A Survey of Item Identifiers in Generative Recommendation: Construction, Alignment, and Generation

Taiyan Zhang*, Hongtao Wang*, Yunqian Fan†, Kunda Yang‡, Jichuan Zeng§, and Renchi Yang*
*Hong Kong Baptist University, †ShanghaiTech University, ‡University of Michigan, §The Chinese University of Hong Kong,
*{tyzhang,cshtwang,renchi}@comp.hkbu.edu.hk, †fanyq2022@shanghaitech.edu.cn, ‡kunda@umich.edu, §jczeng@cse.cuhk.edu.hk,

***Abstract*—The paradigm shift from discriminative to generative recommendation has revolutionized the field, empowering systems to autoregressively generate the next item in a user's interaction sequence. Central to this new paradigm is the representation of items, specifically, the item identifier (item ID), which acts as the fundamental bridge connecting discrete items in a catalog to the continuous generation space of *Large Language Models* (LLMs) and Transformer-based architectures. This survey provides a comprehensive and systematic review of item IDs in *Generative Recommender Systems* (Gen-RecSys), covering both LLM-as-Gen-RecSys and Semantic ID (SID)-based Gen-RecSys. We propose a novel taxonomy that categorizes existing works based on the lifecycle of Item IDs: Construction, Alignment, and Generation. We first posit six critical quality criteria for ideal item IDs, including uniqueness, semantics, etc., and use them to evaluate various ID types, ranging from numeric and textual IDs to learned semantic codes. Subsequently, we delve into the technical methodologies for constructing SIDs, analyzing diverse tokenization schemes such as hierarchical clustering and codebook-based quantization (e.g., VQ, RQ, and PQ). Furthermore, we systematically review strategies for aligning these IDs with collaborative and multi-modal signals, as well as techniques for ensuring validity and accelerating decoding during the generation phase. Finally, we highlight open challenges regarding ID collision, semantic flatness, and inference efficiency for SIDs, and outline promising future directions to advance the design of robust and scalable item IDs. The related source can be found at https://github.com/HKBU-LAGAS/Awesome-Item-ID-Gen-RecSys.**

***Index Terms*—Recommendation Systems, Generative Recommendation, Tokenization**

I. Introduction

Traditional recommender systems, which predominantly operate by learning to discriminate or rank a set of candidate items based on historical user interactions, have long been the industry standard. Despite being powerful, this discriminative paradigm faces inherent constraints in cold start and cascading bottlenecks caused by the multi-stage filtering [1], [2], particularly in sequential recommendations, where the goal is to predict the next item in a user's interaction sequence. The surge of *Large Language Models* (LLMs) has catalyzed a revolutionary shift, inspiring the emergence of *generative recommender systems* (Gen-RecSys) for sequential recommendation tasks [3], [4]. Instead of calculating a ranking score for every item in a catalog or candidate set, a generative model autoregressively produces the next item identifier (item ID) [5], directly generating the recommendation, based on a profound insight that a user's sequential behavior, i.e., a sequence of past interactions, can be regarded as a "sentence" describing their preference history. This unlocks the potential for more context-aware and semantically coherent recommendations. Within generative recommendation, two primary research directions [6], [7] have garnered significant attention:

- **LLM-as-Gen-RecSys** paradigm directly harnesses *pre-trained* LLMs[1] as the core reasoning engine to output a recommended list of textually-represented items, based on the input sequence containing profile description, behavior prompt, and task instruction, and the vast knowledge and strong reasoning ability [8] of LLMs.
- **SID-based Gen-RecSys** involves first encoding items into sequences of discrete tokens, creating a "vocabulary" for items. A model with an LLM-like architecture is then trained to generate these sequences, mastering the syntax of user behavior to predict the next item.

A critical role in both categories of generative recommendation paradigms is item IDs [76]. In conventional recommender systems, we uniquely identify items by their respective ID embeddings that are usually learned from collaborative signals underlying user-item interactions [77]. In contrast, both LLM-as-Gen-RecSys and SID-based Gen-RecSys require representing items in a form compatible with LLMs and the Transformer architecture, which typically model items as *token sequences* [5]. For instance, an e-commerce product with ID "product_5876" can be represented by a sequence of tokens like ⟨product⟩⟨_⟩⟨58⟩⟨76⟩ [10]. As pinpointed in [3], the tokenization and identification of items have a significant impact on the effectiveness of generative recommendations. To enable effective and efficient recommendations with such models, the representation of item IDs thus acts as a linchpin. As a fundamental topic, there has been an influx of research on the tokenization of

[1]For ease of exposition, we refer to all pre-trained language models as LLMs throughout this survey.

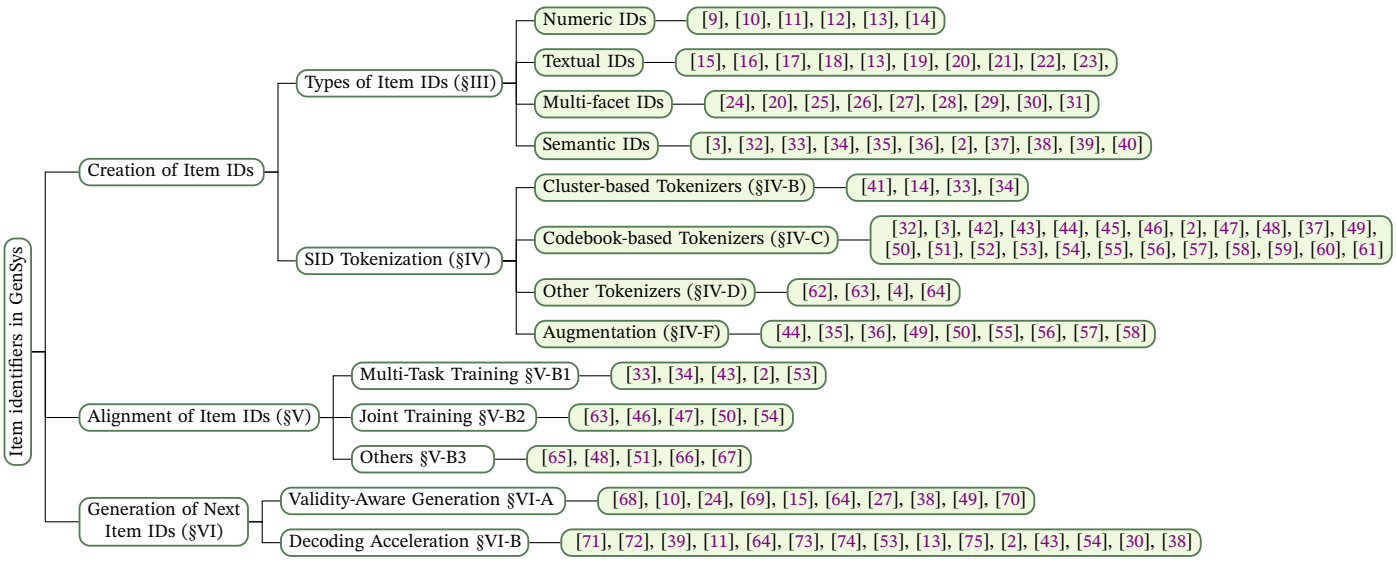

Fig. 1: The proposed taxonomy of item IDs in Gen-RecSys.

items and construction item IDs in recent years [78], [3], [36], [65], [37], [46], [4].

LLM-as-Gen-RecSys primarily relies on two fundamental steps to bridge the item and the language spaces in recommendation [18], [28]: (i) item indexing utilizes item IDs to represent items in the language space, and (ii) generation grounding associates LLMs' generated token sequences to in-catalog items. As the main focus, LLM-as-Gen-RecSys typically indexes items by assigning each item a textual representation, e.g., numeric ID, item title, or descriptions in natural language, and subsequently converts it into text tokens as its item ID inside LLMs. Accordingly, the user's historical interactions are then mapped into a sequence of such item IDs. To adapt LLMs for sequential recommendations with these item IDs, considerable early efforts have been invested in exploring prompt engineering and in-context learning(ICL) [16], [24], fine-tuning [30], [27], [20], multi-tasking [79], [10], preference learning [80], and reinforcement learning [81], [82]. Another research line aligns LLMs for recommendations by augmenting original textual item IDs with auxiliary information, such as pre-trained collaborative embeddings [18], [28], [12], external item tokens [14], [31], [83], and multi-modal content [84].

Recent studies [3], [32] propose to represent each item using content-derived (typically from the textual and visual features of items), semantically meaningful, and compositional tokens shared across items, referred to as *semantic IDs* (SIDs). Instead of relying on ad-hoc vocabulary indexing of items or noisy, verbose item titles/descriptions, SIDs encode items as short, compressed token sequences, enhancing inference speed and storage efficiency [2], [3]. In addition, by capitalizing on multi-modal data, SIDs carry semantical information instead of being arbitrary identifiers [11], which directly tackles the notorious cold-start and long-tail problems [3], [85], [66].

The emergence of SIDs has inspired a large body of subsequent studies in recent years towards building SID-based Gen-RecSys [36], [65], [2], [86], [67], [38], [66], [50], [58], [40], yielding multiple applications in industry, e.g., online retail [87], [88], video [2], [11], music recommendations [89], [90], and local life services [53], [91]. Amid these works, a majority of them focus on the construction of SIDs, involving the design and development of various tokenizers [14], [33], [44], [3], [32], [37], [51] with hierarchical clustering or vector quantization, and their augmentation techniques [36], [50], [92]. Another problem widely studied in these works is the alignment of semantics underlying SIDs with user behaviors in the course of recommender training for more effective generative recommendations [93], [43], [40].

Since both LLM-as-Gen-RecSys and SID-based Gen-RecSys rely on auto-regressive modeling of user sequences that enables the direct generation of next item IDs in the form of tokens, this necessitates additional means to ground item IDs to valid items in the item catalog. To address this challenge, various strategies have been explored in the literature [94], [81], [52]. Moreover, the auto-regressive decoding also poses grand challenges in efficiency and scalability in the generation of next item IDs, making the practical deployment in industrial scenarios problematic, particularly for the SID-based Gen-RecSys, motivating a series of studies on decoding acceleration [71], [72], [38].

Our Contributions. In view of the foundational role of item IDs in Gen-RecSys and substantial related research, this survey is to offer a systematic review of existing studies revolving around item IDs and explore potential directions for future research. Our focus is on the taxonomy, comparative analysis, and open challenges across three dimensions: ID design and construction, augmentation and alignment, and ID generation in both LLM-as-Gen-RecSys and SID-based Gen-RecSys. Particularly, our contributions can be

summarized as follows:

- We are the first survey offering the taxonomy (Fig. 1) of existing Gen-RecSys works from the perspective of item IDs.
- We posit six quality criteria for item IDs in Gen-RecSys (Table III), which facilitate the comparison of various types of IDs (Table IV), and the clarification of the design objectives of related studies.
- We present a taxonomy and comparison of existing LLM-as-GenRecSys (Table I) and SID-based GenRecSys (Table II) frameworks, respectively, based on their categories in terms of ID construction, alignment, and generation.
- We investigate the tokenization schemes for SID creation, analyze their limitations (Table V), and introduce augmentation techniques.
- We categorize and compare various strategies for aligning item IDs and SIDs for recommendation in LLM-as-GenRecSys and SID-based GenRecSys.
- We examine two key problems, i.e., item validity and efficient decoding, in generating next item IDs in Gen-RecSys, and provide a categorization and comparison of their related solutions.
- We articulate several open research challenges and problems in the construction, alignment, and generation of item IDs in Gen-RecSys.

Comparison with Existing Surveys. The majority of prior surveys primarily center on various roles and usage of LLMs in recommender systems [76], [95], [96], [97], [5], [98], [99], [100], including in ranking, rating prediction, feature extraction, conversational recommendation, and explanation generation, etc., and categorizing existing works based on the schemes of integrating or adapting LLMs. Recent surveys [76], [101], [7], [102], [103] on Gen-Recsys partially cover the topics summarized as follows:

- [76] provides a unified framework for both generative search and recommendation, discussing identifiers only as a sub-component within a broader paradigm shift.
- [101] concentrates on LLM-based recommendation, organizing methods by application stages like recall and ranking.
- [7] proposes a "Data-Model-Task" taxonomy that reviews the entire generative ecosystem, including data augmentation and novel tasks like conversation.
- [102] offers a comprehensive review of various generative architectures, including GANs and Diffusion models, and their information fusion mechanisms.
- [103] adopts a tri-decoupled perspective covering tokenization, architecture, and optimization. While they recognize tokenization as a pillar, our survey makes item IDs the sole subject.

Overall, existing surveys predominantly categorize the field based on model architectures or application stages, treating item identifiers as a data processing step or submodule. In contrast, our survey is the first to propose a taxonomy centered entirely on the item ID, systematically deconstructing its lifecycle, i.e., construction, alignment, and generation, to analyze it as the critical bridge between discrete items and continuous generative spaces.

Survey Organization. The overall structure of this survey is as follows. Apart from this introductory section, §II begins with a formal introduction and overview of the foregoing two generative recommendation paradigms, i.e., LLM-as-Gen-RecSys and SID-based Gen-RecSys. The third section (§III) is concerned with the extant item identification approaches used in the LLM-as-GenSys. In §IV, we delineate the tokenization methods for constructing SIDs and augmentation techniques. §V and §VI summarize existing studies pertaining to the alignment of item IDs with auxiliary information for recommendation and the generation of *next* item IDs, respectively. §VII provides a detailed discussion regarding challenges, open problems, and future directions. Finally, §VIII concludes this survey.

II. Generative Recommendation Paradigms

In general, there are two major paradigms for generative recommendation. The first methodology *LLM-as-Gen-RecSys* frames recommendation as an NLP task by inputting items as textual representations to LLMs, and seeks to leverage the world knowledge, text generative capacity, and intrinsic reasoning ability of LLMs to directly produce recommendations [104], [79], [10], [105]. In contrast, *SID-based Gen-RecSys* learns an item-token vocabulary (i.e., SIDs) beforehand, and trains a Transformer-based recommendation model with the *Next Token Prediction* mechanism [3]. In what follows, we briefly recap the common implementations of these two generative recommendation paradigms in § II-A and § II-B.

A. LLM-as-Gen-RecSys

An overview of this paradigm is illustrated in Fig. 2(a). Let $\mathcal{I}$ be the catalog of all items in recommender systems. The LLM-as-Gen-RecSys paradigm typically involves two main steps:

Item Identification. This step assigns a natural language identifier v_j to each item $i_j \in \mathcal{I}$, forming a corpus $\mathcal{V}$ of item IDs. These identifiers can be numerical IDs, item titles, and so on. Based thereon, a user u's historical interactions $[i_1, \dots, i_h]$ are transformed into a textual sequence $S_u = [v_1, \dots, v_h]$ of item IDs. Inside the LLM, items will be represented by its native tokenized text vocabulary.

Item Generation. This step is also referred to as *generation grounding*. At first, the LLM, conditioned on the user's historical interaction sequence as context, auto-regressively generates a sequence $T^{\text{item}}_{\text{next}}$ of text tokens within the language space. This generated sequence is subsequently grounded. That is, it is reverse-mapped to specific items within the corpus $\mathcal{V}$ via methods such as exact matching or distance-based matching, thereby producing the final recommendations.

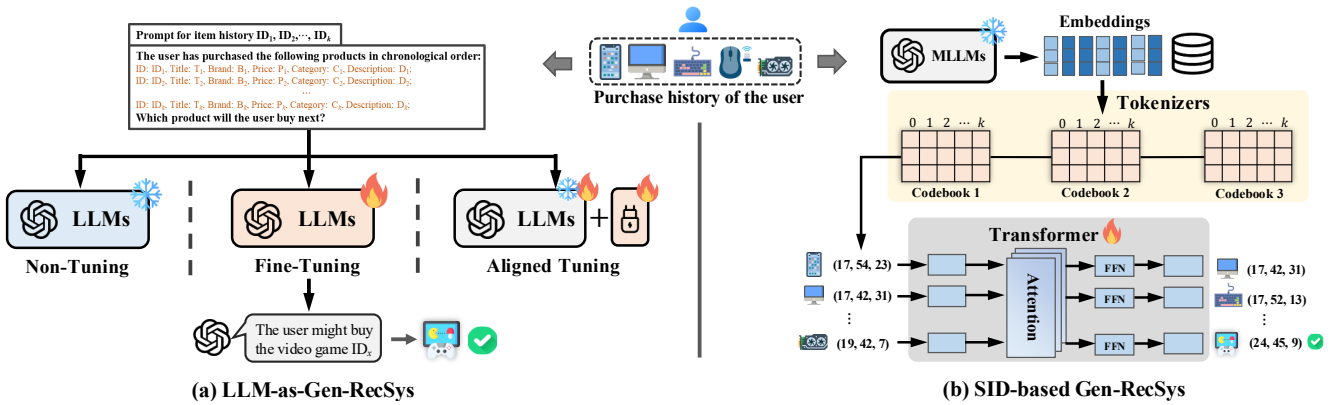

Fig. 2: An overview of existing Gen-RecSys paradigms.

Formally, this generative process can be formulated as follows. Given a user u and its historical identifier sequence S_u

$$T^{\text{item}}_{\text{next}} = \text{LLM}(P, T^{\text{user}}_u, S_u) \text{ and } v_j = \text{Ground}(T^{\text{item}}_{\text{next}}, \mathcal{V}), \quad (1)$$

where P signifies the prompt (e.g., task instruction) and T^{user}_u is the tokens for user u, $\text{LLM}(\cdot)$ and $\text{Ground}(\cdot)$ stand for the pre-trained LLM and item grounding method, respectively.

According to the ways of adapting LLMs for Gen-RecSys, existing LLM-as-Gen-RecSys works can be broadly classified into three categories of paradigms, including *non-tuning*, *fine-tuning*, and *aligned tuning*. Table I lists representative LLM-as-Gen-RecSys works in the literature that align with these categories. In what follows, we briefly revisit non-tuning and fine-tuning. We defer the discussion of aligned tuning paradigm to §V as it involves the alignment or augmentation of item IDs for recommendations.

1) **Non-Tuning:** Most early efforts for LLM-as-GenSys resort to the *non-tuning* paradigm, which simply utilizes the pre-trained LLM with its all parameters frozen for recommendations. Without parameter updates, this paradigm relies solely on prompting techniques and potentially *in-context learning* [106] to condition the LLM's behavior and elicit desired outputs.

Early studies explored open-set direct generation, attempting to generate recommendations directly from the LLM's vocabulary without any candidate set, e.g., via simple prompting [107] or by analyzing text completions [68]. However, this approach was found to perform poorly. A more effective methodology is *candidate-based generative selection* [23]. To be specific, the LLM is provided with a pre-defined candidate set $\mathcal{C} \subset \mathcal{I}$, and is instructed to selectively output a list $\mathcal{R} \subset \mathcal{C}$ of the best items for recommendations. For example, [107] uses a multi-step prompt that first guides the LLM to summarize user preferences, and then explicitly instructs it to generate the top-k list, which is then extracted using a simple rule-based approach. In [24], an external recommender system is employed to generate such a candidate set $\mathcal{C}$.

2) **Fine-Tuning:** *Fine-tuning* is to adapt a general-purpose LLM to recommendation by updating its parameters on task-specific recommendation data to align the model with recommendation objectives. A commonly-adopted methodology is *instruction tuning*, where the recommendation data (e.g., user history and target items) is formatted into natural language instructions.

More precisely, given an instruction input χ that contains a designed P, T^{user}_u, and S_u, and an expected instruction output y containing the next item ID, the LLM parameters are adjusted by minimizing the negative log-likelihood of the target sequence:

$$\mathcal{L} = -\sum_{(x,y)\in\mathcal{D}_{\text{instruct}}} \sum_{\ell=1}^{L} \log \mathbb{P}(\vec{\mathbf{y}}^{(\ell)} | \chi, \{\vec{\mathbf{y}}^{(1)}, \cdots, \vec{\mathbf{y}}^{(\ell-1)}\}), \quad (2)$$

where $\mathcal{D}_{\text{instruct}}$ is the set of instruction input-output pairs, $\vec{\mathbf{y}}^{(\ell)}$ is the ℓ-th token of the target output y. In particular, by the parameter update strategies, the tuning methods can be further categorized into *full fine-tuning* (FFT), *parameter-efficient fine-tuning* (PEFT), and *data-efficient fine-tuning* (DEFT).

Specifically, FFT updates all parameters of the backbone LLM. A foundational example [10] pre-trains and fully fine-tunes a T5 model on personalized prompts to unify diverse tasks. [20] framed recommendation as an instruction-following task, fine-tuning Flan-T5-XL on a large dataset to align with diverse user needs. [30], [27] applied full *supervised fine-tuning* (SFT) to a Qwen-7B model, BART-large, or other models. In contrast, PEFT updates only a small fraction of parameters in the LLM through efficiency techniques like *Low-Rank Adaptation* (LoRA) [108], as in [79], [17], [15], [25], [26]. DEFT [109] is a data pruning strategy using a surrogate model (e.g., `SASRec`), influence, and effort scores to select fine-tuning data for efficient post-training of LLM-based Gen-RecSys. Along another line, recent studies within this paradigm focus on optimizing training objectives with preference learning and reinforcement learning, e.g., *Direct Preference Optimization* (DPO) [110].

B. SID-based Gen-RecSys

Distinct from the LLM-as-Gen-RecSys paradigm that primarily relies on the built-in text vocabulary of the LLM and

TABLE I: Summarization of representative LLM-as-Gen-RecSys works. (■: Direct Generation, ♦: Closed-Set Generation, ▲: Post-Grounding, ★: Constrained Decoding; △: Speculative Decoding, □: Pruned Search, ◇: Parallel Decoding, ○: General Optimizations)

Gen-RecSys	Time	LLM	Paradigm	Item ID Type	Item Type	Item Generation	
						Validity	Acceleration
Pre-ChatGPT-News [23]	Jun 2023	ChatGPT	Non-Tuning	Textual	News	♦	-
Chat-REC [24]	Apr 2023	GPT-3.5	Non-Tuning	Multi-facet	Movie	♦	-
Zero-Shot NIR [107]	Apr 2023	GPT-3	Non-Tuning	Textual	Movie	♦	-
P5 [10]	Sep 2022	T5	Fine-Tuning	Multi-facet	Products, Businesses	■	□
TALLRec [24]	Sep 2023	LLaMA-7B	Fine-Tuning	Textual	Movie, Book	♦	○
GenRec [17]	Jul 2023	LLaMA-7B	Fine-Tuning	Textual	Movie, Toys	■	○
URM [30]	Feb 2025	Qwen-7B	Fine-Tuning	Multi-facet	Sports, Games, Beauty, Toys	♦	□ + ○
LLM4POI [25]	Jul 2024	LLaMA-7B	Fine-Tuning	Multi-facet	POI	■	○
TransRec [27]	Aug 2024	BART, LLaMA-7B	Fine-Tuning	Multi-facet	Beauty, Toys, Businesses	★	□
DEALRec [109]	Jul 2024	BIGRec, TIGER	Fine-Tuning	Multi-facet	Games, Micro-videos, Books	▲	○
InstructRec [20]	May 2023	Flan-T5-XL	Fine-Tuning	Multi-facet	Games, CDs	♦	-
PALR [19]	Jun 2023	LLaMa-7B	Fine-Tuning	Multi-facet	Movie, Beauty	♦	-
BIGRec [15]	Dec 2023	LLaMA-7B	Fine-Tuning	Textual	Movie, Game	▲	□
GNPR-SID [26]	Aug 2025	LLaMA3-8B	Fine-Tuning	Multi-facet	POI	▲	○
S-DPO [80]	Nov 2024	Llama2-7B	Fine-Tuning	Textual	Movie, Book, Music	♦	-
A-LLMRec [13]	Aug 2024	OPT-6.7B	Aligned Tuning	Multi-facet	Movies, Games, Beauty, Toys	♦	○
E4SRec [12]	May 2024	LLaMA-13B	Aligned Tuning	Numeric	Products, Businesses	♦	○
LLaRA [18]	Jul 2024	LLaMA-7B	Aligned Tuning	Multi-facet	Movies, Games, Music Artists	♦	-
SETRec [64]	May 2025	T5, Qwen	Aligned Tuning	Multi-facet	Sports, Games, Beauty, Toys	▲	◇
CoLLM [28]	Jun 2025	Vicuna-7B	Aligned Tuning	Multi-facet	Movies, Books, Games, CDs	♦	○
LC-Rec [83]	Apr 2024	LLaMA-7B	Aligned Tuning	Multi-facet	Instruments, Arts, Games, CDs	▲	-
CLLM4Rec [31]	Apr 2024	GPT-2, T5, LLaMA-7B	Aligned Tuning	Multi-facet	Sports, Beauty, Toys, Businesses	♦	○
I-LLMRec [111]	Mar 2025	Sheared LLaMA-2.7B	Aligned Tuning	Multi-facet	Sports, Grocery, Art, Phone	♦	○
IDLE-Adapter [29]	Oct 2025	BGE, OPT, SBERT	Aligned Tuning	Multi-facet	Movie, Music, Clothing	♦	○
Align^{3}GR [112]	Nov 2025	Llama2-7B	Aligned Tuning	Multi-facet	Instruments, Beauty, Businesses	▲	○
ContRec [113]	Apr 2025	Llama-3.2-1B-Instruct	Aligned Tuning	Multi-facet	Music, Movie, Games, Beauty	▲	○
EAGER-LLM [114]	Apr 2025	Llama-7b	Aligned Tuning	Multi-facet	Beauty, Sports, Instruments	▲	□
Ef-ID-Gen [11]	Sep 2025	PaLM 2	Aligned Tuning	Numeric	Office, Pets, Instruments, Arts	▲	□

its generative abilities (natural-language prompting with textual outputs), SID-based Gen-RecSys requires learning task-specific item-token vocabulary to construct SIDs (i.e., item token sequences) for all items in $\mathcal{I}$, and training a generative recommender that can generate SIDs for recommendations over this item-token space. More concretely, the SID-based Gen-RecSys workflow (Fig 2(b)) can be divided into three main steps:

SID Construction. For each item $i_j \in \mathcal{I}$, the goal of this step is to convert i_j into a sequence $v_j = \{\vec{\mathbf{t}}_j^{(1)}, \cdots, \vec{\mathbf{t}}_j^{(L)}\}$ of L discrete tokens as its *deterministic* SID. The tokenization is typically performed on various input feature modalities of items, such as interaction, textual descriptions, and multimodal information, which are first encoded via pre-trained models, e.g., collaborative filtering models, LLMs, and multi-modal LLMs. Accordingly, each user u's session $[i_1, \ldots, i_h]$ is represented using a sequence $S_u = [v_1, \ldots, v_h]$ of SIDs, i.e., token sequences, input to the recommendation models.

Recommender Training. At this step, a Transformer-based recommender model is trained with next-token prediction over the SID vocabulary. Suppose the SID of the target item i_t is $v_t = \{\vec{\mathbf{t}}_t^{(1)}, \cdots, \vec{\mathbf{t}}_t^{(L)}\}$. Its training objective can be formulated as:

$$\mathcal{L} = -\sum_{\ell=1}^{L} \log \mathbb{P}(\vec{\mathbf{t}}_t^{(\ell)} | S_u, \{\vec{\mathbf{t}}_t^{(1)}, \cdots, \vec{\mathbf{t}}_t^{(\ell-1)}\}) \tag{3}$$

As such, the tokens of the target item i_t will be generated autoregressively. The SID-based Gen-RecSys solutions developed by academia and industry either train their Transformer-based recommendation models (e.g., with the T5-like architecture) from scratch [3], [2] or leverage pre-trained LLMs as base models to facilitate subsequent recommendation training [65], [37].

In addition to the next-token prediction, recent works have explored various alternative or auxiliary training objectives. For instance, Ref. [38] substitutes the multi-token prediction for the next-token prediction. [2], [67] and [118] enhanced the recommender training with additional reinforced preference optimizations, such as DPO and *Group Relative Policy Optimization* (GRPO) [119]. On top of that, to better align SIDs for downstream recommendations in the training stage, a series of alignment strategies have been developed in the literature [33], [43], [34], [46], [50], [47], as surveyed in §V.

Item Generation. In the inference stage, an SID, i.e., token sequence $\{\vec{\mathbf{t}}_t^{(1)}, \cdots, \vec{\mathbf{t}}_t^{(L)}\}$, is generated by autoregressive decoding from logits in the recommender. In most SID-based Gen-RecSys [41], [14], [33], [63], [44], [35], [36], [49], a *constrained generation* strategy [120] will be employed to ensure that the generated SID $\{\vec{\mathbf{t}}_t^{(1)}, \cdots, \vec{\mathbf{t}}_t^{(L)}\}$ can be mapped to a valid item i_t in $\mathcal{I}$. This constrained generation is also deprecated in some studies [116], [3], as it often brings a prefix tree that hinders parallel decoding, and invalid SIDs are rare and can be simply filtered.

In Table II, we present a comprehensive categorization of existing SID-based Gen-RecSys works in terms of their adopted backbone models, SID construction, SID alignment, and item generation schemes.

III. Categorization of Item IDs

In this section, we will first examine the quality standards for item IDs in Gen-RecSys, followed by delineating various types of item IDs used in Gen-RecSys. Lastly, we compare these types of IDs in terms of the quality criteria.

TABLE II: Summarization of SID-based Gen-RecSys. (VQ, RQ, PQ, TQ, DQ = Vanilla, Residual, Product, Tree, Disentangled Quantization; : interactions, : text, : multi-modal data; MTT: Multi-Task Training; JT: Joint Training; ■: Direct Generation, ♦: Closed-Set Generation, ▲: Post-Grounding, ★: Constrained Decoding; △: Speculative Decoding, □: Pruned Search, ◇: Parallel Decoding, ○: General Optimizations)

Gen-RecSys	Time	Backbone	SID Construction			SID Alignment	Item Generation	
			Tokenizer	Augmentation	Data		Validity	Acceleration
SEATER [41]	Sep 2023	Transformer	Cluster-based	-		-	★	□
CID [14]	Nov 2023	T5		Collaborative (C)	,	-	★	□
ColaRec [33]	Mar 2024	T5		-	,	MTT	★	□
EAGER [115]	Jun 2024	Transformer		-	,	MTT	★	□
GPTRec [62]	Jun 2023	GPT-2	Others	-		-	▲	-
IDGenRec [63]	Mar 2024	T5		-	,	JT	★	□
ActionPiece [4]	Feb 2025	T5		Collaborative	,	-	▲	○
SETRec [64]	Feb 2025	T5&Qwen2.5		Collaborative	,	-	▲	◇
VQ-Rec [32]	Oct 2022	Transformer	Codebook-based	PQ	-		-	-	-
TIGER [3]	May 2023	T5X		RQ	-		-	▲	-
LMINDEXER [44]	Oct 2023	T5		VQ	-		-	★	□
CoST [42]	Apr 2024	Transformer		RQ	-		-	▲	○
MMGRec [35]	Apr 2024	Transformer		RQ	C & MM	, ,	-	★	□
MBGen [43]	May 2024	Transformer		RQ	-	,	MTT	▲	○
UIST [45]	May 2024	DCN		RQ	-	,	-	▲	○
LETTER [36]	May 2024	Tiger&LC-Rec		RQ	Collaborative	,	-	★	□
TokenRec [65]	Jun 2024	T5		VQ	-		Others	-	-
ETEGRec [46]	Sep 2024	T5		RQ	-	,	JT	-	-
OneRec [2]	Feb 2025	Transformer		RQ	-		MTT	▲	○
COBRA [47]	Mar 2025	Transformer		RQ	-		JT	-	-
MTGRec [116]	Apr 2025	T5		RQ	-		-	▲	◇
BBQRec [48]	Apr 2025	Transformer		RQ	C & MM	, ,	Others	▲	○
UTGRec [37]	Apr 2025	T5		TQ	C & MM	, ,	-	-	-
DiscRec [49]	Jun. 2025	T5		RQ	Collaborative	,	-	★	○
RPG [38]	Jun 2025	Transformer		PQ	-		-	★	◇
MMQ [50]	Aug. 2025	HeterRec&PPNet		VQ	Multimodal (MM)	, ,	JT	-	-
DQRec [51]	Aug 2025	dual-tower		DQ	-	,	Others	-	-
SaviorRec [66]	Aug 2025	Transformer		RQ	C & MM	, ,	Others	-	-
OneLoc [53]	Aug 2025	Transformer		RQ	-		MTT	♦	○
PCR-CA [54]	Aug 2025	MLP		VQ	-	,	JT	-	○
HiD-VAE [52]	Aug 2025	Transformer		RQ	-		-	★	□
COSETTE [55]	Aug 2025	Transformer		RQ	Collaborative	,	-	▲	○
PSRQ [56]	Aug 2025	MLP		RQ	Multimodal	,	-	-	-
FORGE [39]	Sep 2025	LLM		RQ	C & MM	, ,	-	▲	○
OneSearch [57]	Sep 2025	Transformer		RQ, PQ	Collaborative	, ,	-	-	-
PLUM [40]	Oct 2025	Transformer		RQ	C & MM	, ,	-	▲	-
MiniOneRec [67]	Oct 2025	Qwen2.5		RQ	-		Others	★	□
MMQ-v2 [58]	Oct 2025	REG4Rec&PPNet		VQ	C & MM	, ,	-	-	-
STORE [92]	Nov 2025	MOBA [117]		VQ	Collaborative	, ,	-	-	○
H^2Rec [59]	Dec 2025	GRU4Rec&BERT4Rec		RQ	Collaborative	, ,	-	-	-
ReaSeq [60]	Dec 2025	DIN		RQ	Collaborative	,	-	-	○
HiGR [61]	Dec 2025	Transformer		RQ	Collaborative	,	-	-	◇

TABLE III: Criteria of "good" Item IDs in Gen-RecSys

Criterion	Definition
Uniqueness	Items are uniquely identified and distinguishable with item IDs.
Semantics	Meaningful signals from their meta-data
Generalization	Generalization to unseen/long-tail items
Context	Sequential context and collaborative signals
Space Efficiency	Manageable storage overhead of item IDs
Creation Efficiency	Low computational cost of creating item IDs

A. Quality Criteria for Item IDs

As remarked in the preceding section, item IDs are essentially represented by token sequences within Gen-RecSys models, the quality of which is crucial for the downstream recommendation performance. In Table III, we posit six common criteria or requirements for ideal item IDs in Gen-RecSys.

In the first place, recent studies [5], [76], [27], [27] emphasize two core requirements for item IDs: *uniqueness* and *semantics*. The former is to ensure items are uniquely identified and distinguishable from each other, which is well supported in traditional recommender systems through embedding IDs. The latter requires that the tokens of the item ID should capture the rich semantics underlying the item's metadata and content, which are typically grounded in multimodal evidence, e.g., text, images, audio, and structured product attributes. Such semantics encode meaningful signals of items and their domain-specific relatedness, and thus, are conducive to better recommendations. In the meantime, they allow us to tap into the knowledge of LLMs for cold-start and cross-domain recommendations. The ability that enables models to recommend new, rare, or unseen items with high accuracy, and support cross-domain or zero-shot recommendation is referred to as *generalization* [11], which also acts as a key criterion for good item IDs.

Another aspect is the preservation of *context*, i.e., signals around the item underlying user behaviors, including *sequential context* [4], [93] and *collaborative context* [13]. Sequential context refers to the ordered user interaction history (typically used in sequential recommender sys-

tems), while collaborative context means the user–item co-interaction patterns encoded via collaborative filtering. Ideally, items that are adjacent in user action sequences [4], share similar interaction patterns [41], or frequently co-occur [14] should be represented by similar or closely related item IDs in Gen-RecSys.

Real-world item catalogs often consist of millions of raw discrete item identifiers, so assigning a single and unique token or embedding as the ID to each item imposes colossal space overhead [11], [121], [122]. Ideally, the storage cost of item IDs should be manageable in practice, i.e., being *space efficient*. By representing item IDs with a few tokens from a small and compact vocabulary in Gen-RecSys, this scheme offers massive capacity, for example, a $1,000$-token vocabulary with 10-token item ID yields up to $1000^{10} = 10^{30}$ distinct items [5]. However, this naive composition can still introduce item collisions that undermine distinctiveness of items [3], [83]. At last, the creation of item IDs is also expected to be efficient, given the sheer amount of items in industrial-scale recommender systems.

B. Item ID Types

LLM-as-Gen-RecSys represents the entire item catalog by mapping each item to a sequence of tokens from the model's built-in tokenized text vocabulary such that items are natively representable and generable within its language space. Particularly, there are primarily three types of item IDs, i.e., numeric, textual, and multi-facet IDs, as exemplified in Fig. 3. In Table I, we summarize the types of item IDs used in existing LLM-as-Gen-RecSys.

1) **Numeric IDs:** A simple and straightforward way is to use each item's raw, unique numerical ID (e.g., "product_5876" or "5876") as its identifier input to LLMs [9], [10]. This proffers perfect uniqueness as each ID is exactly associated a unique item in the catalog. However, this scheme suffers from several deficiencies. Firstly, the numeric IDs are not directly compatible with LLMs. By default, the tokenization in LLMs will split the ID "5876" into sub-tokens like "58" and "76", reducing the uniqueness of the item IDs [10]. One simple treatment is to regard each item ID as a new, indivisible "token" in LLMs. Unfortunately, doing so inflates the LLM's vocabulary to catalog scale, often millions, making full softmax decoding a dominant compute bottleneck and increasing the size of the output projection proportionally to the vocabulary [11]. As a remedy, [11] propose a novel two-level softmax mechanism, which combines clustering with *approximate nearest neighbor* (ANN) search for efficient training and inference over this massive vocabulary.

To capture collaborative context and meanwhile retain uniqueness, several studies [12], [13] harness external traditional recommenders to construct ID embeddings from collaborative signals to retain uniqueness and collaborative context, which are later injected into the LLM's input space via a trainable, lightweight adapter or projection layer. In another vein, [10] leverage sequential indexing (e.g., "11138", "11139") for adjacent items in a user's history to reflect their co-occurrences, whilst [14] obtain indexing based on item categories or co-occurrence matrices. However, [3] pinpoints that the naive sequential schemes can inadvertently leak information when numbering correlates according to time.

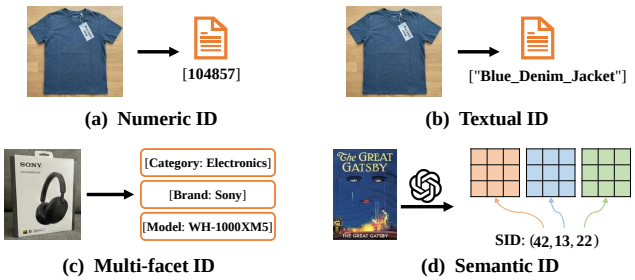

Fig. 3: Illustration of various types of item IDs.

Last but not least, numeric IDs lack semantic information and LLMs cannot comprehend these strings (e.g., "5876"), rendering them unable to tap into the world knowledge of LLMs for recommendations, particularly in cold-start scenarios.

2) **Textual IDs:** This strategy utilizes each item's textual metadata (e.g., titles and descriptions) to identify it in LLMs. A popular choice is item titles [15], [16], [17], [18], [13]. Examples include product names [19], [20], book names [21], news headlines [22], [23], or abstractive text [63]. Such textual IDs embody rich semantics and information, enabling the use of the vast world knowledge of LLMs to understand the item characteristics for recommendations, which have been demonstrated to yield gains in cold-start settings [15], [27].

However, textual IDs incur poor distinctiveness as item titles could be ambiguous, e.g., multiple items with the same name, or non-descriptive. In turn, it is hard to capture the fine-grained, ID-based collaborative signals [13], [18]. On top of that, LLMs may undergo hallucination and out-of-corpus issues when dealing with textual meta-data, especially when it is lengthy. The generated item titles or descriptions can be plausible but fail to match any valid items in the catalog [15], [83]. Even if the LLM is able to generate reasonable textual titles or descriptions, grounding them to in-catalog items is still challenging, as it reduces to a matching problem demanding the calculation of matching scores between generated text and textual meta-data of items in the catalog [5]. Ref. [17], [15] propose to first generate textual titles/descriptions, followed by grounding their tokens to real items via the embedding-based distance. A few studies [23], [24], [107] resort to closed-set generation, which circumvent the above-said issues by retrieving a candidate set and including the textual IDs (i.e., titles or descriptions) of these items to the prompt, before instructing the LLM to make the selection.

3) **Multi-facet IDs:** The multi-facet scheme [27] attempts to overcome the limitations of numeric and textual IDs through a combination of both, i.e., using a composite ID. The resulting multi-facet IDs are anticipated to retain uniqueness as numeric IDs, capture collaborative signals, and encode semantics of items as in textual IDs, simulta-

TABLE IV: Comparison of Item IDs in Gen-RecSys

RecSys	ID Type	Uniqueness	Semantics	Generalization	Context	Creation Efficiency	Space Efficiency
Discriminative RecSys	Embedding IDs	High	None	Low	High	Low/Medium	Low
LLM-as-Gen-RecSys	Numeric IDs	High	None	Low	None	High	High
	Textual IDs	Low	Medium	Medium	None	Medium	Medium
	Multi-facet IDs	Medium	High	Medium	Low	Low/Medium	Medium
SID-based Gen-RecSys	Semantic IDs	Medium	High	High	Low/Medium	Low/Medium	Meidum/High

neously. A standard approach is to explicitly concatenate multiple pieces of information (e.g., ID: "5876", Title: "...") as an item ID in the input prompt [10]. Ref. [24], [20] combine different facets by mixing user/item metadata (e.g., titles, preferences), while [25], [26], [27] include the time, categories, attributes, and numeric IDs of items.

Other ways to increase uniqueness, semantics and inject contextual information are to incorporate collaborative embeddings of items [13], [28], [18], [29] or multi-modal features [111], [30] into the LLM via trainable adapter or projectors, or expand its inherent vocabulary with customized discrete tokens created from user behaviors [31], [83], to augment the original text tokens of item IDs.

4) **Semantic IDs:** Distinct from the above item IDs used in LLM-as-Gen-Recsys, SID-based Gen-Recsys builds its own token vocabulary from the semantic content of items, such that each item is represented by a unique short sequence of semantically grounded tokens, i.e., SID [3], [32]. The semantic content used for extracting these semantic codes can be interaction, textual, and multi-modal data of items, which are first converted into embedding vectors through pre-trained encoders, such as LLMs and multi-modal LLMs [35], [48], [37], [66], [39], [40]. A tokenizer (e.g., `RQ-VAE` [123], [124]) will be subsequently applied to such embedding vectors to derive the token vocabulary, and thus, the SIDs of the items. The learned SIDs can be perceived as a specialized and compact language for the item catalog, whose tokens are thus made catalog-aligned and interpretable. We defer the elaboration on the tokenization and augmentation of SIDs in the next section.

C. Comparison of Item IDs

Table IV compares the foregoing three categories of item IDs in LLM-as-Gen-RecSys and SID in SID-based Gen-RecSys in terms of the six criteria defined in §III-A. To facilitate clear comparison, we additionally include embedding IDs used in traditional discriminative recommender systems. In the following, we carry out a comparative analysis of five categories of item IDs in terms of each quality dimension.

Uniqueness. Firstly, embedding IDs in traditional recommender systems and numeric IDs in LLM-as-Gen-RecSys provide perfect distinctiveness [10], [14], [27], as each item is assigned a distinct vector, integer, or hash, ensuring no collisions. Textual IDs (e.g., product titles and descriptions) usually carry ambiguity or duplication that cause collisions [27]. Multi-facet IDs, which combine several attributes, increase uniqueness but may still face overlap if attribute combinations are not exclusive. SIDs, generated by quantizing content embeddings, intentionally allow slight collisions for semantically similar items to trade off strict uniqueness for semantic grouping and lower space overhead. Note that the collisions can be further controlled by encoding collaborative signals into SIDs.

Semantics and Generalization. As for semantics, embedding and numeric IDs do not encode any semantic information about the item beyond its identity. Textual IDs are rich in textual semantics, as they directly represent item content (e.g., titles and descriptions). Multi-facet IDs further enhance semantics by combining multiple content features (text, image, category), capturing more nuanced item meaning. SIDs are explicitly designed to encode high-level and fine-grained semantics relationships among items, grouping similar ones together in the ID space.

Due to the lack of semantic information and reliance on historical interactions, embedding and numeric IDs generalize poorly to new or long-tail items. Textual and multi-facet IDs generalize better, as they can represent unseen items based on their content features. SIDs achieve the best generalization since they leverage shared semantic features and are obtained by grouping similar items, which overcome the memorization limits of random IDs and enable transfer to new or rare items, i.e., alleviating cold-start and long-tail problems.

Context. Since embedding IDs directly encode user-item interaction patterns, they are highly effective at capturing contextual information (e.g., collaborative signals). By contrast, textual and multi-facet IDs may dilute sequential and collaborative context, as they focus more on item content. SIDs can struggle to capture collaborative signals unless specifically constructed from or aligned with interaction data.

Creation and Space Efficiency. Numeric IDs are the most efficient to create, requiring only the assignment of a new integer or index. Such integers or indices are typically mapped to a small number of tokens in LLMs. In a few works [11], [12], they propose to maintain a customized token for each numeric ID to enhance performance, resulting in significantly higher storage costs. Although embedding IDs can be initialized with random values, they often require time-consuming training with collaborative filtering models to inject signals from user-item interactions. Notice that each item is represented by a distinct embedding vector; the total space consumption for the items in the catalog is thus tremendous.

Textual IDs [79], [15], [80] require textual feature extrac-

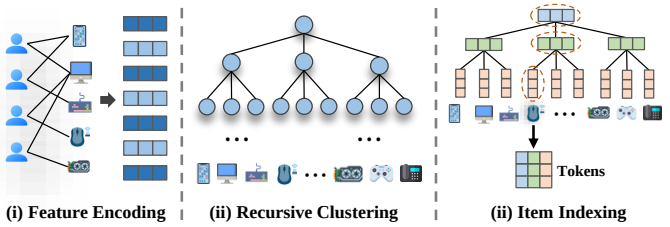

Fig. 4: Illustration of Cluster-based Tokenization

tion, cleaning, and tokenization, rendering their creation slower. In comparison, multi-facet IDs [27], [18], [31] are even more complex, as they demand extracting and combining multiple features, often involving pre-trained models. As for space usage, both textual and multi-facet IDs incur relatively higher space overhead due to variable-length strings and large token vocabularies. The construction of SIDs is usually computationally intensive since they need to go through encoders and quantization processes. However, they utilize compact codebooks but need multiple codewords per item, and, in turn, entail medium space overhead.

In sum, in LLM-as-Gen-RecSys, multi-facet IDs can increase semantics, generalization, and context, while achieving moderate uniqueness, at the expense of creation and space efficiency. SIDs [3], [35], [66] present high semantics and generalization, slightly compromised uniqueness and space efficiency, and medium context preservation and creation efficiency, rendering them a balanced choice for item identification.

IV. Construction of Semantic IDs

As remarked earlier, the SID-based Gen-RecSys typically begins with a front-mounted tokenization stage that encodes items into SIDs, i.e., token sequences, derived from their associated semantic content, such as textual, attribute, and multimodal signals. Based on their tokenization mechanisms, we categorize these tokenizers into three groups: *cluster-based tokenizers*, *codebook-based tokenizers*, and *other tokenizers*. Table II provides the data modalities, tokenizers, and augmentations employed in existing SID-based Gen-RecSys frameworks. In the remainder of this section, we will first summarize common data modalities for tokenization (§IV-A) and review the three categories of tokenizers (§IV-B-IV-D). Lastly, we will examine common strategies employed during tokenization to augment the SIDs in existing Gen-RecSys frameworks (§ IV-F).

A. Data Modalities for Tokenization

There are mainly four types of data modalities used for creating SIDs in the Gen-RecSys, as outlined below.

Interaction Data. Interactions encode relational signals among items and contextual information (sequential and collaborative context), which play a significant role in shaping effective item-ID representations and downstream recommendations. Although interaction data is majorly exploited in the training of recommendation models, it is also widely utilized for the construction of SIDs in Gen-RecSys research [48], [62], [115], [114], [93]. Generally, there are mainly two types of interaction data: *user-item interactions* (a.k.a. user behaviors) and *item-item associations*.

User-item interactions contain explicit and implicit feedback from users on items, such as users' purchasing history, click sequences, ratings, likes, etc., which can be materialized in the forms of rating matrices, bipartite graphs, or item sequences. Considerable research on Gen-RecSys has been done towards leveraging such information to inject contextual information into SIDs [33], [63], [34], [66], [35], [51]. `GPTRec` [62] applies an SVD over the user-item interaction matrix, followed by an quantization to create SIDs, while [115], [114] directly quantize the item embeddings learned from user-item interactions to build the IDs. Item-item associations usually can be derived from user-item interactions and sequences. Examples include co-clicks, co-views, and co-purchases, which can typically be structured as a graph describing the relations among items. A few studies [125], [65], [40], [37] have explored the utilization of these relations to create SIDs.

Textual Data. The most common data modality for SID creation in the literature [3], [47], [49], [116], [38] is textual data. Example textual information that items associate with include titles, descriptions, tags, reviews, captions, transcripts, etc. One special textual data is properties, which are structured attributes that describe an item's categorical and numeric characteristics, such as brand, material, style, function, color, and price. These textual properties provide a concise semantic summary of the item and can be encoded via language models.

Multimodal Data. In modern recommender systems, items are often endowed with multiple modalities, e.g., text, images, audio, video, and geo-spatial signals, each of which provide complementary information about the item [126]. To leverage multi-modal data for SID construction, these modalities are first encoded into embedding vectors with modality-specific encoders (e.g., LLMs, ViT [127], wav2vec [128]), and are later aligned and fused into unified representations as in CLIP [129], LLaVA [130], and other multi-modal LLMs [131]. In particular, numerous studies [35], [50], [48], [66], [39] derive visual features from products or content images to create SIDs. Ref. [56], [89], [35], [132] obtain audio cues of music or sound-centric items, while [2], [53], [133] extract features capturing frames and motions from videos. A few works [53], [26] turn to geographic positions or mobility context.

B. Cluster-based Tokenizers

As depicted in Fig. 4, most cluster-based tokenization methods adopt a three-step pipeline composed of (i) feature encoding, (ii) recursive clustering, and (iii) item indexing, to construct the token sequence $\{\vec{\mathbf{t}}_j^{(1)}, \cdots, \vec{\mathbf{t}}_j^{(L)}\}$ for each item i_j in item catalog $\mathcal{I}$. The basic idea is to build a hierarchical vocabulary $\mathcal{H}$ through a coarse-to-fine process, wherein items are recursively merged from small units up to larger

clusters by their similarities, thereby creating common representations and meaningful tokens.

Feature Encoding. The first step is to encode each catalog item i_j as a compact feature vector $\vec{\mathbf{z}}_j$, referred to as item embedding. One of the popular ways seeks to extract interaction patterns from user behaviors over items. For example, `CID` [14] computes Laplacian eigenvectors of the item co-occurrence graph as item embeddings to reflect their affinity, whereas `SEATER` [41] and `ColaRec` [33] harness pretrained recommendation models, e.g., `SASRec` [134] and `LightGCN` [135], to create item embeddings that embody collaborative information, eliminating the need for further alignment. Some works additionally incorporate content semantics into item embeddings. Particularly, `Eager` [115] and `Eager-LLM` [114] construct two embeddings for capturing behaviors and semantics, respectively, via a two-tower model consisting of the `DIN` [136] for behavioral inputs and a general modality/text encoder (e.g., Sentence-T5, Llama) for semantic inputs.

Recursive Clustering. As the linchpin, this step focuses on recursively clustering the embeddings of items to construct the hierarchical tokenized vocabulary $\mathcal{H}$. That is, similar items in $\{\vec{\mathbf{z}}_1, \cdots, \vec{\mathbf{z}}_{|\mathcal{I}|}\}$ will be grouped into the same clusters $\{\mathcal{C}_1^{(L)}, \cdots, \mathcal{C}_{n_L}^{(L)}\}$, and similar clusters in $\{\mathcal{C}_1^{(\ell)}, \cdots, \mathcal{C}_{n_\ell}^{(\ell)}\}$ at ℓ-th level can be merged into larger ones $\{\mathcal{C}_1^{(\ell-1)}, \cdots, \mathcal{C}_{n_{\ell-1}}^{(\ell-1)}\}$, forming an $(L+1)$-level hierarchical tree $\mathcal{H}$ (containing grounded items) as shown in Fig. 4. In particular, the leaf nodes in the tree are items and the root $\mathcal{C}_1^{(1)}$ represents the entire item set. The non-leaf nodes at the ℓ-th level stand for clusters $\{\mathcal{C}_1^{(\ell)}, \cdots, \mathcal{C}_{n_\ell}^{(\ell)}\}$, each of which can be represented by the cluster center or centroid $\vec{\mathbf{c}}_j^{(\ell)}$ and used as a token embedding. Mathematically,

$$\{\vec{\mathbf{c}}_1^{(\ell-1)}, \cdots, \vec{\mathbf{c}}_{n_{\ell-1}}^{(\ell-1)}\} = \mathsf{Cluster}\left(\vec{\mathbf{c}}_1^{(\ell)}, \cdots, \vec{\mathbf{c}}_{n_\ell}^{(\ell)}\right). \tag{4}$$

Specifically, `CID` [14] applies spectral clustering to recursively split the item set into disjoint clusters such that each child has at most k children. `SEATER` [41] and `ColaRec` [33] employ constrained `KMeans` [137] to build a balanced k-ary tree by recursively partitioning the item set into k groups with nearly-equal sizes until each group has less than k items. [115], [114] recursively group item embeddings into k child clusters with comparable sizes until each leaf contains exactly one item. [14] manually organizes items into a hierarchical structure as per their category text.

Item Indexing. Lastly, for each item i_j, we obtain its corresponding token sequence via a combination (e.g., concatenation) of the token embeddings of its ancestors $\mathcal{C}_1^{(1)}, \mathcal{C}_{j_2}^{(2)}, \cdots, \mathcal{C}_{j_L}^{(L)}$ at all the L levels on the tree, i.e.,

$$\{\vec{\mathbf{t}}_j^{(1)}, \vec{\mathbf{t}}_j^{(2)} \cdots, \vec{\mathbf{t}}_j^{(L)}\} = \{\vec{\mathbf{c}}_1^{(1)}, \vec{\mathbf{c}}_{j_2}^{(2)}, \cdots, \vec{\mathbf{c}}_{j_L}^{(L)}\}.$$

[41], [33] simply assign a unique token to each node (i.e., cluster) in the hierarchical tree $\mathcal{H}$, whereas `CID` [14] introduces b ($b \geq |\mathcal{I}|$) extra tokens for non-leaf nodes to avoid sibling collisions. [115], [114] regard item embeddings $\{\vec{\mathbf{z}}_1, \cdots, \vec{\mathbf{z}}_{|\mathcal{I}|}\}$ at the leaf level as unique tokens and additionally include them in the token sequences of respective items.

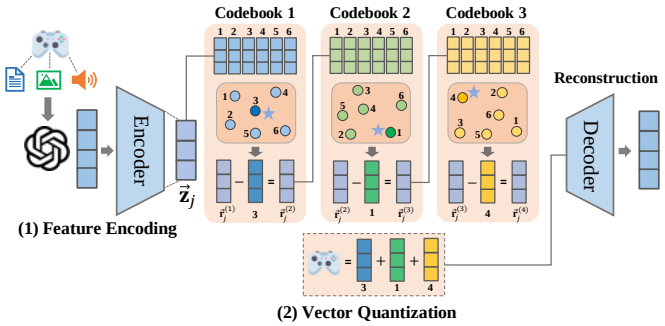

Fig. 5: Illustration of Codebook-based Tokenization

C. Codebook-based Tokenizers

The majority of SID-based Gen-RecSys resort to codebook-based methods for constructing SIDs. This methodology converts each item $i_j \in \mathcal{I}$ into a sequence $\{\vec{\mathbf{t}}_j^{(1)}, \cdots, \vec{\mathbf{t}}_j^{(L)}\}$ of L discrete tokens, each of which is drawn from one of the L distinct learned *codebooks* $\{\mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \cdots, \mathbf{C}^{(L)}\}$. Each codebook $\mathbf{C}^{(\ell)} \in \mathbb{R}^{d \times K}$ contains a set of K distinct codes. The construction can be streamlined into two major steps as illustrated in Fig. 5: feature encoding and vector quantization, where the former is to map the input features $\vec{\mathbf{x}}_j$, extracted from textual, visual, or acoustic data of each item i_j by pre-trained models (e.g., MLLMs), to the embedding(s) $\vec{\mathbf{z}}_j$ in a unified latent space:

$$\vec{\mathbf{z}}_j = \mathsf{Encoder}(\vec{\mathbf{x}}_j), \tag{5}$$

while the latter seeks to draw L codes that are close to $\vec{\mathbf{z}}_j$ from codebooks $\{\mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \cdots, \mathbf{C}^{(L)}\}$, respectively. There exists a large body of literature on vector quantization techniques for the tokenization of multi-modal embeddings, as surveyed in [138]. Here, we mainly elucidate on representative approaches used or devised in existing Gen-RecSys studies for SID creation, including *vanilla*, *residual*, *product*, *tree*, and *decomposition quantization*.

1) **Vanilla Quantization (VQ)***:* The basic idea of VQ is to integrate vector quantization with *variational autoencoders* (VAEs), which has been a popular methodology to learn discrete latent representations for multi-modal data in recent years [139], [140]. The representative VQ technique is `VQ-VAE` [139]. It simply maintains L independent and parallel codebooks $\{\mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \cdots, \mathbf{C}^{(L)}\}$, and searches for the nearest codes to the given item embeddings in the L codebooks separately to form the SIDs. More specifically,

- **Quantization**: The key step is to quantize each of these transformed embeddings with codebooks $\{\mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \cdots, \mathbf{C}^{(L)}\}$. Let $\vec{\mathbf{c}}_k^{(\ell)}$ be the k-th code (i.e., k-th column vector) in codebook $\mathbf{C}^{(\ell)}$. For each item i_j, its

ℓ-th token $\vec{\mathbf{t}}_j^{(\ell)}$ is obtained by looking up the codebook $\mathbf{C}^{(\ell)}$ such that

$$\vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}} \|\vec{\mathbf{z}}_j - \vec{\mathbf{c}}_k^{(\ell)}\|^2. \tag{6}$$

- **Reconstruction**: In lieu of using pre-computed or randomly initialized codebooks, most studies conduct the quantization with learnable codebooks via reconstruction [139]. Formally, they reconstruct the original input as $\vec{\mathbf{x}}'_j$ for each item i_j from each of its corresponding codes $\vec{\mathbf{c}}_{k_j}^{(\ell)}$, through a decoder:

$$\vec{\mathbf{x}}'_j = \text{Decoder}(\vec{\mathbf{c}}_{k_j}^{(\ell)}). \tag{7}$$

The optimization of the codebook-based tokenization consists of three joint objectives:

$$\mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{code}} + \mathcal{L}_{\text{cmit}}, \tag{8}$$

which are formulated using three loss functions. The first one is a reconstruction loss:

$$\mathcal{L}_{\text{recon}} = \sum_{i_j \in \mathcal{I}} \|\vec{\mathbf{x}}_j - \text{Decoder}(\vec{\mathbf{c}}_{k_j}^{(\ell)} + \text{sg}[\vec{\mathbf{c}}_{k_j}^{(\ell)} - \vec{\mathbf{z}}_j])\|^2, \tag{9}$$

where $\text{sg}[\cdot]$ indicates the *stop-gradient* in *straight-through estimator* (STE) [141] for handling non-differentiable objectives (e.g., argmin). Namely, the gradient of the variable in $\text{sg}[\cdot]$ is set to zero in the back-propagation phase. The other two losses aim to enforce the transformed embedding $\vec{\mathbf{z}}_j$ of each item $i_j \in \mathcal{I}$ close to each of its corresponding codes $\vec{\mathbf{c}}_{k_j}^{(\ell)}$:

$$\mathcal{L}_{\text{code}} = \sum_{i_j \in \mathcal{I}} \|\text{sg}[\vec{\mathbf{z}}_j] - \vec{\mathbf{c}}_{k_j}^{(\ell)}\|^2, \ \mathcal{L}_{\text{cmit}} = \sum_{i_j \in \mathcal{I}} \|\vec{\mathbf{z}}_j - \text{sg}[\vec{\mathbf{c}}_{k_j}^{(\ell)}]\|^2.$$

Amid Gen-RecSys works based on VQ, LMINDEXER [44] fetches the nearest code in each codebook via a dot-product lookup and normalized softmax score for each item $\vec{\mathbf{z}}_j$, whilst PCR-CA [54] picks the k nearest codes to $\vec{\mathbf{z}}_j$ in each codebook. Instead of using a single item embedding $\vec{\mathbf{z}}_j$, TokenRec [65] employs L distinct encoders to transform input features $\vec{\mathbf{x}}_j$ into $\vec{\mathbf{z}}_j^{(1)}, \cdots, \vec{\mathbf{z}}_j^{(L)}$, each of which will be used to pair their nearest codes in their corresponding codebooks. MMQ [50] fuses the text embedding $\vec{\mathbf{z}}_j^t$, visual embedding $\vec{\mathbf{z}}_j^v$, and their concatenation $\vec{\mathbf{z}}_j^t \,\|\, \vec{\mathbf{z}}_j^v$ via different modality experts and a gating network, followed by the cosine similarity-based codebook lookups. STORE [92] also applies multiple expert encoders to encode the initial input. To capture the diverse and non-redundant aspects of the original item, both MMQ and STORE impose an orthogonal regularization to make the experts disentangled. However, VQ-VAE inherently suffers from the *codebook collapse* or *codebook underutilization* issue [144]. FSQ [145] substitutes *finite scalar quantization* for the original vector quantization operation in VQ-VAE, which projects the latent representation down to a small number of dimensions (typically less than 10), and each dimension is quantized to a fixed set of values. The codebook is implicitly defined as the product of these sets, matching the codebook size of VQ-VAE but with a much simpler structure. [146] propose a differentiable vector quantization technique (DiVeQ) to enable end-to-end training, while eliminating the need for auxiliary losses (e.g., $\mathcal{L}_{\text{code}}$ and $\mathcal{L}_{\text{cmit}}$) in VQ. SimVQ [142] replaces VQ-VAE's disjoint codebook updates with a single linear layer that jointly optimizes all code vectors, dramatically improving codebook utilization and preventing codebook collapse. It is further adopted in implementing the TQ-based approach UTGRec [37] in §IV-C4.

Further, as revealed in [3], VQ-VAE fails to capture hierarchical or multi-level semantic relationships between items, leading to poor performance for cold-start scenarios and recommendation diversity. CofiRec [147] constructs hierarchical SIDs by manually annotating semantic levels, i.e, *category, title*, and *description*.

2) **Residual Quantization (RQ)***:* RQ is first proposed in RQ-VAE by [123], [124] and is subsequently introduced to Gen-RecSys in the seminal work TIGER [3]. Due to the success of TIGER and its follow-up works [2], [39], RQ has become the mainstream solution to constructing SIDs in SID-based Gen-RecSys. It differs from VQ by using a sequence of residual vector-quantization stages, each of which quantizes the residual vectors from the previous stage by minimizing the remaining reconstruction error. Particularly, it is observed that RQ-VAE is able to achieve a lower reconstruction error, compared to VQ-VAE [3].

Analogous to VQ, RQ-VAE needs to learn L levels of codebooks $\{\mathbf{C}^{(1)}, \mathbf{C}^{(2)}, \cdots, \mathbf{C}^{(L)}\}$. Given the item embedding $\vec{\mathbf{z}}_j$, RQ-VAE derives its L tokens or codes $\{\vec{\mathbf{t}}_j^{(1)}, \vec{\mathbf{t}}_j^{(2)} \cdots, \vec{\mathbf{t}}_j^{(L)}\}$ sequentially. Let $\vec{\mathbf{r}}_j^{(\ell)}$ denote the residual vector of item i_j at the ℓ-th level, and the initial residual vector is $\vec{\mathbf{r}}_j^{(1)} = \vec{\mathbf{z}}_j$. The quantization process will proceed as follows:

$$\begin{cases} \vec{\mathbf{t}}_j^{(\ell)} = \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}} \|\vec{\mathbf{r}}_j^{(\ell)} - \vec{\mathbf{c}}_k^{(\ell)}\|^2, \\ \vec{\mathbf{r}}_j^{(\ell+1)} = \vec{\mathbf{r}}_j^{(\ell)} - \vec{\mathbf{t}}_j^{(\ell)}. \end{cases} \tag{10}$$

A figurative illustration can be found in Fig. 5. In a nutshell, RQ will iteratively use the residual vectors for subsequent lookups in the remaining codebooks, rather than using the item embedding for parallel codebook lookups as in VQ. Doing so iteratively encodes the remaining reconstruction error, and thus, can yield finer approximations. In TIGER, an extra token or code without semantic meaning is injected into the item token sequence to increase the distinctiveness of the SID, and hence, alleviate the item collision problem.

Building on RQ-VAE, MBGen [43] generates the first token using RQ-VAE and the remaining ones by applying K-Means over the residual representations, while MTGRec [116] and PCTX augment the RQ-VAE with a novel *multi-identifier item tokenization* scheme, as the authors believe the strict one-to-one mapping for item tokenization limits the diversity of the items and fails to capture the personalized context. CoST [42] includes a contrastive loss into the learning objectives (Eq. (8)) of RQ-VAE by contrasting each item $\vec{\mathbf{x}}_j$ with its reconstructed feature vector $\vec{\mathbf{x}}'_j$ (positive) and those of other items (negative). Another popular RQ technique for SID construction

TABLE V: Comparison of representative codebook-based tokenizers for SID creation.

Category	Codebook	Tokenizer	Limitation					Gen-RecSys Instances
			Semantic Loss	Codebook Collapse	ID Collision	Semantic Flatness	Creation Inefficiency	
VQ	Single/Parallel	VQ-VAE [139]	✔	✔	✔	✔	✔	[50], [65]
		SimVQ [142]	✗	✗	-	✔	✔	[37]
RQ	Multi-level	RQ-VAE [124]	✗	✔	✔	✗	✔	[3], [36], [40], [60]
		RQ-KMeans [87]	-	✗	✔	✗	✗	[53], [87], [2]
		RQ-OPQ [57]	-	✗	✗	✗	✗	[57]
		PSRQ [56]	✗✗	✔	✔	✗	✔	[56]
		HiD-VAE [52]	✗✗	✗	✗	✗✗	✔	[52]
PQ	Multi-dimensional	OPQ [143]	-	✗	✗	✔	✗	[32], [57], [38]
TQ	Tree	UIT [37]	-	✗	-	✗	✔	[37]
DQ	Multi-dimensional	DQ-VAE [51]	-	-	✗	✔	✔	[51]

is RQ-KMeans, which is adopted in [53], [87], [2] to counteract the issues of the *hourglass phenomenon* [148] (a.k.a. codebook collapse), which signifies unbalanced code distribution, and costly training in RQ-VAE. In comparison, instead of learning codebooks based on reconstruction with VAEs, RQ-KMeans applies the *balanced KMeans* algorithm [2] to update codebooks. Let $\mathcal{S}_1^{(\ell)}, \cdots, \mathcal{S}_K^{(\ell)}$ be the K clusters of items and $\mathbf{C}_k^{(\ell)}$ be the k-th code (i.e., k-th column) in $\mathbf{C}^{(\ell)}$. More concretely, RQ-KMeans will partition residual vectors $\{\vec{\mathbf{r}}_1^{(\ell)}, \cdots, \vec{\mathbf{r}}_{|\mathcal{I}|}^{(\ell)}\}$ into K disjoint clusters, such that (i) each cluster $\mathcal{S}_k^{(\ell)}$ contains exactly $|\mathcal{I}|/K$ items, and (ii) its cluster centroid embedding is the code $\mathbf{C}_k^{(\ell)}$ in the codebook $\mathbf{C}^{(\ell)} \in \mathbb{R}^{d \times K}$. Formally,

$$\mathcal{S}_k^{(\ell)} = \underset{i_j \in \mathcal{I} \setminus \{\mathcal{S}_1^{(\ell)}, \cdots, \mathcal{S}_{k-1}^{(\ell)}\}}{\operatorname{argtop}} \frac{|\mathcal{I}|}{K} -\|\vec{\mathbf{r}}_j^{(\ell)} - \mathbf{C}_k^{(\ell)}\| \text{ and } \mathbf{C}_k^{(\ell)} = \frac{K}{|\mathcal{I}|} \sum_{i_j \in \mathcal{S}_k^{(\ell)}} \vec{\mathbf{r}}_j^{(\ell)}$$

The cluster assignments and codebook embeddings are iteratively updated until convergence. Since each code in a codebook is forced to associate with $|\mathcal{I}|/K$ items, RQ-KMeans is able to alleviate the hourglass phenomenon problem.

As validated in several studies [149], [53], [87], compared to RQ-VAE, RQ-KMeans is able to consistently achieve better recommendation performance, while being significantly faster since the computation merely involves the K-*nearest neighbor* (KNN) search and averaging embedding vectors. On the contrary, a recent study [39] makes an opposite observation on their open-sourced industry dataset by equipping RQ-VAE with a simple collision mitigation strategy.

In [57], the authors pinpoint that VQ-VAE, RQ-VAE, and RQ-KMeans tend to encode shared signals among items using a reduced and fixed vocabulary, losing distinctive attributes of each item, which further leads to ID collisions and curtails the performance of Gen-RecSys. As a remedy, they propose to upgrade RQ-KMeans to RQ-OPQ through a combination with the *optimized product quantization* (OPQ) technique. That is, RQ-OPQ additionally creates two SIDs by product quantization, and integrates them with those produced by RQ-KMeans to preserve the distinctive features of items and increase the *independent coding rate* [57].

Using the *lookup free quantization* (LFQ) [150], ResidualLFQ [151], which accumulates the quantization residual to the next layer, while halving the range of the quantization value, e.g., $\{-1, 1\}$. BSQ [152] further constrains LFQ to a unit hypersphere, enabling faster convergence.

Compared to RQ-KMeans, PSRQ [56] concatenates each residual vector in $\{\vec{\mathbf{r}}_1^{(\ell)}, \cdots, \vec{\mathbf{r}}_{|\mathcal{I}|}^{(\ell)}\}$ with an additional residual before partitioning them into K disjoint clusters. To be precise, the residual vector is updated as

$$\vec{\mathbf{r}}_j^{(\ell)} = \vec{\mathbf{r}}_j^{(\ell)} \,\|\, (\vec{\mathbf{r}}_j^{(1)} - \vec{\mathbf{r}}_j^{(\ell)}), \tag{11}$$

where $\vec{\mathbf{r}}_j^{(1)} = \vec{\mathbf{z}}_j$. This modification aims at retaining the original semantic information in item embeddings $\{\vec{\mathbf{z}}_1, \cdots, \vec{\mathbf{z}}_{|\mathcal{I}|}\}$. It is shown that PSRQ reduces information loss and better preserves the semantic structure of multi-modal inputs (e.g., audio, lyrics, user behaviors).

3) **Product Quantization (PQ)***:* VQ-rec [32] is the first to apply PQ [153], [143] for item tokenization in Gen-RecSys. Unlike prior quantization solutions, PQ first evenly partitions each item embedding $\vec{\mathbf{z}}_j \in \mathbb{R}^d$ into L equal-sized and disjoint sub-vectors $\vec{\mathbf{z}}_{j,1}, \cdots, \vec{\mathbf{z}}_{j,L} \in \mathbb{R}^{\frac{d}{L}}$. As such, for all items in the catalog $\mathcal{I}$ and any integer $1 \le \ell \le L$, we can extract K centroid embeddings from the set of sub-vectors $\{\vec{\mathbf{z}}_{1,\ell}, \cdots, \vec{\mathbf{z}}_{|\mathcal{I}|,\ell}\}$, constituting the ℓ-th codebook $\mathbf{C}^{(\ell)} \in \mathbb{R}^{\frac{d}{L} \times K}$. Similar to Eq. (6) in standard pipeline, the ℓ-th code/token $\vec{\mathbf{t}}_j^{(\ell)}$ of item i_j can be obtained by

$$\vec{\mathbf{t}}_j^{(\ell)} = \underset{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}^{(\ell)}}{\arg\min} \|\vec{\mathbf{z}}_{j,\ell} - \vec{\mathbf{c}}_k^{(\ell)}\|^2. \tag{12}$$

The SID thus can be constructed by optimizing centroid embeddings with *optimized product quantization* (OPQ) [143]. On top of that, VQ-rec will re-train the codebooks and subsequently learn a permutation matrix $\mathbf{\Pi}^{(\ell)} \in \{0, 1\}^{K \times K}$ to align codes in each codebook and indices in the downstream domains.

RPG [38] applies OPQ to generate a long sequence of tokens to enhance the expressiveness of item identifiers, and aggregates the token embeddings via mean or max pooling. The token sequence is regarded as an *unordered* set, and the objective is the multi-token prediction, which renders the training and inference highly parallelizable and hence efficient. RQ-OPQ [57] combines the SID created by RQ-KMeans and OPQ.

4) **Tree Quantization (TQ)***:* As pinpointed in [37], RQ-based approaches suffer from two problems: (i) items with the same prefix codes often share similar subsequent ones, and (ii) they tend to undergo codebook collapse issues for multi-modal inputs. As a remedy, `UTGRec` [37] harnesses the *Universal Item Tokenization* (`UIT`) method to maintain a root codebook $\mathbf{C}_{\text{root}}$ encoding common features of items and a leaf codebook $\mathbf{C}_{\text{leaf}}$ encapsulating incremental features over prefix codes.

The key differences from the standard pipeline in VQ and RQ are as follows. First, TQ requires L input embeddings $\vec{\mathbf{z}}_j^1, \cdots, \vec{\mathbf{z}}_j^L$ for each item, and then applies a residual operation for $2 \leq \ell \leq L$

$$\hat{\vec{\mathbf{z}}}_j^{\ell+1} = \vec{\mathbf{z}}_j^{\ell+1} - \vec{\mathbf{z}}_j^{\ell} \text{ and } \hat{\vec{\mathbf{z}}}_j^1 = \vec{\mathbf{z}}_j^1, \tag{13}$$

engendering a new embedding sequence $\hat{\vec{\mathbf{z}}}_j^1, \cdots, \hat{\vec{\mathbf{z}}}_j^L$. During quantization, the first code/token for item i_j is taken from the root codebook $\mathbf{C}_{\text{root}}$, while the remaining $L-1$ codes/tokens are from the same leaf codebook $\mathbf{C}_{\text{leaf}}$, by the following quantization rules:

$$\begin{aligned} \vec{\mathbf{t}}_j^{(1)} &= \arg\min_{\vec{\mathbf{c}}_k^{(1)} \in \mathbf{C}_{\text{root}}} \|\hat{\vec{\mathbf{z}}}_j - \vec{\mathbf{c}}_k^{(1)} \mathbf{W}_{\text{root}}\|^2, \\ \vec{\mathbf{t}}_j^{(\ell)} &= \arg\min_{\vec{\mathbf{c}}_k^{(\ell)} \in \mathbf{C}_{\text{leaf}}} \|\hat{\vec{\mathbf{z}}}_j - \vec{\mathbf{c}}_k^{(\ell)} \mathbf{W}_{\text{leaf}}\|^2 \ \forall 2 \leq \ell \leq L. \end{aligned} \tag{14}$$

Taking inspirations from `SimVQ` [142], The learnable weights $\mathbf{W}_{\text{root}}$ and $\mathbf{W}_{\text{leaf}}$ are introduced to alleviate codebook collapse.

5) **Disentangled Quantization (DQ)***:* In `DQRec` [51], the authors develop `DQ-VAE` to decouple dimensions of tokens and enhance the interpretability and modularity of the tokenization.

Concretely, given input feature matrix $\mathbf{X} \in \mathbb{R}^{|\mathcal{J}| \times dL}$ for items (or users), it is first centered by subtracting the mean as in *principal component analysis*. Let $\mathbf{X} = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$ be the SVD of $\mathbf{X}$. The right singular vectors (i.e., columns) in $\mathbf{V}$ will be partitioned into disjoint L sub-matrices $\{\mathbf{V}_1, \cdots, \mathbf{V}_L\}$ such that

$$\min_{\{\mathbf{V}_1, \cdots, \mathbf{V}_L\}} \sum_{\ell_1 \neq \ell_2} \left(\sum_{\vec{\mathbf{v}}_k \in \mathbf{V}_{\ell_1}} \mathbf{\Sigma}[k,k]^2 - \sum_{\vec{\mathbf{v}}_k \in \mathbf{V}_{\ell_2}} \mathbf{\Sigma}[k,k]^2 \right), \tag{15}$$

where $\vec{\mathbf{v}}_k$ denotes the k-th singular vector in $\mathbf{V}$. Each submatrix $\mathbf{V}_\ell$ corresponds to a subspace formed by the L orthonormal bases (i.e., singular vectors). Based thereon, `DQ-VAE` projects $\vec{\mathbf{x}}$ to L embeddings $\vec{\mathbf{z}}_j^1, \cdots, \vec{\mathbf{z}}_j^L \in \mathbb{R}^d$ for item i_j by $\vec{\mathbf{z}}_j^\ell = \vec{\mathbf{x}}\mathbf{V}_\ell \ \forall 1 \leq \ell \leq L$.

Similar to Eq. (6), $\vec{\mathbf{z}}_j^1, \cdots, \vec{\mathbf{z}}_j^L$ can be quantized to L codes in $\mathbf{C}^{(1)}, \cdots, \mathbf{C}^{(L)}$, respectively. Since dimensions are "mutually independent", the resulting codebooks are thus disentangled and multi-dimensional. Unlike the reconstruction in VQ and RQ, `DQ-VAE` fixes both the encoder and decoder but updates parameters in codebook embeddings $\mathbf{C}^{(1)}, \cdots, \mathbf{C}^{(L)}$, leading to two simplified joint objectives:

$$\mathcal{L}_{\text{recon}} = \sum_{i_j \in \mathcal{J}} \|\vec{\mathbf{x}} - \vec{\mathbf{x}}'\|^2, \mathcal{L}_{\text{cmit}} = \sum_{i_j \in \mathcal{J}} \sum_{l=1}^{L} \|\vec{\mathbf{z}}_j - \vec{\mathbf{c}}_k^{(\ell)}\|^2. \tag{16}$$

D. Other Tokenizers

Instead of using multi-modal content, `GPTRec` [62] first applies truncated SVD over the user-item interaction matrix, followed by quantizing the resulting item embeddings, and then mapping these quantized values to discrete sub-ID tokens. `IDGenRec` [63] utilizes `SentencePiece` [154] as the tokenizer to compress the item's metadata into natural language tokens compatible with LLMs. It frames the ID generation as a text-to-text generation task and additionally develops a diverse ID generation algorithm to ensure that generated IDs are distinct for every item. Akin to [115], `SETRec` [64] conducts separate tokenization of the collaborative embeddings via a pre-trained recommendation model and the semantic embeddings of items via a pre-trained feature extractor. Each item is then represented as an order-agnostic set of collaborative and semantic tokens. `Actionpiece` [4] views each user's history as a sequence of actions and requires the SID of each item to capture its action context. Accordingly, its tokenization begins by representing each item as an *unordered* set of feature embeddings, followed by dynamically and iteratively merging frequently co-occurring feature embeddings to form a vocabulary of tokens.

E. Comparison of Codebook-based Tokenizers

In Table V, we compare the above representative tokenizers in terms of their categories, codebook types, limitations, and the Gen-RecSys frameworks where they are adopted. Fig. 6 depicts the codebook structures employed in these tokenization schemes. Next, we focus on discussing their major deficiencies of these tokenizers, the mitigation methods, and related measures proposed in the literature [57], [39] for evaluating the tokenization effectiveness pertaining to these deficiencies (summarized in Table VI).

1) **Semantic Loss***:* First, *semantic loss* refers to the loss or degradation of meaningful semantics during the quantization process. In [39], *feature fidelity* (FF) is adopted for measuring semantic loss. Let $\vec{\mathbf{x}}_j$ be the input feature vector (i.e., original semantic information) of the j-th item, and $\vec{\mathbf{x}}'_j$ be the reconstructed feature vector from its codes or tokens. The FF quantifies the semantic loss during the tokenization process:

$$\text{FF} = \max\left(0,\ 1 - \left\|\vec{\mathbf{x}}_j - \vec{\mathbf{x}}'_j\right\|_2 / \left\|\vec{\mathbf{x}}_j\right\|_2\right) \tag{17}$$

Compared to `VQ-VAE`, both `SimVQ` and `RQ-VAE` achieve lower reconstruction errors, i.e., improved semantic fidelity, as validated in their empirical analyses [142], [3]. In the follow-up works, `PSRQ` [56] further optimizes the preservation of original semantics during quantization through constraining residual quantization with prefix semantics, while `HiD-VAE` [52] employs hierarchically-supervised quantization that aligns the discrete codes from each layer with multi-level item tags, which, in turn, improves semantic fidelity.

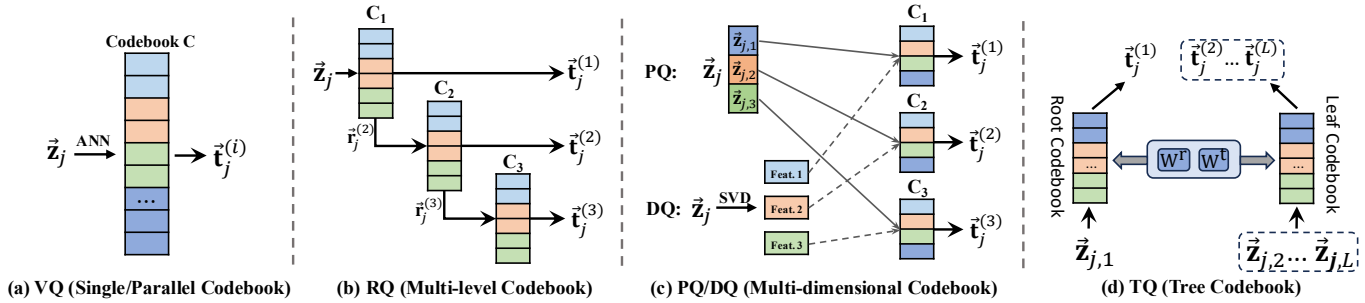

Fig. 6: An illustration of codebook structures.

2) **Codebook Collapse***: Codebook collapse* [155], [148] (a.k.a. *hourglass phenomenon* or *representation collapse*) is a well-documented issue in vector quantization, especially as codebook size increases, which occurs when only a small subset of the codebook entries are used, limiting the diversity and expressiveness of the learned representations.

This issue can be quantified by *codebook utilization rate* (CUR) and *independent coding rate* (ICR) [57], whose definitions are as follows:

$$\text{CUR} = \frac{|\#\text{non-empty codebooks}|}{|\#\text{total codebooks}|}, \tag{18}$$

The CUR measures the proportion of codebooks that are actively used to represent items, while the ICR quantifies the proportion of unique SIDs that are mapped to the distinct items. Let $\mathcal{C}$ denote the set of all the items that are assigned a SID in the system, and the independent set $\mathcal{C}_{ind} \subseteq \mathcal{C}$ as those SIDs assigned to exactly one item. The ICR is formally defined as:

$$\text{ICR} = \frac{|\mathcal{C}_{ind}|}{|\mathcal{C}|} \tag{19}$$

where $|\mathcal{C}_{ind}|$ represents the number of the SIDs with a one-to-one item relationship, and $|\mathcal{C}|$ represents the total number of the items.

From a different angle, since multiple semantically similar items are often mapped to the same SID, while some SIDs remain unused, the *Gini coefficient* (Gini), which is widely used to assess the distribution within a population, is adopted by [39] to quantify this imbalance of SID usage. This metric indirectly assesses the codebook collapse. Let $\mathcal{I}(\text{SID}^{(j)})$ be the set of items that associate with the j-th SID $\text{SID}^{(j)}$. Given the set of all SIDs $\{\text{SID}^{(1)}, \text{SID}^{(2)}, \cdots, \text{SID}^{(n)}\}$,

$$\text{Gini} = \frac{2}{n}\sum_{i=1}^{n}\left(\frac{i}{n} - \frac{\sum_{j=0}^{i}|\mathcal{I}(\text{SID}^{(j)})|}{\sum_{j=0}^{n}|\mathcal{I}(\text{SID}^{(j)})|}\right), \tag{20}$$

which can be regarded as the difference of the cumulative distribution function between the uniform and the real SID distributions.

To alleviate this issue in vector quantization models, substantial efforts, including stochastic quantization [156], distribution penalty [157], and codebook reset [158], have been made [142]. `SimVQ` (and also `UIT` [37]) mitigates this issue by reparameterizing all codebook vectors through a learnable linear transformation over a latent basis, thereby jointly optimizing the entire codebook space. In comparison, `RQ-KMeans` directly partitions residual vectors into equal-sized clusters to form codebooks to avoid codebook collapse, and `OPQ` divides the high-dimensional space into several lower-dimensional subspaces for independent quantization, which inherently reduces the risk of collapse. `RQ-OPQ` [57] integrates `RQ-KMeans` with `OPQ` to achieve the collapse mitigation. [36] groups code embeddings by constrained `K-Means`, and applies a diversity regularization to render the code embeddings from the same cluster closer and those from different clusters distant.

3) **ID Collision***: ID collision* (a.k.a. semantic conflict or representation entanglement) [52], [159], [52] is another key issue that codebook-based tokenization suffers from, which describes a situation where multiple distinct items may be assigned similar or overlapping IDs.

In e-commerce, it's common to find items from different shops that are essentially the same, which are either identical in style (e.g., size, category) or derived from the same source (via data augmentation like image editing or text modification). The *style/source consistency* (SC) metric [39] evaluates whether items identified as belonging to the same style or source are assigned to the same SIDs, i.e., ID collisions:

$$\text{SC} = \frac{|\{(i,j) \in \mathcal{S} | \text{SID}^{(i)} = \text{SID}^{(j)}\}|}{|\mathcal{S}|}, \tag{21}$$

where $\mathcal{S}$ stands for the set of item pairs with the same styles or sources.

By decomposing high-dimensional feature spaces into multiple subspaces and quantizing each subspace separately, `OPQ` and `RQ-OPQ` directly reduce the probability of ID collisions. `HiD-VAE` introduces a "uniqueness loss" that penalizes overlap in the latent space, directly targeting and resolving ID collisions. `DQ-VAE` decomposes pre-trained embeddings from user-item interactions and attributes into distinct dimensions and then quantizes these dimensions to create disentangled and distinctive SIDs. Additional mitigation strategies primarily rely on structural or distributional adjustments: (i) suffix appending, which adds non-semantic distinguishing tokens or random indices to force uniqueness [3], [14], [35]; (ii) distribution alignment, utilizing algorithms like *Sinkhorn-Knopp* [83] or random sampling [39] to rebalance code assignments; and (iii) imposing additional regularization in the tokenization process. For example, [44] adds a contrastive objective to

TABLE VI: Quality measures for SID tokenization

Category	Measure	Definition
Codebook Collapse	Codebook Utilization Rate [57], [39]	Eq. (18)
	Independent Coding Rate [57]	Eq. (19)
	Gini coefficient [39]	Eq. (20)
Collaborative Context	Embedding HitRate [39]	Eq. (22)
Semantic Loss	Feature Fidelity [39]	Eq. (17)
ID collision	Style/Source Consistency [39]	Eq. (21)

distinguish the different documents that share the same prefix, while HiD-VAE [52] includes a *disentanglement uniqueness loss* to push away the different items that are assigned the same SID.

4) **Semantic Flatness***:* The fourth limitation is *semantic flatness* [52], [36], which refers to the lack of a meaningful, interpretable hierarchy and inability to reflect multi-level or structured semantic relationships, which often results from flat, unsupervised approaches like VQ-VAE. PQ and DQ build multi-dimensional codebooks to capture semantics in independent subspaces, neglecting the hierarchy. To mitigate this issue, RQ approaches create hierarchical and multi-level codes. HiD-VAE [52] addresses this issue by introducing hierarchically-supervised quantization, aligning codes with multi-level item tags and enabling interpretable, structured semantic representations.

5) **Creation Efficiency***:* As for the construction efficiency, most tokenizers rely on optimizing the auto-encoders for reconstruction, which entails costly training overhead. By contrast, RQ-KMeans, OPQ, and RQ-OPQ are computationally faster, due to their reliance on efficient clustering and codebook partitioning rather than end-to-end neural network optimization. HiGR [61] proposes a new coarse-to-fine construction approach, which sequentially sketches a preference embedding for each item to guide the construction of the SIDs.

6) **Others***:* In addition to the aforementioned five common defects, some researchers seek to ameliorate existing tokenizers in terms of tokenization instability [160], [161], which causes inconsistencies and reduced performance and can result from external tokenizers not aligning with the LLM's internal schema, the lack of collaborative signals [36], [65], and others [50], [56].

Particularly, *embedding HitRate* (HR@K) is used in [39] to assess the capturing of collaborative signals. Concretely, it measures if a user's interacted items from their current session appear in the recommender's retrieved results. With top-K items ranked by the predicted scores as the retrieval results, HR@K is formulated as

$$\text{HR@}K = \frac{1}{n_{\text{test}}} \sum_{i=1}^{n_{\text{test}}} \frac{|\mathcal{I}_i^{\text{pred}} \cap \mathcal{I}_i^{\text{click}}|}{|\mathcal{I}_i^{\text{click}}|} \tag{22}$$

where n_{test} is the total number of test user samples, $\mathcal{I}_i^{\text{click}}$ stands for the set of items that are actually clicked by the i-th user, and $\mathcal{I}_i^{\text{pred}}$ denotes the set of top-K retrieved items by the recommender. Notice that $\mathcal{I}_i^{\text{pred}}$ is obtained based on the similarities of the multi-modal features of items for the i-th user here.

F. SID Augmentation

Along another line, considerable research [66], [55], [35], [83], [36], [46], [48], [49], [43], [53] focuses on introducing additional losses, tasks, and modules in the training of RQ-VAE to augment or align the SIDs with ancillary information for stronger uniqueness and semantics.

1) **Collaborative Augmentation:** It is contended that an ideal semantic ID should not only encapsulate an item's inherent features but also recognize the associations between frequently co-occurring items [40], [61] Moreover, building SIDs solely on textual information often leads to ID collisions, where items with similar features are assigned the same ID. Next, we detail the emerging approaches that augment SIDs with collaborative signals for enhanced uniqueness.

Amid cluster-based tokenizers, after obtaining the token sequence $\{\vec{\mathbf{t}}_j^{(1)}, \vec{\mathbf{t}}_j^{(2)} \cdots, \vec{\mathbf{t}}_j^{(L)}\}$ as aforementioned, CID and Eager-LLM combine such token sequences from distinct sources (e.g., semantic and behavior data) as item IDs. [33], [115] jointly optimize the item IDs together with the primary recommendation task by introducing additional training objectives, e.g., item-item indexing task and contrastive loss.

Among codebook-based tokenization approaches, many of them [66], [55], [35], [36], [48], [49], [43], [47], [40], [85], [59] inject collaborative signals into SID construction in RQ-VAE. More specifically, methods like [55], [66], [36], [40], [61] follow the idea of contrastive learning for the augmentation. For instance, Savior-Rec [66] involves an extra contrastive objective, taking the frequent co-click patterns as positive pairs and using InfoNCE loss to align the collaborative signals and the semantic embeddings into a unified resulting embedding space. COSETTE [55] mines the co-occurrence pairs as positive pairs and applies the *sigmoid contrastive loss* [162] to guide the tokenization. [36], [40] introduce an extra collaborative regularization on the standard RQ-VAE process to inject collaborative signals in the item tokenization process. DAS [125] conducts multiple contrastive tasks to augment the SID with collaborative signals, including the user-item relationship and the co-occurrence signals, from both the user and the advertisement side. HiGR [61] collects semantic neighbors and high co-occurrence signals as positive pairs to mitigate the ID entanglement phenomenon.

Other works [35], [48], [49], [43], [53], [93] resort to structural design to incorporate collaborative signals. [35], [43], [147] adds an extra token to the semantic IDs to represent the behavior signal. [49] designs an additional independent tokenizer to process the collaborative signal, and then fuses with the SID by a *dynamic fusion gate*. [48] aligns the text/image signals and behavior signals with a specially designed alignment encoder, then quantizes the encoded text, image, and a learnable ID signal into a behavior-aligned identifier via a shared RQ-VAE codebook, and concatenates the three distinct identifiers into a final item representation. [53] captures the user behavior patterns with a specially designed self-attention module,

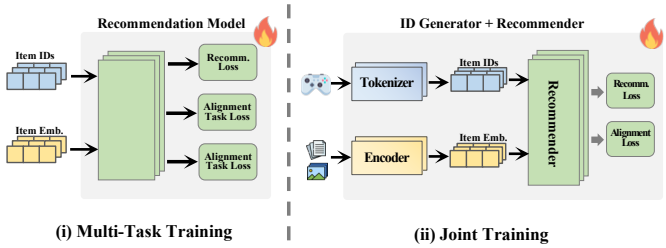

Fig. 7: Alignment Schemes in SID-based Gen-RecSys

according to the user's real-time location. [93] trains an additional sequential model to capture the user context feature and then adopts a data augmentation approach to concatenate this context feature with the item's own feature and train with RQ-VAE.

Furthermore, a few works [47], [85] have attempted to combine semantic IDs and traditional learnable ID embeddings to form a unified tokenization. COBRA [47] generates the item identifier with a cascade structure. First, COBRA generates a sparse semantic ID for each item with RQ-VAE, based on the items' property information. Then it generates a dense vector for each item that contains the textual information of the item. While [85] captures the uniqueness of items by assigning a reduced ID embedding, and learns the item characteristics by RQ-VAE. It then trained the model in an end-to-end manner, optimizing the tokenization loss and recommendation loss jointly.

2) **Multi-Modal Augmentation:** While the majority of methods for SID creation rely on textual descriptions to capture an item's intrinsic features [41], [14], [34], [3], [44], [32], [42], [116], items are often represented by multiple modalities, including visual and acoustic data. MMQ [50] introduces a multi-expert architecture, where modality-specific experts for text and image capture modality-dependent information, while modality-shared experts model synergistic information across modalities. MMQ-v2 [58] inherits this architecture and further concentrates on how to adaptively align, denoise, and amplify the multimodal signals. To be specific, it fuses the dynamic behavioral modality with the static multi-modal modality, and applies a behavior-guided dynamic router to calibrate the importance scores of the behavioral SIDs. PSRQ [56] concatenates the multimodal embedding and obtains the multimodal SIDs with a Modal-joint codebook. It then applies the attention mechanism to the collaborative embedding sequence to integrate the collaborative information and obtain the user's interest representation.

V. Aligning Item IDs for Recommendations

Since item IDs are derived from textual or multi-modal content that lacks recommendation-oriented context, almost all the extant LLM-as-Gen-RecSys and SID-based Gen-RecSys frameworks [12], [28], [18], [163], [63], [46] need to align these IDs for downstream recommendations, typically with historical user-item interactions. In Table VII, we summarize these alignment schemes.

A. Alignment in LLM-as-Gen-RecSys

In the LLM-as-Gen-RecSys paradigm, item IDs are often represented by textual tokens that lack the collaborative and behavioral signals inherent in traditional recommender systems. To bridge this gap, *alignment* strategies are employed to inject external information into the LLM. We categorize these strategies into *Aligned Tuning*, where lightweight modules are trained to map external features into the LLM's input space.

Aligned tuning seeks to inject external information, e.g., collaborative embeddings, extra tokens, and multi-modal features, to align pre-trained LLMs for recommendations. This paradigm often adds lightweight adapters or projectors to the backbone LLM, which requires training additional parameters in such adapters or projectors, while the core parameters in LLMs can be either frozen or fine-tuned. We categorize existing works into three groups based on the source of aligned information:

1) **Collaborative Alignment:** This category focuses on injecting collaborative signals ranging from static ID embeddings to dynamic behavioral patterns derived from user-item interactions into the LLM. A primary approach harnesses pre-trained traditional recommendation models (e.g., SASRec [134], LightGCN [135]) to extract ID embeddings. For instance, E4SRec [12] freezes the LLM but incorporates an input linear projection for ID embeddings alongside a LoRA adapter. Similarly, models like CoLLM [28] and A-LLMRec [13] employ trainable projectors (e.g., MLP) to map these collaborative embeddings into the LLM's input space, ensuring the model captures latent user-item interactions. IDLE-Adapter [29] further addresses the ID-language barrier by aligning ID embeddings with the LLM's semantic space. Beyond static embeddings, this category also includes strategies that inject dynamic behavioral patterns or soft tokens to represent user interests. For example, LLaRA [18] employs a curriculum learning strategy to introduce projected behavioral tokens alongside textual prompts progressively. [64] propose representing each item by an order-agnostic set of collaborative filtering and semantic tokens, treating item features as an unordered set to better align with user preferences. Furthermore, recent work like Align^{3}GR [112] introduces a multi-level alignment framework, integrating dual SCID tokenization with multi-task supervised fine-tuning to align behavior modeling with user preferences.

2) **Multi-Modal Alignment:** To enhance item understanding and reduce hallucination, this approach augments textual IDs with rich modality features. I-LLMRec [111] inputs visual features extracted from item images to the LLM via a learnable adapter. By mapping visual embeddings into the text token space, the LLM can leverage visual semantics for more accurate recommendations, particularly in cold-start scenarios.

Aligned tuning seeks to inject external information, e.g., collaborative embeddings, extra tokens, and multi-modal features, to align pre-trained LLMs for recommendations. This paradigm often adds lightweight adapters or projectors

TABLE VII: Summarization of Alignment Schemes in Gen-RecSys.

Gen-RecSys	Tuning/Training Scheme	Trainable Components	Optional Training Objectives	
			Type	Objective
LLM-as-Gen-RecSys	Aligned Tuning	LLM + adaptor/projector	Behavior	collaborative embedding projection [13], sequential alignment [29], bidirectional semantic/preference alignment [29], curriculum learning [18], discriminative ID prediction [12]
			Semantic	visual feature projection [111], multimodal prompt alignment [84], ID-Text contrastive learning [29]
SID-based Gen-RecSys	Multi-Task Training	decoder, recommender	Behavior	user-item/item-item/item-ID alignment [33], target-behavior/behavior-specific/behavior item prediction [43]
			Semantic	semantic guided transfer [34], text/query alignment [163]
	Joint Training	tokenizer, encoder, recommender	Semantic	multimodal reconstruction [50]
			Behavior	behavior-aware fine-tuning [50], sequence-item/preference-semantic alignment [46]
	Others	recommender	behavior-aligned multi-modal quantization [48]	

to the backbone LLM, which requires training additional parameters in such adapters or projectors, while the core parameters in LLMs can be either frozen or fine-tuned.

`E4SRec` [12] freezes the LLM but incorporates three trainable components: an input linear projection for ID embeddings, a LoRA adapter for personalization, and an item linear projection layer for output. Several models [13], [28], [18], [29] harness a pre-trained traditional recommendation model (e.g., `SASRec` [134] and matrix factorization) as an encoder to obtain collaborative embeddings. A trainable projector (e.g., MLP) is then included to map these collaborative embeddings into LLMs' input space. `LLaRA` [18] employs a curriculum learning strategy to introduce projected behavioral tokens into the LLM, while [64] proposes to represent each item by an order-agnostic set of collaborative filtering and semantic tokens. Apart from collaborative signals and extra tokens, `I-LLMRec` [111] inputs visual features extracted from item images to the LLM via a learnable adapter.

B. Alignment in SID-based Gen-RecSys

Most SID-based Gen-RecSys works [33], [43], [34], [63], [50], [46], [47], [48] typically introduce alignment strategies consisting of *joint training*, *multi-task training*, and other schemes, during recommender training to reconcile multi-modal features with collaborative signals in the item ID spaces. The alignment strategy adopted in each work is given in Table II.

1) **Multi-Task Training:** Based on the item IDs and embeddings previously obtained from ID tokenizers and other models, the *multi-task training* is to align them for recommendation by multi-tasking the recommender training [33], [43], [34], [163]. Fig. 7 presents a figurative illustration of this strategy.

For instance, `ColaRec` [33] achieves the alignment through three carefully-designed alignment tasks over user embeddings, item embeddings and IDs. Therein, the user-item alignment task seeks to predict the next item and item rankings in user interaction sequences, while the item-item alignment task contrast items based on their ID prefix. The item-ID alignment task predicts item IDs from their corresponding text descriptions and relevant users. `MBGen` [43] performs the alignment upon ⟨behavior, item⟩ pairs with three tasks, including target-behavior item prediction, behavior-specific item prediction, and behavior-item prediction. In the former two tasks, a behavior token is given to the model for learning to predict items conditioned on the fixed target behavior and various specific behavior types, respectively, whilst in the third task, the model jointly predicts behavior types and the corresponding items. `EAGER` [34] adopts two parallel decoders to decode item IDs derived from pre-trained multi-modal and collaborative embeddings, respectively, in the recommender, where the hidden states of the former are used to guide the reconstruction of masked collaborative codes in the latter decoder, thereby achieving an alignment. `SemanticConvergence` [163] conducts the alignment in its fine-tuning stage by designing a suite of supervised tasks to synchronize collaborative signals with language semantics. In addition to the next-item prediction task, it incorporates a language-semantic alignment task, learns to map item descriptions and user search queries to specific item tokens. Notably, it enriches the sequential alignment with a negative sampling strategy, training the model to predict items a user would negatively interact with to mitigate overfitting and sample selection bias.

2) **Joint Training:** As depicted in Fig. 7, instead of using the cascaded pipeline where the item ID construction and recommendation model training are decoupled, the *joint training* strategy integrates them into a joint and end-to-end framework such that item IDs are constructed and meanwhile aligned for recommendations.

One way to combine the item ID generator and recommender is to alternate the training of these two models for stable training process. `IDGenRec` [63] employs this strategy by feeding the output hidden states of the LLM-based ID decoder to the base recommendation model so as to ensure that produced IDs can be effectively interpreted by the base recommender and final next token prediction by the recommendation pipeline is accurate. `ETEGRec` [46]

treats these two stages as parallel dual encoder–decoders, which are trained alternatively with an additional KL divergence aiming at aligning token distributions from the ID encoder and embeddings from the recommender encoder, and an additional contrastive loss aligning reconstructed embeddings by the ID decoder and those by the recommender decoder.

One alternative is to enable gradient flow through the item ID construction process. MMQ [50] adopts the STE strategy in the quantization step when computing the similarity between multi-modal embeddings and codebooks. During backpropagation, gradients flow through soft indices (i.e., the softmax outputs of similarity scores), making the quantization step differentiable. COBRA [47] represents each item using an SID from a frozen RQ tokenizer and its text embedding output by a trainable encoder, allowing gradients from the downstream recommender to propagate to the upstream encoder through the embedding part.

3) **Others:** BBQRec [48] aligns the item IDs for recommendations by incorporating the element-wise products of their quantized similarity embeddings into the original attention scores in the recommender model. TokenRec [65] derives SIDs from the collaborative embeddings output by *graph neural networks* [135], [164], and further aligns the SIDs with the textual representations of the collaborative information through an MLP projector.

VI. Generation of Next Item IDs

This section summarizes works pertinent to generating next item IDs in the Gen-RecSys, which mainly involve the topics of ensuring the validity of generated item IDs and speeding up the decoding stage, particularly in inference. In Tables I and II, we present the validity and decoding techniques employed in existing works on LLM-as-Gen-RecSys and SID-based Gen-RecSys, respectively.

A. Validity-Aware Generation

Since the Gen-RecSys requires directly generating the item IDs for recommendations, one simple approach is the *free generation* [76], wherein at each step, the model searches over the entire vocabulary and selects the tokens with the top-k highest probabilities as the subsequent input for the next step's generation. This leads to one of the key challenges in Gen-RecSys, i.e., the hallucination or out-of-corpus issue [14], where the model may generate next item IDs that cannot match any valid items in the actual item catalog [17]. In response, a wealth of research has been done to ensure the validity of the next item IDs in Gen-RecSys with the following constrained generation methods.

1) **Direct Generation (DG):** In some scenarios with LLM-as-Gen-RecSys, the recommendation is framed as a pure text-generation task, and the generated text itself (e.g., movie titles) is regarded as a recommended item, which eliminates the need for additional mappings from item IDs to items [68], [10], [17], [25], [109].

TABLE VIII: Summary of constrained generation

Scheme	Method
Post-Grounding	Dense Retrieval [15], [64]
Constrained Decoding	Trie [70], [14] FM-Index [27] Codebook Graph [38]

2) **Closed-Set Generation (CSG):** A number of LLM-as-GenRecSys solutions operate over a closed set of items, i.e., a predefined list of candidate items [24], [16]. In some of them [24], [69], [79], [20], [80], [13], [18], [28], [29], the LLM is explicitly instructed to generate the recommendation items within the candidate list provided in the input prompt. On top of that, another line of works [16], [107], [12], [31], [113] instruct the LLM to generate scores [16], text, or structured output [111], which are subsequently processed by certain heuristic rules or models to facilitate a selection from the candidate list. For instance, CLLM4Rec [31] adds an item prediction head that computes the likelihood over each valid item ID from the closed set.

3) **Post-Grounding (PG):** Under the open-set setting [68], [10], the Gen-RecSys model is allowed to generate arbitrary token sequences (i.e., item IDs). The simplest way is to just filter out the invalid generations [3]. Other solutions include additionally carrying out a grounding step so as to map the generated token sequences to valid items in the corpus. For example, BIGRec [15] grounds generated textual tokens to items in the catalog according to the L_2 distances of their embeddings, i.e., dense retrieval. OneRec [165] introduces an extra format reward for legal generation. SETRec [64] utilizes each generated token embedding to compute a set of scores across all items in the catalog. These scores are then aggregated into a final score (e.g., via a weighted sum) for recommending each item, which explicitly balances the influence of collaborative filtering and semantic attributes.

4) **Constrained Decoding (CD):** The idea of this methodology is to impose constraints over token orders with additional trie (a.k.a. prefix tree), tree, or graph structures (summarized in Table VIII) during sequence decoding such that the model is forced to generate valid item IDs [41], [14], [33], [35], [44], [36], [67], [52], [49], [166], [38], [167], [168]. IDGenRec [63] employs tries to accommodate all candidate IDs, ensuring its autoregressively generated textual ID always maps to a valid item in the catalog. TransRec [27] generates multi-facet IDs containing numeric IDs, titles, and attributes, equipped with an FM-index to map their tokens to in-catalog items. RPG [38] connects similar SIDs through a codebook graph, where items are interconnected by weighted edges, thereby enabling propagation between items with confidence scores once an initial item is selected.

DiscRec [49] models tokens in an SID as nodes in a Trie structure and items as paths from root to leaf nodes, ensuring all partial generated sequences are valid prefixes, and hence, full sequences are valid items. [70] and [14] also leverage Trie for constrained generation, where the

generated ID is guaranteed to be valid. However, Trie strictly generates the valid identifier from the first token, where the accuracy of the recommendation depends highly on the accuracy of the first several generated tokens.

B. Decoding Acceleration Techniques

In practice, the Gen-RecSys suffers from efficiency bottlenecks in the generation of next item IDs, especially in the inference stage, due to the enormous model sizes and sequential token-wise decoding that incurs a quadratic computational complexity in terms of the length of sequences of item IDs or user interactions. To tackle these challenges, substantial ongoing efforts [71], [72], [168], [64], [13], [74] have been invested in devising techniques to accelerate the autoregressive decoding (i.e., from model logits to tokens), which consist of the following specialized techniques and general optimizations.

1) **Speculative Decoding (SD):** *Speculative decoding* [169], [170] is a popular technique invented by Google and DeepMind for decoding acceleration in LLMs, whose basic idea is to leverage small models to quickly draft a batch of next tokens and have the larger target model verify them in one forward pass, thereby averting costly generation of tokens by the larger target model directly. Common LLMs only prioritize the top-1 prediction, whereas recommender systems often require predicting the next top-k ($k > 1$) items, leading to a wide adoption of this approach in Gen-RecSys [71], [38], [171], [72]. For instance, `ATSpeed` [71] introduces a novel loss function to align draft model outputs with the target model by minimizing their output differences, while [171] employ retrieval models as draft models to select candidates until the top-k ones are obtained. [72] adopts a self-drafting architecture [172] to empower the target LLM itself to generate candidate tokens.

2) **Pruned Search (PS):** The high-level idea of this methodology is to prune the search space during translating logits into item IDs in the large item catalog. A category of methods [47], [39], [173], [174], [87], [11], [171], [168] facilitates fast filtering by organizing the items or codebooks into certain structures. For example, `FORGE` [39] represents each item using tokens in merely three layers of codebooks, each of which contains 8,192 tokens, and all of which can cover 10^{11} items. [11] propose to organize item embeddings into clusters, such that a two-level softmax can be used to efficiently locate the top-k items within the cluster, while `URM` [168] first produces an initial item subset via subset sampling before expanding to cover probable neighbors.

Other works [47], [49], [39], [166], [175], [2], [38], [94], [167], [168] dynamically prune the search space in decoding based on partial outputs obtained. Representative approaches include constrained sampling techniques and the prominent *beam search* [176], whose idea is to retain only a small number of potential item sequences during multi-step decoding to effectively bypass paths in advance [47].

3) **Parallel Decoding (PD):** Instead of autoregressively generating tokens, this strategy [38] seeks to produce unordered and long semantic IDs, such that the digits can be predicted in parallel. In the inference, it constructs an undirected graph of SIDs of items based on their similarities and leverages these connections to iteratively refine the maintained beam of SIDs. `SETRec` [64] implements simultaneous item generation by guiding the LLM with a set of learnable vectors, where each query vector corresponds to a token's generation. Since query operations can be executed simultaneously, this allows the model to generate the tokens for all dimensions in parallel within a single step. Instead of building upon auto-regressive models, `LLaDA-Rec` [73] resorts to the diffusion model, whose generation merely requires T steps to generate a total of M tokens ($M \gg T$). Therein, each step generates M tokens in parallel and selects the top-$\frac{M}{T}$ confident tokens.

4) **General Optimizations (GO):** In addition to the above optimizations specialized for decoding, considerable research on Gen-RecSys has developed general model optimizations that indirectly speed up the generation of next item IDs, including compressions, caching strategies, and engineering tricks.

- **Compressions:** This category consists of the *sequence compression* and *low-precision computation*. The former is to reduce latency by decreasing the lengths of sequences in the Transformer model [13], [4], [42], [177], [74]. For instance, [74] merges each item-action token pair into a single token, thereby significantly reducing redundancies in conventional *item-action-item-action* sequences. The low-precision computation typically explores well-established model quantization techniques to speed up recommendations [25], [53], [2].
  [25] implements 4-bit quantized training and inference to considerably reduce GPU memory consumption, while `OneLoc` [53] employs the mixed-precision serving to lower inference latency.
- **Caching Strategies:** To enable real-time recommendations at scale in practice, it is essential to implement caching optimizations in recommender systems. Apart from the traditional strategy of caching pre-processed user and item features [48], [178], [49], [175], [179], [180], [25], [53], [86], [54], [45], recent works [13], [2], [30] have explored the KV caching optimizations in Gen-RecSys. KV caching reuses model-specific attention states (keys/values) for shared prefixes, eliminating redundant computation in autoregressive decoding and improving real-time efficiency. This is particularly helpful when multiple recommendations start from the same user history or partial interaction sequence.
- **Engineering Tricks:** For the practical deployment of Gen-RecSys at an industrial scale, various engineering tricks are needed. Such optimizations span across multiple dimensions of the recommendation pipeline and computational infrastructure, comprising algorithmic approximations, architectural, and kernel optimizations. Architectural optimizations directly adjust the neural

model architecture [74], [43], [2], [54], while kernel optimizations accelerate specific computational components through high-performance implementations, e.g., Flash Attention [74], [25]. Algorithmic approximations trade exactness for speed by approximately implementing compute-intensive operations such as sorting and top-k selection [71].

VII. Challenges and Future Directions

A. LLM-as-Gen-RecSys

The LLM-as-Gen-RecSys paradigm still faces significant friction when transitioning from research prototypes to industrial deployment [97] due to its item identifications and representations.

Semantic-Collaborative Gap in Item IDs. As reviewed in §V-A, to bridge the fundamental misalignment between the general semantic space of LLMs and the collaborative space of recommendations, current methods largely rely on naive instruction tuning or external adapters [28], [13]. However, these training methodologies are arguably suboptimal since simple NTP may not be the most effective objective for recommendation tasks compared to traditional discriminative losses [181], [80].

Future work can explore novel loss functions beyond NTP, such as ranking objectives employed in recent works [181], [80] for distinguishing preferred items from negatives. Furthermore, we anticipate a surge in research focusing on fusing behavioral semantics (user action sequences) with textual and multimodal semantics (images, audio), as made in recent attempts [35], [66], [87].

Validity of Generated Item IDs. The out-of-vocabulary or hallucination problem remains a persistent issue in LLM-as-Gen-RecSys [14]. While constrained decoding (see §VI-A) helps ensure validity, it often limits the model's creativity, reasoning potential, and parallelization capabilities [94]. Currently, mapping generated text back to specific items in a catalog of millions involves error-prone distance-based matching or entity resolution, which can fail to distinguish fine-grained semantic differences [27], [15]. Recent empirical studies [182] also highlight that generalization to novel queries remains inconsistent across different reranking paradigms. One promising direction to go is to develop mechanisms such that the model can verify its own outputs, such as the self-reflection mechanism [183], drafter-verifier architecture [171], and closed-loop optimization [81].

B. SID-based Gen-RecSys

In what follows, we delineate the challenges and opportunities arising from SID representation and construction, alignment, and downstream generation.

1) **SID Representation and Construction:** As discussed in §III-C and §IV, existing SID representation and tokenization approaches present distinct upsides and downsides regarding various aspects.

Lacking Comprehensive Benchmarking. Despite the multitude of item tokenization solutions being proposed, this research task still lacks a comprehensive benchmark for their fair, rigorous evaluation and in-depth analyses. Existing benchmarking efforts [149], [39] are often limited in their scope, both in terms of the methodologies and datasets considered. This partial evaluation can lead to contradictory conclusions, for example, the superiority of `RQ-VAE` and `RQ-KMeans`. Moreover, both benchmarks neglect other competitive tokenization methods, such as hierarchical clustering, alternative codebook-based approaches, and numerous well-established quantization techniques [138] for visual or multi-modal data. Therefore, the development of a comprehensive benchmark that incorporates a wide array of methods and diverse datasets is crucial for advancing the field.

Lacking Quality Assessment. Most extant research defaults to measuring the performance of downstream recommendation tasks. But there is a lack of a systematic analysis, quantitative evaluation, and comparison of existing tokenization approaches and their resulting SIDs in terms of the quality aspects listed in Table V and Table IV. The empirical effectiveness of various tokenization tricks pertinent to many of these aspects still remains unclear. Although [39] introduced a suite of metrics as listed in Table VI, only a paucity of tokenizers are empirically evaluated, and meanwhile, other quality criteria, including generalization, context, semantic flatness, creation, and space efficiency, remain untouched.

Hierarchical vs. Flat Representations. Hierarchical approaches, such as RQ and hierarchical clustering, align well with the natural taxonomy of items or underlying hierarchical semantic relationships. However, this rigidity often induces the hourglass effect. Conversely, flat and disentangled approaches, including VQ, PQ, and DQ, maximize codebook utilization and enable parallel decoding for higher efficiency by treating features as independent subspaces. Yet, these methods suffer from "semantic flatness", struggling to capture the inter-dependencies and hierarchical relationships. Thus, a promising future direction lies in the hybrid combination of hierarchical and flat representations, or exploring topologically complex ID structures like trees in [37] or graphs in [38].

Order-dependent vs. Order-agnostic Tokens. The majority of Gen-RecSys works adopt ordered token sequences as SIDs. Recently, `SETRec` [64], `Actionpiece` [4], and `RPG` [38] claim that each token should characterize unique features of the item. Accordingly, they propose to employ the order-agnostic token sequence as the SID for each item. Doing so allows the predictions of multiple next tokens, and hence, facilitates the training of the recommendation models in parallel through multi-token prediction or query-guided generation [64]. However, constructing order-agnostic tokens is incompatible with the RQ tokenization

scheme. In other words, it is hard to inject the hierarchical semantics into the tokens. Still now, limited research [184], [57] has made the attempt to combine the merits of both.

Uniqueness vs. Semantics. A persistent challenge in discrete tokenization is ID collisions. As reviewed in §IV-E, existing mitigation strategies primarily rely on structural or distributional adjustments. A critical limitation of these approaches is the compromise of semantic integrity, as they often distort the learned representation space to achieve distinctiveness. A promising future direction lies in *purely semantic indexing*, which integrates collision avoidance directly into the quantization process rather than applying post-hoc patches. Recent innovations like *exhaustive candidate matching* and *recursive residual searching* [159] demonstrate that collisions can be resolved by exploring alternative, collision-free paths within the residual search space (via top-K ranking or backtracking).

2) **SID Alignment:** The challenges in SID alignment lie in the fusion of heterogeneous information and the optimization of alignment with downstream tasks.

As discussed in §IV-F and §V-B, existing multi-modal fusion strategies [56], [125], [34], [48] either trade uniqueness for better synergy, or vice versa, incurring suboptimal performance. Drawing from the *partial information decomposition* (PID) theory [185], which deconstructs multi-modal information into *redundancy*, *uniqueness*, and *synergy*, a poorly designed alignment method may overamplify redundant signals common across modalities while suppressing the unique, valuable information exclusive to each one. It is worth exploring to align the heterogeneous signals from the PID perspective.

The cascaded workflow (item tokenization + recommender training) of the SID-based Gen-RecSys creates a fundamental misalignment between the SID tokenizer and the downstream recommender, as the objectives in the tokenization stage are often not directly optimized for the final recommendation goal. Despite various joint or end-to-end training strategies [46], [50], [186] being proposed as summarized in §V-B, the effective optimization of multiple tasks/objectives for better recommendations remains a key direction for future research.

3) **Next SID Generation:** Real-time inference of next SIDs is still a primary challenge for industrial deployment of the SID-based Gen-RecSys. In addition to the decoding acceleration techniques in §VI, there are some other promising directions. The first one is to explore the order-agnostic token representations or new designs for SIDs so as to facilitate parallel generation. Another way is new model paradigms for recommenders, including mixture-of-experts [187], decoder-only architectures [86], and diffusion frameworks [73], [188], [113], particularly for ultra-long behavior sequences in real scenarios.

C. Item IDs in Future Gen-RecSys

As compared in §III, contemporary item representations (textual or multi-facet IDs) used in LLM-as-Gen-RecSys and SIDs used in SID-based Gen-RecSys offer different benefits and drawbacks. The former are inherently interpretable, easily extendable, and facilitate the recommender implementation through fine-tuning and integration with external agents, while the latter can integrate heterogeneous information sources but cannot directly benefit from the model's pre-trained knowledge.

Given this dichotomy, the ideal item identification should synthesize the strengths of both approaches and meet the quality criteria in Table 3. It should be capable of leveraging the rich knowledge of pre-trained LLMs, remain versatile for integration with external information like manual tags or agents, and uniquely represent items. Crucially, it must balance semantic richness and contextual signals with computational efficiency for both training and inference. A promising direction lies in the principled integration of textual and semantic representations. This could be achieved by establishing a "bridge" between the item's semantic space (represented by the efficient SID) and the textual space (represented by descriptive attributes or tags). Such a hybrid representation would ground the abstract semantic IDs in human-readable text, unlocking the LLM's reasoning capabilities while retaining the efficiency required for modern recommendation scenarios.

VIII. Conclusion

This survey reviewed the critical role of item IDs in Gen-RecSys, proposing a taxonomy that distinguishes LLM-as-Gen-RecSys from SID-based Gen-RecSys. Our analysis highlights that while traditional textual and numeric identifiers offer interpretability, SIDs, constructed via advanced tokenization, provide superior generalization and efficiency, despite challenges like codebook collapse. We also examined key processes for aligning IDs with user behavior and accelerating generation. Future progress depends on addressing ID collisions, developing end-to-end training frameworks, and establishing rigorous benchmarking for tokenization quality. We hope this work serves as a foundational reference for advancing item representation in the next generation of recommender systems.

References

[1] L. Zhang, K. Song, Y. Q. Lee, W. Guo, H. Wang, Y. Li, H. Guo, Y. Liu, D. Lian, and E. Chen, "Killing two birds with one stone: Unifying retrieval and ranking with a single generative recommendation model," in *SIGIR*, 2025, pp. 2224–2234.

[2] J. Deng, S. Wang, K. Cai, L. Ren, and Q. Hu, "Onerec: Unifying retrieve and rank with generative recommender and iterative preference alignment," Feb. 2025.

[3] S. Rajput, N. Mehta, A. Singh, R. Hulikal Keshavan, and T. Vu, "Recommender systems with generative retrieval," in *NeurIPS*, vol. 36. Curran Associates, Inc., 2023, pp. 10 299–10 315.

[4] Y. Hou, J. Ni, Z. He, N. Sachdeva, W.-C. Kang, E. H. Chi, J. McAuley, and D. Z. Cheng, "Actionpiece: Contextually tokenizing action sequences for generative recommendation," in *ICML*.

[5] L. Li, Y. Zhang, D. Liu, and L. Chen, "Large language models for generative recommendation: A survey and visionary discussions," in *LREC-COLING*, 2024, pp. 10 146–10 159.

[6] J. Liu, L. Collins, J. Tang, T. Zhao, N. Shah, and C. M. Ju, "Understanding generative recommendation with semantic ids from a model-scaling view," *arXiv preprint arXiv:2509.25522*, 2025.

[7] M. Hou, L. Wu, Y. Liao, Y. Yang, Z. Zhang, C. Zheng, H. Wu, and R. Hong, "A survey on generative recommendation: Data, model, and tasks," *arXiv preprint arXiv:2510.27157*, 2025.

[8] J. Huang and K. C.-C. Chang, "Towards reasoning in large language models: A survey," in *ACL Findings*, 2023, pp. 1049–1065.

[9] Z. Li, J. Ji, Y. Ge, W. Hua, and Y. Zhang, "Pap-rec: Personalized automatic prompt for recommendation language model," *arXiv preprint arXiv:2402.00284*, 2024.

[10] S. Geng, S. Liu, Z. Fu, Y. Ge, and Y. Zhang, "Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5)," in *RecSys*, 2022, pp. 299–315.

[11] A. Singh, T. Vu, N. Mehta, R. Keshavan, and M. Sathiamoorthy, "Better generalization with semantic ids: A case study in ranking for recommendations," in *RecSys*, 2024, pp. 1039–1044.

[12] X. Li, C. Chen, X. Zhao, Y. Zhang, and C. Xing, "E4srec: An elegant effective efficient extensible solution of large language models for sequential recommendation," Dec. 2023.

[13] S. Kim, H. Kang, S. Choi, D. Kim, M. Yang, and C. Park, "Large language models meet collaborative filtering: An efficient all-round llm-based recommender system," in *KDD*, 2024, pp. 1395–1406.

[14] W. Hua, S. Xu, Y. Ge, and Y. Zhang, "How to index item ids for recommendation foundation models," in *SIGIR*, 2023, pp. 195–204.

[15] K. Bao, J. Zhang, W. Wang, Y. Zhang, and Z. Yang, "A bi-step grounding paradigm for large language models in recommendation systems," *TORS*, vol. 3, no. 4, pp. 1–27, 2025.

[16] Y. Hou, J. Zhang, Z. Lin, H. Lu, R. Xie, J. McAuley, and W. X. Zhao, "Large language models are zero-shot rankers for recommender systems," in *ECIR*. Springer, 2024, pp. 364–381.

[17] J. Ji, Z. Li, S. Xu, W. Hua, Y. Ge, J. Tan, and Y. Zhang, "Genrec: Large language model for generative recommendation," in *ECIR*. Springer, 2024, pp. 494–502.

[18] J. Liao, S. Li, Z. Yang, J. Wu, and Y. Yuan, "Llara: Large language-recommendation assistant," in *SIGIR*, 2024, pp. 1785–1795.

[19] F. Yang, Z. Chen, Z. Jiang, E. Cho, X. Huang, and Y. Lu, "Palr: Personalization aware llms for recommendation," *arXiv preprint arXiv:2305.07622*, 2023.

[20] J. Zhang, R. Xie, Y. Hou, X. Zhao, and L. Lin, "Recommendation as instruction following: A large language model empowered recommendation approach," *TOIS*, vol. 43, no. 5, pp. 1–37, 2025.

[21] Z. Li, Y. Chen, X. Zhang, and X. Liang, "Bookgpt: A general framework for book recommendation empowered by large language model," *Electronics*, vol. 12, no. 22, p. 4654, 2023.

[22] X. Li, Y. Zhang, and E. C. Malthouse, "Pbnr: Prompt-based news recommender system," *arXiv preprint arXiv:2304.07862*, 2023.

[23] ——, "A preliminary study of chatgpt on news recommendation: Personalization, provider fairness, fake news," *arXiv preprint arXiv:2306.10702*, 2023.

[24] Y. Gao, T. Sheng, Y. Xiang, Y. Xiong, and H. Wang, "Chat-rec: Towards interactive and explainable llms-augmented recommender system," Apr. 2023.

[25] P. Li, M. de Rijke, H. Xue, S. Ao, Y. Song, and F. D. Salim, "Large language models for next point-of-interest recommendation," in *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2024, pp. 1463–1472.

[26] D. Wang, Y. Huang, S. Gao, Y. Wang, and C. Huang, "Generative next poi recommendation with semantic id," in *KDD*, 2025, pp. 2904–2914.

[27] X. Lin, W. Wang, Y. Li, F. Feng, and S.-K. Ng, "Bridging items and language: A transition paradigm for large language model-based recommendation," in *KDD*, 2024, pp. 1816–1826.

[28] Y. Zhang, F. Feng, J. Zhang, K. Bao, and Q. Wang, "Collm: Integrating collaborative embeddings into large language models for recommendation," *TKDE*, vol. 37, no. 5, pp. 2329–2340, May 2025.

[29] X. Yu, L. Zhang, X. Zhao, and Y. Wang, "Break the id-language barrier: An adaption framework for sequential recommendation," *arXiv preprint arXiv:2411.18262*, 2024.

[30] J. Jiang, Y. Huang, B. Liu, X. Kong, Z. Xu, H. Zhu, J. Xu, and B. Zheng, "Large language models are universal recommendation learners," *arXiv preprint arXiv:2502.03041*, 2025.

[31] Y. Zhu, L. Wu, Q. Guo, L. Hong, and J. Li, "Collaborative large language model for recommender systems," in *TheWebConf*, 2024, pp. 3162–3172.

[32] Y. Hou, Z. He, J. McAuley, and W. X. Zhao, "Learning vector-quantized item representation for transferable sequential recommenders," in *TheWebConf*, 2023, pp. 1162–1171.

[33] Y. Wang, Z. Ren, W. Sun, J. Yang, and Z. Liang, "Content-based collaborative generation for recommender systems," in *CIKM*, 2024, pp. 2420–2430.

[34] Y. Wang, J. Xun, M. Hong, J. Zhu, and T. Jin, "Eager: Two-stream generative recommender with behavior-semantic collaboration," in *KDD*. Barcelona Spain: ACM, Aug. 2024, pp. 3245–3254.

[35] H. Liu, Y. Wei, X. Song, W. Guan, and Y.-F. Li, "Mmgrec: Multimodal generative recommendation with transformer model," Apr. 2024.

[36] W. Wang, H. Bao, X. Lin, J. Zhang, and Y. Li, "Learnable item tokenization for generative recommendation," in *CIKM*. Boise ID USA: ACM, Oct. 2024, pp. 2400–2409.

[37] B. Zheng, H. Lu, Y. Chen, W. X. Zhao, and J.-R. Wen, "Universal item tokenization for transferable generative recommendation," 2025.

[38] Y. Hou, J. Li, A. Shin, J. Jeon, and A. Santhanam, "Generating long semantic ids in parallel for recommendation," in *KDD*, 2025, pp. 956–966.

[39] K. Fu, T. Zhang, S. Xiao, Z. Wang, and X. Zhang, "Forge: Forming semantic identifiers for generative retrieval in industrial datasets," Sep. 2025.

[40] R. He, L. Heldt, L. Hong, R. Keshavan, and S. Mao. (2025) Plum: Adapting pre-trained language models for industrial-scale generative recommendations.

[41] Z. Si, Z. Sun, J. Chen, G. Chen, X. Zang, K. Zheng, Y. Song, X. Zhang, J. Xu, and K. Gai, "Generative retrieval with semantic tree-structured item identifiers via contrastive learning," *arXiv preprint arXiv:2309.13375*, 2023.

[42] J. Zhu, M. Jin, Q. Liu, Z. Qiu, and Z. Dong, "Cost: Contrastive quantization based semantic tokenization for generative recommendation," in *RecSys*, 2024, pp. 969–974.

[43] Z. Liu, Y. Hou, and J. McAuley, "Multi-behavior generative recommendation," in *CIKM*, 2024, pp. 1575–1585.

[44] B. Jin, H. Zeng, G. Wang, X. Chen, T. Wei, R. Li, Z. Wang, Z. Li, Y. Li, H. Lu *et al.*, "Language models as semantic indexers," in *ICML*, 2024, pp. 22 244–22 259.

[45] Q. Liu, H. Hu, J. Wu, J. Zhu, and M.-Y. Kan, "Discrete semantic tokenization for deep ctr prediction," in *TheWebConf*, 2024, pp. 919–922.

[46] E. Liu, B. Zheng, C. Ling, L. Hu, and H. Li, "Generative recommender with end-to-end learnable item tokenization," in *SIGIR*, 2025, pp. 729–739.

[47] Y. Yang, Z. Ji, Z. Li, Y. Li, and Z. Mo, "Sparse meets dense: Unified generative recommendations with cascaded sparse-dense representations," Mar. 2025.

[48] K. Li, R. Xiang, Y. Bai, Y. Tang, and Y. Cheng, "Bbqrec: Behavior-bind quantization for multi-modal sequential recommendation," Apr. 2025.

[49] C. Liu, Y. Bai, X. Zhao, Y. Zhang, and F. Feng, "Discrec: Disentangled semantic-collaborative modeling for generative recommendation," Jun. 2025.

[50] Y. Xu, M. Zhang, C. Li, Z. Liao, and H. Xing, "Mmq: Multimodal mixture-of-quantization tokenization for semantic id generation and user behavioral adaptation," Aug. 2025.

[51] Y. Luo, Y. Jiang, G. Chen, J. Wang, and S. Wang, "Representation quantization for collaborative filtering augmentation," Aug. 2025.

[52] D. Fang, J. Gao, C. Zhu, Y. Li, and X. Zhao, "Hid-vae: Interpretable generative recommendation via hierarchical and disentangled semantic ids," Sep. 2025.

[53] Z. Wei, K. Cai, J. She, J. Chen, and M. Chen, "Oneloc: Geo-aware generative recommender systems for local life service," Aug. 2025.

[54] B. Tan, W. Ge, Y. Wang, X. Liu, and J. Burtoft, "Pcr-ca: Parallel codebook representations with contrastive alignment for multiple-category app recommendation," Sep. 2025.

[55] S. Lepage, J. Mary, and D. Picard, "Closing the performance gap in generative recommenders with collaborative tokenization and efficient modeling," 2025.

[56] S. Wang, T. Ouyang, Q. Xiao, D. Wang, Y. Ren, S. Xu, D. Guo, and C. Luo, "Progressive semantic residual quantization for multimodal-joint interest modeling in music recommendation," in *CIKM*, 2025, pp. 6119–6127.

[57] B. Chen, X. Guo, S. Wang, Z. Liang, Y. Lv, Y. Ma, X. Xiao, B. Xue, X. Zhang, Y. Yang *et al.*, "Onesearch: A preliminary exploration of the unified end-to-end generative framework for e-commerce search," *arXiv preprint arXiv:2509.03236*, 2025.

[58] Y. Xu, M. Zhang, C. Fan, J. Hu, and X. Li. (2025) Mmq-v2: Align, denoise, and amplify: Adaptive behavior mining for semantic ids learning in recommendation.

[59] Z. Liu, Y. Wang, Q. Liu, Z. Zhang, C. Chen, W. Huang, and X. Zhao, "The best of the two worlds: Harmonizing semantic and hash ids for sequential recommendation," *arXiv preprint arXiv:2512.10388*, 2025.

[60] J. Tang, C. Wang, G. Yang, H. Wu, J. Yu, J. Wu, J. Hu, J. Zheng, L. Li, S. Xiao, X. Kong, Y. Yang, Y. Jiang, A. Nurlanbek, B. Cao, B. Zheng, F. Zhu, G. Zhou, H. Yi, H. Chu, J. Huang, J. Shan, K. Cui, L. Li, S. Zhou, W. Chen, X. Ming, X. Gao, X. Yao, X. Wen, Y. Zhang, Y. Hu, Y. Wang, Z. Bao, and Z. Wu, "Reaseq: Unleashing world knowledge via reasoning for sequential modeling," 2025.

[61] Y. Pang, Z. Liu, Y. Li, S. Zhu, Z. Luo, C. Yu, S. Wu, S. Shen, C. Xu, B. Wang, K. Jiang, H. Yu, C. Zhuo, and Z. Li, "Higr: Efficient generative slate recommendation via hierarchical planning and multi-objective preference alignment," 2025.

[62] A. V. Petrov and C. Macdonald, "Generative sequential recommendation with gptrec," 2023.

[63] J. Tan, S. Xu, W. Hua, Y. Ge, Z. Li, and Y. Zhang, "Idgenrec: Llm-recsys alignment with textual id learning," in *SIGIR*, 2024, pp. 355–364.

[64] X. Lin, H. Shi, W. Wang, F. Feng, and Q. Wang, "Order-agnostic identifier for large language model-based generative recommendation," in *SIGIR*, 2025, pp. 1923–1933.

[65] H. Qu, W. Fan, Z. Zhao, and Q. Li, "Tokenrec: Learning to tokenize id for llm-based generative recommendations," *TKDE*, 2025.

[66] Y. Yao, Z. Li, S. Xiao, B. Du, and J. Zhu, "Saviorrec: Semantic-behavior alignment for cold-start recommendation," Aug. 2025.

[67] X. Kong, L. Sheng, J. Tan, Y. Chen, and J. Wu. (2025) Minionerec: An open-source framework for scaling generative recommendation.

[68] D. Sileo, W. Vossen, and R. Raymaekers, "Zero-shot recommendation as language modeling," in *ECIR*, 2022, pp. 223–230.

[69] S. Dai, N. Shao, H. Zhao, W. Yu, Z. Si, C. Xu, Z. Sun, X. Zhang, and J. Xu, "Uncovering chatgpt's capabilities in recommender systems," in *RecSys*, 2023, pp. 1126–1132.

[70] Z. Chu, H. Hao, X. Ouyang, S. Wang, Y. Wang, Y. Shen, J. Gu, Q. Cui, L. Li, S. Xue *et al.*, "Leveraging large language models for pre-trained recommender systems," *arXiv preprint arXiv:2308.10837*, 2023.

[71] X. Lin, C. Yang, W. Wang, Y. Li, and C. Du, "Efficient inference for large language model-based generative recommendation," in *ICLR*, 2025.

[72] Y. Wang, S. Zhou, J. Lu, Z. Liu, L. Liu, M. Wang, W. Zhang, F. Li, W. Su, P. Wang *et al.*, "Nezha: A zero-sacrifice and hyperspeed decoding architecture for generative recommendations," *arXiv preprint arXiv:2511.18793*, 2025.

[73] T. Shi, C. Shen, W. Yu, S. Nie, and C. Li. (2025) Llada-rec: Discrete diffusion for parallel semantic id generation in generative recommendation.

[74] Y. Huang, Y. Chen, X. Cao, R. Yang, and M. Qi, "Towards large-scale generative ranking," May 2025.

[75] J. Li, J. Xu, S. Huang, Y. Chen, W. Li, J. Liu, Y. Lian, J. Pan, L. Ding, H. Zhou *et al.*, "Large language model inference acceleration: A comprehensive hardware perspective," *arXiv preprint arXiv:2410.04466*, 2024.

[76] Y. Li, X. Lin, W. Wang, F. Feng, L. Pang, W. Li, L. Nie, X. He, and T.-S. Chua, "A survey of generative search and recommendation in the era of large language models," *arXiv preprint arXiv:2404.16924*, 2024.

[77] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua, "Neural collaborative filtering," in *WWW*, 2017, pp. 173–182.

[78] Y. Hou, A. Zhang, L. Sheng, J. Wu, X. Wang, T.-S. Chua, and J. McAuley, "Towards large generative recommendation: A tokenization perspective," in *CIKM*, 2025, pp. 6821–6824.

[79] K. Bao, J. Zhang, Y. Zhang, W. Wang, and F. Feng, "Tallrec: An effective and efficient tuning framework to align large language model with recommendation," in *RecSys*, 2023, pp. 1007–1014.

[80] Y. Chen, J. Tan, A. Zhang, Z. Yang, and L. Sheng, "On softmax direct preference optimization for recommendation," in *NeurIPS*, 2025.

[81] J. Lin, T. Wang, and K. Qian, "Rec-r1: Bridging generative large language models and user-centric recommendation systems via reinforcement learning," May 2025.

[82] J. Tan, Y. Chen, A. Zhang, J. Jiang, B. Liu, Z. Xu, H. Zhu, J. Xu, B. Zheng, and X. Wang, "Reinforced preference optimization for recommendation," *arXiv preprint arXiv:2510.12211*, 2025.

[83] B. Zheng, Y. Hou, H. Lu, Y. Chen, W. X. Zhao, M. Chen, and J.-R. Wen, "Adapting large language models by integrating collaborative semantics for recommendation," in *ICDE*, 2024, pp. 1435–1448.

[84] S. Geng, J. Tan, S. Liu, Z. Fu, and Y. Zhang, "Vip5: Towards multimodal foundation models for recommendation," in *ACL Findings*, 2023, pp. 9606–9620.

[85] G. Lin, Z. Hua, T. Feng, S. Yang, and B. Long, "Unified semantic and id representation learning for deep recommenders," Feb. 2025.

[86] G. Zhou, H. Hu, H. Cheng, H. Wang, and J. Deng, "Onerec-v2 technical report," Sep. 2025.

[87] X. Luo, J. Cao, T. Sun, J. Yu, R. Huang, W. Yuan, H. Lin, Y. Zheng, S. Wang, Q. Hu *et al.*, "Qarm: Quantitative alignment multi-modal recommendation at kuaishou," in *CIKM*, 2025, pp. 5915–5922.

[88] L. Ma, W. Zhang, K. Zhao, A. Kulkarni, L. Morishetti, A. Ganesh, A. Ranjan, A. Padmanabhan, J. Xu, J. H. Cho *et al.*, "Grace: Generative recommendation via journey-aware sparse attention on chain-of-thought tokenization," in *RecSys*, 2025, pp. 135–144.

[89] M. J. Mei, F. Henkel, S. E. Sandberg, O. Bembom, and A. F. Ehmann, "Semantic ids for music recommendation," in *RecSys*, 2025, pp. 1070–1073.

[90] S. Xu, S. Wang, D. Guo, X. Guo, Q. Xiao, B. Huang, G. Wu, and C. Luo, "Climber: Toward efficient scaling laws for large recommendation models," in *CIKM*, 2025, pp. 6193–6200.

[91] H. Jiang, G. Wang, D. Zhou, S. Yu, and Y. Zeng, "Llm-aligned geographic item tokenization for local-life recommendation," Nov. 2025.

[92] Y. Xu, C. Fan, J. Hu, Y. Zhang, Z. Xiaoyi, and J. Zhang, "Store: Semantic tokenization, orthogonal rotation and efficient attention for scaling up ranking models," *arXiv preprint arXiv:2511.18805*, 2025.

[93] Q. Zhong, J. Su, Y. Ma, J. McAuley, and Y. Hou, "Pctx: Tokenizing personalized context for generative recommendation," *arXiv preprint arXiv:2510.21276*, 2025.

[94] E. Palumbo, G. Penha, A. Damianou, J. L. R. García, and T. C. Heath, "Text2tracks: Prompt-based music recommendation via generative retrieval," Apr. 2025.

[95] J. Lin, X. Dai, Y. Xi, W. Liu, B. Chen, H. Zhang, Y. Liu, C. Wu, X. Li, C. Zhu *et al.*, "How can recommender systems benefit from large language models: A survey," *TOIS*, vol. 43, no. 2, pp. 1–47, 2025.

[96] L. Wu, Z. Zheng, Z. Qiu, H. Wang, H. Gu, T. Shen, C. Qin, C. Zhu, H. Zhu, Q. Liu *et al.*, "A survey on large language models for recommendation," *WWW*, vol. 27, no. 5, p. 60, 2024.

[97] Z. Zhao, W. Fan, J. Li, Y. Liu, X. Mei, Y. Wang, Z. Wen, F. Wang, X. Zhao, J. Tang *et al.*, "Recommender systems in the era of large language models (llms)," *TKDE*, vol. 36, no. 11, pp. 6889–6907, 2024.

[98] Q. Wang, J. Li, S. Wang, Q. Xing, R. Niu, H. Kong, R. Li, G. Long, Y. Chang, and C. Zhang, "Towards next-generation llm-based recommender systems: A survey and beyond," *arXiv preprint arXiv:2410.19744*, 2024.

[99] Y. Deldjoo, Z. He, J. McAuley, A. Korikov, and S. Sanner, "A review of modern recommender systems using generative models (gen-recsys)," in *KDD*. Barcelona Spain: ACM, Aug. 2024, pp. 6448–6458.

[100] Q. Liu, X. Zhao, Y. Wang, Y. Wang, Z. Zhang, Y. Sun, X. Li, M. Wang, P. Jia, C. Chen *et al.*, "Large language model enhanced recommender systems: Taxonomy, trend, application and future," *arXiv e-prints*, pp. arXiv–2412, 2024.

[101] Z. Yang, H. Lin, Z. Zhang *et al.*, "Gr-llms: Recent advances in generative recommendation based on large language models," *arXiv preprint arXiv:2507.06507*, 2025.

[102] Y. Zhao, C. Tan, L. Shi, Y. Zhong, F. Kou, P. Zhang, W. Chen, and C. Ma, "Generative recommender systems: A comprehensive survey on model, framework, and application," *Information Fusion*, p. 103919, 2025.

[103] X. Li, B. Chen, J. She, S. Cao, Y. Wang, Q. Jia, H. He, Z. Zhou, Z. Liu, J. Liu, Z. Zhang, Y. Zhou, G. Tang, Y. Yang, C. Guo, S. Dong, K. Cai, P. Jia, M. Wang, and X. Zhao, "A survey of generative recommendation from a tri-decoupled perspective: Tokenization, architecture, and optimization," 11 2025.

[104] J. Li, W. Zhang, T. Wang, G. Xiong, A. Lu, and G. Medioni, "Gpt4rec: A generative framework for personalized recommendation and user interests interpretation," in *eCom@ SIGIR*, 2023.

[105] W. Wei, X. Ren, J. Tang, Q. Wang, L. Su, S. Cheng, J. Wang, D. Yin, and C. Huang, "Llmrec: Large language models with graph augmentation for recommendation," in *WSDM*, 2024, pp. 806–815.

[106] Q. Dong, L. Li, D. Dai, C. Zheng, J. Ma, R. Li, H. Xia, J. Xu, Z. Wu, B. Chang *et al.*, "A survey on in-context learning," in *EMNLP*, 2024, pp. 1107–1128.

[107] L. Wang and E.-P. Lim, "Zero-shot next-item recommendation using large pretrained language models," Apr. 2023.

[108] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, W. Chen *et al.*, "Lora: Low-rank adaptation of large language models." *ICLR*, vol. 1, no. 2, p. 3, 2022.

[109] X. Lin, W. Wang, Y. Li, S. Yang, and F. Feng, "Data-efficient fine-tuning for llm-based recommendation," in *SIGIR*. Washington DC USA: ACM, Jul. 2024, pp. 365–374.

[110] R. Rafailov, A. Sharma, E. Mitchell, C. D. Manning, and S. Ermon, "Direct preference optimization: Your language model is secretly a reward model," *NeurIPS*, vol. 36, pp. 53 728–53 741, 2023.

[111] K. Kim, S. Kim, H. Kang, J. Kim, H. Noh, Y. In, K. Yoon, J. Oh, and C. Park, "Image is all you need: Towards efficient and effective large language model-based recommender systems," *arXiv preprint arXiv:2503.06238*, 2025.

[112] W. Ye, M. Sun, S. Chen, W. Wu, and P. Jiang, "Align^{3}gr: Unified multi-level alignment for llm-based generative recommendation," *arXiv preprint arXiv:2511.11255*, 2025.

[113] H. Qu, W. Fan, and S. Lin, "Generative recommendation with continuous-token diffusion," Apr. 2025.

[114] M. Hong, Y. Xia, Z. Wang, J. Zhu, Y. Wang, S. Cai, X. Yang, Q. Dai, Z. Dong, Z. Zhang, and Z. Zhao, "Eager-llm: Enhancing large language models as recommenders through exogenous behavior-semantic integration," in *TheWebConf*, ser. WWW '25, 2025, p. 2754–2762.

[115] Y. Wang, J. Xun, M. Hong, J. Zhu, T. Jin, W. Lin, H. Li, L. Li, Y. Xia, Z. Zhao *et al.*, "Eager: Two-stream generative recommender with behavior-semantic collaboration," in *KDD*, 2024, pp. 3245–3254.

[116] B. Zheng, E. Liu, Z. Chen, Z. Ma, and Y. Wang, "Pre-training generative recommender with multi-identifier item tokenization," May 2025.

[117] E. Lu, Z. Jiang, J. Liu, Y. Du, T. Jiang, C. Hong, S. Liu, W. He, E. Yuan, Y. Wang *et al.*, "Moba: Mixture of block attention for long-context llms," *arXiv preprint arXiv:2502.13189*, 2025.

[118] Z. Liu, S. Wang, X. Wang, R. Zhang, and J. Deng, "Onerec-think: In-text reasoning for generative recommendation," Oct. 2025.

[119] Z. Shao, P. Wang, Q. Zhu, R. Xu, and J. Song. (2024) Deepseekmath: Pushing the limits of mathematical reasoning in open language models.

[120] Y. Wang, Y. Hou, H. Wang, Z. Miao, S. Wu, Q. Chen, Y. Xia, C. Chi, G. Zhao, Z. Liu *et al.*, "A neural corpus indexer for document retrieval," *NeurIPS*, vol. 35, pp. 25 600–25 614, 2022.

[121] U. Gupta, C.-J. Wu, X. Wang, M. Naumov, B. Reagen, D. Brooks, B. Cottel, K. Hazelwood, M. Hempstead, B. Jia *et al.*, "The architectural implications of facebook's dnn-based personalized recommendation," in *HPCA*. IEEE, 2020, pp. 488–501.

[122] Z. Xu, D. Li, W. Zhao, X. Shen, T. Huang, X. Li, and P. Li, "Agile and accurate ctr prediction model training for massive-scale online advertising systems," in *SIGMOD*, 2021, pp. 2404–2409.

[123] N. Zeghidour, A. Luebs, A. Omran, J. Skoglund, and M. Tagliasacchi, "Soundstream: An end-to-end neural audio codec," *TASLPRO*, vol. 30, pp. 495–507, 2021.

[124] D. Lee, C. Kim, S. Kim, M. Cho, and W.-S. Han, "Autoregressive image generation using residual quantization," in *CVPR*, 2022, pp. 11 513–11 522.

[125] W. Ye, M. Sun, S. Shi, P. Wang, W. Wu, and P. Jiang, "Das: Dual-aligned semantic ids empowered industrial recommender system," in *CIKM*, 2025, pp. 6217–6224.

[126] Z. Yuan, F. Yuan, Y. Song, Y. Li, J. Fu, F. Yang, Y. Pan, and Y. Ni, "Where to go next for recommender systems? id-vs. modality-based recommender models revisited," in *SIGIR*, 2023, pp. 2639–2649.

[127] A. Dosovitskiy, "An image is worth 16x16 words: Transformers for image recognition at scale," *arXiv preprint arXiv:2010.11929*, 2020.

[128] A. Baevski, Y. Zhou, A. Mohamed, and M. Auli, "wav2vec 2.0: A framework for self-supervised learning of speech representations," *NeurIPS*, vol. 33, pp. 12 449–12 460, 2020.

[129] A. Radford, J. W. Kim, C. Hallacy, A. Ramesh, G. Goh, S. Agarwal, G. Sastry, A. Askell, P. Mishkin, J. Clark *et al.*, "Learning transferable visual models from natural language supervision," in *ICML*, 2021, pp. 8748–8763.

[130] H. Liu, C. Li, Q. Wu, and Y. J. Lee, "Visual instruction tuning," *NeurIPS*, vol. 36, pp. 34 892–34 916, 2023.

[131] Z. Yi, Z. Long, I. Ounis, C. Macdonald, and R. Mccreadie, "Large multi-modal encoders for recommendation," *arXiv preprint arXiv:2310.20343*, 2023.

[132] J. Huang, Z. Novack, P. Long, Y. Hou, K. Chen, T. Berg-Kirkpatrick, and J. McAuley, "Musetok: Symbolic music tokenization for generation and semantic understanding," *arXiv preprint arXiv:2510.16273*, 2025.

[133] H. Wang, J. Wang, H. Li, F. Yi, and M. Fu, "Serendipitous recommendation with multimodal llm," Sep. 2025.

[134] W.-C. Kang and J. McAuley, "Self-attentive sequential recommendation," in *ICDM*, 2018, pp. 197–206.

[135] X. He, K. Deng, X. Wang, Y. Li, Y. Zhang, and M. Wang, "Lightgcn: Simplifying and powering graph convolution network for recommendation," in *SIGIR*, 2020, pp. 639–648.

[136] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *KDD*, 2018, pp. 1059–1068.

[137] P. S. Bradley, K. P. Bennett, and A. Demiriz, "Constrained k-means clustering," *Microsoft Research, Redmond*, vol. 20, no. 0, p. 0, 2000.

[138] J. Li, Y. Fu, J. Liu, L. Cao, W. Ji, M. Yang, I. King, and M.-H. Yang, "Discrete tokenization for multimodal llms: A comprehensive survey," *arXiv preprint arXiv:2507.22920*, 2025.

[139] A. Van Den Oord, O. Vinyals *et al.*, "Neural discrete representation learning," *NeurIPS*, vol. 30, 2017.

[140] A. Razavi, A. Van den Oord, and O. Vinyals, "Generating diverse high-fidelity images with vq-vae-2," *NeurIPS*, vol. 32, 2019.

[141] Y. Bengio, N. Léonard, and A. Courville, "Estimating or propagating gradients through stochastic neurons for conditional computation," *arXiv preprint arXiv:1308.3432*, 2013.

[142] Y. Zhu, B. Li, Y. Xin, Z. Xia, and L. Xu, "Addressing representation collapse in vector quantized models with one linear layer," in *ICCV*, 2025, pp. 22 968–22 977.

[143] T. Ge, K. He, Q. Ke, and J. Sun, "Optimized product quantization," *TPAMI*, vol. 36, no. 4, pp. 744–755, 2013.

[144] W. Zhao, Q. Zou, R. Shah, and D. Liu, "Representation collapsing problems in vector quantization," *arXiv preprint arXiv:2411.16550*, 2024.

[145] F. Mentzer, D. Minnen, E. Agustsson, and M. Tschannen, "Finite scalar quantization: Vq-vae made simple," *arXiv preprint arXiv:2309.15505*, 2023.

[146] M. H. Vali, T. Bäckström, and A. Solin, "Diveq: Differentiable vector quantization using the reparameterization trick," *arXiv preprint arXiv:2509.26469*, 2025.

[147] T. Wei, X. Ning, X. Chen, R. Qiu, Y. Hou, Y. Xie, S. Yang, Z. Hua, and J. He, "Cofirec: Coarse-to-fine tokenization for generative recommendation," *arXiv preprint arXiv:2511.22707*, 2025.

[148] Z. Kuai, Z. Chen, H. Wang, M. Li, and D. Miao, "Breaking the hourglass phenomenon of residual quantization: Enhancing the upper bound of generative retrieval," in *EMNLP*. Miami, Florida, US: Association for Computational Linguistics, 2024, pp. 677–685.

[149] C. M. Ju, L. Collins, L. Neves, B. Kumar, L. Y. Wang, T. Zhao, and N. Shah, "Generative recommendation with semantic ids: A practitioner's handbook," in *CIKM*, 2025, pp. 6420–6425.

[150] L. Yu, J. Lezama, N. B. Gundavarapu, L. Versari, K. Sohn, D. Minnen, Y. Cheng, A. Gupta, X. Gu, A. G. Hauptmann *et al.*, "Language model beats diffusion-tokenizer is key to visual generation," in *ICLR*, 2024.

[151] G. Penha, E. D'Amico, M. De Nadai, E. Palumbo, and A. Tamborrino, "Semantic ids for joint generative search and recommendation," in *RecSys*, 2025, pp. 1296–1301.

[152] Y. Zhao, Y. Xiong, and P. Krähenbühl, "Image and video tokenization with binary spherical quantization," *arXiv preprint arXiv:2406.07548*, 2024.

[153] H. Jegou, M. Douze, and C. Schmid, "Product quantization for nearest neighbor search," *TPAMI*, vol. 33, no. 1, pp. 117–128, 2010.

[154] T. Kudo and J. Richardson, "Sentencepiece: A simple and language independent subword tokenizer and detokenizer for neural text processing," in *EMNLP*, 2018, pp. 66–71.

[155] A. Roy, A. Vaswani, A. Neelakantan, and N. Parmar, "Theory and experiments on vector quantized autoencoders," *arXiv preprint arXiv:1805.11063*, 2018.

[156] Y. Takida, T. Shibuya, W. Liao, C.-H. Lai, J. Ohmura, T. Uesaka, N. Murata, S. Takahashi, T. Kumakura, and Y. Mitsufuji, "Sq-vae: Variational bayes on discrete representation with self-annealed stochastic quantization," in *ICML*, 2022, pp. 20 987–21 012.

[157] T.-L. Vuong, T. Le, H. Zhao, C. Zheng, M. Harandi, J. Cai, and D. Phung, "Vector quantized wasserstein auto-encoder," in *ICML*, 2023, pp. 35 223–35 242.

[158] C. Zheng and A. Vedaldi, "Online clustered codebook," in *ICCV*, 2023, pp. 22 798–22 807.

[159] R. Zhang, J. Li, J. McAuley, and Y. Hou, "Purely semantic indexing for llm-based generative recommendation and retrieval," *arXiv preprint arXiv:2509.16446*, 2025.
[160] R. Chen, M. Ju, N. Bui, D. Antypas, and S. Cai, "Enhancing item tokenization for generative recommendation through self-improvement," Dec. 2024.
[161] C. Zheng, M. Huang, D. Pedchenko, K. Rangadurai, and S. Wang, "Enhancing embedding representation stability in recommendation systems with semantic id," in *RecSys*, 2025, pp. 954–957.
[162] X. Zhai, B. Mustafa, A. Kolesnikov, and L. Beyer, "Sigmoid loss for language image pre-training," in *ICCV*, 2023, pp. 11 975–11 986.
[163] G. Li, X. Zhang, Y. Zhang, Y. Yin, G. Yin, and W. Lin, "Semantic convergence: Harmonizing recommender systems via two-stage alignment and behavioral semantic tokenization," in *AAAI*, vol. 39, no. 11, 2025, pp. 12 040–12 048.
[164] W. Fan, X. Liu, W. Jin, X. Zhao, J. Tang, and Q. Li, "Graph trend filtering networks for recommendation," in *SIGIR*, 2022, pp. 112–121.
[165] G. Zhou, J. Deng, J. Zhang, K. Cai, L. Ren, Q. Luo, Q. Wang, Q. Hu, R. Huang, S. Wang *et al.*, "Onerec technical report," *arXiv preprint arXiv:2506.13695*, 2025.
[166] T. Shi, J. Xu, X. Zhang, X. Zang, and K. Zheng, "Unified generative search and recommendation," Apr. 2025.
[167] J. Chen, X. Jiang, Z. Wang, Q. Zhu, and J. Zhao, "Unisearch: Rethinking search system with a unified generative architecture," Sep. 2025.
[168] J. Jiang, Y. Huang, B. Liu, X. Kong, and X. Li, "Large language model as universal retriever in industrial-scale recommender system," May 2025.
[169] Y. Leviathan, M. Kalman, and Y. Matias, "Fast inference from transformers via speculative decoding," in *ICML*, 2023, pp. 19 274–19 286.
[170] C. Chen, S. Borgeaud, G. Irving, J.-B. Lespiau, L. Sifre, and J. Jumper, "Accelerating large language model decoding with speculative sampling," *arXiv preprint arXiv:2302.01318*, 2023.
[171] Y. Ding, Y. Hou, J. Li, and J. McAuley, "Inductive generative recommendation via retrieval-based speculation," Oct. 2024.
[172] T. Cai, Y. Li, Z. Geng, H. Peng, J. D. Lee, D. Chen, and T. Dao, "Medusa: Simple llm inference acceleration framework with multiple decoding heads," in *ICML*, 2024, pp. 5209–5235.
[173] J. Chen, L. Chi, S. Xu, S. Ran, and B. Peng, "Hllm-creator: Hierarchical llm-based personalized creative generation," Aug. 2025.
[174] R. Han, B. Yin, S. Chen, H. Jiang, F. Jiang, X. Li, C. Ma, M. Huang, X. Li, C. Jing *et al.*, "Mtgr: Industrial-scale generative recommendation framework in meituan," in *CIKM*, 2025, pp. 5731–5738.
[175] M. Pang, C. Yuan, X. He, Z. Fang, D. Xie, F. Qu, X. Jiang, C. Peng, Z. Lin, Z. Luo *et al.*, "Generative retrieval and alignment model: A new paradigm for e-commerce retrieval," in *TheWebConf*, 2025, pp. 413–421.
[176] P. S. Ow and T. E. Morton, "Filtered beam search in scheduling," *IJPR*, vol. 26, no. 1, pp. 35–62, 1988.
[177] C. Yang, X. Lin, W. Wang, Y. Li, and T. Sun, "Earn: Efficient inference acceleration for llm-based generative recommendation by register tokens," in *KDD*, 2025, pp. 3483–3494.
[178] E. Liu, B. Zheng, W. X. Zhao, and J.-R. Wen, "Bridging textual-collaborative gap through semantic codes for sequential recommendation," in *KDD*, 2025, pp. 1788–1798.
[179] H. Luo, B. Wu, H. Jia, Q. Zhu, and L. Shan, "Llm-cot enhanced graph neural recommendation with harmonized group policy optimization," Oct. 2025.
[180] J. Chen, L. Chi, B. Peng, and Z. Yuan, "Hllm: Enhancing sequential recommendations via hierarchical large language models for item and user modeling," Sep. 2024.
[181] W.-S. Chao, Z. Zheng, H. Zhu, and H. Liu, "Make large language model a better ranker," in *ACL Findings*, 2024, pp. 918–929.
[182] A. Abdallah, B. Piryani, J. Mozafari, M. Ali, and A. Jatowt, "How good are llm-based rerankers? an empirical analysis of state-of-the-art reranking models," in *ACL Findings*, 2025, pp. 5693–5709.
[183] W. Xu, Y. Shi, Z. Liang, X. Ning, K. Mei, K. Wang, X. Zhu, M. Xu, and Y. Zhang, "iagent: Llm agent as a shield between user and recommender systems," *arXiv preprint arXiv:2502.14662*, 2025.
[184] T. Zhang, J. Pan, J. Wang, Y. Zha, and T. Dai, "Towards scalable semantic representation for recommendation," Oct. 2024.
[185] P. L. Williams and R. D. Beer, "Nonnegative decomposition of multivariate information," *arXiv preprint arXiv:1004.2515*, 2010.
[186] F. Shi, Z. Luo, Y. Ge, Y. Yang, Y. Shan, and L. Wang, "Scalable image tokenization with index backpropagation quantization," in *ICCV*, 2025, pp. 16 037–16 046.
[187] X. Miao, G. Oliaro, Z. Zhang, X. Cheng, H. Jin, T. Chen, and Z. Jia, "Towards efficient generative large language model serving: A survey from algorithms to systems," *CSUR*, vol. 58, no. 1, pp. 1–37, 2025.
[188] Z. Liu, Y. Zhu, Y. Yang, G. Tang, R. Huang, Q. Luo, X. Lv, R. Tang, K. Gai, and G. Zhou, "Diffgrm: Diffusion-based generative recommendation model," *arXiv preprint arXiv:2510.21805*, 2025.