DISCRETE DIFFUSION FOR BUNDLE CONSTRUCTION

Anonymous authors
Paper under double-blind review

ABSTRACT

As a central task in product bundling, bundle construction aims to select a subset of items from huge item catalogs to complete a partial bundle. Existing methods often rely on the sequential construction paradigm that predicts items one at a time, nevertheless, this paradigm is fundamentally unsuitable for the essentially unordered bundles. In contrast, the non-sequential construction paradigm models bundle as a set, while it still faces two dimensionality curses: the combination complexity is exponential to the catalog size and bundle length. Accordingly, we identify two technical challenges: 1) how to effectively and efficiently model the higher-order intra-bundle relations with the growth of bundle length; and 2) how to learn item embeddings that are sufficiently discriminative while maintaining a relatively smaller search space other than the huge item set.

To address these challenges, we propose DDBC, a Discrete Diffusion model for Bundle Construction. DDBC leverages a masked denoising diffusion process to build bundles non-sequentially, capturing joint dependencies among items without relying on certain pre-defined order. To mitigate the curse of large catalog size, we integrate residual vector quantization (RVQ), which compresses item embeddings into discrete codes drawn from a globally shared codebook, enabling more efficient search while retaining semantic granularity. We evaluate our method on real-world bundle construction datasets of music playlist continuation and fashion outfit completion, and the experimental results show that DDBC can achieve more than 100% relative performance improvements compared with state-of-the-art baseline methods. Ablation and model analyses further confirm the effectiveness of both the diffusion backbone and RVQ tokenizer, where the performance gain is more significant for larger catalog size and longer bundle length. Our code is available at `https://anonymous.4open.science/r/DDBC-44EE`.

1 INTRODUCTION

Product bundling has been a pervasive business strategy, which originates from conventional retailing, evolves to e-commerce, and is further adopted by generic online services, such as music and video streaming (Chang et al., 2020). A product bundle is a set of relevant items assembled to satisfy users' needs (*e.g.,* games, outfits, playlists, meal kits) (Sun et al., 2024) and promote sales regarding sellers' pursuit. Bundle construction, *i.e.,* select a subset of items from the large item pools to build an entire bundle or complete a partial bundle, is the first and foremost problem among various bundle-centric studies, such as personalized bundle recommendation (Ma et al., 2022).

Existing studies, either specifically designed for bundle construction (Han et al., 2017; Bai et al., 2019; Gong et al., 2019; Deng et al., 2021) or general sequential recommendation (Kang & McAuley, 2018; Sun et al., 2019), have a fatal yet ever-overlooked flaw: most of them are based on a sequential construction paradigm, *i.e.,* predict the next item only rather than all the items in the entire bundle, however, such a sequential construction paradigm is essentially not suitable for bundle construction. Intuitively, a bundle is not a sequence of user's interacted items, and a user is not necessarily to follow a certain sequential order to consume the items within a bundle [1]. Thereby, sequential dependencies barely exist between consecutive items in a bundle, and sequential models bring marginal benefits to bundle construction. Delving deep into the technical foundations, consider N as the total number of items (item catalog size) and k as the bundle length (number of items

[1]Some bundles may have a sequential order by design, while here we focus on the general scenarios.

Highlighting that retrievaling is not suitable for bundle construction.
Generative paradigm is more practical.

within a bundle), the theoretical space of modeling a bundle as a sequence is the permutation, *i.e.,* $P(N,k)$, while modeling it as a set by relaxing the sequential constraint will significantly downgrade the space to the combination, *i.e.,* $C(N,k)$. Nonetheless, simply discarding the sequential construction paradigm, *e.g.,* existing non-sequential construction methods (Tomasi et al., 2024; Yang et al., 2024), only partially addresses the problem, since the space of the combination is still exponential to item catalog size N and bundle length k, which we call the two dimensionality curses.

These two dimensionality curses induce two technical challenges: First, bundle construction requires to model the intra-bundle relations, such as similarity, compatibility, composability, *etc.*, among any possible combinations of items, *e.g.,* pair-wise, tripartite, and quaternary (Chang et al., 2020). Thus, how to effectively and efficiently preserve these higher-order relations, considering that the complexity increases exponentially with the linear growth of k, remains the first key challenge. Second and more seriously, the item catalogs, from which we draw items to build the bundle, are often huge. For example, N could be tens of thousands or even millions on some online platforms such as Spotify or Amazon. Conventional approaches typically leverage one embedding for each item (Ma et al., 2024c), consequently, it is highly difficult to navigate through the huge candidate space and precisely pick the desired item for a certain bundle. Therefore, how to learn item embeddings that are sufficiently discriminative regarding different bundling functions while maintaining a relatively small search space poses the second technical challenge.

To tackle the above two challenges, we propose a method that leverages ***D**iscrete **D**iffusion for **B**undle **C**onstruction*, named as **DDBC**. Specifically, to model the higher-order intra-bundle item relations, we introduce diffusion model as the backbone to replace the previous sequential or non-sequential solutions. Basically, the diffusion model features with a non-sequential construction paradigm, where it picks items according to the learned strategies regarding the entire bundle structure instead of following a certain pre-defined left-to-right sequential order. In terms of the second challenge caused by huge item catalog size, we leverage the residual vector quantization tokenizer (RVQ) to quantize the continuous item embedding into multiple discrete codes (Rajput et al., 2023). The codes of each item are selected from a globally shared codebook that is significantly smaller than the original item set, remarkably relaxing the dimensionality curse caused by N. By integrating the RVQ tokenizer into the diffusion backbone, we design our discrete diffusion model DDBC. Concretely, we treat a complete bundle $\bar{\mathbf{b}}$ as the clean state at $t{=}0$; at each forward step we randomly mask a subset of positions, eventually reaching an all-`[MASK]`. The training objective is to learn the reverse denoising dynamics $p_\theta(\mathbf{b}_{t-1} \mid \mathbf{b}_t, t)$. During inference, given a partial bundle with unknown slots marked `[MASK]`, we iteratively denoise until the bundle is fully recovered. Importantly, random masking exposes the model to rich contexts during training, thereby approximating the joint distribution modeling over bundle items and providing the flexibility to accommodate different decoding priors. Also, the item codes learned by RVQ have different levels of semantic granularity, therefore, the diffusion model can learn the bundling strategy more fine-grainedly.

Our contributions include: (1) We emphasize bundle construction should follow a non-sequential construction paradigm and instantiate it with a masked denoising process. (2) We operate the diffusion model in a vector-quantized discrete space using RVQ, which relaxing the dimensionality curse caused by huge item catalog size. (3) We provide extensive and detailed empirical evidence that our approach outperforms baselines in bundle construction, with benefits especially pronounced on larger bundles and larger item catalogs. Comprehensive ablation and model studies further verify the contribution of each component.

2 Related Work

We review three lines of literature most relevant to our work: bundle construction, generative recommendation, and discrete diffusion models.

Bundle construction is the task of selecting a subset of items from the large item pools to build an entire bundle or complete a partial bundle. It typically comprises two parts: (1) an encoder for users, items, and bundles, and (2) a bundle generator. On the encoder side, early advances fuse semantics feature (often via multimodal encoders *e.g.,* Elizalde et al. (2023)/Li et al. (2023)) with collaborative signals (Sarwar et al., 2001; He et al., 2020) to learn stronger item embeddings (Ma et al., 2022; 2024a;c;b; Salganik et al., 2024). However, these encoders do not directly capture bundle-level structure; semantically similar items may still not co-occur, whereas real user-constructed bundles balance relevance, exhibit diversity, and maintain complementarity (Sun et al., 2024). Most bundle generators still follow a sequential construction paradigm (Chen et al., 2019; Bai et al., 2019; Chang et al., 2021; Deng et al., 2021; Liu et al., 2025; Han et al., 2017), however, the item order within a bundle does not necessarily reflect how users construct or consume bundles. Relying on a fixed order can introduce unnecessary order bias and harm generalization by overfitting to dataset-specific sequences (Yang et al., 2024). Several non-sequential approaches have been proposed. Wei et al. (2022) predicts all bundle items in parallel with a contrastive non-auto-regressive decoder, but it relies on predefined templates and fixed object types. Tomasi et al. (2024) uses a continuous-space diffusion model but only accepts text prompts as input. Yang et al. (2024) outputs an order-agnostic set in one shot; however, it lacks explicit intra-bundle relations interactions during generation. More importantly, all of these works generate bundles from scratch and do not address partial-bundle completion. Our approach targets this gap via discrete masked denoising, completing the missing items in an order-agnostic manner.

Generative recommendation is a paradigm that reframes recommendation as generating target item IDs rather than full-rank retrieving. (Wu et al., 2024; Li et al., 2024a) The line originates from generative retrieval (Rajput et al., 2023): using residual vector quantization (**?**Lee et al., 2022), items are quantized into multiple semantic IDs from coarse to fine, and an auto-regressive decoder generates these IDs conditioned on context. Subsequent work extends this paradigm to multimodal settings (Liu et al., 2024) and LLM backbones (Zheng et al., 2024; Zhai et al., 2025). As previously discussed, sequential construction paradigm is not suitable for bundling tasks; accordingly, we retain the multiple semantic-ID idea but replace AR backbone with a discrete diffusion model.

Diffusion models learn to generate data by inverting a forward noise process; in continuous spaces they have been widely used for images, audio, and trajectories (Ho et al., 2020; Janner et al., 2022; Kong et al., 2021; Liu et al., 2023; Yang et al., 2022). For discrete data, diffusion extends to categorical tokens by corrupting symbols (often to an absorbing [MASK]) and denoising to reconstruct them (Austin et al., 2021; Sahoo et al., 2024). Within recommendation, diffusion has largely been applied to sequential next-item prediction, operating on item latent embedding space (Wang et al., 2023; Yang et al., 2023; Li et al., 2024b), while discrete diffusion remains comparatively underexplored (Lin et al., 2024; Ju et al., 2025). In this work, we adopt an MDLM-style discrete diffusion backbone and leverage its order-agnostic nature to better model bundles. To the best of our knowledge, we are the first to study bundle construction with discrete diffusion.

3 Method

Our framework consists of two key components: (1) RVQ to discretize item embeddings, and (2) a DDM that operates over the code tokens for full bundle construction. The overall architecture is illustrated in Figure 1.

3.1 Problem Formulation

Let $\mathcal{I} = \{i_1, i_2, \ldots, i_N\}$ denote the item catalog and $\mathcal{B} = \{\mathbf{b}_1, \mathbf{b}_2, \ldots, \mathbf{b}_M\}$ the collection of bundles, where N and M denote the number of items and bundles, respectively. Each bundle $\mathbf{b} \in \mathcal{B}$ is set of items, $\mathbf{b} = \{i_{j_1}, i_{j_2}, \ldots, i_{j_{|\mathbf{b}|}}\},\ \{j_1, \ldots, j_{|\mathbf{b}|}\} \subseteq [N],\ |\mathbf{b}| \geq 2$. Each item $i \in \mathcal{I}$ is mapped by a feature extractor to a latent vector $E(i)$, where the feature often encapsulates semantic signals and collaborative-filtering signals depending on the datasets. We formulate the bundle construction task as: for a bundle $\bar{\mathbf{b}}$, given a non-empty partial bundle (a subset of the entire bundle) $\mathbf{b}_x \subseteq \bar{\mathbf{b}}$, predict the rest part of the bundle (the complementary item set) $\mathbf{b}_y = \bar{\mathbf{b}} \setminus \mathbf{b}_x$.

3.2 Residual Quantization of Item Embeddings

To make masked discrete diffusion feasible on large catalogs, we first discretize continuous item embeddings into a compact, hierarchical code space via RVQ. We apply an L-level RVQ to obtain a tuple of discrete code indices $\mathbf{z}(i) = (z^{(1)}(i), \ldots, z^{(L)}(i))$. The last level is a dedup code that carries no semantics and acts purely as an auto-increment field to ensure a one-to-one mapping from a code tuple back to a unique item ID. Formally, $z^{(\ell)}(i) \in \{1, \ldots, C_\ell\}$ indexes a codeword in

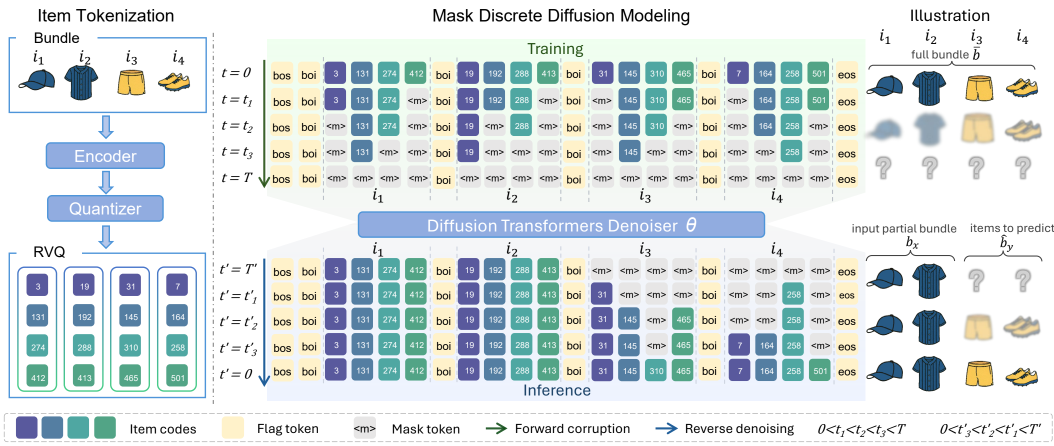

Figure 1: The overall framework of DDBC. The left side illustrates the item tokenization process via RVQ. The right side visualizes the training and inference stages of the masked discrete diffusion modeling. To be noted, we only show the forward process of the training stage, of which the backward process is just a reverse of the forward and omitted for simplicity.

codebook $\mathcal{C}^{(\ell)} = \{\mathbf{e}_1^{(\ell)}, \ldots, \mathbf{e}_{C_\ell}^{(\ell)}\} \subset \mathbb{R}^d$. Let the residual be $\mathbf{r}^{(0)} = E(i)$ and, for $\ell = 1, \ldots, L-1$,

$$z^{(\ell)}(i) = \arg\min_{c \in \{1,\ldots,C_\ell\}} \left\|\mathbf{r}^{(\ell-1)} - \mathbf{e}_c^{(\ell)}\right\|_2^2, \qquad \mathbf{r}^{(\ell)} = \mathbf{r}^{(\ell-1)} - \mathbf{e}^{(\ell)}_{z^{(\ell)}(i)}. \tag{1}$$

The reconstruction uses only the semantic codebooks: $\hat{E}(i) = \sum_{\ell=1}^{L-1} \mathbf{e}^{(\ell)}_{z^{(\ell)}(i)}$. Early codebooks capture coarse semantics; later residual codebooks refine details, inducing semantic smoothness among similar items. We train the codebooks with an RVQ loss that combines a reconstruction term and a codebook commitment term:

$$\mathcal{L}_{\text{RVQ}} = \left\|E(i) - \hat{E}(i)\right\|_2^2 + \beta \sum_{\ell=1}^{L-1} \left(\left\| \text{sg}[\mathbf{r}^{(\ell-1)}] - \mathbf{e}^{(\ell)}_{z^{(\ell)}(i)} \right\|_2^2 + \left\| \mathbf{r}^{(\ell-1)} - \text{sg}[\mathbf{e}^{(\ell)}_{z^{(\ell)}(i)}] \right\|_2^2 \right), \tag{2}$$

where $\text{sg}[\cdot]$ denotes stop-gradient and β balances the commitment loss. We use a straight-through estimator for the discrete assignment; codebooks are updated via gradient descent. For a bundle $\mathbf{b} = \{i_1, \ldots, i_{|b|}\}$, RVQ yields a token matrix $\mathbf{Z}^{(0)} \in \mathbb{N}^{|b| \times L}$ with entries $z_{j\ell} \in \{1, \ldots, C_\ell\}$; the j-th row contains the L codes representing item i_j.

Among many quantization strategies, we choose RVQ for three reasons. (1) Vocabulary compression. Its theoretical capacity is $\prod_{\ell=1}^{L} C_\ell$, enabling a small per-level vocabulary to index a very large item universe. (2) Denser supervision. Each item contributes L code tokens; at level ℓ, a code typically aggregates roughly N/C_ℓ items (for N total items), so code-level supervision is markedly denser than item-ID supervision. We quantify the increase in effective supervision in Section 4. (3) Coarse-to-fine granularity of semantics. Hierarchical residual codebooks perform implicit clustering at multiple granularities, which benefits downstream denoising. Additionally, when a candidate set is available in real applications, the finest semantic level and the disambiguation index may be unnecessary at inference time: one can decode with fewer RVQ levels and map prefixes to candidate items via an inverted index over code prefixes. We study this design choice in Section 4.

3.3 Masked Discrete Diffusion over Code Tokens

We cast bundle construction as a masked discrete denoising process. In contrast to sequential construction paradigm, this formulation avoids imposing any arbitrary order on the items and instead allows the model to leverage the full set context in an order-agnostic manner. Importantly, our discrete diffusion does not directly impose a set-invariant objective: items are still flattened into a list and we do not explicitly enforce permutation invariance. Nevertheless, it serves as a good approximation to set modeling: at each training step, the already-revealed context and the positions to be revealed are randomized, providing rich, position-agnostic supervision so that the learned features do not rely on absolute positions. To realize this idea, we adopt a discrete diffusion framework with an absorbing-mask corruption mechanism. The key components are: an input tokenization design for bundles, a forward corruption process that incrementally masks tokens, a bidirectional Transformer as the reverse denoiser, and an order-agnostic inference procedure with token-level validity constraints.

Input tokenization for bundles. We serialize a bundle by inserting `<boi>` before each item and wrapping with `<bos>` and `<eos>`. Let $z_{j,\ell}$ denote the ℓ-th code token of item j (j=$1,\ldots,|\mathbf{b}|$, ℓ=$1,\ldots,L$). We use exactly two index sets:

$$\Omega_{\text{flag}} = \{\texttt{<bos>}, \texttt{<boi>}, \texttt{<eos>}\}, \qquad \Omega_{\text{code}} = \{(j,\ell) : j = 1,\ldots,|\mathbf{b}|,\ \ell = 1,\ldots,L\}. \quad (3)$$

Tokens in Ω_{flag} are never masked; corruption and prediction operate only on $z_{j,\ell}$ with $(j,\ell) \in \Omega_{\text{code}}$.

Forward corruption. We use an absorbing-mask Markov chain as in masked discrete diffusion. At each step $t \in \{1,\ldots,T\}$, each currently unmasked token $z_{j,\ell}$ with $(j,\ell) \in \Omega_{\text{code}}$ is independently replaced by `[MASK]` with probability $\beta_t \in (0,1)$:

$$q\Big(z_{j,\ell}^{(t)} = u \,\Big|\, z_{j,\ell}^{(t-1)} = u\Big) = 1 - \beta_t, \quad q\Big(z_{j,\ell}^{(t)} = \texttt{[MASK]} \,\Big|\, z_{j,\ell}^{(t-1)} \neq \texttt{[MASK]}\Big) = \beta_t, \quad (4)$$

for any token value $u \neq \texttt{[MASK]}$, with the absorbing condition $q(z_{j,\ell}^{(t)} = \texttt{[MASK]} \mid z_{j,\ell}^{(t-1)} = \texttt{[MASK]}) = 1$ and independence across (j,ℓ). Let $\alpha_t = \prod_{s=1}^{t}(1-\beta_s)$ be the survival probability. The closed-form transition from t=0 to t is:

$$q\Big(z_{j,\ell}^{(t)} = v \mid z_{j,\ell}^{(0)} = u\Big) = \alpha_t\, \mathbf{1}[v = u] \;+\; (1-\alpha_t)\, \mathbf{1}[v = \texttt{[MASK]}], \quad (5)$$

i.e., after t steps a token either survives with probability α_t or is masked with probability $1 - \alpha_t$.

Reverse denoising. A bidirectional Transformer θ is trained to predict the original token values from a corrupted sequence. At inference, it produces a categorical distribution for each masked position conditioned on the current noisy tokens, the timestep t:

$$p_\theta\big(z_{j,\ell}^{(0)} \mid \mathbf{C}^{(t)}, t\big) \in \Delta^{C_\ell - 1}, \quad \text{where} \;\; \Delta^{C_\ell-1} \triangleq \{\, p \in [0,1]^{C_\ell} \;:\; \mathbf{1}^\top p = 1 \,\}. \quad (6)$$

Following common practice in Sahoo et al. (2024), we use the "simple" reconstruction objective that trains p_θ to predict the original token $z_{j,\ell}^{(0)}$ directly from a state $\mathbf{Z}^{(t)}$ corrupted at a random timestep t. Crucially, tokens that are unmasked, either because they belong to b_x or because they have already been generated in a previous step, are treated as clamped observations; they are never masked again and remain fixed in all subsequent steps. This mechanism enables the model to unmask items in any order during generation, without ever overwriting a code once it's decided. [2]

Training objective. Let $\mathcal{M}_t \subseteq \Omega_{\text{code}}$ be the set of positions masked by the forward process at step t. The discrete diffusion variational objective reduces to a weighted masked-token cross-entropy:

$$\mathcal{L}_{\text{NELBO}} = \mathbb{E}_{t\sim\mathcal{U}\{1,\ldots,T\}}\, \mathbb{E}_{\mathcal{M}_t} \sum_{(j,\ell)\in\mathcal{M}_t} -\log\, p_\theta\big(z_{j,\ell}^{(0)} \mid \mathbf{Z}^{(t)}, t\big). \quad (7)$$

Inference. We model bundle construction as iterative denoising of a partially masked token matrix. Let the observed set be $\mathbf{b}_x$ and the unknown complementary set be $\mathbf{b}_y$ with $|\mathbf{b}_y|$ items. Denote by Ω_x the positions (including all RVQ levels) that belong to items in $\mathbf{b}_x$, by Ω_y the positions that belong to items in $\mathbf{b}_y$, and by Ω_{flag}. We construct an input sequence by flattening tokens row-wise and inserting `<boi>` before each item's L tokens to mark boundaries. Formally, the initial state $\mathbf{Z}$ is:

$$z_u = \mathbf{1}[u \in \Omega_x \cup \Omega_{\text{flag}}]\, x_u \;+\; \mathbf{1}[u \in \Omega_y]\; \texttt{[MASK]}. \quad (8)$$

Here, x_u represents the given token at position u, with Ω_{flag} kept unmasked so the model knows item segmentation. The tokens of $\mathbf{b}_x$ remain clamped throughout. After decoding, a predicted item $\hat{i}_j$ is obtained by mapping its token tuple back to the catalog or to a reconstruction $\hat{E}_j = \sum_{\ell=1}^{L-1}$.

[2] While diffusion can be extended to allow re-masking to revise earlier decisions, we do not enable that option here and leave it to future work.

Token-validation constraints. At generation time, we constrain the model's predictions to ensure that each set of L code tokens corresponds to a valid item from the catalog. Formally, for each position (j, ℓ) (code level ℓ of item j), we restrict the predicted token to the prefix-consistent subset $\mathcal{V}^{(\ell)}_{\text{valid}}(j;\, \mathbf{z}_{j,<\ell}) \subseteq \{1, \ldots, C_\ell\}$ and set the logits of any token not in $\mathcal{V}^{(\ell)}_{\text{valid}}(j;\, \mathbf{z}_{j,<\ell})$ to $-\infty$ before the softmax, thereby preventing the model from selecting an invalid code combination. This validation step ensures that the generated code tuples always decode to legitimate items, which is crucial for maintaining recommendation feasibility.

4 EXPERIMENT

4.1 EXPERIMENTAL SETTINGS.

Model settings. We use CLHE (Ma et al., 2024c) as the item encoder, i.e., $E(i) = \text{CLHE}(i)$. Unless otherwise noted, our RVQ uses $L = 4$ levels with fixed per-level codebook size $C_\ell \equiv C$; the diffusion horizon is $T = \ell\,(L + 1)$, where the per-item token length (including the boundary marker) is denoted by ℓ. We utilize a lightweight DDiT architecture for our Diffusion backbone, with 6 transformer blocks, each with a hidden size of 64 and 8 self-attention heads. The model operates with a linear noise scheduler $\alpha(t) = 1-t$. All experiments are performed on four NVIDIA A40 GPUs, and all models are trained in 20,000 steps.

Datasets. Following prior research on bundle construction Ma et al. (2024c); Liu et al. (2025), we evaluate on two representative datasets, Spotify (Chen et al., 2018) and POG (Chen et al., 2019). Unlike these works, our discrete diffusion model currently requires a fixed number of tokens per instance, so we truncate bundles to a target length. For the Spotify playlist dataset, we create three subsets by capping playlist length at 30/60/90 items (Spotify$_{k=30,60,90}$). For the POG fashion dataset, whose average bundle length is small, we start from its denser variant and derive a fixed-length version with four items (denoted POG$_{k=4}$). Unless noted, the input-predict ratio of the bundle, $|\mathbf{b}_x| : |\mathbf{b}_y|$, are set as $1 : 1$, see Table 2 for other settings. Samples shorter than the target length are dropped. Each dataset is split into train/validation/test with non-overlapping bundles. We also perform data augmentation by swapping items within the bundle, and the details are describe in Appendix B.

Candidate size. To standardize candidate pool, we set a candidate ratio ρ and construct a shortlist $\mathcal{C}$ of size $\rho\,|\mathbf{b}_y|$ by augmenting the ground-truth targets with randomly sampled non-targets: $\mathcal{C} = \mathbf{b}_y \cup \text{Random}_{(\rho-1)\,|\mathbf{b}_y|}\big(I \setminus \mathbf{b}\big)$. Unless otherwise stated, we fix $\rho = 100$ in all experiments.

4.2 BASELINES.

We consider both non-sequential and sequential construction methods as baselines. To be fair, all the baselines use the same item features, *i.e.,* pre-trained embeddings via CLHE (Ma et al., 2024c).

Non-sequential construction methods. They input the partial bundle $\mathbf{b}_x$ and predict all the items in the complementary set at once. *CLHE* (Ma et al., 2024c): A method that leverages contrastive learning and hierarchical encoder to learn item and bundle representations. To be noted, CLHE was not originally designed to predict all the items in the complementary set, while it follows the typical top-k recommendation paradigm and evaluation protocol. We re-evaluate it against our metrics that are pertinent to entire bundle construction. *BundleNAT* (Yang et al., 2024): A non-auto-regressive generator that predicts a set of items in one shot using preference/compatibility signals. It was originally used for the task of personalized bundle recommendation instead of bundle construction, we adapt it for our task by removing the user inputs.

Sequential construction methods. They follow an auto-regressive construction strategy: initialize $\mathbf{s}_0 = \mathbf{b}_x$; for $j = 0, \ldots, |\mathbf{b}_y| - 1$, choose $\hat{i}_j = \arg\max_{i \notin \mathbf{s}_j} \pi(i \mid \mathbf{s}_j)$ and update $\mathbf{s}_{j+1} = \mathbf{s}_j \cup \{\hat{i}_j\}$ until $|\mathbf{s}_{|\mathbf{b}_y|}| = |\bar{\mathbf{b}}|$. *Bi-LSTM* (Han et al., 2017): It uses bi-directional LSTM to model the bundle as a sequence. *SASRec* (Kang & McAuley, 2018): A Transformer-based sequential recommender trained for next-item prediction. *TIGER* (Rajput et al., 2023): It generates items as discrete semantic token sequences with an auto-regressive decoder. *BundleMLLM* (Liu et al., 2025): It finetunes a multimodal LLM for bundle construction. Its original evaluation is based on the multiple-choice question protocol since it is impossible to input all the candidate items as input due to context

Table 1: Overall performance comparison between our DDBC and baselines. "%Improv." denotes the relative improvement over the strongest baseline. Best in **bold**, second best <u>underlined</u>.

Model (A/beam)	Spotify$_{k=30}$ F1 ↑	Spotify$_{k=30}$ Jacc ↑	Spotify$_{k=30}$ OAS ↑	Spotify$_{k=60}$ F1 ↑	Spotify$_{k=60}$ Jacc ↑	Spotify$_{k=60}$ OAS ↑	Spotify$_{k=90}$ F1 ↑	Spotify$_{k=90}$ Jacc ↑	Spotify$_{k=90}$ OAS ↑	POG$_{k=4}$ F1 ↑	POG$_{k=4}$ Jacc ↑	POG$_{k=4}$ OAS ↑
CLHE	0.071	0.039	0.373	0.100	0.054	<u>0.446</u>	0.119	0.065	<u>0.486</u>	0.140	0.096	0.446
Bi-LSTM	0.124	0.071	<u>0.489</u>	0.062	0.034	0.430	0.047	0.025	0.426	0.035	0.024	0.390
SASRec	0.070	0.043	0.318	0.089	0.054	0.310	0.050	0.029	0.285	<u>0.169</u>	<u>0.114</u>	0.468
TIGER	0.093	0.053	0.329	<u>0.129</u>	<u>0.076</u>	0.413	<u>0.123</u>	<u>0.070</u>	0.480	**0.213**	**0.157**	**0.546**
BundleNAT	<u>0.153</u>	<u>0.090</u>	0.454	0.101	0.056	0.438	0.095	0.052	0.446	0.145	0.097	0.462
BundleMLLM	0.046	0.024	0.296	0.045	0.024	0.324	0.052	0.027	0.355	0.070	0.047	0.322
DDBC	**0.282**	**0.178**	**0.618**	**0.296**	**0.185**	**0.668**	**0.287**	**0.177**	**0.684**	0.139	0.098	<u>0.526</u>
%Improv. +	84.3%	97.8%	26.4%	129.5%	143.4%	49.8%	133.3%	152.9%	40.7%	–	–	–

limitation of LLMs. Even though this setting is easier than our all-ranking setting, to be simple, we follow this paradigm and set the candidate set as 20.

To be noted, many other recommendation models can be adapted as baselines by following the paradigm of either the sequential or non-sequential construction. For example, the baselines implemented in Ma et al. (2024c): MultiDAE (Wu et al., 2016), MultiVAE (Liang et al., 2018), Hypergraph (Yu et al., 2022), and Transformer (Wei et al., 2023), *etc.* or the other advanced sequential recommendation models. However, we do not include them because they either underperform CLHE or not highly relevant to the bundle scenario. We implement the most relevant and strongest baselines to the best of our knowledge, and more baselines could be implemented upon request.

4.3 Evaluation Metrics

We report retrieval-based metrics F1 and Jaccard (Jacc) (Manning et al., 2008; Ding et al., 2023), as well as a latent-space similarity-based metric OAS (Salton et al., 1975). Higher $\mathrm{F1}$, Jacc, and OAS indicate better performance. Let $\hat{\mathbf{b}}_y$ denote the set of predicted items, these metrics are calculated by:

$$\mathrm{F1} := \frac{2PR}{P+R}, \quad \mathrm{Jacc} := \frac{|\hat{\mathbf{b}}_y \cap \mathbf{b}_y|}{|\hat{\mathbf{b}}_y \cup \mathbf{b}_y|}, \quad \mathrm{OAS} := \frac{1}{|\mathbf{b}_y|} \max_{M} \sum_{(\alpha,\beta)\in M} \cos\big(E(\alpha), E(\beta)\big), \tag{9}$$

where $P = \frac{|\hat{\mathbf{b}}_y \cap \mathbf{b}_y|}{|\hat{\mathbf{b}}_y|}, R = \frac{|\hat{\mathbf{b}}_y \cap \mathbf{b}_y|}{|\mathbf{b}_y|}$, and M is the optimal matching between items in $\hat{\mathbf{b}}_y$ and $\mathbf{b}_y$ and $\cos(\cdot,\cdot)$ denotes cosine similarity. Previous methods in bundle construction use popular next-item recommendation metrics, such as recall, ndcg, or hit rate (Ma et al., 2024c). However, these metrics are not suitable in the scenario of full bundle construction, which needs to assess the quality of the predicted entire item set instead of single item. Therefore, we propose these three metrics to collaboratively measure the performance of bundle construction, offering a comprehensive and consistent benchmark setting for future studies.

4.4 Overall Performance Comparison

Table 1 shows the overall performance of DDBC compared with baseline methods. First, among the baselines, BundleNAT and TIGER achieve the strongest performance. These results respectively highlight two key component of our model: the non-sequential construction paradigm and the advantages of discretizing items into multiple codes. Second, on the Spotify dataset series, DDBC clearly outperforms all baselines, achieving a 153% improvement in Jacc on Spotify$_{k=90}$. Moreover, the performance gain of DDBC becomes more pronounced as the bundle sequence length increases. These results demonstrate that DDBC effectively captures the higher-order intra-bundle item relations, particularly for long-sequence bundles with rich structural dependencies. Third, on POG$_{k=4}$, our model does not outperform TIGER. In fact, since we only predict two items, the task, to some extent reduces to a next-item prediction scenario, where auto-regressive methods such as TIGER have a clear advantage.

Table 2: Effect of input-predict ratio on Spotify$_{k=60}$. Best in **bold**, second best underlined.

Model	5/55			10/50			30/30			45/15		
	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑
BundleNAT	0.106	0.059	0.443	0.128	0.072	0.463	0.101	0.056	0.438	0.084	0.046	0.359
SASRec	0.119	0.070	0.425	0.131	0.078	0.442	0.089	0.054	0.310	0.095	0.055	0.315
TIGER	0.087	0.050	0.365	0.100	0.059	0.381	0.129	0.076	0.413	0.154	0.091	0.426
DDBC	**0.237**	**0.144**	**0.637**	**0.268**	**0.164**	**0.664**	**0.296**	**0.185**	**0.668**	**0.260**	**0.161**	**0.614**
Improv. +	99.2%	105.7%	43.8%	104.6%	110.3%	43.4%	129.5%	143.4%	52.5%	68.8%	76.9%	44.1%

Table 3: Effect of candidate ratio on Spotify$_{k=60}$. Best in **bold**, second best underlined.

Model	ρ=10			ρ=20			ρ=50			ρ=100		
	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑
BundleNAT	0.266	0.163	0.508	0.210	0.124	0.485	0.153	0.088	0.464	0.101	0.056	0.438
SASRec	0.292	0.194	0.519	0.200	0.126	0.474	0.200	0.126	0.475	0.089	0.054	0.310
TIGER	0.191	0.151	0.326	0.107	0.080	0.295	0.108	0.081	0.296	0.129	0.076	0.413
DDBC	**0.599**	**0.447**	**0.763**	**0.503**	**0.355**	**0.727**	**0.380**	**0.250**	**0.689**	**0.296**	**0.185**	**0.668**
Improv. +	105.1%	130.4%	47.0%	139.5%	181.7%	49.9%	90.0%	98.4%	45.1%	129.5%	143.4%	52.5%

4.5 Model Study

Effect of input-predict ratio. We conduct experiments with different input-predict ratio on Spotify$_{k=60}$ and report results in Table 2. We observe that DDBC outperforms all baselines across different partial bundle sizes and exhibits a relatively consistent performance, demonstrating its robustness in scenarios with limited known items. Specifically, when the known partial bundles are small (*e.g.,* 5/55, 10/50, 30/30), DDBC achieves substantial improvements over the best baseline by 106%, 110%, 143% on Jaccard, respectively. Although the performance gap narrows down as the number of input items grow, our method continues to maintain a leading position. These results highlight that DDBC is capable of generating coherent and distribution-aware bundles even when only a small subset of items is provided, validating the effectiveness of our masked denoising formulation and the discrete diffusion mechanism.

Effect of candidate ratio. In addition, we report the results for DDBC and the baselines under different candidate ratios in Table 3. The results indicate that while the absolute values of the evaluation metrics fluctuate as the candidate ratio (ρ) increases, the relative improvements of DDBC over all baselines remain consistently substantial. Interestingly, among baselines, when ρ increases, TIGER start to bypass other baselines on F1 and Jacc (ρ=100). This can be attributed to the fact that RVQ allows precise reconstruction of item IDs, implying the advantages of using RVQ. These findings highlight the adaptability of DDBC under varying candidate pool sizes, demonstrating its ability to maintain strong bundle representations even when the retrieval space becomes more challenging.

Efficiency analysis. We record the inference time and paremeter size of DDBC and the baseline methods, as reported in Figure 2, where the circle radius indicates each model's overall performance. The inference time is measured on Spotify$_{k=60}$. Specifically, although Bi-LSTM has fast inference and smallest parameters, its performance is not competitive (see Table 1). DDBC is highly parameter-efficient, containing only 0.79M parameters, and is significantly smaller than other baselines. Moreover, DDBC's inference speed is comparable to the one-shot generation method BundleNAT and faster than all other baseline models; in particular, it is substantially faster than BundleMLLM, which relies on interactions with large language models.

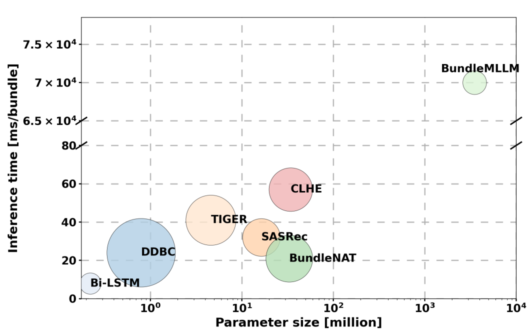

Figure 2: Illustration of the model efficiency comparison. The x-axis is parameter size (millions), y-axis is inference time (milliseconds per bundle), and the bubble radius corresponds to overall performance (larger is better).

4.6 ABLATION STUDY

Key components. To further evaluate the effectiveness of key components of our model, we conduct ablation experiments (Table 4) to assess the contribution of each design choice in DDBC. Considering the resource and time overhead imposed by the extremely large vocabulary in Spotify (254,155), performing ablation studies without RVQ would be prohibitively expensive. Therefore, we adopt Spotify$k = 30$, the shortest sequence setting, as a more practical benchmark for these experiments. We study how each component contributes to performance: hierarchical (coarse-to-fine) decoding, token validity filter, boi token, data augmentation, we also investigate how different RVQ depth affect performance. We derive the following insights. (1) Removing RVQ results in a dramatic performance drop. We encode items using their IDs and initialize their embeddings with CLHE features, results in a dramatic drop in both F1 and Jaccard. This demonstrates RVQ mitigates the dimensionality curse due to N, which is crucial as it permits dense supervision. (2) Discarding the boi token leads to a performance decline. Since Diffusion's generation lacks inherent sequence, integrating the boi token is necessary to guide the model with positional information. (3) Data augmentation shows beneficial for modeling. The results of training on the original dataset show a slight performance reduction in this case, and simple data augmentation remains significant for diffusion modeling because it explicitly provides richer input context. (4) The token validity filter remains essential to guarantee the validity of the generated bundles, despite its removal leading to only a marginal decrease in performance. We evaluated the necessity of the token validity filter during inference. While removing the filter resulted in only a marginal decrease in overall performance, the invalid ratio concurrently rose to 2.5%. Therefore, the filter remains essential to guarantee the validity of the generated bundles.

Table 4: Ablation study of key components.

Variant	F1	Jacc	OAS
Our proposed DDBC	0.166	0.092	0.620
w/o RVQ	0.028	0.015	0.556
w/o boi token	0.116	0.063	0.536
w/o data augmentation	0.152	0.084	0.598
w/o token validity filter	0.163	0.090	–

Effect of RVQ depth. To investigate the impact of the item embedding quantization level on model performance as discussed in method section,with a fixed 4 levels RVQ, we train DDBC with utilizing different levels of RVQ. We report the results in Table 5. As the number of RVQ levels used increases, the model captures increasingly finer-grained item information, leading to substantial improvements in all the evaluation metrics. We state that the current setting represents a favorable trade-off between the representational capacity and compression ratio of RVQ.

Table 5: Effect of RVQ levels.

$\ell \in$	F1	Jacc	OAS
$\{1\}$	0.096	0.051	0.555
$\{1, 2\}$	0.122	0.066	0.591
$\{1, 2, 3\}$	0.148	0.081	0.590
Our proposed DDBC			
$\{1, 2, 3, 4\}$	0.166	0.092	0.620

5 CONCLUSION

We recast bundle construction with a masked discrete diffusion model that progressively resolves unknown items in an order-agnostic manner. Conceptually, the formulation address the dual dimensionality curses: (i) it removes spurious ordering, reducing the search space from permutations to combinations, preserves fine-grained, higher-order item relations, and (ii)shrinks the effective search space by mapping items to codes drawn from a globally shared codebook. Empirically, coupling DDM with RVQ yields consistent gains over prior sequential and non-sequential construction baselines, with especially strong improvements as bundle length grows.

Discussion. Our current instantiation assumes fixed-length bundles, learning when to stop (*i.e.,* variable-length completion and principled halting criteria) remains open. Personalization is mediated by frozen encoders for user–item signals and item semantics; introducing explicit conditioning into the diffusion process (*e.g.,* context features, or user instruction) could yield user-specific bundling. The RVQ design space (*e.g.,* number of levels, codebook sizes, and training regimes) deserves further study to balance identifiability, compression, and semantic smoothness. Finally, diffusion schedules and inference policies merit deeper optimization: adaptive timestep schedules, selective re-masking strategies, and entropy-guided decoding may improve sample efficiency and robustness.

Ethics statement

We affirm compliance with the ICLR Code of Ethics. Our study addresses bundle construction (*e.g.,* playlists and fashion outfits) and uses publicly available research datasets and splits released by prior work; no personally identifiable information (PII) or sensitive attributes are collected, inferred, or released. The inputs consist of item identifiers and non-sensitive metadata, and our models operate on discretized representations without accessing user profiles. Any code and models we release will be for research use only and will not include copyrighted media or proprietary assets.

Reproducibility statement

We make our method reproducible by specifying the full training and evaluation pipeline, including the RVQ configuration, diffusion horizon, architecture, schedulers, and all hyperparameters. We provide an anonymized repository `https://anonymous.4open.science/r/DDBC-44EE`, including the implementation of our model as well as the evaluation scripts for F1, Jaccard, and OAS. Upon publication, we plan to release checkpoints (where licenses permit) to reproduce all main and ablation results.

References

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *NeurIPS*, pp. 17981–17993, 2021.

Jinze Bai, Chang Zhou, Junshuai Song, Xiaoru Qu, Weiting An, Zhao Li, and Jun Gao. Personalized bundle list recommendation. In *WWW*, pp. 60–71, 2019.

Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. Bundle recommendation with graph convolutional networks. In *SIGIR*, pp. 1673–1676. ACM, 2020.

Jianxin Chang, Chen Gao, Xiangnan He, Depeng Jin, and Yong Li. Bundle recommendation and generation with graph neural networks. *TKDE*, 35(3):2326–2340, 2021.

Ching-Wei Chen, Paul Lamere, Markus Schedl, and Hamed Zamani. Recsys challenge 2018: Automatic music playlist continuation. In *RecSys*, 2018.

Wen Chen, Pipei Huang, Jiaming Xu, Xin Guo, Cheng Guo, Fei Sun, Chao Li, Andreas Pfadler, Huan Zhao, and Binqiang Zhao. Pog: personalized outfit generation for fashion recommendation at alibaba ifashion. In *KDD*, pp. 2662–2670, 2019.

Qilin Deng, Kai Wang, Minghao Zhao, Runze Wu, Yu Ding, Zhene Zou, Yue Shang, Jianrong Tao, and Changjie Fan. Build your own bundle-a neural combinatorial optimization method. In *ACM Multimedia*, pp. 2625–2633, 2021.

Yujuan Ding, P. Y. Mok, Yunshan Ma, and Yi Bin. Personalized fashion outfit generation with user coordination preference learning. *Inf. Process. Manag.*, 60(5):103434, 2023.

Benjamin Elizalde, Soham Deshmukh, Mahmoud Al Ismail, and Huaming Wang. Clap learning audio concepts from natural language supervision. In *ICASSP*, pp. 1–5. IEEE, 2023.

Yu Gong, Yu Zhu, Lu Duan, Qingwen Liu, Ziyu Guan, Fei Sun, Wenwu Ou, and Kenny Q. Zhu. Exact-k recommendation via maximal clique optimization. In *KDD*, pp. 617–626. ACM, 2019.

Xintong Han, Zuxuan Wu, Yu-Gang Jiang, and Larry S. Davis. Learning fashion compatibility with bidirectional lstms. In *ACM Multimedia*, pp. 1078–1086. ACM, 2017.

Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *SIGIR*, pp. 639–648, 2020.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *NeurIPS*, 2020.

Michael Janner, Yilun Du, Joshua B. Tenenbaum, and Sergey Levine. Planning with diffusion for flexible behavior synthesis. In *ICML*, volume 162 of *Proceedings of Machine Learning Research*, pp. 9902–9915. PMLR, 2022.

Zheng Ju, Honghui Du, Elias Z. Tragos, Neil Hurley, and Aonghus Lawlor. Diffgr: A discrete diffusion-based model for personalised recommendation by reconstructing user-item bipartite graphs. In *ECIR (3)*, volume 15574 of *Lecture Notes in Computer Science*, pp. 246–254. Springer, 2025.

Wang-Cheng Kang and Julian J. McAuley. Self-attentive sequential recommendation. In *ICDM*, pp. 197–206. IEEE Computer Society, 2018.

Zhifeng Kong, Wei Ping, Jiaji Huang, Kexin Zhao, and Bryan Catanzaro. Diffwave: A versatile diffusion model for audio synthesis. In *ICLR*. OpenReview.net, 2021.

Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *CVPR*, pp. 11523–11532, 2022.

Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. Blip-2: Bootstrapping language-image pre-training with frozen image encoders and large language models. In *ICML*, pp. 19730–19742, 2023.

Lei Li, Yongfeng Zhang, Dugang Liu, and Li Chen. Large language models for generative recommendation: A survey and visionary discussions. In *LREC/COLING*, pp. 10146–10159. ELRA and ICCL, 2024a.

Zihao Li, Aixin Sun, and Chenliang Li. Diffurec: A diffusion model for sequential recommendation. *ACM Trans. Inf. Syst.*, 42(3):66:1–66:28, 2024b.

Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *WWW*, pp. 689–698, 2018.

Xiao Lin, Xiaokai Chen, Chenyang Wang, Hantao Shu, Linfeng Song, Biao Li, and Peng Jiang. Discrete conditional diffusion for reranking in recommendation. In *WWW (Companion Volume)*, pp. 161–169. ACM, 2024.

Han Liu, Yinwei Wei, Xuemeng Song, Weili Guan, Yuan-Fang Li, and Liqiang Nie. Mmgrec: Multimodal generative recommendation with transformer model. *CoRR*, abs/2404.16555, 2024.

Haohe Liu, Zehua Chen, Yi Yuan, Xinhao Mei, Xubo Liu, Danilo Mandic, Wenwu Wang, and Mark D. Plumbley. Audioldm: Text-to-audio generation with latent diffusion models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 21450–21474. PMLR, 2023. URL https://proceedings.mlr.press/v202/liu23f.html.

Xiaohao Liu, Jie Wu, Zhulin Tao, Yunshan Ma, Yinwei Wei, and Tat-Seng Chua. Fine-tuning multimodal large language models for product bundling. In *KDD (1)*, pp. 848–858. ACM, 2025.

Yunshan Ma, Yingzhi He, An Zhang, Xiang Wang, and Tat-Seng Chua. Crosscbr: Cross-view contrastive learning for bundle recommendation. In *KDD*, pp. 1233–1241, 2022.

Yunshan Ma, Yingzhi He, Xiang Wang, Yinwei Wei, Xiaoyu Du, Yuyangzi Fu, and Tat-Seng Chua. Multicbr: Multi-view contrastive learning for bundle recommendation. *ACM Transactions on Information Systems*, 42(4):1–23, 2024a.

Yunshan Ma, Yingzhi He, Wenjun Zhong, Xiang Wang, Roger Zimmermann, and Tat-Seng Chua. CIRP: cross-item relational pre-training for multimodal product bundling. In *ACM Multimedia*, pp. 9641–9649. ACM, 2024b.

Yunshan Ma, Xiaohao Liu, Yinwei Wei, Zhulin Tao, Xiang Wang, and Tat-Seng Chua. Leveraging multimodal features and item-level user feedback for bundle construction. In *WSDM*, pp. 510–519, 2024c.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008.

Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Mahesh Sathiamoorthy. Recommender systems with generative retrieval. In *NeurIPS*, 2023.

Subham S. Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *NeurIPS*, 2024.

Rebecca Salganik, Xiaohao Liu, Yunshan Ma, Jian Kang, and Tat-Seng Chua. LARP: language audio relational pre-training for cold-start playlist continuation. In *KDD*, pp. 2524–2535. ACM, 2024.

Gerard Salton, Anita Wong, and Chung-Shu Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

Badrul Munir Sarwar, George Karypis, Joseph A. Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *WWW*, pp. 285–295. ACM, 2001.

Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *CIKM*, pp. 1441–1450. ACM, 2019.

Meng Sun, Lin Li, Ming Li, Xiaohui Tao, Dong Zhang, Peipei Wang, and Jimmy Xiangji Huang. A survey on bundle recommendation: Methods, applications, and challenges. *CoRR*, abs/2411.00341, 2024.

Federico Tomasi, Francesco Fabbri, Mounia Lalmas, and Zhenwen Dai. Diffusion model for slate recommendation. *CoRR*, abs/2408.06883, 2024.

Wenjie Wang, Yiyan Xu, Fuli Feng, Xinyu Lin, Xiangnan He, and Tat-Seng Chua. Diffusion recommender model. In *SIGIR*, pp. 832–841. ACM, 2023.

Penghui Wei, Shaoguo Liu, Xuanhua Yang, Liang Wang, and Bo Zheng. Towards personalized bundle creative generation with contrastive non-autoregressive decoding. In *SIGIR*, pp. 2634–2638. ACM, 2022.

Yinwei Wei, Xiaohao Liu, Yunshan Ma, Xiang Wang, Liqiang Nie, and Tat-Seng Chua. Strategy-aware bundle recommender system. In *SIGIR*, pp. 1198–1207, 2023.

Likang Wu, Zhi Zheng, Zhaopeng Qiu, Hao Wang, Hongchao Gu, Tingjia Shen, Chuan Qin, Chen Zhu, Hengshu Zhu, Qi Liu, Hui Xiong, and Enhong Chen. A survey on large language models for recommendation. *World Wide Web (WWW)*, 27(5):60, 2024.

Yao Wu, Christopher DuBois, Alice X. Zheng, and Martin Ester. Collaborative denoising auto-encoders for top-n recommender systems. In *WSDM*, pp. 153–162. ACM, 2016.

Ling Yang, Zhilong Zhang, Yang Song, Shenda Hong, Runsheng Xu, Yue Zhao, Yingxia Shao, Wentao Zhang, Ming-Hsuan Yang, and Bin Cui. Diffusion models: A comprehensive survey of methods and applications. *CoRR*, abs/2209.00796, 2022.

Wenchuan Yang, Cheng Yang, Jichao Li, Yuejin Tan, Xin Lu, and Chuan Shi. Non-autoregressive personalized bundle generation. *Inf. Process. Manag.*, 61(5):103814, 2024.

Zhengyi Yang, Jiancan Wu, Zhicai Wang, Xiang Wang, Yancheng Yuan, and Xiangnan He. Generate what you prefer: Reshaping sequential recommendation via guided diffusion. In *NeurIPS*, 2023.

Zhouxin Yu, Jintang Li, Liang Chen, and Zibin Zheng. Unifying multi-associations through hypergraph for bundle recommendation. *Knowledge-Based Systems*, 255:109755, 2022.

Jianyang Zhai, Zi-Feng Mai, Chang-Dong Wang, Feidiao Yang, Xiawu Zheng, Hui Li, and Yonghong Tian. Multimodal quantitative language for generative recommendation. In *ICLR*, 2025.

Bowen Zheng, Yupeng Hou, Hongyu Lu, Yu Chen, Wayne Xin Zhao, Ming Chen, and Ji-Rong Wen. Adapting large language models by integrating collaborative semantics for recommendation. In *ICDE*, pp. 1435–1448. IEEE, 2024.

A THE USE OF LARGE LANGUAGE MODELS

We used large language models (*e.g.,* ChatGPT 5, Claude) as an assistive tool for writing polish (grammar, phrasing, and LaTeX formatting), troubleshooting LaTeX errors, and scaffolding non-critical scripts (plotting and small utilities). LLMs did not contribute novel scientific ideas, data collection, or result selection, and any code snippets suggested by an LLM were reviewed, rewritten where necessary, and validated by the authors. All technical claims, mathematical formulations, and empirical results are the authors' responsibility. LLMs are not listed as authors and have no authorship rights.

B IMPLEMENTATION DETAILS

Dataset statistics. The statistics for the datasets used in our experiments are summarized in Table 6. All Spotify series dataset share a large item catalog size (N) of 254,155, with a massive candidate space. Density Spotify$_{k=30}$ > Spotify$_{k=60}$ > Spotify$_{k=90}$. Conversely, the POG$_{k=4}$ dataset features a substantially smaller item catalog size (N) of 31,217 and contains a more consistent number of bundles across the splits (*e.g.,* $\mathrm{M}_{train} = 29,704$). This variation in item catalog size and bundle count allows for a comprehensive evaluation of our method's scalability and performance under different data density conditions.

Data augmentation. To improve the model's robustness and prevent the bundle overfit to the default sequential item order of the bundle, we employ a data augmentation strategy based on item swapping. Specifically, for each original item sequence, we performed a series of adjacent item swaps on a copy of the sequence. For the POG dense dataset, we set swap ratio 0.8, and for the Spotify datasets, we used swap ratio 0.4. Subsequently, we randomly sampled a fixed-length subsequence (sequence length) from the perturbed sequence, creating a new augmented training instance. This data augmentation is an enhancement to the diffusion model, fulfilling the non-sequential modeling objective while countering the potential issue of overfitting to the certain given sequential order in the bundle.

Over-retrieval. To standardize the evaluation of generative models which can produce multiple possible outputs, we employ an Over-Retrieval Strategy. This strategy aggregates the results from multiple generation attempts, effectively forming the union $\hat{B}_y$ used in the retrieval-based metrics. For generative sampling models, we evaluate performance under a varying number of attempts $b \in \{1, 5, 10, 20, 50\}$ (denoted as Multiple Sampling, MS). For auto-regressive baselines that use beam search, we report the results using beam width $b \in \{1, 3, 5, 10, 20, 50\}$, mapping the beam width to the number of attempts ($K = b$) for a fair comparison of computational cost.

Table 6: Dataset statistics. N is the catalog size (total number of items in the dataset); $M_{\text{train/val/test}}$ are the number of bundles in train/val/test sets.

Dataset	N	M_{train}	M_{val}	M_{test}
Spotify$_{k=30}$	254,155	321,929	1,374	2,744
Spotify$_{k=60}$	254,155	253,358	798	1,582
Spotify$_{k=90}$	254,155	188,618	463	969
POG$_{k=4}$	31,217	29,704	1,303	2,521

Input tokenization for bundles (details). Given $b = \{i_1, \ldots, i_{|b|}\}$ with item codes $\mathbf{z}(i_j) = (z_{j,1}, \ldots, z_{j,L})$, the serialized sequence is

$$\mathbf{x} = (\texttt{<bos>}, \texttt{<boi>}, z_{1,1}, \ldots, z_{1,L}, \texttt{<boi>}, z_{2,1}, \ldots, z_{2,L}, \ldots, \texttt{<boi>}, z_{|b|,1}, \ldots, z_{|b|,L}, \texttt{<eos>}). \tag{10}$$

Its length is $U = 2 + |b|(L+1)$. Define the index map for item j and level ℓ:

$$u(j, 0) = 1 + (j-1)(L+1) + 1, \qquad u(j, \ell) = u(j, 0) + \ell, \;\; \ell \in \{1, \ldots, L\}. \tag{11}$$

Algorithm 1 RVQ-ENCODE for an item embedding

Require: Item embedding $E(i) \in \mathbb{R}^d$; semantic codebooks $\{\mathcal{C}^{(1)}, \dots, \mathcal{C}^{(L-1)}\}$ with $\mathcal{C}^{(\ell)} = \{\mathbf{e}_c^{(\ell)}\}_{c=1}^{C_\ell} \subset \mathbb{R}^d$; dedup indexer $\textsc{Dedup}(i) \in \{1, \dots, C_L\}$
Ensure: Code indices $\mathbf{z}(i) = (z^{(1)}(i), \dots, z^{(L)}(i))$ and reconstruction $\hat{E}(i)$

- $\mathbf{r} \leftarrow E(i)$; $\hat{E} \leftarrow \mathbf{0}_d$
- **for** $\ell = 1$ **to** $L-1$ **do** ▷ semantic levels
  - $z^{(\ell)}(i) \leftarrow \underset{c \in \{1,\dots,C_\ell\}}{\operatorname{argmin}} \|\mathbf{r} - \mathbf{e}_c^{(\ell)}\|_2^2$ *(tie-break: smallest index)*
  - $\hat{E} \leftarrow \hat{E} + \mathbf{e}^{(\ell)}_{z^{(\ell)}(i)}$; $\mathbf{r} \leftarrow \mathbf{r} - \mathbf{e}^{(\ell)}_{z^{(\ell)}(i)}$
- **end for**
- $z^{(L)}(i) \leftarrow \textsc{Dedup}(i)$ ▷ non-semantic dedup level
- **return** $\mathbf{z}(i)$, $\hat{E}(i) = \hat{E}$

Algorithm 2 Constraint-aware order-agnostic decoding (inference)

Require: Observed item set $\mathbf{b}_x$; index maps $u(j, \ell)$ and $\textsc{InvIdx}(u) \to (j, \ell)$; code-domain valid sets $\{\mathcal{V}^{(\ell)}_{\text{valid}}(j; \mathbf{z}_{j,<\ell})\}$; diffusion model p_θ; horizon T
Ensure: Clean token matrix $\mathbf{Z}^{(0)}$ and completed bundle $\hat{\mathbf{b}}$

- Initialize $\mathbf{Z}$ with tokens for Ω_x and for Ω_{flag}; set $z_u \leftarrow$ `[MASK]` for all $u \in \Omega_y$
- **while** there exists $u \in \Omega_y$ with $z_u =$ `[MASK]` **do**
  - Choose a timestep $t \in \{1, \dots, T\}$ (*e.g.*, $t = T - s + 1$ at step s, or by a schedule)
  - **for all** $u \in \Omega_y$ with $z_u =$ `[MASK]` **do**
    - $(j, \ell) \leftarrow \textsc{InvIdx}(u)$
    - $\pi \leftarrow p_\theta(\cdot \mid \mathbf{Z}^{(t)} = \mathbf{Z}, t)$ *categorical over* $\{1, \dots, C_\ell\}$ *(no* `[MASK]`*)*
    - **mask out invalids:** $\pi[c] \leftarrow 0$ for $c \notin \mathcal{V}^{(\ell)}_{\text{valid}}(j; \mathbf{z}_{j,<\ell})$; $\pi \leftarrow \pi / \sum_c \pi[c]$
    - $P(u) \leftarrow \pi$
  - **end for**
  - Select a reveal set $S \subseteq \{u \in \Omega_y : z_u =$ `[MASK]`$\}$ (*e.g.*, top-k by $\max P(u)$, lowest-entropy, or reveal ratio η)
  - **for all** $u \in S$ **do**
    - **decode:** $z_u \leftarrow \arg\max P(u)$ *(or sample with temperature/top-p)*
    - **clamp:** z_u stays unmasked thereafter
  - **end for**
- **end while**
- $\mathbf{Z}^{(0)} \leftarrow \mathbf{Z}$
- **return** $\mathbf{Z}^{(0)}$, $\hat{\mathbf{b}}$ via $\hat{i}_j = \textsc{Code2Item}(z_{j,1:L})$ for all j

We then specify exactly two sets:

$$\Omega_{\text{flag}} = \{1, U\} \cup \{u(j,0)\}_{j=1}^{|b|}, \qquad \Omega_{\text{code}} = \{u(j,\ell) : j = 1, \dots, |b|, \ \ell = 1, \dots, L\}. \tag{12}$$

By construction, $\Omega_{\text{flag}} \cap \Omega_{\text{code}} = \varnothing$ and $\Omega_{\text{flag}} \cup \Omega_{\text{code}} = [U]$. Positions in Ω_{flag} are never masked; corruption and prediction operate only on Ω_{code} (including the dedup level $\ell = L$).

RVQ encoding pseudocode. We elucidate the pseudocode in Algorithm 1, specifying the encoding for an item embedding.

Evaluation metric setting. To quantify the similarity between p_{Predict} and p_{Target}, we compute the pairwise similarity $S(t_i, \hat{t}_j)$ for all tracks $t_i \in p_{\text{Target}}$ and $t_j \in p_{\text{Predict}}$. This setup forms a bipartite graph, where the nodes correspond to tracks in the two playlists, and the edge weights represent their pairwise similarity scores. The total similarity is defined as the *Optimal Weighted Bipartite Matching*:

$$M^* = \arg\max_M \sum_{(t_i, t_j) \in M} S(t_i, t_j), \tag{13}$$

where M is a bijective mapping between p_{Target} and p_{Predict}.

Hungarian algorithm. We employ the Hungarian algorithm to solve the optimal matching problem. The steps are detailed with pseudocode format in Algorithm 3.

Algorithm 3 OAS via Hungarian Algorithm (maximize sum of cosine similarities)

Require: Predicted set $\hat{\mathbf{b}}_y = \{\hat{i}_1, \ldots, \hat{i}_{\hat{n}}\}$ (duplicates removed); ground-truth set $\mathbf{b}_y = \{i_1, \ldots, i_n\}$; embeddings $E(\cdot)$
Ensure: Optimal matching $M \subseteq \{1, \ldots, \hat{n}\} \times \{1, \ldots, n\}$ and OAS

build similarity: $S \in \mathbb{R}^{\hat{n} \times n}$ with $S[a, b] \leftarrow \cos\big(E(\hat{i}_a), E(i_b)\big)$
$m \leftarrow \max(\hat{n}, n)$
build square cost: $\tilde{C} \in \mathbb{R}^{m \times m}$ ▷ convert max-sim to min-cost
for $a = 1$ to m **do**
  for $b = 1$ to m **do**
    if $a \leq \hat{n}$ **and** $b \leq n$ **then**
      $\tilde{C}[a, b] \leftarrow 1 - S[a, b]$ ▷ cost $\in [0, 2]$ since cos $\in [-1, 1]$
    else
      $\tilde{C}[a, b] \leftarrow 1$ ▷ dummy pairs have similarity 0
    end if
  end for
end for
row reduction: $\tilde{C}[a, \cdot] \leftarrow \tilde{C}[a, \cdot] - \min_b \tilde{C}[a, b]$ for all a
column reduction: $\tilde{C}[\cdot, b] \leftarrow \tilde{C}[\cdot, b] - \min_a \tilde{C}[a, b]$ for all b
repeat
  Cover all zeros in $\tilde{C}$ by the minimum number of horizontal/vertical lines
  if (#lines $< m$) **then**
    $\Delta \leftarrow \min\{\tilde{C}[a, b] : \tilde{C}[a, b] \text{ is uncovered}\}$
    Subtract Δ from every *uncovered* entry
    Add Δ to every *doubly-covered* entry
    (singly-covered entries unchanged)
  end if
until #lines $= m$
extract assignment: find m independent zeros (no two share a row/column) to form an optimal assignment $\tilde{M} \subseteq \{1, \ldots, m\}^2$
restrict to real items: $M \leftarrow \{(a, b) \in \tilde{M} : a \leq \hat{n},\ b \leq n\}$
$S_{\text{sum}} \leftarrow \sum_{(a,b) \in M} S[a, b]$
OAS $\leftarrow \dfrac{S_{\text{sum}}}{n}$ ▷ denominator is $|\mathbf{b}_y| = n$
return M, OAS

C Additional Results

Comparison across datasets using Jaccard. We compare the results crosss datasets using Jaccard with multiple attempts. As Table 7 shows, among the established baselines, BundleNAT generally achieves the best Jaccard performance across the Spotify datasets. This suggests that BundleNAT's non-auto-regressive architecture is particularly effective at generating relevant set-based results compared to the sequential models. What's more, our proposed method, DDBC, consistently and significantly outperforms all baselines across every dataset and attempt level. On Spotify$_{k=30}$, the DDBC model achieves a Jaccard@1 of 0.164, nearly doubling the performance of the best baseline (BundleNAT at 0.090). This performance gap confirms the efficacy and advanced capability of our model, especially when generating predictions with multiple attempts.

Table 7: Comparison across datasets using Jaccard with $A \in \{1, 5, 20\}$. Best in **bold**, second best underlined.

Model (A/beam)	Spotify$_{k=30}$			Spotify$_{k=60}$			Spotify$_{k=90}$			POG$_{k=4}$		
	Jacc@1	Jacc@5	Jacc@20	Jacc@1	Jacc@5	Jacc@20	Jacc@1	Jacc@5	Jacc@20	Jacc@1	Jacc@5	Jacc@20
CLHE	.009	.005	.002	.008	.004	.002	.007	.003	.001	.059	.048	.020
SASRec	.043	.023	.009	.054	.030	.013	.029	.013	.005	.114	.066	.024
TIGER	.053	.028	.011	.076	.036	.013	.070	.034	.013	**.157**	**.094**	**.038**
BundleNAT	.090	.076	.033	.056	.055	.027	.052	.052	.026	.097	.055	.023
DDBC	**.164**	**.130**	**.053**	**.185**	**.137**	**.055**	**.177**	**.132**	**.054**	.098	.073	.032

Latent-space quality. To better explore the latent-space quality of the items generated by our method, we report the OAS metric at $A = 50$, as shown in Table 8. For the Spotify dataset series, when the bundle length (k) increases from 30 to 90, the latent-space quality improves substantially and stable. Specifically, the average OAS decreases consistently from 0.397 (Spotify$_{k=30}$) to 0.338 (Spotify$_{k=60}$) and finally to 0.315 (Spotify$_{k=90}$). Since a lower OAS score indicates lower distance and higher similarity, this trend suggests that our method has an improved capacity to model longer bundles.

Table 8: Latent-space quality at A=50. We report the $\{\min, \text{avg}, \max, \text{var}\}$ OAS over test bundles, corresponding to minimal, average, maximal, and variance, respectively.

Dataset	min	avg	max	var
Spotify$_{k=30}$	0.477	0.603	0.717	0.003
Spotify$_{k=60}$	0.582	0.662	0.733	0.001
Spotify$_{k=90}$	0.625	0.685	0.738	0.001
POG$_{k=4}$	0.246	0.525	0.791	0.018

Additional ablation study results. We report ablation study results on Spotify$_{k=30}$ (A=1 or A=10) in Table 9. The results obtained at A=10 are consistent with those observed at A=1. (1) The Residual Vector Quantization (RVQ) component exhibits to be absolutely indispensable. This result confirms the substantial mechanism of RVQ to mitigate the dimensionality curse caused by the huge item catalog size (N). (2) The boi token could provide positional guidance and improve latent space quality largely. Removing the boi token results in a significant performance degradation. (3) Simple data augmentation (item swapping) proves to be a beneficial technique for enhancing order-agnostic modeling and improving model robustness by mitigating overfitting to specific bundle arrangements. (4) Although the Token Validity Filter yields only a marginal performance improvement, its inclusion remains necessary to guarantee the validity of the generated bundles during inference.

Table 9: Ablation study on Spotify$_{k=30}$ (A=1 vs. A=10). "Proposed" is our model DDBC; each of the other variants changes exactly one component that is removed from the proposed method.

Variant	A=1			A=10		
	F1 ↑	Jacc ↑	OAS ↑	F1 ↑	Jacc ↑	OAS ↑
Our proposed DDBC	0.282	0.178	0.618	0.166	0.092	0.620
w/o RVQ	0.021	0.011	0.557	0.028	0.015	0.556
w/o token validity filter	0.276	0.173	–	0.163	0.090	–
w/o boi token	0.176	0.104	0.538	0.116	0.063	0.536
w/o data augmentation	0.254	0.158	0.599	0.152	0.084	0.598