i, \mathbf{y}_m)/\iota)}, \quad (18)$$

where $(\hat{\mathbf{y}}_i, \mathbf{y}_m)$ represents sample pairs, including both the positive pair and all negative pairs. $\mathcal{D}(\cdot)$ is the squared ℓ_2 distance, and ι is a temperature parameter for contrastive learning, which also controls the learning speed of the diffusion process in our cases.

Despite the potential of contrastive learning as a regularizer, directly applying it would require negative sampling, which is computationally intensive in the diffusion process. Therefore, an emerging technique known as Dispersive Loss [43] is introduced to encourage internal representations in the diffusion process to disperse within the hidden space. It is analogous to the repulsive effect of standard contrastive learning, while it requires no positive sample pairs and therefore does not interfere with the sampling process used for regression. As the original regression loss in diffusion models naturally serves as an alignment mechanism, defining positive pairs as required in standard contrastive learning would be unnecessary. Let h_i denote the internal representations of the feed-forward layer in our diffusion model for an input sample user u_i, we define our Dispersive Loss here as:

$$\mathcal{L}_{\text{disp}} = \log \mathbb{E}_{i,b}[\exp(-\mathcal{D}(\mathbf{h}_i, \mathbf{h}_b)/\iota)], \quad (19)$$

where $(\mathbf{h}_i, \mathbf{h}_b)$ represents any pair within a batch of samples. In essence, Dispersive Loss functions as a "contrastive loss without positive pairs." Detailed derivations can be found in Appendix B.

3.5 Training and Inference

3.5.1 **Training**. In our model, σ-VAE Tokenizer is responsible for mapping users and items into continuous tokens, while the hybrid LLM-Diffusion recommender backbone explores user preferences and generates personalized recommendations. To optimize the whole model, we first train the Tokenizer using a self-supervised reconstruction objective to minimize $\mathcal{L}_{\text{vae}}$ as shown in Eq. (12). We then train our LLM-Diffusion backbone on recommendation data and tasks, keeping the pre-trained tokenizer components fixed throughout this stage. In this stage, three objectives are optimized jointly to build an end-to-end generative recommendation model combining the capabilities of LLMs and Diffusion. The integrated objective is defined as:

$$\mathcal{L} = \mathcal{L}_{\text{llm}} + \gamma_1(\mathcal{L}_{\text{diff}} + \gamma_2 \mathcal{L}_{\text{disp}}), \quad (20)$$

where γ_1 controls the trade-off between discrete output and continuous output, and γ_2 controls the impacts of the dispersive regularization. Notably, we incorporate a small proportion of dummy conditions during the training of our denoising network to enhance the robustness and overall performance of our generative model.

3.5.2 **Inference**. For the Diffusion process [20] during inference, we introduce a hyperparameter ω that controls the strength of conditioning to modulate the effect of the guidance signal $\mathbf{c}_i$. We use ω to modify the noise prediction in Eq. (17). While a higher value of ω can enhance personalized guidance, it may also reduce the

generalization ability of the diffusion process, potentially deteriorating the quality of the generated oracle item. Mahematically, the modified noise prediction process can be formulated as follows:

$$\tilde{\epsilon}_\theta(\mathbf{y}_i^t, \mathbf{c}_i, t) = (1 + \omega)\epsilon_\theta(\mathbf{y}_i^t, \mathbf{c}_i, t) - \omega\epsilon_\theta(\mathbf{y}_i^t, \Phi, t), \qquad (21)$$

where Φ is a dummy conditioning token randomly sampled from a normal distribution.

For personalized recommendation inference, we present a **hybrid retrieval paradigm** that considers not only the textual reasoning output from LLM but also the continuous representations $\hat{\mathbf{y}}_i$ of user preference derived from a denoising network [7, 34]. In the retrieval process, we first measure the similarity score between the $\hat{\mathbf{y}}_i$ and the predefined item representation $\hat{\mathbf{q}}_j$ for each candidate item. Meanwhile, we extract semantic information for the predicted recommendation items from the textual reasoning output of the LLM, such as category and brand. We introduce a Boolean indicator π and combine the two parts to obtain the final similarity score for each user-item pair as follows:

$$s_{ij} = \frac{\hat{\mathbf{y}}_i \mathbf{q}_j}{\|\hat{\mathbf{y}}_i\|\|\mathbf{q}_j\|} \cdot (1 + \pi). \qquad (22)$$

For any item (v_j) in the candidate pool, if it is with the same semantic features as predicted by LLM, we set π to a positive value (e.g., 0.05) and otherwise to 0. By ranking the items based on similarity scores calculated by Eq. (22), ContRec seamlessly delivers Top-K item recommendations through a single inference process, transitioning from LLM to Diffusion. In brief, the LLM provides a general description of user preferences, while the diffusion model generates implicit representations accordingly.

4 Experiment

4.1 Experimental Settings

4.1.1 Datasets. We conducted a series of comprehensive experiments across four benchmark datasets to evaluate the effectiveness of the proposed ContRec and core modules in it. First, LastFM, sourced from the *HetRec 2011*[1] repository, encompasses music artist listening data from users within the Last.fm online music platform. Moreover, the ML1M dataset offers a collection of movie ratings made by *MovieLens*[2] users. The last two datasets—Beauty and Video Games—are sourced from the *Amazon Review Data*[3] repository. These datasets have undergone a reduction process to derive their 5-core versions, ensuring that each user and item retains a minimum of five reviews. Table 1 details the characteristics of these datasets, with the maximum item sequence length capped at 20 to align with a 1024-token input constraint. In our cases, each item's title, ID, brand, and category are used, while only the ID is considered on the user side. Moreover, our dataset division follows the Split-by-Timepoint policy [24], in which all interactions occurring after a predefined time point are designated as test instances, while those before the time point are used for training. To ensure that all items appear at least once in the training corpus, we split the recommendation data in chronological order at the 90th and 95th percentiles to create the training, validation, and testing sets.

---

[1]https://grouplens.org/datasets/hetrec-2011/
[2]https://grouplens.org/datasets/movielens/1m/
[3]https://nijianmo.github.io/amazon/

Table 1: Basic statistics of benchmark datasets.

Datasets	Statistics				
	#Users	#Items	#Interactions	Length	Density (%)
LastFM	1,091	3,685	52,670	48.3	1.3101
ML1M	6,040	3,416	447,294	165.5	2.1679
Beauty	22,363	12,101	278,641	8.9	0.0734
Games	47,568	16,834	266,139	9.5	0.0332

4.1.2 Compared Models. Here, we compare our approach with four representative sequential recommendation methods and five recently developed LLM-based RecSys. The four sequential recommendation methods are **GRU4Rec** [16], **SASRec** [25], **SSD4Rec** [56], and **DreamRec** [52], with DreamRec being a diffusion-based method. The five LLM-based models are: (i) **P5** [13] reformats recommendation tasks into a text-to-text format and finetunes a small pretrained language model (i.e., T5) to capture more profound semantics for recommendation. (ii) **CoLLM** [55] incorporates collaborative representations of items into LLMs using a projection layer. (iii) **TIGER** [36] employs residual vector quantization to distill large volumes of text into a compact set of semantic identifiers, and then leverages a Transformer-based architecture trained on these semantic ID sequences to perform sequential recommendation. (iv) **LLaRA** [29] uses a projector to align item embeddings with textual item information and implements a curriculum learning approach for finetuning. (v) **TokenRec** [34] proposes a novel quantized tokenizer to encode users and items along with their collaborative knowledge into discrete tokens that are compatible with LLMs.

4.1.3 Evaluation Metrics. Two commonly used metrics are used: Top-K Hit Ratio (HR@K) and Top-K Normalized Discounted Cumulative Gain (NDCG@K) [15], where higher values indicate superior performance. The evaluation entails presenting the average metrics for all users in the test set. Furthermore, the values of K are specified as 10 and 20, with 10 serving as the default setting for ablation experiments and parameter analyses.

4.1.4 Configurations. We implement all models using Python 3.11 and PyTorch 2.5.1 on an NVIDIA A800 (80 GB) GPU. We use AdamW with learning rates and weight decays of 0.00001/0.0001 for the backbone and 0.0001/0.001 for the tokenizer. The maximum length of the item sequence is configured to 20. The masking ratio ρ is searched from 0 to 1 in 0.2 increments, while the number of sub-encoders is selected from {1, 2, 3, 4, 5}. Taking into account both performance and computational resources, we opt to utilize the *Llama-3.2-1B-Instruct*[4] as our LLM backbone, which is renowned as one of the most popular pre-trained language models globally. The LLM backbone operates with a default temperature of 0. We create ten prompt templates and three response templates for the input and output of the LLM backbone, respectively. A random selection from these templates is made in each batch to enhance the model's capacity to generalize across various template structures. The two loss weights γ_1 and γ_2 are sequential searched in the range of {0, 0.5, 1.0, 1.5, 2.0}. The basic embeddings for users and items are derived from both textual information encoded by SentenceBERT [38] and collaborative knowledge learned through LightGCN [15]. Our noise schedule consists of 1000 steps during training; during inference, it is resampled with fewer steps (100). We fix the unconditional training probability ζ as 0.1 suggested by [20], and the diffusion

---

[4]https://huggingface.co/meta-llama/Llama-3.2-1B-Instruct

Table 2: Performance comparison between representative baselines and ContRec across four commonly used datasets. The best and second-best results are highlighted in bold and underlined fonts, respectively. For ContRec, we conduct inference five times and report the mean and standard deviation. The improvements over baselines are statistically significant ($p < 0.01$).

Dataset	Metric	GRU4Rec	SASRec	SSD4Rec	DreamRec	P5	POD	CoLLM	TIGER*	TokenRec	LLaRA	ContRec	Improve.*
Beauty	HR@10	0.0394	0.0407	0.0412	0.0423	0.0411	0.0414	0.0429	0.0439	0.0438	<u>0.0442</u>	**0.0473**$_{\pm 0.0017}$	7.7449%
	HR@20	0.0595	0.0611	0.0628	0.0642	0.0594	0.0606	0.0609	0.0620	0.0619	<u>0.0631</u>	**0.0669**$_{\pm 0.0024}$	7.9032%
	NDCG@10	0.0204	0.0211	0.0218	0.0222	0.0236	0.0227	0.0237	0.0243	0.0226	<u>0.0237</u>	**0.0251**$_{\pm 0.0011}$	3.2922%
	NDCG@20	0.0253	0.0258	0.0256	0.0265	0.0258	0.0254	0.0266	0.0269	0.0273	<u>0.0283</u>	**0.0295**$_{\pm 0.0012}$	9.6654%
Games	HR@10	0.0732	0.0783	0.0875	0.0915	0.0839	0.0858	0.0928	0.0958	<u>0.1018</u>	0.0997	**0.1041**$_{\pm 0.0036}$	8.6639%
	HR@20	0.1139	0.1231	0.1345	0.1427	0.1269	0.1286	0.1398	0.1438	<u>0.1506</u>	0.1471	**0.1606**$_{\pm 0.0074}$	11.6829%
	NDCG@10	0.0365	0.0447	0.0468	0.0489	0.0430	0.0426	0.0491	0.0546	<u>0.0570</u>	0.0568	**0.0588**$_{\pm 0.0017}$	7.6923%
	NDCG@20	0.0467	0.0536	0.0545	0.0626	0.0554	0.0542	0.0618	0.0679	<u>0.0705</u>	0.0690	**0.0704**$_{\pm 0.0013}$	3.6819%
LastFM	HR@10	0.0340	0.0353	0.0358	0.0359	0.0386	0.0402	0.0468	0.0467	0.0489	<u>0.0525</u>	**0.0539**$_{\pm 0.0034}$	15.4176%
	HR@20	0.0467	0.0470	0.0469	0.0493	0.0555	0.0672	0.0732	0.0749	0.0755	<u>0.0827</u>	**0.0890**$_{\pm 0.0067}$	18.8251%
	NDCG@10	0.0191	0.0202	0.0195	0.0198	0.0190	0.0219	0.0228	0.0226	0.0238	<u>0.0244</u>	**0.0250**$_{\pm 0.0010}$	10.6195%
	NDCG@20	0.0239	0.0237	0.0235	0.0244	0.0225	0.0268	0.0305	0.0306	0.0311	<u>0.0325</u>	**0.0338**$_{\pm 0.0015}$	10.4575%
ML1M	HR@10	0.0877	0.0964	0.0951	0.0944	0.0867	0.0886	0.0923	0.0954	<u>0.1076</u>	0.1012	**0.1099**$_{\pm 0.0066}$	15.1992%
	HR@20	0.1250	0.1382	0.1345	0.1371	0.1248	0.1277	0.1499	0.1548	0.1506	**0.1672**	<u>0.1666</u>$_{\pm 0.0084}$	7.6227%
	NDCG@10	0.0379	0.0399	0.0396	0.0403	0.0381	0.0373	0.0456	0.0472	<u>0.0542</u>	0.0532	**0.0561**$_{\pm 0.0039}$	18.8559%
	NDCG@20	0.0487	0.0510	0.0511	0.0540	0.0486	0.0487	0.0620	0.0644	<u>0.0716</u>	0.0698	**0.0717**$_{\pm 0.0035}$	11.3354%

temperature ι to 0.5 following [43]. We search for the personalized guidance strength ω in the range of {1, 2, 4, 6, 8, 10}. More implementation details are presented in Appendix C.

4.2 Recommendation Performance

Table 2 presents the overall performance comparison between ContRec and the baselines over four investigated datasets. From the table, we have the following observations:

- The proposed model achieves superior performance, outperforming all representative baselines in the majority of cases. On average, ContRec significantly exceeds the typical baseline with discrete tokens (TIGER) by 11.76% on HR@10 and 10.11% on NCDG@10 across the four datasets. When compared to the strongest baseline (TokenRec and LLaRA), the enhancements amount to 5.67% and 4.33% correspondingly. Such improvement demonstrates the effectiveness of our proposed method and the great potential of exploring diffusion continuous tokens in LLM-based recommender systems.

- Leveraging its exceptional reasoning capability developed from million/billion-level parameters and extensive training data, the LLM-based methods, with personalized fine-tuning on these datasets, empirically surpass the collaborative filtering methods and sequential recommendation models.

- Among the LLM-based models, those incorporating continuous tokens for representing users/items (i.e., CoLLM and LlaRA) outperform the discrete method (i.e., P5 and TIGER), indicating the advantage of using continuous tokens. Nevertheless, the inherent disparity between discrete and continuous token spaces impedes the performance of these models, as evidenced by the improved outcomes achieved by TokenRec, which introduces an advanced tokenizer to index users and items with several discrete tokens. To operate with continuous tokens, our method adopts the proposed continuous tokenizer and dispersive diffusion, enabling LLM-based RecSys to benefit from non-quantized user/item representations and attain higher-quality generation.

- Additionally, among deep learning methods, DreamRec outperforms traditional sequential recommendation approaches such as GRU4Rec, SASRec, and SSD4Rec. These results highlight the

effectiveness of diffusion models in generating precise user preferences through step-by-step denoising diffusion.

4.3 In-depth Analysis

*4.3.1 **Model Optimization Robustness Analysis**.* We assess the sensitivity of the optimization process of ContRec's with regard to two hyperparameters, as defined in the end-to-end learning loss in Eq. (20). γ_1 controls the trade-off between LLM and Diffusion parts, and γ_2 determines the impact of dispersive loss specifically in the Diffusion. Using a sequential grid search over {0, 0.5, 1, 1.5, 2}, we first vary γ_1 with γ_2 fixed at 0, then assess the effect of γ_2 using the optimal γ_1 from the first step. As illustrated in Figure 3, the results show that the model performance stays relatively stable with the shift of the two hyperparameters. This demonstrates the robustness of ContRec's learning process. Moreover, optimal performance for γ_1 is achieved when its value is set to 1, while the optimal value for γ_2 is 0.5.

*4.3.2 **Effect of Key Hyperparameters**.* This section examines three critical hyperparameters in our approach: the classifier-free guidance ω, the masking ratio ρ, and the token number K for each user/item. Figure 4 illustrates the NDCG@10 and HR@10 change observed in ContRec. First, the hyperparameter ρ controls the masking ratio of our encoder for robustly tokenizing users and items with continuous tokens. The results show that minor incorporation of masking during training yields performance enhancements, with a ratio of 0.2 emerging as the optimal value in our experiments. Furthermore, our experimental findings demonstrate that the recommendation performance deteriorates for masking ratios $\rho \geq 0.4$, underscoring the importance of avoiding excessive masking practices. Second, the figure shows that indexing users and items with three or four tokens yields the best results for datasets such as Beauty, while two tokens are sufficient for ML1M due to its relatively smaller number of users and items. This observation is generally consistent with findings reported in previous studies, e.g., TIGER [36], TokenRec [34], and OneRec [5]. Finally, Classifier-free guidance [20] is widely used in diffusion models to balance fidelity and realism by scaling the difference between conditional and unconditional predictions. Following previous studies [28, 30, 41], we
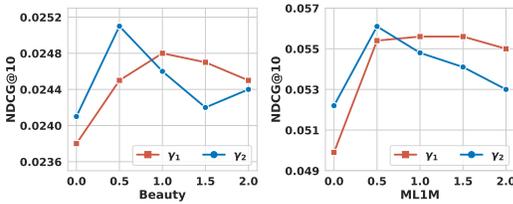
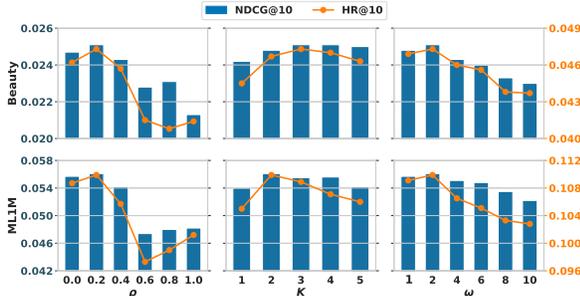Figure 3: Optimization robustness on the Beauty and ML1M datasets.

Figure 4: Hyper-parameter tuning on the Beauty and ML1M datasets.

search the hyperparameter ω in the range of {0, 2, 4, 6, 8, 10}. The results show that employing a small CFG ($\omega = 2$) improves overall performance across both datasets, while larger values may lead to degradation in output quality.

4.3.3 **Reconstruction Evaluation**. The reconstruction performance of a model largely reflects the quality of its learned representation, which in turn affects its performance on downstream tasks [4, 42], such as the item/user tokenizer in our recommendation task. To investigate the effectiveness of continuous tokens in recommendation scenarios, we evaluate various methods for the item embedding reconstruction task on the Beauty dataset and report all the performance in Figure 1. In our experimental setting, each model takes the item embeddings learned by LightGCN [15] as input and encodes them into tokens, which are then decoded to reconstruct the input (for diffusion models, this reconstruction process includes adding and removing noise). We can see that, although advanced quantization-based methods (RQ- and MQ-VAE) demonstrate improvements compared to VAE and VQ-VAE, methods with continuous tokens and decoding using the diffusion model achieve superior reconstruction performance and loss convergence. This can be attributed to the ability of continuous tokens and diffusion models to model subtle differences.

4.4 Ablation Study

We conducted ablation experiments on the Beauty and ML1M datasets, where each key component was eliminated separately as follows: (i) w/o Masking: The masking operation in the encoder is deactivated. (ii) w/o User: The user tokens are removed, leaving only item tokens in our prompt. (iii) w/o σ: The regularization coefficient in Eq. (11) is removed. (iv) w/o Tokenizer: The proposed tokenizer is replaced with a projection layer. (v) w/o Text: The basic embeddings exclude textual item-side information. (vi) w/o Collaborative: The basic embeddings are constructed without collaborative embeddings. (vii) w/ RQ-VAE: The continuous σ-VAE is substituted with a discrete RQ-VAE. (viii) w/ VQ-VAE: The σ-VAE

Table 3: Results of ablation studies. NDCG@10 is abbreviated as NG@10 throughout the table.

Module	Conponent	Beauty		ML1M	
		HR@10	NG@10	HR@10	NG@10
ContRec	Full	0.0473	0.0251	0.1099	0.0561
Tokenization	w/o Masking	0.0462	0.0247	0.1087	0.0557
	w/o User	0.0461	0.0246	0.1066	0.0555
	w/o σ	0.0457	0.0242	0.1051	0.0536
	w/o Tokenizer	0.0438	0.0237	0.1018	0.0511
	w/o Text	0.0456	0.0242	0.1070	0.0550
	w/o Collaborative	0.0448	0.0236	0.1030	0.0534
	w/ RQ-VAE	0.0438	0.0236	0.0996	0.0508
	w/ VQ-VAE	0.0426	0.0236	0.0974	0.0488
Generation	w/o π	0.0463	0.0244	0.1063	0.0544
	w/o $\mathcal{L}_{\text{disp}}$	0.0448	0.0241	0.1042	0.0522
	w/o Diffusion	0.0431	0.0238	0.1007	0.0499

is replaced with a vanilla VQ-VAE. (viv) w/o π: The LLM reasoning result is excluded from the retrieval process. (vv) w/o $\mathcal{L}_{\text{disp}}$: The dispersive loss is removed from the overall objective function in Eq. (20). (vvi) w/o Diffusion: The diffusion head is removed and replaced with a projection layer. From the ablation results presented in Table 3, we can draw several key observations. First, all major components of our approach contribute to overall performance, as removing any single component leads to a decrease in effectiveness. Second, the continuous σ-VAE consistently outperforms its discrete counterparts, RQ-VAE and VQ-VAE. Finally, eliminating either the dispersive loss or the diffusion module results in a substantial drop in performance, highlighting the critical importance of the proposed diffusion-based continuous token generation in generative recommendation systems. Further results, including inference costs, case analyses, and generalizability, are provided in Appendix C.

5 Conclusion

To overcome the limitations of lossy tokenization and suboptimal learning in quantization methods, we introduced ContRec, a novel framework that integrates continuous tokenization into LLM-based recommender systems. By leveraging a σ-VAE Tokenizer and a Dispersive Diffusion module, ContRec effectively encodes users and items as continuous tokens and captures implicit user preferences through conditional diffusion processes. Extensive experiments on four benchmark datasets demonstrate that ContRec consistently outperforms both traditional and state-of-the-art LLM-based recommender systems. These results highlight the significant potential of continuous tokenization and generative modeling in advancing the next generation of recommender systems.

Acknowledgments

References

[1] Keqin Bao, Jizhi Zhang, Yang Zhang, Wenjie Wang, Fuli Feng, and Xiangnan He. 2023. Tallrec: An effective and efficient tuning framework to align large language model with recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*. 1007–1014.

[2] Yoshua Bengio, Nicholas Léonard, and Aaron Courville. 2013. Estimating or propagating gradients through stochastic neurons for conditional computation. *arXiv preprint arXiv:1308.3432* (2013).

[3] Xuheng Cai, Chao Huang, Lianghao Xia, and Xubin Ren. 2023. LightGCL: Simple Yet Effective Graph Contrastive Learning for Recommendation. In *The Eleventh International Conference on Learning Representations*.

[4] Hao Chen, Ze Wang, Xiang Li, Ximeng Sun, Fangyi Chen, Jiang Liu, Jindong Wang, Bhiksha Raj, Zicheng Liu, and Emad Barsoum. 2025. Softvq-vae: Efficient 1-dimensional continuous tokenizer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 28358–28370.

[5] Jiaxin Deng, Shiyao Wang, Kuo Cai, Lejian Ren, Qigen Hu, Weifeng Ding, Qiang Luo, and Guorui Zhou. 2025. Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment. *arXiv preprint arXiv:2502.18965* (2025).

[6] Lijie Fan, Tianhong Li, Siyang Qin, Yuanzhen Li, Chen Sun, Michael Rubinstein, Deqing Sun, Kaiming He, and Yonglong Tian. 2025. Fluid: Scaling Autoregressive Text-to-image Generative Models with Continuous Tokens. In *The Thirteenth International Conference on Learning Representations*.

[7] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li. 2024. A survey on rag meeting llms: Towards retrieval-augmented large language models. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*. 6491–6501.

[8] Wenqi Fan, Xiaorui Liu, Wei Jin, Xiangyu Zhao, Jiliang Tang, and Qing Li. 2022. Graph trend filtering networks for recommendation. In *Proceedings of the 45th international ACM SIGIR conference on research and development in information retrieval*. 112–121.

[9] Wenqi Fan, Yao Ma, Qing Li, Yuan He, Eric Zhao, Jiliang Tang, and Dawei Yin. 2019. Graph neural networks for social recommendation. In *The world wide web conference*. 417–426.

[10] Wenqi Fan, Yao Ma, Qing Li, Jianping Wang, Guoyong Cai, Jiliang Tang, and Dawei Yin. 2020. A graph neural network framework for social recommendations. *IEEE Transactions on Knowledge and Data Engineering* 34, 5 (2020), 2033–2047.

[11] Wenqi Fan, Yi Zhou, Shijie Wang, Yuyao Yan, Hui Liu, Qian Zhao, Le Song, and Qing Li. 2025. Computational Protein Science in the Era of Large Language Models (LLMs). *arXiv preprint arXiv:2501.10282* (2025).

[12] Christopher Fifty, Ronald Guenther Junkins, Dennis Duan, Aniketh Iyengar, Jerry Weihong Liu, Ehsan Amid, Sebastian Thrun, and Christopher Re. [n. d.]. Restructuring Vector Quantization with the Rotation Trick. In *The Thirteenth International Conference on Learning Representations*.

[13] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. 2022. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). In *Proceedings of the 16th ACM Conference on Recommender Systems*. 299–315.

[14] Zhiye Guo, Jian Liu, Yanli Wang, Mengrui Chen, Duolin Wang, Dong Xu, and Jianlin Cheng. 2024. Diffusion models in bioinformatics and computational biology. *Nature reviews bioengineering* 2, 2 (2024), 136–154.

[15] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[16] Balázs Hidasi and Alexandros Karatzoglou. 2018. Recurrent neural networks with top-k gains for session-based recommendations. In *Proceedings of the 27th ACM international conference on information and knowledge management*. 843–852.

[17] Irina Higgins, Loic Matthey, Arka Pal, Christopher Burgess, Xavier Glorot, Matthew Botvinick, Shakir Mohamed, and Alexander Lerchner. 2017. beta-vae: Learning basic visual concepts with a constrained variational framework. In *International conference on learning representations*.

[18] Julia Hirschberg and Christopher D Manning. 2015. Advances in natural language processing. *Science* 349, 6245 (2015), 261–266.

[19] Jonathan Ho, Ajay Jain, and Pieter Abbeel. 2020. Denoising diffusion probabilistic models. *Advances in neural information processing systems* 33 (2020), 6840–6851.

[20] Jonathan Ho and Tim Salimans. [n. d.]. Classifier-Free Diffusion Guidance. In *NeurIPS 2021 Workshop on Deep Generative Models and Downstream Applications*.

[21] Wenyue Hua, Shuyuan Xu, Yingqiang Ge, and Yongfeng Zhang. 2023. How to index item ids for recommendation foundation models. In *Proceedings of the Annual International ACM SIGIR Conference on Research and Development in Information Retrieval in the Asia Pacific Region*. 195–204.

[22] Jiani Huang, Shijie Wang, Liang-bo Ning, Wenqi Fan, Shuaiqiang Wang, Dawei Yin, and Qing Li. 2025. Towards Next-Generation Recommender Systems: A Benchmark for Personalized Recommendation Assistant with LLMs. *arXiv preprint arXiv:2503.09382* (2025).

[23] Minyoung Huh, Brian Cheung, Pulkit Agrawal, and Phillip Isola. 2023. Straightening out the straight-through estimator: Overcoming optimization challenges in vector quantized networks. In *International Conference on Machine Learning*. PMLR, 14096–14113.

[24] Yitong Ji, Aixin Sun, Jie Zhang, and Chenliang Li. 2023. A critical study on data leakage in recommender system offline evaluation. *ACM Transactions on Information Systems* 41, 3 (2023), 1–27.

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

[26] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[27] Diederik P Kingma and Max Welling. 2014. Auto-Encoding Variational Bayes. *stat* 1050 (2014), 1–14.

[28] Tianhong Li, Yonglong Tian, He Li, Mingyang Deng, and Kaiming He. 2024. Autoregressive image generation without vector quantization. *Advances in Neural Information Processing Systems* 37 (2024), 56424–56445.

[29] Jiayi Liao, Sihang Li, Zhengyi Yang, Jiancan Wu, Yancheng Yuan, Xiang Wang, and Xiangnan He. 2024. Llara: Large language-recommendation assistant. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 1785–1795.

[30] Run Luo, Yunshui Li, Longze Chen, Wanwei He, Ting-En Lin, Ziqiang Liu, Lei Zhang, Zikai Song, Hamid Alinejad-Rokny, Xiaobo Xia, et al. 2025. DEEM: Diffusion models serve as the eyes of large language models for image perception. In *The Thirteenth International Conference on Learning Representations*.

[31] Alexander Quinn Nichol and Prafulla Dhariwal. 2021. Improved denoising diffusion probabilistic models. In *International conference on machine learning*. PMLR, 8162–8171.

[32] Liangbo Ning, Ziran Liang, Zhuohang Jiang, Haohao Qu, Yujuan Ding, Wenqi Fan, Xiao-yong Wei, Shanru Lin, Hui Liu, Philip S Yu, et al. 2025. A survey of webagents: Towards next-generation ai agents for web automation with large foundation models. In *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 2*. 6140–6150.

[33] Aaron van den Oord, Yazhe Li, and Oriol Vinyals. 2018. Representation learning with contrastive predictive coding. *arXiv preprint arXiv:1807.03748* (2018).

[34] Haohao Qu, Wenqi Fan, Zihuai Zhao, and Qing Li. 2025. TokenRec: Learning to Tokenize ID for LLM-Based Generative Recommendations. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[35] Haohao Qu, Liangbo Ning, Rui An, Wenqi Fan, Tyler Derr, Hui Liu, Xin Xu, and Qing Li. 2024. A survey of mamba. *arXiv preprint arXiv:2408.01129* (2024).

[36] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2023. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2023), 10299–10315.

[37] Ali Razavi, Aäron van den Oord, and Oriol Vinyals. 2019. Generating diverse high-fidelity images with VQ-VAE-2. In *Proceedings of the 33rd International Conference on Neural Information Processing Systems*. 14866–14876.

[38] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. 3982–3992.

[39] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*. PMLR, 1278–1286.

[40] Chitwan Saharia, William Chan, Huiwen Chang, Chris Lee, Jonathan Ho, Tim Salimans, David Fleet, and Mohammad Norouzi. 2022. Palette: Image-to-image diffusion models. In *ACM SIGGRAPH 2022 conference proceedings*. 1–10.

[41] Yutao Sun, Hangbo Bao, Wenhui Wang, Zhiliang Peng, Li Dong, Shaohan Huang, Jianyong Wang, and Furu Wei. 2024. Multimodal Latent Language Modeling with Next-Token Diffusion. *arXiv preprint arXiv:2412.08635* (2024).

[42] Aaron van den Oord, Oriol Vinyals, and Koray Kavukcuoglu. 2017. Neural discrete representation learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 6309–6318.

[43] Runqian Wang and Kaiming He. 2025. Diffuse and Disperse: Image Generation with Representation Regularization. *arXiv preprint arXiv:2506.09027* (2025).

[44] Shijie Wang, Wenqi Fan, Yue Feng, Lin Shanru, Xinyu Ma, Shuaiqiang Wang, and Dawei Yin. 2025. Knowledge Graph Retrieval-Augmented Generation for LLM-based Recommendation. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 27152–27168.

[45] Shijie Wang, Jiani Huang, Zhikai Chen, Yu Song, Wenzhuo Tang, Haitao Mao, Wenqi Fan, Hui Liu, Xiaorui Liu, Dawei Yin, et al. 2025. Graph machine learning in the era of large language models (llms). *ACM Transactions on Intelligent Systems and Technology* 16, 5 (2025), 1–40.

[46] Xiaoyang Wang, Yao Ma, Yiqi Wang, Wei Jin, Xin Wang, Jiliang Tang, Caiyan Jia, and Jian Yu. 2020. Traffic flow prediction via spatial temporal graph neural network. In *Proceedings of the web conference 2020*. 1082–1092.

[47] Xinyou Wang, Zaixiang Zheng, Dongyu Xue, Shujian Huang, Quanquan Gu, et al. 2025. DPLM-2: A Multimodal Diffusion Protein Language Model. In *The Thirteenth International Conference on Learning Representations*.

[48] Xinyou Wang, Zaixiang Zheng, Fei Ye, Dongyu Xue, Shujian Huang, and Quanquan Gu. 2024. Diffusion language models are versatile protein learners. In *Proceedings of the 41st International Conference on Machine Learning*. 52309–52333.

[49] Zhaoqing Wang, Xiaobo Xia, Runnan Chen, Dongdong Yu, Changhu Wang, Mingming Gong, and Tongliang Liu. 2025. Lavin-dit: Large vision diffusion transformer. In *Proceedings of the Computer Vision and Pattern Recognition Conference*. 20060–20070.

[50] Wei Wei, Xubin Ren, Jiabin Tang, Qinyong Wang, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Llmrec: Large language models with graph augmentation for recommendation. In *Proceedings of the 17th ACM International Conference on Web Search and Data Mining*. 806–815.

[51] Xu Xie, Fei Sun, Zhaoyang Liu, Shiwen Wu, Jinyang Gao, Jiandong Zhang, Bolin Ding, and Bin Cui. 2022. Contrastive learning for sequential recommendation. In *2022 IEEE 38th international conference on data engineering (ICDE)*. IEEE, 1259–1273.

[52] Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. 2023. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. *Advances in Neural Information Processing Systems* 36 (2023), 24247–24261.

[53] Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. 2025. Representation Alignment for Generation: Training Diffusion Transformers Is Easier Than You Think. In *International Conference on Learning Representations*.

[54] Zhenrui Yue, Sara Rabhi, Gabriel de Souza Pereira Moreira, Dong Wang, and Even Oldridge. 2023. LlamaRec: Two-stage recommendation using large language models for ranking. *arXiv preprint arXiv:2311.02089* (2023).

[55] Yang Zhang, Fuli Feng, Jizhi Zhang, Keqin Bao, Qifan Wang, and Xiangnan He. 2025. Collm: Integrating collaborative embeddings into large language models for recommendation. *IEEE Transactions on Knowledge and Data Engineering* (2025).

[56] Yifeng Zhang, Haohao Qu, Liangbo Ning, Wenqi Fan, and Qing Li. 2025. SSD4Rec: A Structured State Space Duality Model for Efficient Sequential Recommendation. *ACM Transactions on Information Systems* 44, 2 (2025), 1–26.

[57] Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223* (2023).

[58] Zihuai Zhao, Wenqi Fan, Jiatong Li, Yunqing Liu, Xiaowei Mei, Yiqi Wang, Zhen Wen, Fei Wang, Xiangyu Zhao, Jiliang Tang, et al. 2024. Recommender systems in the era of large language models (llms). *IEEE Transactions on Knowledge and Data Engineering* (2024).

[59] Bowen Zheng, Hongyu Lu, Yu Chen, Wayne Xin Zhao, and Ji-Rong Wen. 2025. Universal Item Tokenization for Transferable Generative Recommendation. *arXiv preprint arXiv:2504.04405* (2025).

A Related Work

A.1 LLM-based Recommender Systems

The rapid advancement of Large Language Models (LLMs), such as ChatGPT and GPT-4, has significantly transformed natural language processing [11, 26, 32, 35]. As a result, LLMs have increasingly been integrated into recommender systems, enhancing their capabilities [22, 44, 58]. To effectively leverage LLMs in recommendation tasks, two primary challenges must be addressed: the tokenization of users and items, and the generation of recommendations. Most research has focused on adapting recommender systems to the language domain using discretization techniques, which align with the discrete nature of natural language. A notable example is P5 [13], which unifies various recommendation tasks into natural language processing through multi-task prompt-based pre-training, achieving impressive zero-shot generalization with personalized prompts. Following this, simple tokenization methods like title indexing, independent indexing, and sequential indexing have become common [21], as seen in studies like TallRec [1] and LLMRec [50]. However, these methods struggle to capture complex collaborative knowledge and are limited by LLMs' input length constraints. To overcome these limitations, researchers have developed robust tokenizers that convert users, items, and their interactions into discrete tokens. Examples include TIGER [36] and TokenRec [34], which use vector-quantized tokenizers to encode textual information and

collaborative representations from graph neural network-based methods [15]. Despite the performance improvements offered by advanced discrete-valued tokenizers, they are hindered by information loss during compression or quantization. Consequently, there is growing interest in continuous-valued tokens, which have shown promise in representation learning [29, 54, 55]. However, this approach is still in its early stages of development. Such a direct approach faces challenges in bridging the substantial gap between the continuous collaborative knowledge that GNNs capture from user-item interactions and the semantic information that LLMs learn from discrete natural language.

A.2 Diffusion Models with Continuous Tokens

Denoising diffusion probabilistic models, also known as DDPM or diffusion models, function based on the concept of learning to reverse a gradual noise-addition process [19, 31]. In contrast to language models that commonly operate with discrete tokens, diffusion models work with continuous vectors, rendering them well-suited for tasks that involve continuous data, such as images. For example, palette [40] demonstrates strong performance across a range of image-to-image generation tasks by employing a general diffusion model, eliminating the need for task-specific fine-tuning. Given the success of diffusion models in continuous data generation, an increasing number of studies have proposed combining large language models with diffusion models in cross-modal modeling, with both continuous and discrete data as inputs and targets. Examples include VAE-MAR [28] and Next-Token Diffusion [41], which employ a variational autoencoder (VAE) to represent continuous image data as latent vectors and introduce denoising diffusion for autoregressive generation of these vectors. In these studies, language models continue their imposition on discrete language generation and comprehension, while diffusion models are incorporated as generation heads to operate within continuous domains. Despite that, the integration of diffusion models with auto-regressive models is currently in its nascent phase.

B Further Explanation of ContRec

B.1 Algorithm of the Proposed Framework

The pseudo-code of the proposed framework during training and inference is described in Algorithm 1. The ultimate goal is to develop a comprehensive understanding of user preferences for generating effective and accurate recommendations. Top-K relevant items can be retrieved from the ranking list as the recommended output.

B.2 Derivation of Dispersive Loss

In our case, we derive the dispersive loss [43] from the typical contrastive learning loss, namely InfoNCE loss. Formally, the original InfoNCE loss [33] can be interpreted as a categorical cross-entropy objective that encourages high similarity between positive pairs and low similarity between negative pairs, as shown in Eq. (18). We can rewrite this equation as:

$$\mathcal{L}_{\text{contrast}} = \mathcal{D}(\hat{y}_i, y_i)/\tau + \log \sum_m \exp(-\mathcal{D}(\hat{y}_i, y_m)/\tau), \quad (23)$$

where the numerator involves only the positive pair, whereas the denominator includes all pairs in the batch. Here, the first term

---

Algorithm 1 ContRec: Pseudo-code.

Init: Embeddings bases for users $\{\mathbf{p}_i\}^n$ and items $\{\mathbf{q}_j\}^m$
Training:
Phase1: σ-VAE Tokenizer
1: **for** each $\mathbf{p}_i$ or $\mathbf{q}_j$ **do**
2: 　　Encoding: $\mu_k = \text{Enc}^k(\text{Mask}(\mathbf{x}, \rho))$;
3: 　　Not-quantized tokenization: $\mathbf{z}_k = \mu_k + \sigma_k \odot \epsilon$;
4: 　　Decoding: $\hat{\mathbf{x}} = \text{Dec}(\text{Concat}\{\mathbf{z}_k\}^K)$;
5: 　　Calculating the reconstruction loss based on Eq. (12).
6: **end for**
7: **return** K continuous tokens for each user and item.
Phase2: LLM and Diffusion
8: **for** each u_i with item interactions $\mathcal{V}_{u_i}$ **do**
9: 　　Prompting to $\mathcal{X}_i$ as expressed in Eq. (13);
10: 　　Auto-regressive generation via LLM: $\mathcal{Y}_i = \text{LLM}(\mathcal{X}_i)$;
11: 　　Continuous generation via Diffusion: $\mathbf{y}_i = \text{Diffusion}(\mathbf{c}_i, \epsilon)$;
12: 　　Gradient backpropagation through Eq. (20).
13: **end for**
14: ————————————————————————————
Inference:
15: **for** each $u_i \in \mathcal{U}$ **do**
16: 　　$\mathcal{T}_j : (|\mathcal{V}_{u_i}|, K, D) = \{\{\mathbf{z}_k^{v_j}\}^K | v_j \in \mathcal{V}_{u_i}\} \leftarrow \text{Tokenizer}(\mathbf{q}_j)$;
17: 　　$\mathcal{T}_i : (K, D) = \{\mathbf{z}_k^u\}^K \leftarrow \text{Tokenizer}(\mathbf{p}_i)$;
18: 　　Prompting $\mathcal{X}_i : (L_{in}, D) \leftarrow \mathcal{P}(\mathcal{T}_i, \{\mathcal{T}_j | v_j \in \mathcal{V}_{u_i}\})$;
19: 　　Reasoning $\mathcal{Y}_i : (L, D) = \text{LLM}(\mathcal{X}_i)$;
20: 　　Conditioning $\mathbf{c}_i : (D) = f(\mathbf{y}_i^1, \ldots, \mathbf{y}_i^{L-1})$;
21: 　　Denoising $\mathbf{y}_i : (d) = \text{Diffusion}(\mathbf{c}_i, \epsilon)$;
22: 　　Hybrid scoring $s_{ij} \leftarrow \text{Eq.}(22)$;
23: **end for**
24: **return** Scores for all users towards all items $\{s_{ij}\} \in \mathbb{R}^{|\mathcal{U}| \times |\mathcal{V}|}$.

---

is similar to a regression objective, which minimizes the distance between the prediction $\hat{\mathbf{y}}_i$ and its target $\mathbf{y}_i$. On the other hand, the second term encourages any pair to be as distant as possible.

To construct Dispersive Loss, we conduct the training on the intermediate representations $\mathbf{h}_i$ of the generative model for user u_i as a replacement of $\mathbf{y}_i$, followed by [43]. Building upon this, we keep only the second term in Eq (23):

$$\mathcal{L}_{\text{disp}} = \log \sum_m \exp(-\mathcal{D}(\mathbf{h}_i, \mathbf{h}_m)/\tau). \tag{24}$$

Moreover, to have a form defined on a batch of samples, the formula can be redefined as:

$$\mathcal{L}_{\text{disp}} = \log \mathbb{E}_{i,b}[\exp(-\mathcal{D}(\mathbf{h}_i, \mathbf{h}_b)/\tau)]. \tag{25}$$

This loss function yields the same value for all samples within a batch and is computed only once per batch. By removing the alignment term on the intermediate representations, it further emphasizes the regularization aspect. Notably, the loss does not need to explicitly exclude the pair $\mathcal{D}(\mathbf{h}_i, \mathbf{h}_b)$ when $i = b$, as it maintains a unified view of all samples in each batch.

B.3　Prompt Templates

Inspired by the success of in-context learning in large language models, we employ two widely used prompting techniques, namely

role play and few-shot learning. Let "$\langle\cdot\rangle$" represents a continuous encoded token for users or items. A detailed example of our prompt template is outlined in the colorbox. Please refer to our code for the detailed prompts.

$\Longrightarrow$ **In-context Learning**:
You are the expert in {dataset} products, entrusted with the responsibility of recommending the perfect products to our users.

Here is an example format for recommendations:

Input: Given the following purchase history: [item side information]. I wonder what the user will like. Can you help me decide?
Response: The interaction history shows that the user might like [item_brand/item_category] products.'

Now, please provide your recommendations based on the following content:

- -

$\Longrightarrow$ **Sample Content**:
Input: Considering the user
$\langle\text{z\_start}\rangle\ \langle u_i^1\text{-emb}\rangle\ \langle u_i^2\text{-emb}\rangle\ \langle u_i^3\text{-emb}\rangle\ \langle\text{z\_end}\rangle$
has interacted with:
itemtitle $\langle\text{z\_start}\rangle\ \langle v_1^1\text{-emb}\rangle\ \langle v_1^2\text{-emb}\rangle\ \langle v_1^3\text{-emb}\rangle\ \langle\text{z\_end}\rangle$,
itemtitle $\langle\text{z\_start}\rangle\ \langle v_2^1\text{-emb}\rangle\ \langle v_2^2\text{-emb}\rangle\ \langle v_2^3\text{-emb}\rangle\ \langle\text{z\_end}\rangle$,
itemtitle $\langle\text{z\_start}\rangle\ \langle v_3^1\text{-emb}\rangle\ \langle v_3^2\text{-emb}\rangle\ \langle v_3^3\text{-emb}\rangle\ \langle\text{z\_end}\rangle$,
what are the expected preferences of the user?
Response:

C　Further Analysis of Experimental Results

C.1　Analysis of Success and Failure Cases.

Table 4 shows two success cases (users 2715 and 36183) and a failure case (user 7849) of ContRec. Notably, the failure case for user 7849 demonstrates a scenario in which ContRec makes an incorrect prediction due to the user's shift in type, platform, and category preferences. This highlights a potential limitation of our model: the LLM-based reasoning tends to favor items with categories similar to those in the user's historical interactions, making it challenging to accommodate sudden shifts in user preferences.

C.2　Discussion on Inference Costs

As shown in Table 5, the majority of our model's inference latency is attributed to the LLM's reasoning phase, which accounts for approximately 88.60% of the total inference time across all four datasets. Nevertheless, in line with the "deep thinking" paradigm adopted by recent advanced large language models such as DeepSeek-R1 and GPT-5, we suggest that this "latency" is essential, as it provides transparency into how the model generates recommendations and thereby fosters user trust. Given these considerations, we propose that the most suitable application scenarios for our model are in

Table 4: Examples of two success cases (users 2715 and 36183) and a failure case (user 7849). The target item is underlined.

User ID	Item Sequence ↓	Title	Type	Platform	Category
2715 (S)	523	Namco Museum 64	Retro Gaming & Microconsoles	Nintendo 64	Games
	552	Jungle Gree	Retro Gaming & Microconsoles	Nintendo 64	Games
	6386	Video Cable Cord	Retro Gaming & Microconsoles	Trenro	Accessories
	12289	OSTENT Wired Controller Gamepad Joystick Joypad	Retro Gaming & Microconsoles	Nintendo 64	Accessories
	_9142_	Booster Pack (Jumper Pak)	Retro Gaming & Microconsoles	Nintendo 64	Accessories
36183 (S)	13596	Battlefield 1	Electronic Arts	PlayStation 4	Games
	1257	Dualshock 2 Analog Wired Controller SCPH-10010	Retro Gaming & Microconsoles	PlayStation 2	Accessories
	9470	SimCity [Instant Access]	Electronic Arts	PC	Games
	8491	SimCity: Limited Editio	Electronic Arts	PC	Games
	_789_	Fallout 1 / Fallout 2 Bundle (Jewel Case) - PC	Interplay	PC	Games
7849 (F)	3488	Wall Charger for Nintendo DS Lite	Chargers	Nintendo DS	Accessories
	1887	Rollercoaster Tycoon 2: Triple Thrill Pack - PC	/	PC	Games
	2869	Nintendo DS Lite Polar White	/	Nintendo DS	Consoles
	6952	Wii Remote Plus - Black	/	Nintendo	Accessories
	_7550_	Kinect Disneyland Adventures - Xbox 360	Kids & Family	Xbox 360	Games

Table 7: Statistics of the trainable parameters.

	Name	Type	Params
1	model	PeftModelForCausalLM	760 M
2	diffusion_net	Transformer	5.8 M
3	item_tokenizer	MLP	2.6 M
4	user_tokenizer	MLP	2.6 M
5	condition_net	MLP	1.1 M

Table 5: Average inference time (seconds) per batch.

Dataset	LlaRA	TIGER	TokenRec	ContRec	LLM	Diffusion
LastFM	0.4255	0.3502	0.0240	1.6110	1.5244	0.0866
ML1M	0.7370	0.5990	0.0367	2.4264	2.0994	0.3271
Games	0.5281	0.4316	0.0273	1.7338	1.4920	0.2418
Beauty	0.5423	0.4388	0.0292	1.7643	1.5385	0.2258

Table 6: Generalizability of the representative generative recommendation models to unseen users (5%).

Model	Beauty		Games	
	HR@10	NDCG@10	HR@10	NDCG@10
P5	0.0255	0.0156	0.0643	0.0368
CoLLM	0.0263	0.0171	0.0663	0.0373
TokenRec	0.0374	0.0197	0.0747	0.0414
TIGER	0.0384	0.0195	0.0759	0.0429
LlaRA	0.0380	0.0198	0.0770	0.0436
ContRec	0.0391	0.0205	0.0787	0.0450

personalized conversational recommendation systems with user interfaces, rather than as a back-end algorithm for large-scale online recommendation tasks involving billions of web requests.

C.3 Analysis of Generalizability

To assess the Generalizability of our approach and its LLM-based counterparts, we utilize the Beauty and Games datasets. In these datasets, we exclude the 5% of users with the least interaction history from the training split, treating them as newly added users in the evaluation set. The results demonstrate that advanced LLM-based methods (i.e., TokenRec, TIGER, LLaRA, and ContRec) maintain strong performance in this scenario. This can be attributed to their ability to leverage textual information (e.g., title and category) as supplementary cues for inferring user preferences when interaction history is limited. Furthermore, our hybrid retrieval process encourages ContRec to predict items with relevant categories and brands, making it particularly effective in cases involving previously unseen users.

C.4 Implementation Details

The proposed model is developed based on Hugging Face and PyTorch, leveraging multiple NVIDIA A800 GPUs (80GB). We employ the *meta-Llama/Llama-3.2.-1B-Instruct* model as the foundation of our LLM, which is subsequently quantized into a 4-bit format to facilitate efficient fine-tuning. In ContRec, the trainable parameters are divided into three parts, with their respective sizes detailed in Table 7. Additionally, the total model parameter size is estimated to be 3,058.967 MB. To maximize the utilization of our computing hardware, the NVIDIA A800 GPU (80GB), we have configured a batch size of 24 and a maximum of 200 epochs. The training process on the four datasets (ML1M, Games, Beauty, and LastFM) demands considerable time, exceeding 72 hours, 60 hours, 50 hours, and 24 hours on a single A800 GPU, respectively. For all baselines, we conduct negative sampling from the uniform distribution at the ratio of 1:1, which is not conducted in DreamRec and ContRec.