# 第 9 章 生成式推荐模块

在上一章节中，我们系统介绍了级联式推荐系统中的混排模块。混排模块从多个业务系统出发，在统一序列空间中完成跨业务内容的融合与排序，实现了从“单业务最优”到“多业务协同优化”的进一步扩展。然而，从系统演进的角度来看，无论是召回、排序还是混排，本质上仍然依赖于“候选生成 + 多阶段筛选”的级联式范式，其核心思想是逐步缩小候选空间并进行局部最优决策。

随着大语言模型与生成式建模方法的快速发展，推荐系统开始出现一种新的研究范式：直接将用户兴趣建模为一个序列生成问题，从而在统一模型中完成候选生成与排序决策的融合。这种范式被称为**生成式推荐（Generative Recommendation）**。与传统级联式架构相比，生成式推荐试图打破多阶段管线的设计方式，将推荐过程统一为基于序列建模的自回归生成问题，从而为推荐系统的统一建模提供新的可能性。

在这一背景下，本章将围绕生成式推荐的基本概念与核心思想展开介绍，并进一步分析其与传统推荐范式之间的关系与差异。

## 9.1 生成式推荐概述

随着生成式模型（Generative Model）以及大语言模型（Large Language Model, LLM）的快速发展，推荐系统领域近年来也开始积极探索将生成式建模思想引入传统推荐架构之中。特别是在 Scaling Law、Transformer 架构以及 Next Token Prediction（NTP）训练范式取得巨大成功之后，越来越多的研究工作开始思考：推荐问题是否也能够像自然语言生成一样，被统一建模为一个序列生成问题。

传统推荐系统通常采用“召回-粗排-精排-重排”的级联式架构，通过多个模块逐步缩小候选集合并提升排序精度。而生成式推荐（Generative Recommendation）的核心思想则是：

**定义 9.1**

将推荐任务直接转化为一个序列生成任务：给定用户历史行为序列、上下文环境以及用户画像等信息，由模型直接生成用户未来最可能感兴趣的内容标识（Item ID），从而完成推荐决策。

然而，与自然语言中的词汇 Token 不同，推荐系统中的 Item 规模往往达到百万甚至亿级别，直接将 Item ID 作为生成目标会导致词表规模过大，训练和推理成本极高。因此，目前主流的生成式推荐工作通常会首先对 Item 进行语义离散化编码（Semantic Tokenization），将每个 Item 映射为一个或多个语义 ID（Semantic ID）。这些语义 ID 通常通过 Residual Quantization（RQ）[13]、VQ-VAE [19]、Product Quantization（PQ）[9] 等向量量化技术从 Item Embedding 中学习得到。例如，一个视频内容可能被编码为：

$$Item_A \rightarrow [s_1, s_2, s_3, s_4] \tag{9.1}$$

，其中 $s_i$ 表示不同层级的语义编码。这样，原本庞大的 Item 空间便被映射到一个相对紧凑的语义 Token 空间之中。

在训练阶段，系统将用户行为序列转换为语义 ID 序列：

$$[s_1, s_2, s_3] \rightarrow [s_4] \tag{9.2}$$

或者：

$$Item_1 \rightarrow Item_2 \rightarrow Item_3 \rightarrow ? \tag{9.3}$$

并采用与大语言模型类似的 Next Token Prediction（NTP）目标进行训练。模型通过学习用户历史消费序列中的模式，预测用户下一步最可能感兴趣的语义 Token。训练完成后，模型便具备了根据用户行为历史直接生成未来兴趣内容的能力。

在线推理阶段，生成式推荐模型接收用户最近浏览的视频、点击行为、停留时长、搜索记录以及上下文特征等信息作为输入，通过自回归生成（Autoregressive Generation）的方式逐步生成语义 ID 序列。例如：

$$[s_1, s_2] \rightarrow [s_3] \rightarrow [s_4] \tag{9.4}$$

。当生成完整语义 ID 后，系统再通过预先构建的语义 ID 到 Item ID 映射表完成检索，从而获得对应的推荐内容集合。

## 9.2 生成式推荐架构

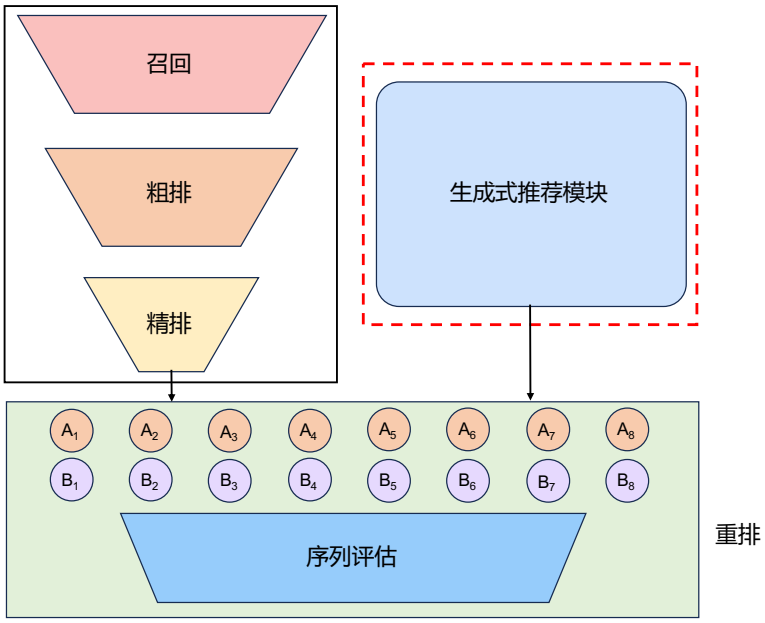


**图 9.1:** 生成式推荐系统架构图。

从系统架构角度来看，生成式推荐模块在工业级推荐系统中的位置如图9.1所示。与传统推荐架构不同，生成式推荐并不严格遵循“召回-粗排-精排”的逐层筛选流程，而是利用大模型的生成能力直接产生候选内容，因此其在整体架构中通常作为传统推荐链路之外的一条独立推荐通路存在。

目前工业界主要存在两种典型的部署方式。第一种方式是**并行候选生成架构（Parallel Candidate Generation Architecture）**。在这种模式下，生成式推荐模块与传统的“召回-粗排-精排”模块并行运行。传统推荐链路负责产生大规模高覆盖率候选集，而生成式推荐模块则利用大模型直接生成一批高质量候选内容。两部分候选集合最终汇合进入重排模块进行统一排序。此时，生成式推荐模块本质上可以被视为一条特殊的召回通路。与传统基于 Embedding 向量召回、图网络召回等召回策略相比，生成式推荐能够直接利用用户行为序列中的长期兴趣和复杂上下文信息生成候选内容，因此通常被认为是一条质量较高的“直通重排召回”。这种传统推荐与生成式推荐并行执行的架构具有较好的工程兼容性。一方面可以充分利用生成式模型的表达能力；另一方面即使生成式模型出现异常，传统推荐链路仍然能够作为兜底方案保证系统正常运行，因此目前许多工业界落地案例都采用这种方式进行部署。

第二种方式是**流量隔离架构（Traffic Isolation Architecture）**。在这种模式下，平台会从整体流量中划分出一定比例的实验流量，例如：$5\% \sim 30\%$ 作为生成式推荐流量。当用户命中生成式推荐实验流量时，请求将直接进入生成式推荐模块，不再执行传统的召回、粗排和精排流程，而是由生成式模型直接生成候选 Item 集合，并送入重排模块完成最终排序。在这种情况下，整个推荐流程可以表示为：

$$User \rightarrow Generative\ Recommendation \rightarrow Reranking \rightarrow Result \tag{9.5}$$

。而对于未命中实验流量的用户，则仍然采用传统级联架构：

$$User \rightarrow Retrieve \rightarrow Coarse\ Ranking \rightarrow Fine\ Ranking \rightarrow Reranking \rightarrow Result \tag{9.6}$$

这种部署方式的优势在于能够更加准确地评估生成式推荐对整体推荐效果的增益情况。通过 AB 实验，平台可以直接比较生成式推荐架构与传统级联架构在 CTR、观看时长、用户留存以及商业化指标上的差异，从而评估生成式推荐的实际业务价值。

需要注意的是，即使在采用生成式推荐架构的场景下，大多数工业系统仍然会保留重排（Re-ranking）模块作为最后一道决策环节。原因在于生成式模型虽然能够生成用户可能感兴趣的内容，但仍然难以同时兼顾广告收益、直播曝光、多样性约束、生态治理以及流量调控等复杂业务目标。因此，重排模块仍然承担着多目标优化与最终决策的重要职责。

从当前 LLM4Rec 研究的发展趋势来看，生成式推荐更像是在传统推荐架构之上增加了一层强大的候选生成能力，而不是完全替代整个推荐链路。未来推荐系统的发展方向，很可能是生成式模型与传统级联架构长期共存，并逐步形成融合式推荐范式（Hybrid Recommendation Architecture），从而同时兼顾模型能力、系统稳定性以及复杂业务目标的优化需求。

在生成式推荐模块内部，通常包含三个核心组成部分：**语义 ID 生成（Semantic ID Construction）**、**生成式推荐模型（Generative Recommendation Model）**以及**物品 ID 检索（Item Retrieval）**，整体架构如图9.2所示。注意图9.2为了简化起见，残差连接在图中并未展示出来。

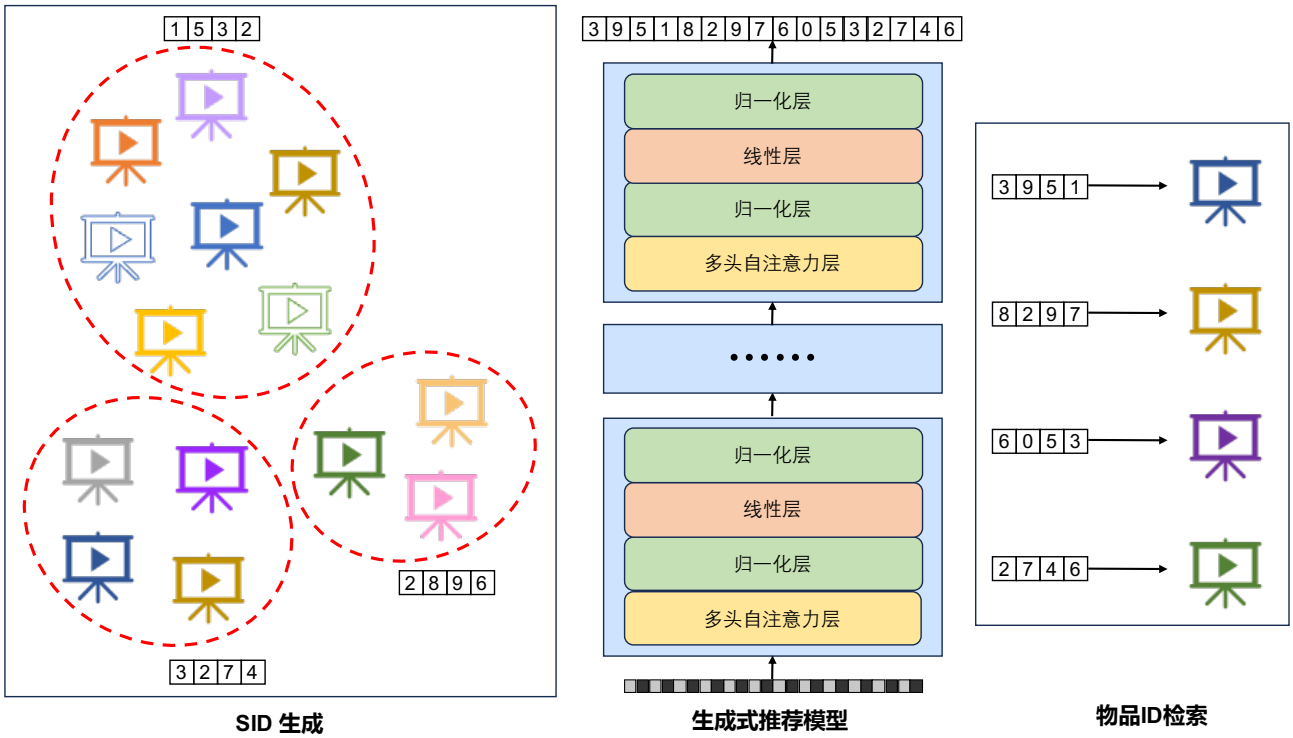


图 9.2: 生成式推荐模块内部结构。

其中，语义 ID 生成模块通常属于离线预处理阶段，其核心目标是将候选物品空间映射到一个结构化的离散语义空间。该过程通常采用 Residual Quantization（RQ-VAE）、RQ-KMeans 等向量量化方法，对物品 Embedding 进行多层次聚类建模，从而得到多维度的 codebook 表示。以四层 codebook 结构为例，每一层的取值空间大小为 512，则理论上可表示的组合空间规模为 $512^4 \approx 6.87 \times 10^{10}$，能够覆盖极大规模的物品集合，从而在有限的离散空间中实现对海量物品的有效编码与表达。

生成式推荐模型通常基于 Transformer Decoder-only 架构构建，其整体结构与大语言模型高度一致，主要由多层自注意力机制（Multi-Head Self-Attention）、前馈网络（Feed Forward Network）以及 Layer Normalization 归一化层等组件堆叠而成。通过对大规模用户行为序列进行自回归建模，模型能够学习用户在语义 Token 空间中

的动态偏好演化模式，从而实现对下一步语义 ID 的预测。

在模型完成语义 ID 序列生成之后，系统将按照预设的 codebook 维度对连续生成的 token 进行分组重构，例如每四个语义 ID 组成一个完整的物品表示，随后通过反向索引表完成从语义 ID 空间到物品 ID 空间的映射检索，最终生成可用于推荐展示的物品 ID 列表。

## 9.3 语义 ID 生成

语义 ID 生成阶段是生成式推荐框架中的基础环节，其核心目标是将高维连续的物品表示空间映射到离散的语义 Token 空间，从而为后续的序列建模提供统一的离散化输入表示。在工业实践中，物品通常首先通过双塔模型或表征学习模型映射为低维 Embedding 向量，该向量能够表达物品的语义信息、用户反馈信息以及上下文统计特征。然而，直接在连续空间上进行生成式建模存在词表不可控与计算复杂度过高的问题，因此需要通过向量量化（Vector Quantization）方法对物品空间进行离散化处理。

常见的方法包括 Residual Quantization Variational AutoEncoder（RQ-VAE）、RQ-KMeans 以及 Product Quantization（PQ）等。这类方法通过逐层残差编码的方式，将一个物品 Embedding 分解为多个离散码本 (codebook) 索引，从而形成层次化语义表示结构。具体而言，给定物品 Embedding $\mathbf{e}_i$，语义编码过程可以表示为：

$$\mathbf{e}_i \rightarrow (s_i^1, s_i^2, ..., s_i^L) \tag{9.7}$$

其中 $L$ 表示码本层数，每一层 $s_i^l$ 均从对应的离散字典空间中选取。通过该方式，原本连续且不可控的物品表示空间被转化为结构化的离散语义空间，为生成式推荐模型提供了统一的 token 输入表示基础。

### 9.3.1 RQ-VAE 算法

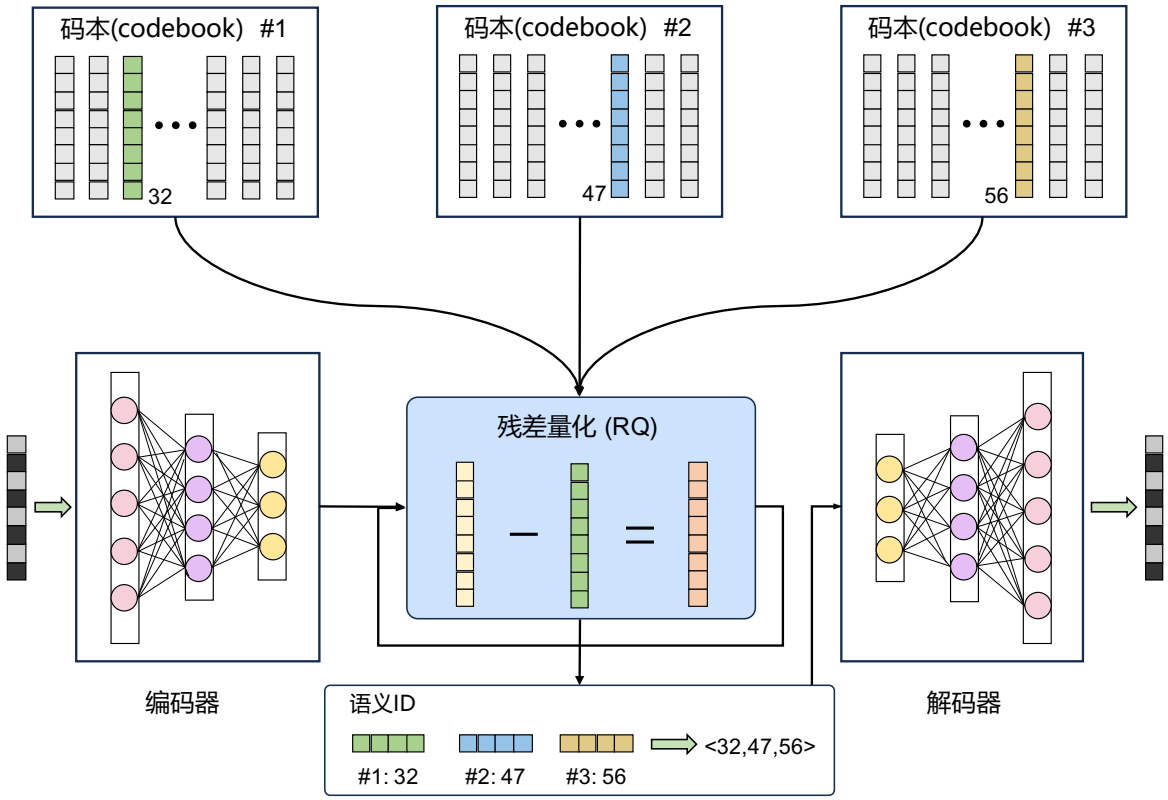


图 9.3: RQ-VAE 模型结构。

这里我们介绍一下 RQ-VAE 这种经典的算法。RQ-VAE 的整体结构如图9.3所示，其核心思想是在标准 VAE 框架中引入残差量化 (Residual Quantization, RQ) 机制，将连续表示逐层分解为多个离散码本索引，从而形成多

层语义编码结构。RQ-VAE 主要由三个部分组成：编码器 (Encoder)、残差量化 (Residual Quantizer) 以及解码器 (Decoder)。

**（1）Encoder 阶段**

给定输入物品的特征表示（如 ID embedding、内容理解特征等），Encoder 网络将其映射为连续潜在表示：

$$\mathbf{z}_0 = f_\theta(\mathbf{x}) \tag{9.8}$$

其中 $\mathbf{x}$ 表示原始物品特征，$\mathbf{z}_0$ 表示通过 Encoder 网络映射到隐空间中的初始潜在表示 (latent representation)。

**（2）RQ 阶段**

RQ-VAE 的核心在于逐层残差量化过程。对于第 $l$ 层量化过程，定义：

$$\mathbf{z}_l = \mathbf{z}_{l-1} - \mathbf{e}_{s^l} \tag{9.9}$$

其中 $\mathbf{e}_{s^l}$ 表示第 $l$ 层 codebook 中选中的向量。对应的离散编码过程为：

$$s^l = \arg\min_{k \in \mathcal{C}^l} \|\mathbf{z}_{l-1} - \mathbf{e}_k\|^2 \tag{9.10}$$

其中 $\mathcal{C}^l$ 表示第 $l$ 层 codebook。这个过程与 K-Means 算法中寻找聚类簇中心的过程非常相似。然后，通过逐层残差分解，原始向量被表示为：

$$\mathbf{z}_0 \approx \sum_{l=1}^{L} \mathbf{e}_{s^l} \tag{9.11}$$

**（3）Decoder 与训练目标**

Decoder 从量化后的表示重构输入：

$$\hat{\mathbf{x}} = g_\phi\left(\sum_{l=1}^{L} \mathbf{e}_{s^l}\right) \tag{9.12}$$

重构后的 $\hat{\mathbf{x}}$ 通常用来预测原始输入 $\mathbf{x}$，模型希望在通过压缩编码和解码的过程中还能够还原回原始输入。因此，整体训练目标通常包含两部分：

$$\mathcal{L} = \mathcal{L}_{recon} + \lambda \sum_{l=1}^{L} \|\mathbf{z}_{l-1} - \mathbf{e}_{s^l}\|^2 \tag{9.13}$$

其中，$\mathcal{L}_{recon}$ 表示重构损失，用于保证量化后的表示仍能恢复原始物品语义信息；第二项为量化误差约束，用于保证 codebook 学习的稳定性。

RQ-VAE 算法的优势在于能够通过残差分解机制显著提升 codebook 的表达能力，从而在有限的离散表示空间中实现对复杂物品语义的高精度建模。从方法演进的角度来看，RQ-VAE 是在 VQ-VAE（Vector Quantized Variational AutoEncoder）基础上的重要扩展。VQ-VAE 通过引入向量量化机制，将连续潜在表示映射为离散 codebook 索引，从而将原始的连续表示空间转换为结构化的离散语义空间。这一过程使得模型能够天然适配基于 token 的生成式框架，并广泛应用于序列建模任务中。

然而，VQ-VAE 通常采用单层或扁平化的 codebook 结构，其表达能力受限于单一量化层的粒度，因此在面对大规模、高复杂度的物品语义空间时，容易出现表示能力不足的问题。为此，RQ-VAE 在 VQ-VAE 的基础上引入了 Residual Quantization 机制，通过逐层对残差进行量化，将原始向量分解为多个 codebook 层级的组合表示。相比于单层量化方式，RQ-VAE 能够以多阶段逐步逼近的方式对连续空间进行细粒度建模，从而在表达能力与压缩效率之间取得更优的平衡。此外，多层 codebook 结构使得 RQ-VAE 产出的语义 ID 具备天然的层次化结构特性，不仅增强了表示能力，也提升了语义 Token 空间的可组合性。因此，该类语义 ID 能够与 Transformer 自回归结构高度兼容，直接作为 token 输入用于生成式推荐模型的序列建模过程。

### 9.3.2 RQ-VAE 的工程实践问题

尽管 RQ-VAE 已经成为当前生成式推荐系统中最主流的语义 ID 生成方法之一，但在实际训练与部署过程中仍然存在若干值得关注的问题，包括梯度传播、码本坍缩（Codebook Collapse）以及训练稳定性等。这些问题不仅影响语义 ID 的质量，也会进一步影响后续生成式推荐模型的效果。

首先需要指出的是，尽管 VQ-VAE 与 RQ-VAE 名称中均包含 VAE（Variational AutoEncoder），但二者实际上并不属于严格意义上的变分自编码器。传统 VAE 通过变分推断（Variational Inference）优化证据下界（Evidence Lower Bound, ELBO），即：

$$\mathcal{L}_{ELBO} = \mathbb{E}_{q(z|x)}[\log p(x|z)] - D_{KL}\left(q(z|x) \parallel p(z)\right) \tag{9.14}$$

而 VQ-VAE 与 RQ-VAE 均未引入 KL 散度约束，也未显式学习潜变量分布，而是采用离散码本（Codebook）进行向量量化。从本质上来看，它们更接近于一种带离散信息瓶颈层（Discrete Information Bottleneck）的 AutoEncoder 结构。然而，引入离散码本后会产生新的问题。对于编码器输出：

$$\mathbf{z}_e = f_\theta(x) \tag{9.15}$$

量化过程为：

$$\mathbf{z}_q = \arg\min_{\mathbf{e}_k \in \mathcal{C}} \|\mathbf{z}_e - \mathbf{e}_k\|_2^2 \tag{9.16}$$

其中 $\mathcal{C}$ 表示 Codebook 集合。由于最近邻查找（Nearest Neighbor Search）本质上是离散操作，因此：

$$\frac{\partial \mathbf{z}_q}{\partial \mathbf{z}_e} = 0 \tag{9.17}$$

梯度无法从 Decoder 反向传播到 Encoder。

为了解决这一问题，VQ-VAE 与 RQ-VAE 通常采用 Straight Through Estimator（STE）技巧进行梯度近似：

$$\mathbf{z}_{STE} = \mathbf{z}_e + \mathrm{sg}(\mathbf{z}_q - \mathbf{z}_e) \tag{9.18}$$

其中 $\mathrm{sg}(\cdot)$ 表示 Stop Gradient 操作。在前向传播阶段，有：

$$\mathbf{z}_{STE} = \mathbf{z}_q \tag{9.19}$$

使用量化后的离散向量参与计算；而在反向传播阶段：

$$\frac{\partial \mathbf{z}_{STE}}{\partial \mathbf{z}_e} = 1 \tag{9.20}$$

梯度直接传递给 Encoder 输出，从而实现近似可导训练。尽管 STE 能够有效解决梯度传播问题，但其本质属于梯度近似方法，因此训练过程中仍可能出现梯度偏差较大、收敛不稳定等现象。

另一个常见问题是码本坍缩（Codebook Collapse）。理想情况下，Codebook 中的所有向量都能够被较为均匀地访问和利用。然而在实际训练过程中，模型往往倾向于频繁选择少量 Codebook 向量，而大部分 Codebook 长期处于未使用状态。假设 Codebook 大小为 $K$，第 $k$ 个 Codebook 被选中的概率为：

$$p_k = \frac{n_k}{\sum_{j=1}^{K} n_j} \tag{9.21}$$

其中 $n_k$ 表示第 $k$ 个 Codebook 在统计周期内被访问的次数。若仅有少数 Codebook 被频繁访问，则会导致：

$$p_k \to 0 \tag{9.22}$$

对于大部分 Codebook 成立，从而造成表示能力严重下降。工业界通常采用 Perplexity 指标监控 Codebook 使用情况：

$$Perplexity = \exp\left(-\sum_{k=1}^{K} p_k \log p_k\right) \tag{9.23}$$

其本质上是 Codebook 访问分布熵的指数形式。当所有 Codebook 均匀使用时：

$$p_k = \frac{1}{K} \tag{9.24}$$

此时：

$$Perplexity = K \tag{9.25}$$

达到最大值。而当模型仅使用少数几个 Codebook 时：

$$Perplexity \ll K \tag{9.26}$$

说明已经出现明显的码本坍塌 (Codebook Collapse) 现象。

为了缓解这种码本坍塌问题，工业界通常会引入熵正则项（Entropy Regularization）：

$$\mathcal{L}_{entropy} = -\lambda \sum_{k=1}^{K} p_k \log p_k \tag{9.27}$$

从而鼓励模型更加均衡地使用各个码本。

### 9.3.3 RQ-KMeans 算法

除了 RQ-VAE 之外，工业界还广泛采用 RQ-KMeans [6] 作为语义 ID 生成方案，比如快手提出的 OneRec 生成式推荐模型。RQ-KMeans 可以看作 Residual Quantization（RQ）与 KMeans 聚类算法的结合。与 RQ-VAE 需要训练 Encoder 和 Decoder 网络不同，RQ-KMeans 完全基于聚类方法构建语义 ID，因此训练过程更加简单，计算资源消耗也更低。

具体而言，对于原始 Embedding 向量 $\mathbf{x}_0$，首先执行第一层 KMeans 聚类：

$$c_1 = \arg\min_j \|\mathbf{x}_0 - \mu_j\|_2^2 \tag{9.28}$$

其中，$\mu_j$ 表示聚类中心，$c_1$ 表示的是第一层语义 ID，即聚类中心的 ID。随后计算残差：

$$\mathbf{r}_1 = \mathbf{x}_0 - \mu_{c_1} \tag{9.29}$$

并将残差继续作为下一层 KMeans 的输入：

$$c_2 = \arg\min_j \|\mathbf{r}_1 - \mu_j^{(2)}\|_2^2 \tag{9.30}$$

不断迭代后得到：

$$(c_1, c_2, \cdots, c_L) \tag{9.31}$$

作为最终的语义 ID 表示。对应的向量重构形式为：

$$\mathbf{x} \approx \sum_{l=1}^{L} \mu_{c_l}^{(l)} \tag{9.32}$$

与 RQ-VAE 相比，RQ-KMeans 无需训练神经网络，因此实现简单、训练速度快，并且不存在 STE 带来的梯度近似问题。在一些超大规模推荐系统中，RQ-KMeans 及其改进版本 RQ-KMeans++ 仍然是构建语义 ID 的重要方案之一。其中，RQ-KMeans++ 继承了 KMeans++ 的思想，在每层残差聚类过程中，新的聚类中心选取策略并非按照 KMeans 算法中那样采用随机初始化策略，而是按照样本到已有聚类中心距离平方成比例的概率进行采样。该策略倾向于选择距离已有聚类中心较远的样本作为新的聚类中心，从而使码本能够更加均匀地覆盖向量空间，降低聚类陷入局部最优解的风险，并提升最终语义 ID 的表达能力。

总体而言，RQ-VAE 与 RQ-KMeans 分别代表了神经网络量化方法与传统聚类量化方法两条技术路线。前者具有更强的表达能力，后者则具有更好的训练效率与工程稳定性。工业系统通常会根据数据规模、计算资源以及线上效果等因素综合选择具体方案。

### 9.3.4 其他离散 Tokenizer 方法

除了 RQ-VAE、RQ-KMeans 这两种生成式推荐模型框架中经常使用到的离散 Tokenzier 方法。本节我们将会介绍其他几种常见的离散 Tokenizer 算法 [10]。首先我们重新形式化定义一下离散 Tokenizer 所做的量化 (Quantization) 过程。假设连续的向量 $\mathbf{z} \in \mathbb{R}^d$ 所组成的 Embedding 矩阵为 $\mathbf{Z} \in \mathbb{R}^{n \times d}$，向量量化过程希望去学习一个 Codebook 矩阵 $\mathbf{C} \in \mathbb{R}^{m \times d}$ 包含 $m$ 个离散的编码，其中 $m \ll n$。所以向量量化过程的原理如下：

**定义 9.2**

向量量化 (Vector Quantization) 过程本质上是学习一个从连续矩阵到离散编码 codebook 的映射 $Q: \mathbf{Z} \rightarrow \mathbf{C}$，对输入矩阵 $\mathbf{Z}$ 中的每个向量 $\mathbf{z}$ 找到 codebook 矩阵 $C$ 中最近的离散编码 $\mathbf{c}$:

$$\begin{aligned} [j, \mathbf{c}_j] &= Q(\mathbf{z}) \\ j &= \arg \min_{k=1,\cdots,m} D(\mathbf{z}, \mathbf{c}_k) \end{aligned} \tag{9.33}$$

♣

根据上述关于 VQ 的定义我们知道，实现这种最朴素的 VQ 过程的方法就是传统聚类方法，比如 K-Means 算法。而 RQ-VAE 和 RQ-KMeans 由于引入了 RQ 这种层次化方法，因此属于层次化的向量量化方法 (Level-wise Quantization)。另外一种向量量化的范式是分组量化方法 (Group-level Quantization)，具体原理是将一个 $N$ 维向量从维度上切分成几个组，比如 $K$ 个组，然后分别对这 $K$ 个向量单独去做向量量化来降低量化误差。一个经典的分组量化方法是乘积量化 (Product Quantization, PQ) [9]。PQ 算法通过并行地将高维向量 $\mathbf{x}$ 分解成 $K$ 个正交的子向量，得到:

$$\mathbf{x} = concat(\mathbf{x}_1, \cdots, \mathbf{x}_K) \tag{9.34}$$

而 $\mathbf{x}$ 对应的离散编码 $C(\mathbf{x})$ 为:

$$C(\mathbf{x}) = (C_1(\mathbf{x}_1), \cdots, C_K(\mathbf{x}_K)) \tag{9.35}$$

关于 PQ 算法的具体细节我们在后续第17章向量检索系统中详细介绍。

最后一类向量量化算法是无查找表的量化方法 (Lookup Free Quantization)。其中的一个经典算法就是 Finite Scalar Quantization (FSQ) [17] 算法。FSQ 的核心思想是通过映射 $f$ 将编码之后的表征向量映射到比较少的维度上，然后通过四舍五入的方式进行离散化，从而构成某种隐式的 codebook，具体的向量量化公式如下:

$$Q(\mathbf{z}) = \text{round}(f(\mathbf{z})) \tag{9.36}$$

其中 $\mathbf{z}$ 是连续向量。一个 FSQ 的例子如下：假设向量 $\mathbf{z} \in \mathbb{R}^d$，将每个元素 $\mathbf{z}_i$ 映射到 $L$ 个取值上，就可以采用如下的 $f$ 函数:

$$f(\mathbf{z}_i) = \frac{L}{2} tanh(\mathbf{z}_i) \tag{9.37}$$

这样通过四舍五入之后最多就会有 $L^d$ 种不同的向量。

此外，无查找表的量化方法中还有一种经典方法，即 Lookup-Free Quantization (LFQ) [20] 算法。传统 VQ-VAE 模型在 codebook 规模增大时，不仅容易导致 codebook 利用率下降以及训练不稳定等问题，还会使得模型的生成能力下降。为了解决这一问题，LFQ 直接去除了传统 VQ-VAE 中的嵌入码本，将原来的连续码字矩阵替换为一个一维的离散索引集合:

$$C = \{0, 1, \cdots, K-1\} \tag{9.38}$$

从而完全消除了从 Embedding 表中进行查找的操作。LFQ 的核心思想如下:

**定义 9.3**

LFQ 算法希望将高维离散码本分解为多个独立的一维二值变量的笛卡尔积，然后采用二进制编码 Embedding 向量每个维度的方式来用一维数值 Index 表征最终的向量量化结果。

♣

假设码本大小为 $K$，则对应的潜在空间可以表示为:

$$C = \prod_{i=1}^{\log_2 K} C_i \tag{9.39}$$

其中每个子空间均为二值集合:

$$C_i = \{-1, 1\} \tag{9.40}$$

对于编码器输出的特征向量

$$\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \cdots, \mathbf{z}_{\log_2 K}] \tag{9.41}$$

其每个维度独立进行量化：

$$q(\mathbf{z}_i) = \arg\min_{c \in C_i} \|\mathbf{z}_i - c\| \tag{9.42}$$

由于每个子空间仅包含两个取值 $\{-1, 1\}$，上述最近邻搜索可以进一步简化为符号函数：

$$q(\mathbf{z}_i) = \text{sign}(\mathbf{z}_i) = \begin{cases} -1, & \mathbf{z}_i \leq 0, \\ 1, & \mathbf{z}_i > 0 \end{cases} \tag{9.43}$$

因此，一个长度为 $\log_2 K$ 的二值向量即可唯一对应一个离散 Token，其索引可以表示为：

$$Index(\mathbf{z}) = \sum_{i=1}^{\log_2 K} 2^{i-1} \mathbf{1}(\mathbf{z}_i > 0) \tag{9.44}$$

其中 $\mathbf{1}(\cdot)$ 表示指示函数。可以看出，LFQ 本质上利用二进制编码将连续特征直接映射为离散 Token，而不再依赖传统码本中的向量查找操作。为了避免部分 Token 长期得不到使用，LFQ 在训练过程中还引入了熵正则项（Entropy Penalty）来提高码本利用率：

$$L_{entropy} = \mathbb{E}_x[H(q(\mathbf{z}))] - H\left(\mathbb{E}_x[q(\mathbf{z})]\right) \tag{9.45}$$

其中 $H(q(\mathbf{z})) = \sum_{i=1}^{\log_2 K} H(\hat{q}(c_i|\mathbf{z}_i))$，利用了各个维度独立时熵的可加性。这里 $\hat{q}(\mathbf{z}_i)$ 是一个如下所示的伯努利分布，而不是硬编码之后的 $q(\mathbf{z})$。

$$\hat{q}(c_i|\mathbf{z}_i) = \frac{\exp(-\tau\|c_i - \mathbf{z}_i\|)}{\sum_{c_i' \in C_i} \exp(-\tau\|c_i' - \mathbf{z}_i\|)} \tag{9.46}$$

需要注意的是，$L_{entropy}$ 损失函数中的第一项用于降低单个样本的不确定性，第二项则鼓励不同样本在整个码本空间上均匀分布，从而提高 Token 的利用率，缓解“码本坍塌”（Codebook Collapse）问题。

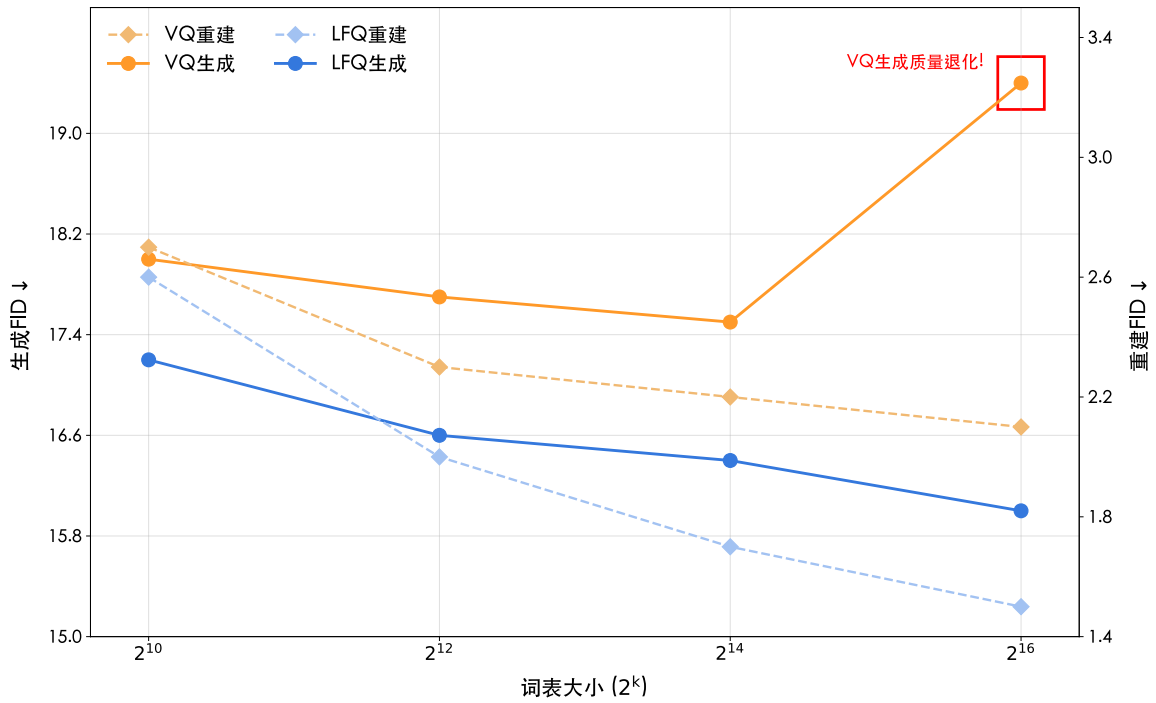


图 9.4: LFQ 与传统 VQ 在生成与重建方面的效果对比。

相比于传统 VQ-VAE 和 RQ-VAE，LFQ 最大的优势在于其不依赖显式码本查找，因此能够支持更大的词表规模。例如，通过增加二值变量的维度即可指数级扩展词表的大小，而计算复杂度仅线性增长。如图9.4所示，LFQ

的论文实验表明，随着词表的持续增大，LFQ 的重建质量（Reconstruction FID，Fréchet Inception Distance）以及后续 Transformer 的生成质量都能够持续提升，而传统 VQ-VAE 方法在码本规模增大之后往往会出现性能饱和甚至退化。

注意这里的 FID 是一种衡量生成图像（或重建图像）的特征分布与真实图像特征分布之间距离的度量指标，一般来讲 FID 越小表明生成或者重建的质量越好。FID 的定义如下：

$$FID = \|\mu_r - \mu_g\|_2^2 + \mathrm{Tr}\left(\Sigma_r + \Sigma_g - 2(\Sigma_r\Sigma_g)^{1/2}\right), \tag{9.47}$$

其中 $\mu_r$、$\Sigma_r$ 是真实图像特征的均值和协方差矩阵，$\mu_g$、$\Sigma_g$ 是生成（或重建）图像特征的均值和协方差矩阵，$Tr(\cdot)$ 是矩阵迹（trace）运算，$(\Sigma_r\Sigma_g)^{1/2}$ 是矩阵平方根运算。

综上所述，LFQ 不仅能够降低量化阶段的计算开销，而且能够提供更大的离散语义空间，为后续基于 Transformer 的生成模型提供更加丰富的 Token 表示能力。目前，LFQ 已成为视觉 Tokenizer 领域的重要研究方向，并被广泛应用于 Visual Autoregressive（VAR）、Vision-Language Model 以及多模态生成模型中。同时，LFQ 的语义 ID 生成技巧也可以无缝迁移与应用到生成式推荐框架之中。

## 9.4 生成式推荐模型

生成式推荐模型（Generative Recommendation Model）是整个生成式推荐框架的核心组成部分，其本质思想是将推荐问题转化为语义 Token 序列生成问题。与传统推荐模型直接预测用户对候选物品的点击率、观看时长或转化率不同，生成式推荐模型试图学习用户行为序列背后的生成规律，并通过自回归（Autoregressive）的方式逐步生成用户未来最可能感兴趣的内容。

目前主流生成式推荐模型大多采用 Transformer Decoder-only 架构，其整体设计与大语言模型中的 GPT 系列模型高度一致。模型输入通常由用户历史行为对应的语义 ID 序列构成。例如，假设每个物品通过 RQ-VAE 被编码为 4 层语义 ID：

$$Item_i \rightarrow [s_i^1, s_i^2, s_i^3, s_i^4] \tag{9.48}$$

那么用户历史行为序列：

$$Item_1 \rightarrow Item_2 \rightarrow Item_3 \tag{9.49}$$

便可以展开为：

$$[s_1^1, s_1^2, s_1^3, s_1^4, s_2^1, s_2^2, s_2^3, s_2^4, s_3^1, s_3^2, s_3^3, s_3^4] \tag{9.50}$$

随后输入 Transformer 模型进行序列建模。首先，每个语义 Token 会被映射到 Embedding 空间：

$$\mathbf{x}_t = Embedding(s_t) + PE(t) \tag{9.51}$$

其中 $PE(t)$ 表示位置编码（Positional Encoding），用于向模型注入序列位置信息。

### 9.4.1 Transformer 模型

Transformer 最核心的组件是多头自注意力机制（Multi-Head Self-Attention，MHA）。对于输入隐藏状态矩阵：

$$H = [h_1, h_2, \cdots, h_n] \tag{9.52}$$

模型首先通过三个不同的线性映射得到查询（Query）、键（Key）和值（Value）向量：

$$Q = HW_Q, \quad K = HW_K, \quad V = HW_V \tag{9.53}$$

其中，$Q$ 表示 Query 矩阵，$K$ 表示 Key 矩阵，$V$ 表示 Value 矩阵，$W_Q$、$W_K$ 和 $W_V$ 均为可学习参数矩阵。对于单个注意力头，其计算过程如下：

$$Attention(Q,K,V) = Softmax\left(\frac{QK^\top}{\sqrt{d_k}}\right)V \tag{9.54}$$

其中，$d_k$ 表示 Key 向量维度，缩放因子 $\sqrt{d_k}$ 用于缓解高维空间下内积值过大的问题，从而保证训练过程的稳定性。

为了同时建模不同语义子空间中的信息交互，Transformer 进一步采用多头注意力机制：

$$head_i = Attention(Q_i, K_i, V_i) \tag{9.55}$$

多个注意力头的输出经过拼接后再进行一次线性变换：

$$MultiHead(Q,K,V) = Concat(head_1, \cdots, head_h)W_O \tag{9.56}$$

其中 $h$ 表示注意力头数量，$W_O$ 表示输出映射矩阵。

通过多头注意力机制，模型能够从不同角度捕获用户行为序列中的依赖关系。例如，一部分 Attention Head 可能更加关注用户长期兴趣偏好，而另一部分 Attention Head 则可能更加关注用户最近的兴趣变化、内容类别迁移等。相比于传统推荐模型依赖固定长度兴趣向量的方式，Transformer 能够动态建模任意位置之间的交互关系，因此更加适合长行为序列场景。

在多头注意力层之后，Transformer 还包含位置独立的前馈神经网络（Feed Forward Network，FFN）：

$$FFN(x) = W_2\sigma(W_1x + b_1) + b_2 \tag{9.57}$$

其中 $\sigma(\cdot)$ 通常采用 ReLU、GELU 等非线性激活函数。

FFN 层负责对 Attention 提取出的上下文特征进行非线性变换，从而提升模型的表达能力。从近年来对大语言模型内部机理的研究结果来看，FFN 层不仅承担特征变换的作用，还可能存储大量与实体、概念以及语义模式相关的知识信息；而 Attention 层则更多负责在当前上下文条件下对这些知识进行动态检索、组合与调用。因此，也有研究将 Attention 视为一种**信息路由（Information Routing）机制**，而将 FFN 视为模型知识存储的重要载体。

为了保证深层网络训练的稳定性，Transformer 还引入了残差连接（Residual Connection）和层归一化（Layer Normalization）机制。对于输入 $x$ 和子层输出 $f(x)$，残差结构通常表示为：

$$y = x + f(x) \tag{9.58}$$

随后再进行归一化操作：

$$LayerNorm(y) = \gamma\frac{y-\mu}{\sqrt{\sigma^2+\epsilon}} + \beta \tag{9.59}$$

其中 $\mu$ 和 $\sigma^2$ 分别表示特征维度上的均值与方差，$\gamma$ 和 $\beta$ 为可学习参数。

关于 Layer Normalization 的具体放置位置，目前主要存在两种实现方式。早期 Transformer 采用 Post-LN 结构，即先执行残差连接再进行归一化：

$$y = LayerNorm(x + f(x)) \tag{9.60}$$

而当前主流的大语言模型（如 GPT-3、LLaMA 等）则普遍采用 Pre-LN 结构，即先归一化再进行子层计算：

$$y = x + f(LayerNorm(x)) \tag{9.61}$$

大量实践表明，Pre-LN 结构能够显著改善深层 Transformer 的梯度传播问题，因此在当前大规模生成式推荐模型和大语言模型中应用更加广泛。

### 9.4.2 生成式推荐训练目标

生成式推荐模型通常采用与大语言模型一致的 Next Token Prediction（NTP）训练范式。给定历史语义 Token 序列：

$$[s_1, s_2, \cdots, s_{t-1}] \tag{9.62}$$

模型预测下一个 Token：

$$P(s_t|s_{<t}) = Softmax(z_{t,s_t}) = \frac{\exp(z_{t,s_t})}{\sum_{k=0}^{K} \exp(z_{t,k})} \tag{9.63}$$

其中 $z_{t,k}$ 表示第 t 个 token 时 Transformer 最后一层输出向量中第 $k$ 个语义 token 对应的 logit 数值，而 $K$ 表示每个码本的大小。整个序列的联合概率可以表示为：

$$P(S) = \prod_{t=1}^{T} P(s_t|s_{<t}) \tag{9.64}$$

因此训练目标为最大化训练样本的似然函数：

$$\max_{\theta} \sum_{t=1}^{T} \log P(s_t|s_{<t}; \theta) \tag{9.65}$$

对应的损失函数即为交叉熵损失：

$$\mathcal{L}_{NTP} = -\sum_{t=1}^{T} \log P(s_t|s_{<t}) \tag{9.66}$$

或者写成期望形式：

$$\mathcal{L}_{NTP} = -\mathbb{E}_S \left[ \sum_{t=1}^{T} \log P(s_t|s_{<t}) \right] \tag{9.67}$$

注意这里面的 $s_t$ 都是对应真实语义 token，实际上相当于是根据已有训练数据去做最大似然估计。通过最小化上述损失函数，模型能够学习用户历史行为序列中的潜在兴趣演化规律，并逐步掌握用户未来行为的生成分布。

与传统推荐系统中的分类或排序建模不同，生成式推荐模型不再直接预测候选物品的相关性分数，而是学习用户行为序列本身的生成过程。因此，推荐任务被统一转化为序列建模问题，从而能够充分利用 Transformer 在长序列建模和生成任务中的优势。

## 9.5 物品 ID 检索

物品 ID 检索（Item ID Retrieval）阶段负责将生成式推荐模型输出的语义 Token 序列映射回实际可展示的物品集合，是连接生成空间与推荐系统输出空间的重要桥梁。在完成自回归生成之后，模型输出的是一段连续的语义 Token 序列，而非传统推荐系统中的物品 ID。由于生成式推荐通常采用多层码本 (Codebook) 构建语义 ID（Semantic ID，SID），因此需要首先将生成得到的 Token 序列按照预定义的编码维度进行重组，从而恢复出完整的语义 ID。

假设模型生成的语义序列为：

$$[s_1, s_2, \cdots, s_n] \tag{9.68}$$

若每个物品由长度为 $L$ 的语义 ID 表示，则每连续 $L$ 个 Token 可组成一个完整的语义编码：

$$(s_i^1, s_i^2, \cdots, s_i^L) \rightarrow SID_i \tag{9.69}$$

随后通过语义 ID 与物品 ID 的映射关系恢复得到对应的物品：

$$SID_i \rightarrow item_i \tag{9.70}$$

物品 ID 检索过程如图 9.5 所示。这里采用了 Transformer Encoder-Decoder 的生成式推荐模型架构，关于生成式推荐模型的 Encoder-Decoder 架构和 Decoder-Only 架构我们在后续。

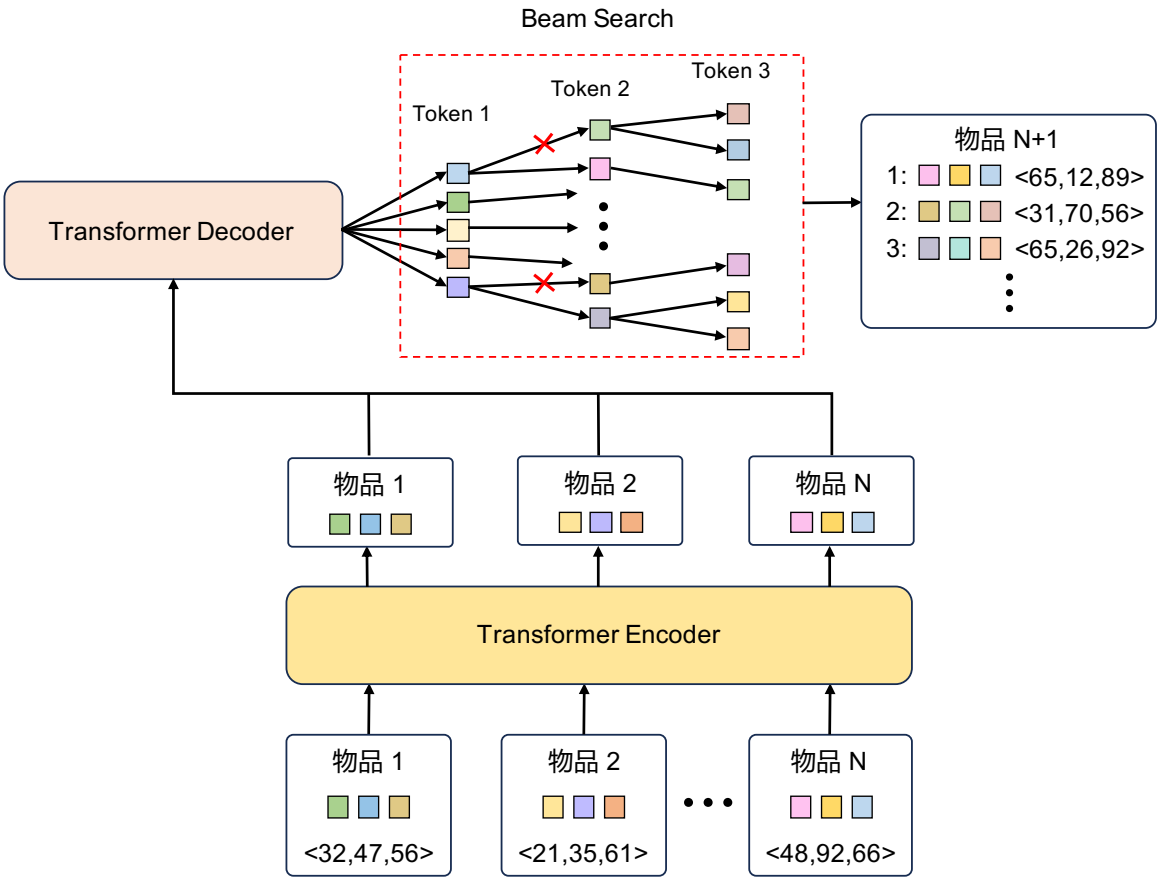


图 9.5: 物品 ID 检索示意图。

需要指出的是，生成式推荐模型在每个生成步输出的并非离散 Token，而是整个码本上的概率分布。因此，在推理阶段通常需要采用采样策略生成最终的 Token 序列。其中最常用的方法是 Beam Search。以三层码本为例，在生成每一层 Token 时，可以保留概率最高的 Top-$B$ 个候选，并不断扩展生成树。随着生成过程的进行，通过剪枝策略持续保留综合得分最高的若干条路径，最终得到 Top-$K$ 个候选语义 ID。该过程本质上是一种 Top-$K$ 搜索过程。

设某个候选语义 ID 对应的 Token 序列为：

$$(s_1, s_2, \cdots, s_L) \tag{9.71}$$

则其生成概率可表示为：

$$P(SID) = \prod_{i=1}^{L} P(s_i | s_{<i}, u) \tag{9.72}$$

实际计算时通常采用对数概率累加的形式作为路径得分：

$$Score(SID) = \sum_{i=1}^{L} \log P(s_i | s_{<i}, u) \tag{9.73}$$

并据此对不同候选路径进行排序和剪枝。为了减轻长度偏置问题，在实际系统中也常采用长度归一化后的分数作为 Beam Search 的评价指标。

在根据语义 ID 检索物品 ID 的过程中，还可能出现两类异常情况。第一类是多个语义 ID 映射到同一个物品，即多对一映射问题。此时需要对重复物品进行去重，仅保留得分最高的候选结果。第二类是生成得到的语义 ID 无法在映射表中找到对应物品，即产生无效语义 ID（Invalid SID）。对于这类情况，系统通常会直接丢弃对应路径，并继续从 Beam Search 的其他候选路径中补充有效结果，以保证最终输出的 Top-$K$ 推荐列表满足数量要求。

在完成语义 ID 到物品 ID 的恢复之后，系统需要借助预先构建的索引结构完成快速查找。最简单的实现方式是哈希表或倒排映射表，通过语义 ID 直接定位对应物品；对于层次化语义 ID，也可以构建 Trie 树进行前缀检索；当允许一个语义 ID 对应多个近邻物品时，还可以采用 ANN（Approximate Nearest Neighbor）索引实现近似匹配。借助这些高效索引结构，系统能够在在线推理阶段以较低的延迟完成从生成空间到实际物品空间的映射，从而实现生成式推荐模型的实时服务。

## 9.6 一阶段生成式推荐

前面几节介绍的生成式推荐方法，大多采用“语义 ID 生成 + 生成式推荐模型”的两阶段框架。该框架首先利用内容理解模型得到物品 Embedding，然后通过 RQ-VAE、RQ-KMeans 等向量量化方法将连续向量压缩为离散语义 ID，最后将语义 ID 序列作为生成式推荐模型的输入与预测目标进行训练。

在这种两阶段架构中，语义 ID 一旦生成之后，在后续生成式推荐模型训练过程中通常保持固定，不再随着下游推荐目标进行更新。这种范式虽然能够有效降低大规模物品空间的生成难度，但也会带来如下问题：

笔记 语义 ID 的学习目标与生成式推荐模型的优化目标存在不匹配的问题。语义 ID 的训练通常关注内容重建质量或聚类质量，而重建质量的提升并不一定能够带来下游推荐性能的提升。此外，语义 ID 在生成式推荐训练阶段被冻结，使得推荐损失无法反向传播到语义 ID 的学习过程，从而无法实现端到端联合优化。

因此，近年来越来越多的研究开始尝试打破传统“两阶段压缩 + 自回归生成”的范式，希望能够实现语义 ID 与生成式推荐模型的联合训练，从而使语义 ID 的学习直接服务于推荐目标。根据优化方式的不同，这一小节我们主要介绍可微语义 ID（Differentiable Semantic ID）和端到端语义 ID 生成这两大类方法。

### 9.6.1 DIGER 算法

Differentiable Semantic ID for Generative Recommendation（DIGER）[7] 是近年来尝试将语义 ID 学习过程纳入生成式推荐优化目标中的代表性工作。传统生成式推荐模型通常采用两阶段范式，首先通过 RQ-VAE 等方法离线生成语义 ID，然后将固定的语义 ID 序列作为生成式推荐模型的输入和预测目标。在这种设置下，语义 ID 在后续训练过程中保持不变，因此推荐损失无法反向传播至量化模块，导致语义 ID 的学习目标与下游推荐目标之间存在优化不匹配的问题。

然而，直接采用 Straight Through Estimator（STE）对离散 Token 进行可微优化，容易导致前几节提到的码本坍塌（Codebook Collapse）问题，即模型过早收敛到少量语义 Token 上，使得大量码本长期得不到使用，最终导致语义空间利用率下降。因此，DIGER 提出了 Diffe**R**entiable Semantic **ID** with Exploratory **L**earning（DRIL）框架，其整体结构如图9.6所示。

DIGER 算法的核心思想如下：

**定义 9.4**

DIGER 算法通过构建可微语义 ID 学习机制，使得推荐损失能够直接参与语义 ID 的优化过程，从而实现语义 ID 与生成式推荐模型的端到端联合训练。

DIGER 算法提出了一种 **Diffe****R**entiable Semantic **ID** with Exploratory **L**earning(DRIL) 的范式。下面对 DRIL 进行一个简要的介绍。对于一个物品 $v$，经过“LLM Encoder + RQ-VAE Encoder”得到连续表示：

$$\mathbf{r}_v = (\mathbf{r}_{v,1}, \mathbf{r}_{v,2}, \cdots, \mathbf{r}_{v,m}), \tag{9.74}$$

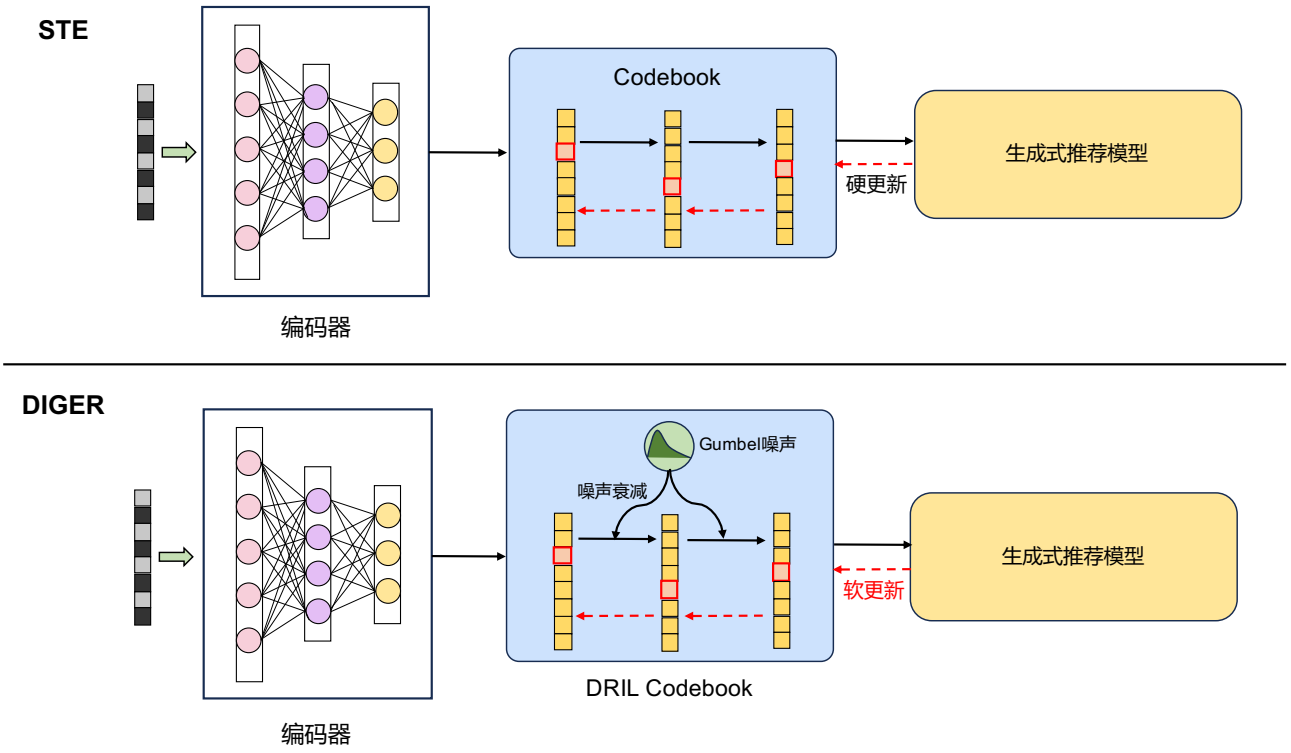


图 9.6: DIGER 方法与朴素 STE 方法的对比。

其中 $m$ 表示语义 ID 的层数。对于第 $j$ 个码本位置，计算其与码本中所有码字 $\{e_i\}_{i=1}^{K}$ 的相似度：

$$\ell_{v,j,i} = sim(\mathbf{r}_{v,j}, e_i), \qquad i \in 1, \cdots, K. \tag{9.75}$$

如果直接采用最大相似度对应的码字作为离散 Token，则梯度无法通过离散选择操作传播。因此，DRIL 在 logits 上引入 Gumbel 噪声：

$$g_{v,j,i} \sim Gumbel(0,1), \tag{9.76}$$

其中，$Gumbel(0,1)$ 表示标准 Gumbel 分布（Type-I Extreme Value Distribution），其累积分布函数（CDF）为：

$$F(g) = \exp(-e^{-g}) \tag{9.77}$$

对应的概率密度函数（PDF）为：

$$p(g) = \exp\left(-(g + e^{-g})\right) \tag{9.78}$$

在实际应用中，通常不直接从 Gumbel 分布进行采样，而是利用逆变换采样（Inverse Transform Sampling）的方法。对于均匀分布随机变量

$$u \sim Uniform(0,1) \tag{9.79}$$

可以通过如下变换得到服从标准 Gumbel 分布的随机变量：

$$g = -\log(-\log u) \tag{9.80}$$

即

$$g \sim Gumbel(0,1) \tag{9.81}$$

相比于常见的高斯噪声，Gumbel 噪声能够更好地模拟离散类别采样过程。事实上，著名的 Gumbel-Max Trick

表明，对于 logits $\ell_1, \ell_2, \cdots, \ell_K$，若为每个 logits 添加独立同分布的 Gumbel 噪声：

$$g_i \sim Gumbel(0, 1) \tag{9.82}$$

则有：

$$\arg\max_i(\ell_i + g_i) \sim Softmax(\ell_i) \tag{9.83}$$

即对 logits 加上 Gumbel 噪声之后再执行 $\arg\max$ 操作，等价于按照 Softmax 概率分布进行采样。

有了通过 Gumbel 分布采样的噪声 $g_{v,j,i}$，DRIL 就可以通过 Gumbel-Softmax 得到连续概率分布：

$$\tilde{y}_{v,j,i} = \frac{\exp\left((\ell_{v,j,i} + g_{v,j,i})/\tau\right)}{\sum_{k=1}^{K} \exp\left((\ell_{v,j,k} + g_{v,j,k})/\tau\right)}, \tag{9.84}$$

其中 $\tau$ 为温度系数。相比起高斯噪声，Gumbel 噪声更符合离散 Token 的概率选择机制：相似度较高的码字会以指数形式获得更大的采样概率，而相似度较低的码字仍有一定概率被探索到，从而提高码本的利用率，并有效缓解“码本坍塌”的问题。

在前向传播阶段，为了保持语义 ID 的离散性，DRIL 仍然采用硬选择（Hard Assignment）：

$$c_{v,j} = \arg\max_i(\ell_{v,j,i} + g_{v,j,i}) \tag{9.85}$$

从而得到最终的语义 ID：

$$z_v = (c_{v,1}, c_{v,2}, \cdots, c_{v,m}) \tag{9.86}$$

该语义 ID 将作为生成式推荐模型中的离散 Token 序列进行索引和生成。然而，在反向传播阶段，DRIL 不再直接对离散 Token 求梯度，而是利用 Gumbel-Softmax 得到的连续概率进行软更新（Soft Update）。具体而言，通过概率加权得到软码字表示：

$$\bar{e}_{v,j} = \sum_{i=1}^{K} \tilde{y}_{v,j,i} e_i \tag{9.87}$$

随后利用软码字参与梯度传播，从而实现推荐损失对于 RQ-VAE 码本的可微更新。

因此，DRIL 实际上采用了“Forward Hard, Backward Soft”的训练机制：

$$\begin{gathered}\text{Hard Semantic ID Forward} \quad + \\ \text{Soft Embedding Backward}\end{gathered}$$

这种机制既保证了推理阶段仍然能够输出离散语义 ID，又允许梯度通过连续概率分布传播，从而实现语义 ID 与生成式推荐模型的联合优化。

此外，Gumbel 噪声所引入的随机扰动天然具有探索的作用。在训练初期，随机采样能够增加不同 Token 被访问的概率，提高码本的利用率；随着训练过程的进行，可以逐渐降低温度参数 $\tau$ 或减弱噪声影响，使模型逐渐从探索转向利用，最终收敛到更加稳定的语义空间。当然，DIGER 方法并没有采用这种策略的方法随着训练 epoch 的增加逐渐调低温度参数 $\tau$，而是引入了两种不确定度衰减的技巧，来实现这种从训练初期偏向探索逐渐过渡到训练后期偏向利用。

**9.6.1.0.1 （1）基于噪声标准差的不确定度衰减（Standard Deviation Uncertainty Decay，SDUD）** Gumbel 噪声的标准差 $\sigma$ 决定了语义 ID 分配过程中的随机性。较大的噪声对应更强的探索能力，而当噪声趋近于零时，语义 ID 的选择逐渐退化为确定性的 $\arg\max$ 操作，更接近推理阶段的行为。为此，DIGER 将生成式推荐损失与噪声尺度 $\sigma$ 耦合，构造辅助优化目标：

$$L_\sigma = \frac{L_{gen}}{2(\sigma + \lambda)^2} + \log(\sigma + \lambda), \tag{9.88}$$

其中，$L_{gen}$ 为 Next-SID Prediction 的生成损失，$\sigma \geq 0$ 表示 Gumbel 噪声的标准差，$\lambda > 0$ 为超参数。

对 $\sigma$ 求偏导并令其为零，可得到平衡点：

$$(\sigma + \lambda)^2 = L_{gen} \tag{9.89}$$

即

$$\sigma^* = \max\left(0, \sqrt{L_{gen}} - \lambda\right) \tag{9.90}$$

因此，随着训练过程中生成损失不断减小，最优噪声强度也会自动减小。当 $L_{gen}$ 接近 $\lambda^2$ 时，$\sigma^*$ 逐渐趋近于零，从而实现从随机探索向确定性选择的平滑过渡，减小训练与推理之间的不匹配。

**9.6.1.0.2 （2）基于频率的不确定度衰减（Frequency-based Uncertainty Decay，FrqUD）** 另一种不确定度衰减策略直接利用码本的使用频率进行调节。其核心思想是：频繁被访问的编码往往存在过度使用的问题，因此需要继续保留随机探索；而低频编码本身利用率较低，则无需额外扰动。

设第 $e$ 个 epoch 中第 $i$ 个编码的使用频率为：

$$f_i^{(e)} = \beta f_i^{(e-1)} + (1-\beta)\hat{f}_i^{(e)} \tag{9.91}$$

其中 $\hat{f}_i^{(e)}$ 为当前 epoch 的统计频率，$\beta$ 为 EMA 系数。在均匀利用情况下，平均频率为：

$$\bar{f} = \frac{1}{K} \tag{9.92}$$

进一步定义热门编码的阈值：

$$\gamma = r\bar{f} = \frac{r}{K} \tag{9.93}$$

其中 $r$ 为超参数。

由此可以将码本划分为高频集合和低频集合：

$$I_{high}^{(e)} = \{i \mid f_i^{(e)} > \gamma\} \tag{9.94}$$

$$I_{low}^{(e)} = \{1, \cdots, K\} \backslash I_{high}^{(e)} \tag{9.95}$$

对于高频编码，继续采用带 Gumbel 噪声的随机采样：

$$y_i = \frac{\exp((l_i + g_i)/\tau)}{\sum_j \exp((l_j + g_j)/\tau)}, \qquad i \in I_{high} \tag{9.96}$$

而对于低频编码，则关闭 Gumbel 噪声，直接使用确定性的 Softmax：

$$y_i = \frac{\exp(l_i/\tau)}{\sum_j \exp(l_j/\tau)}, \qquad i \in I_{low} \tag{9.97}$$

这种方法相当于仅对过度使用的热门编码保留探索能力，而对低频编码采用更加稳定的确定性分配，从而提高码本利用率并减小训练后期的随机性。

从本质上来看，DIGER 将传统固定语义 ID 的两阶段生成式推荐问题转化为一个可微的离散表示学习问题，使得推荐损失能够直接参与语义 ID 的学习过程，从而缓解两阶段方法中的优化目标不匹配问题。相比于简单的 STE 方法，DRIL 在保持端到端可训练性的同时，通过 Gumbel 噪声引入探索机制，有效缓解了“码本坍塌”问题，提高了语义空间利用率，为一阶段生成式推荐框架提供了一种有效的实现路径。

## 9.6.2 ETEGRec 算法

ETEGRec（End-To-End Generative Recommender）是另一种实现生成式推荐端到端联合训练的代表性工作，由中国人民大学高瓴人工智能研究院与快手科技联合提出，源自 SIGIR 2025 论文《Generative Recommender with End-to-End Learnable Item Tokenization》[16]。

与上一节介绍的 DIGER 算法类似，ETEGRec 算法同样关注传统两阶段生成式推荐框架中的优化目标不一

致问题。然而，两者解决问题的思路有所不同。DIGER 算法主要从可微离散表示学习的角度出发，通过 Gumbel-Softmax 实现语义 ID 的可微优化；而 ETEGRec 算法则进一步将物品 Tokenizer 与生成式推荐模型作为一个统一整体进行联合建模，并设计推荐导向（Recommendation-oriented）的表示对齐机制，使 Tokenizer 能够持续受到推荐目标的指导，从而学习更加适合生成式推荐任务的语义 ID。

### 9.6.2.1 ETEGRec 算法的核心框架

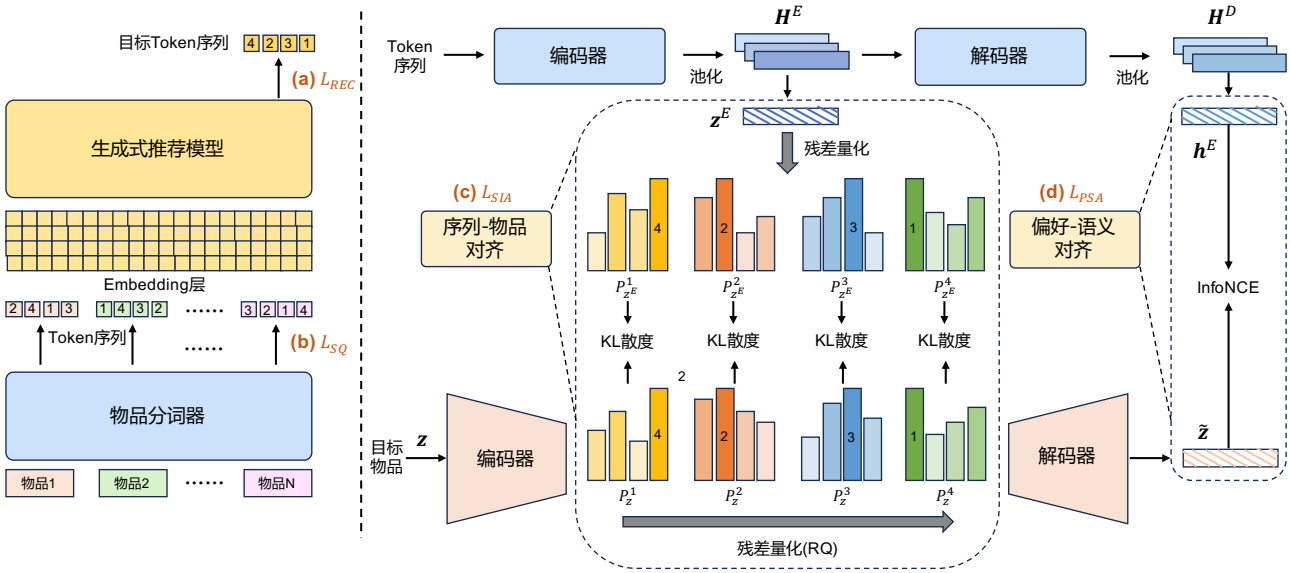


图 9.7: ETEGRec 整体架构示意图。

ETEGRec 整体采用双 Encoder-Decoder（Dual Encoder-Decoder）架构，如图 9.7所示。整个模型主要由两个部分组成：物品 Tokenizer（Item Tokenizer）和生成式推荐模型（Generative Recommender）。其中，Tokenizer 负责将物品映射为多层语义 Token，而生成式推荐模型则负责学习 Token 序列之间的生成关系。与传统方法不同的是，这两个模块在训练过程中并不是彼此独立，而是能够相互协同优化。

具体来说，对于物品 $v$ 而言，ETEGRec 首先利用预训练得到的内容语义表示或协同语义表示作为输入，即 $\mathbf{z}_v \in \mathbb{R}^{d_s}$。随后通过 Tokenizer 中的 Encoder 映射到潜在表示空间 $\mathbf{r}_v = \text{Encoder}_T(\mathbf{z}_v)$，再然后采用 Residual Quantization（RQ）进行逐层量化。假设共有 $L$ 层 Codebook，第 $l$ 层对应的码本表示为 $\mathcal{C}_l = \{\mathbf{e}_1^l, \mathbf{e}_2^l, \cdots, \mathbf{e}_K^l\}$，其中 $K$ 表示每层 Codebook 中的码字数量。则对于第 $l$ 层残差向量 $\mathbf{v}_l$，首先计算其属于各个码字的概率：

$$P(k|\mathbf{v}_l) = \frac{\exp(-\|\mathbf{v}_l - \mathbf{e}_k^l\|_2^2)}{\sum_{j=1}^{K} \exp(-\|\mathbf{v}_l - \mathbf{e}_j^l\|_2^2)} \tag{9.98}$$

然后选择概率最大的码字作为当前层对应的语义 Token：

$$c_l = \arg\max_k P(k|\mathbf{v}_l) \tag{9.99}$$

在完成当前层量化之后，需要继续计算下一层的残差 $\mathbf{v}_{l+1} = \mathbf{v}_l - \mathbf{e}_{c_l}^l$，并不断重复上述过程，最终得到物品对应的多层语义 ID，即 $[c_1, c_2, \cdots, c_L]$。随后，将所有层对应的码字 Embedding 进行累加得到量化表示 $\tilde{\mathbf{r}} = \sum_{l=1}^{L} \mathbf{e}_{c_l}^l$，并利用 Tokenizer 中的 Decoder 网络来重建原始语义 Embedding $\tilde{\mathbf{z}} = \text{Decoder}_T(\tilde{\mathbf{r}})$。为了保证 Tokenizer 能够学习具有良好语义表示能力的离散 Token，ETEGRec 算法采用 RQ-VAE 中的重建目标与残差量化目标来共同训练 Tokenizer，其优化目标可表示为：

$$L_{SQ} = L_{RECON} + L_{RQ} \tag{9.100}$$

其中重建损失定义为：

$$L_{RECON} = \|\mathbf{z} - \tilde{\mathbf{z}}\|_2^2 \tag{9.101}$$

而残差量化损失定义为：

$$L_{RQ} = \sum_{l=1}^{L} \left( \|\text{sg}[\mathbf{v}_l] - \mathbf{e}_{c_l}^l\|_2^2 + \beta \|\mathbf{v}_l - \text{sg}[\mathbf{e}_{c_l}^l]\|_2^2 \right) \tag{9.102}$$

其中 $\text{sg}(\cdot)$ 表示 Stop-Gradient 操作。

在完成 Tokenizer 之后，ETEGRec 算法采用 Transformer Encoder-Decoder 作为生成式推荐模型。假设用户历史交互序列对应的 Token 序列为 $X = [c_1, c_2, \cdots, c_n]$，ETEGRec 算法首先利用 Encoder 学习用户历史行为的表示 $\mathbf{H}_E = \text{Encoder}_R(X)$，随后 Decoder 网络根据 Encoder 网络输出以及已经生成的 Token，以自回归的方式来预测目标物品对应的 Token 序列 $\mathbf{H}_D = \text{Decoder}_R(\mathbf{H}_E, \tilde{Y})$，其中 $\tilde{Y}$ 表示 Decoder 输入序列，即在目标 Token 序列前增加起始 Token（BOS）。最终 ETEGRec 采用标准自回归生成损失进行训练：

$$L_{REC} = -\sum_{j=1}^{L} \log P(Y_j | X, Y_{<j}) \tag{9.103}$$

传统两阶段生成式推荐虽然采用 RQ-VAE 能够学习到具有一定语义结构的物品 Token，但 Tokenizer 与生成式推荐模型始终作为两个相互独立的模块进行优化。Tokenizer 关注的是语义重建，而生成式推荐模型关注的是用户行为的预测，两者之间缺少直接的信息交互。因此，即使生成式推荐模型已经学习到更加准确的用户兴趣表示，也无法进一步指导 Tokenizer 学习更加符合推荐任务需求的语义 ID。在具体工业界落地应用生成式推荐模型的时候也发现，RQ-VAE 训练物品语义 ID 过程的优化目标可能与生成式推荐模型的优化目标存在不一致的问题，导致最终生成式推荐模型的性能存在一定的上限。

### 9.6.2.2 ETEGRec 算法的对齐机制

为了实现 Tokenizer 与生成式推荐模型之间的协同优化，ETEGRec 提出了 Recommendation-oriented Alignment（推荐导向表示对齐）的思想。其核心目标是在推荐模型训练过程中建立用户行为表示与物品语义表示之间的联系，使 Tokenizer 能够持续受到推荐目标的优化约束，从而不断调整语义 Token 的组织方式。ETEGRec 算法分别从 Encoder 端和 Decoder 端设计了两种表示对齐策略，分别为 Sequence-Item Alignment（序列—物品对齐）和 Preference-Semantic Alignment（兴趣—语义对齐）。

**9.6.2.2.1 Sequence-Item Alignment** Sequence-Item Alignment（SIA）的核心思想如下：

**定义 9.5**

用户历史交互序列所蕴含的兴趣表示，应当与用户下一次即将交互物品的语义表示保持一致。 ♣

具体而言，生成式推荐模型的 Encoder 首先根据历史 Token 序列得到用户的行为表示：$\mathbf{H}_E = \text{Encoder}_R(X)$。随后，ETEGRec 算法对所有位置的隐藏状态进行平均池化，并经过一层 MLP 映射得到整个历史序列所对应的语义表示：

$$\mathbf{z}_E = \text{MLP}(\text{MeanPool}(\mathbf{H}_E)) \tag{9.104}$$

另一方面，目标物品本身已经具有对应的协同语义 Embedding：$\mathbf{z}$。将两种表示分别输入 Tokenizer，可以得到每一层 Codebook 对应的 Token 概率分布 $P_{\mathbf{z}}^l$ 以及 $P_{\mathbf{z}_E}^l$，其中 $l$ 表示第 $l$ 层 Codebook。

ETEGRec 算法认为，两种表示本质上都描述了用户下一步最有可能交互的物品，其对应的 Token 分布也应

尽可能一致。因此，ETEGRec 采用了对称的 KL 散度来作为优化目标，即：

$$L_{SIA} = \sum_{l=1}^{L} \left( D_{KL}(P_{\mathbf{z}}^{l} \| P_{\mathbf{z}_E}^{l}) + D_{KL}(P_{\mathbf{z}_E}^{l} \| P_{\mathbf{z}}^{l}) \right) \tag{9.105}$$

这里双向 KL 散度能够同时约束两个概率分布互相靠近，相比于单向 KL 具有更加稳定的优化效果。

除了实现 Tokenizer 与推荐模型之间的信息交互之外，SIA 还有另一个重要作用。由于 Encoder-Decoder 结构中的 Decoder 通常具有较强的自回归建模能力，在训练过程中容易出现 Decoder 绕过 Encoder（Encoder Bypassing）的现象，即 Decoder 更多依赖历史生成 Token，而较少利用 Encoder 输出的信息。SIA 通过直接约束 Encoder 表示与目标物品保持一致，从而进一步增强 Encoder 对于用户兴趣的建模能力。

**9.6.2.2.2 Preference-Semantic Alignment** 除了 Encoder 表示之外，ETEGRec 算法还进一步利用了 Decoder 中的用户兴趣表示构建另一种对齐关系，即 Preference-Semantic Alignment（PSA）。Decoder 在完成 Token 生成之前，会得到一个隐藏状态 $\mathbf{h}_D$，其中 $\mathbf{h}_D$ 对应 Decoder 第一个位置（BOS）的隐藏表示，它融合了用户历史行为信息，可以看作当前用户兴趣的整体表示。另一方面，Tokenizer 经过 RQ-VAE 重建之后，可以恢复得到目标物品对应的语义 Embedding $\tilde{\mathbf{z}}$。ETEGRec 认为，用户兴趣表示应该尽可能接近最终点击物品的语义表示，因此采用对比学习的方法来进一步对两者进行约束。

具体来说，ETEGRec 算法首先利用 MLP 网络来分别映射两种表示，然后采用 InfoNCE 的损失函数进行优化，即：

$$L_{PSA} = -\log \frac{\exp(s(\tilde{\mathbf{z}}, \mathbf{h}_D)/\tau)}{\sum_{\hat{\mathbf{h}} \in B} \exp(s(\tilde{\mathbf{z}}, \hat{\mathbf{h}})/\tau)} - \log \frac{\exp(s(\mathbf{h}_D, \tilde{\mathbf{z}})/\tau)}{\sum_{\hat{\mathbf{z}} \in B} \exp(s(\mathbf{h}_D, \hat{\mathbf{z}})/\tau)} \tag{9.106}$$

其中 $s(\cdot,\cdot)$ 表示余弦相似度，$\tau$ 表示温度参数，$B$ 表示当前 Mini-batch。相比起推荐损失只利用了离散 Token 进行监督，PSA 可以直接利用连续语义 Embedding 进行优化，使得用户兴趣表示与物品语义空间保持一致，同时也能进一步更新 Tokenizer 中的参数，实现两部分模型的联合优化。

### 9.6.2.3 ETEGRec 算法的交替优化策略

在 ETEGRec 算法中，如果直接同时优化 Tokenizer 和生成式推荐模型，由于两部分参数始终处于动态变化过程中，容易导致训练过程震荡甚至无法收敛。因此，ETEGRec 算法并没有采用完全联合训练，而是提出了 Alternating Optimization（交替优化）的策略。当优化 Tokenizer 时，固定生成式推荐模型，仅更新 Tokenizer 参数，其优化目标为：

$$L_{IT} = L_{SQ} + \mu L_{SIA} + \lambda L_{PSA} \tag{9.107}$$

其中，$L_{SQ}$ 表示 RQ-VAE 语义量化损失，$L_{SIA}$ 和 $L_{PSA}$ 分别表示两种表示对齐损失。

ETEGRec 算法随后固定 Tokenizer，只更新生成式推荐模型。其优化目标表示为：

$$L_{GR} = L_{REC} + \mu L_{SIA} + \lambda L_{PSA} \tag{9.108}$$

ETEGRec 的整个训练过程按照多个轮次交替进行。在每个轮次开始的时候，首先优化 Tokenizer 的部分，使得语义 Token 能够适应当前的推荐模型；随后会冻结 Tokenizer 模块，只通过若干个 Epoch 来训练生成式推荐模型的部分。当 Tokenizer 的部分逐渐收敛之后，ETEGRec 算法会将其永久冻结，只继续训练生成式推荐模型直到其最终收敛为止。这种交替优化策略避免了两个模块同时更新带来的训练不稳定问题，同时又能够实现 Tokenizer 与生成式推荐模型之间的持续协同优化。

## 9.6.3 UniSID 算法

除了 DIGER 试图解决“语义 ID 与生成式推荐模型联合训练”的问题之外，在传统两阶段生成式推荐框架中，语义 ID 本身的生成过程通常也是一种级联式架构。一般而言，需要首先利用文本大模型或多模态大模型对物品

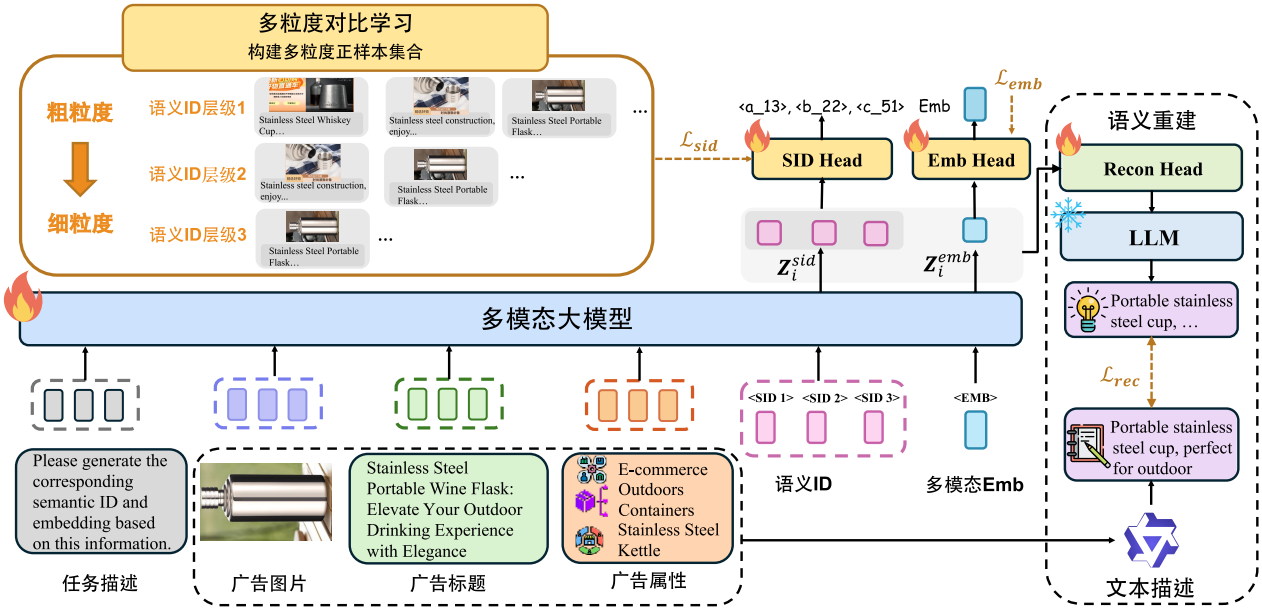


图 9.8: UniSID 算法框架。

内容进行理解，得到 Item Embedding，然后再通过 VQ、RQ-VAE 等向量量化方法将连续 Embedding 压缩为离散语义 ID。因此，整个过程实际上经历了两次信息变换：

$$\text{Raw Item} \rightarrow \text{Embedding} \rightarrow \text{Semantic ID} \tag{9.109}$$

这种两阶段压缩过程不仅会带来量化误差累积，还会导致 Embedding 学习目标与语义 ID 学习目标之间存在优化不一致的问题。由于语义 ID 的生成过程依赖于预训练好的 Embedding，后续语义 ID 的优化无法反向影响底层内容表示，从而造成一定的信息损失。

为此，腾讯在广告推荐场景提出了 End-to-End Semantic ID Generation for Generative Advertisement Recommendation（UniSID）[11]，尝试完全摆脱传统“Embedding → RQ-VAE → Semantic ID”的级联压缩过程，实现真正意义上的一阶段语义 ID 生成框架，如图9.8所示。与传统方法不同，UniSID 利用统一的多模态大模型（MLLM）同时生成 Item Embedding 和 Semantic ID。给定广告物料的输入序列：

$$X_i = [x_{\text{instruction}}, x_{\text{image}}, x_{\text{text}}, x_{\text{attribute}}, x_{\text{sid}}, x_{\text{emb}}] \tag{9.110}$$

通过共享的多模态大模型编码器 (MLLM Encoder) 得到上下文化表示：

$$Z_i = MLLM(X_i) \tag{9.111}$$

然后分别提取语义 ID(Semantic ID, SID) Token 和 Embedding Token 对应位置的隐藏状态 $Z_i^{\text{SID}}$ 与 $Z_i^{\text{Emb}}$，并通过两个独立的线性投影头生成 SID 表示和物品 Embedding：

$$z_i^{\text{SID}} = f_{\text{SID}}(Z_i^{\text{SID}}) \tag{9.112}$$

$$z_i^{\text{Emb}} = f_{\text{Emb}}(Z_i^{\text{Emb}}) \tag{9.113}$$

对于包含 $L$ 层语义层级的 SID，UniSID 将 $z_i^{\text{SID}}$ 分解为多个层级的语义表示：

$$z_i^{\text{SID}} = \left(z_i^1, z_i^2, \cdots, z_i^L\right) \tag{9.114}$$

并通过最大值选择生成对应层级的语义 Token：

$$s_i^l = \arg\max(z_i^l), \quad l = 1, \cdots, L \tag{9.115}$$

最终得到完整的 SID：

$$s_i = (s_i^1, s_i^2, \cdots, s_i^L) \tag{9.116}$$

需要强调的是，UniSID 算法在生成 SID 的时候，实际上也并没有用到 codebook 来做最近邻查找，只是通过自身 embedding 内部的 argmax 来选出取值最大的维度。这种方式得到的语义 ID 只关注不同维度之间取值的大小顺序，而不关心具体的绝对数值。因此，在生成语义 ID 方面，这种基于 $\arg\max$ 操作的方式会具有一定的鲁棒性。

除此之外，相比起传统 RQ-VAE 的残差量化方式，UniSID 实现了

$$\text{Raw Item} \rightarrow \text{Semantic ID} \tag{9.117}$$

的一阶段端到端映射，从而避免了两阶段压缩所带来的信息损失。

为了增强不同层级 ID 的判别能力，UniSID 还引入了多粒度对比学习（Multi-granularity Contrastive Learning）机制。由于不同层级的 SID 对应不同粒度的语义信息，因此不同层级的正样本定义也不相同。浅层 SID 更关注粗粒度语义，而深层 SID 则关注更加细粒度的语义信息。

对于第 $l$ 层 SID，定义正样本集合为 $P_l$，候选集合为 $A_l$，则对应的多粒度对比学习目标为：

$$L_{sid} = \frac{1}{L}\sum_{l=1}^{L}\left(-\frac{1}{|P_l|}\sum_{p\in P_l}\log\frac{\exp(sim(z_i^l z_p^l)/\tau)}{\sum_{a\in A_l}\exp(sim(z_i^l, z_a^l)/\tau)}\right) \tag{9.118}$$

其中 $sim(\cdot,\cdot)$ 表示余弦相似度，$\tau$ 为温度参数。这种多粒度对比学习机制使得不同层级的 SID 分别学习对应粒度的语义信息，从而形成从粗粒度到细粒度的层次化语义结构，避免浅层语义信息泄露到深层语义空间，增强不同层级 SID 的语义解耦能力。

除了 SID 的监督之外，UniSID 对 Embedding 分支同样采用标准对比学习进行优化：

$$L_{emb} = -\log\frac{\exp(sim(z_i^{\text{Emb}}, z_j^{\text{Emb}})/\tau)}{\sum_{k\in N_i}\exp(sim(z_i^{\text{Emb}}, z_k^{\text{Emb}})/\tau)} \tag{9.119}$$

其中 $N_i$ 表示负样本集合。

进一步地，为了使 SID 学习更加抽象的高层语义信息，UniSID 引入了 Summary-based Reconstruction 机制。首先利用冻结的大语言模型对广告属性进行语义摘要：

$$s_i^{\text{sum}} = LLM_{\text{sum}}(\text{Prompt}_{\text{sum}}, x_i^{\text{att}}), \tag{9.120}$$

得到隐藏在原始属性之外的高层语义描述。随后，将 SID 表示和 Embedding 表示进行拼接：

$$Z_i = [Z_i^{\text{SID}}; Z_i^{\text{Emb}}] \tag{9.121}$$

经过重建头得到隐藏状态：

$$h_i^{\text{rec}} = f_{\text{rec}}(Z_i) \tag{9.122}$$

再通过 LLM 在 Next-Token Prediction 范式下重建语义摘要：

$$L_{rec} = \sum_t \log p\left(s_{i,t}^{\text{sum}} \mid h_i^{\text{rec}}, s_{i,<t}^{\text{sum}}\right) \tag{9.123}$$

通过这种摘要重建机制，SID 不再仅仅表示原始属性信息，而是被迫编码更加抽象、更具判别性的高层语义，从而提升复杂广告场景下的语义表达能力。

最终，UniSID 将多粒度 SID 对比学习、Embedding 对比学习以及摘要重建任务进行联合优化：

$$L_{total} = L_{sid} + L_{emb} + \lambda L_{rec} \tag{9.124}$$

其中 $\lambda$ 为重建损失权重。总体来看，UniSID 实现了从原始广告特征到语义 ID 与内容 Embedding 的端到端式联合训练，使得底层内容理解、语义 ID 生成以及高层语义表达能够在统一框架下协同优化，从而避免传统两阶段压缩带来的目标不一致和信息损失问题，为一阶段生成式推荐框架提供了一种新的实现思路。

本节中介绍的 DIGER 算法和 UniSID 算法分别从两种不同的视角下来进行一阶段式的生成式推荐训练，包

括从大模型内容 Embedding 到语义 ID 的统一训练，再到语义 ID 生成与生成式推荐模型的一阶段式训练，生成式推荐框架正在逐渐从“先压缩、后生成”的级联范式演进到真正意义上的端到端联合优化范式。这种一阶段生成式推荐框架使得语义 ID 的学习目标能够与下游推荐目标保持一致，有望进一步释放生成式推荐模型的表达能力。目前，这一方向仍处于快速发展阶段，如何在保持语义 ID 可解释性和高利用率的同时实现稳定训练和高效推理，仍然是未来生成式推荐领域的重要研究方向。

## 9.7 生成式推荐落地问题

尽管生成式推荐模型能够以端到端的方式直接生成推荐结果，但在实际工业落地过程中仍面临诸多挑战，包括曝光偏差导致的数据循环问题、候选集聚集度过高的问题以及自回归生成带来的推理效率问题等。首先，生成式推荐系统容易受到曝光偏差（Exposure Bias）所带来的数据循环（Data Feedback Loop）问题的影响，如下图9.9所示。由于模型会倾向于生成历史上表现较好的内容，当某一类物品（例如以点击率或停留时长为优化目标的营销号内容）在模型中获得较高概率时，这类内容便更容易在推荐结果中获得曝光。随着这些内容不断被用户消费，其行为反馈又会进一步进入用户历史序列，从而在下一轮模型训练或推理过程中继续强化模型对该类内容的偏好。长期迭代之后，不仅用户容易陷入信息茧房（Filter Bubble），模型本身也会陷入数据循环，逐渐丧失对内容空间的探索能力，形成越来越严重的学习偏差。

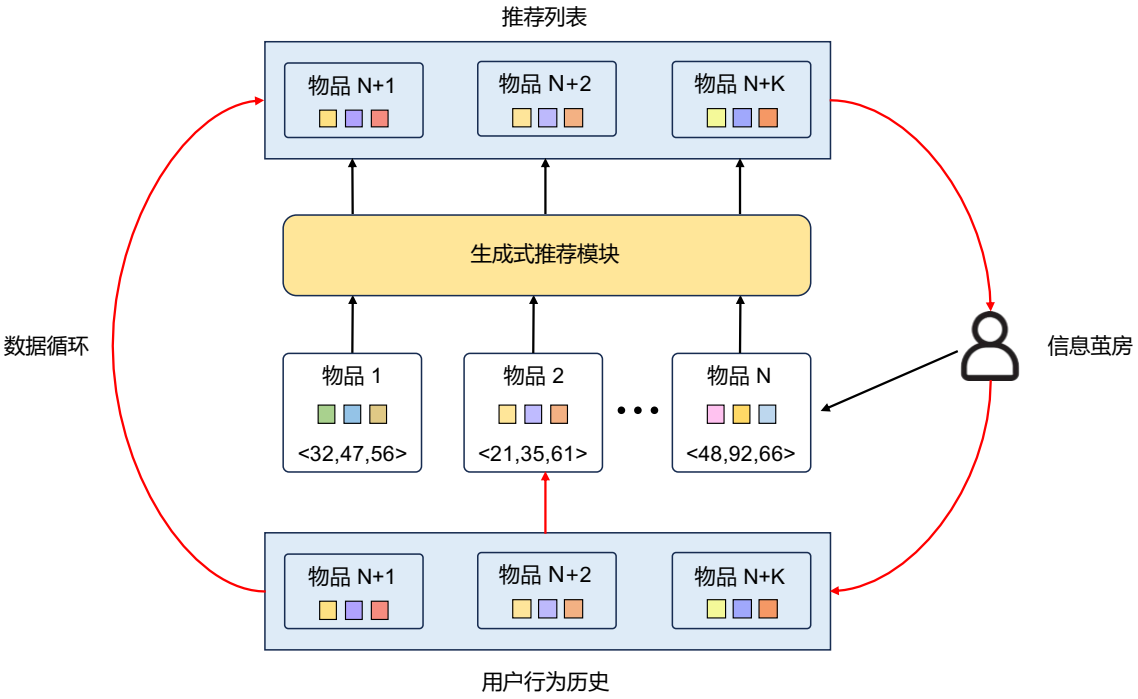


**图 9.9:** 生成式推荐的数据循环问题。

因此，在生成式推荐模型的训练过程中，不能简单地将用户行为序列中的高反馈物品直接视为正样本，而需要同时考虑曝光偏差、探索与利用（Exploration and Exploitation）之间的平衡关系，以及平台内容生态的整体价值。例如，需要进一步区分内容垂类、营销号属性、内容质量等因素，通过更加细粒度的样本构建策略和目标设计，降低模型在训练和推理阶段产生的偏差，从而提升推荐结果的长期用户价值和生态价值。

除此之外，生成式推荐链路还容易面临候选集聚集度过高的问题。在传统级联式推荐系统中，多样性约束通常在精排、重排以及后处理模块中完成，通过多样性打散、类别约束等策略保证最终结果的丰富性。而生成式推荐模型往往采用端到端的方式直接生成候选集，缺少显式的多样性控制机制，因此更容易出现生成结果集中于少数类别或相似内容的问题。

一种常见的解决方案是在 Decoder 的生成过程以及后续的 Beam Search 搜索过程中引入多样性约束。例如，

可以适当增大搜索的束大小，扩大搜索空间；也可以在路径评分过程中加入类别覆盖率、新颖性、多样性等约束项，并结合剪枝策略保留更加丰富的候选路径。从策略算法的角度来看，生成式推荐模型实际上给出了语义 ID 的概率分布，而 Beam Search 则提供了一种在概率空间中搜索最优解的方法。因此，将传统推荐系统中的多样性打散策略、探索策略以及业务约束融入 Beam Search 的搜索过程，本质上能够在一定程度上缓解候选集聚集度过高的问题。

除了算法层面的挑战之外，生成式推荐模型在工程实现方面也面临着较大的实时性压力。目前大部分生成式推荐模型借鉴了大语言模型的结构，采用 Next-Token Prediction（NTP）的训练范式，并通过自回归方式逐步生成 Token 序列。例如生成长度为 $T$ 的序列，生成式推荐模型就需要执行 $T$ 次前向计算，其推理延迟与生成长度线性相关。在聊天机器人 (AI ChatBot) 等场景下，比如用户在使用 ChatGPT 或者 DeepSeek 的时候，通常可以接受数秒甚至十几秒的响应时间。但工业级推荐系统对于延迟的要求通常只有几百毫秒，而单个推荐模块的耗时往往需要控制在几十毫秒以内。因此，自回归逐 Token 生成的方式在推理速度上存在天然劣势，很难直接满足工业级推荐系统对于低延迟服务的要求。

近年来，一类基于扩散模型的生成方法：Diffusion Language Model（DLM）[18] 开始受到广泛关注。与自回归模型按照时间顺序逐个生成 Token 不同，DLM 借鉴图像扩散模型的思想，将离散 Token 序列的生成过程建模为一个逐步去噪（Denoising）过程。具体而言，在训练阶段，首先向真实 Token 序列不断注入噪声，得到不同噪声程度下的中间状态；而在推理阶段，则从随机噪声或者完全 Mask 的序列开始，通过多个去噪步骤逐渐恢复出最终序列：

$$x_T \to x_{T-1} \to \cdots \to x_1 \to x_0 \tag{9.125}$$

其中 $x_0$ 表示最终生成的 Token 序列。

由于每一步去噪过程都可以同时预测多个位置上的 Token，因此 DLM 天然具备并行生成能力，不需要像自回归模型一样严格按照 Token 顺序逐个生成。相比于 Next-Token Prediction，扩散语言模型能够显著减少推理过程中的串行依赖，提高生成速度。然而，由于离散 Token 上的扩散过程设计较为复杂，同时推理过程仍然需要经历多个去噪步骤，因此目前 DLM 仍处于快速发展阶段，在生成质量和推理效率之间尚未达到最优平衡。

相比于完全放弃自回归结构，另一条更加务实的技术路线是多 Token 预测（Multi-token Prediction, MTP）[8]。MTP 的核心思想是在保留自回归框架的基础上，让模型在一次前向传播中同时预测多个未来 Token，从而减少生成所需的前向计算次数。传统 Next-Token Prediction 的训练目标为：

$$L_{NTP} = -\log P(x_t | x_{<t}) \tag{9.126}$$

而 Multi-token Prediction 则同时预测未来 $K$ 个 Token：

$$L_{MTP} = -\sum_{i=0}^{K-1} \log P(x_{t+i} | x_{<t}) \tag{9.127}$$

即模型不仅学习下一个 Token 的概率分布，还学习未来多个 Token 的联合预测能力。

在推理阶段，传统自回归模型每次仅生成一个 Token：

$$x_t \to x_{t+1} \to x_{t+2} \tag{9.128}$$

而 MTP 模型则可以一次生成多个 Token：

$$x_t \to (x_{t+1}, x_{t+2}, ..., x_{t+K}) \tag{9.129}$$

从而将原本需要 $T$ 次前向传播的生成过程缩减为约 $\frac{T}{K}$ 次，大幅降低推理延迟。

近年来，诸如 Medusa [3]、Hydra [1]、Eagle [14]、DeepSeek-V3 [15] 等模型都在不同程度上采用了多 Token 预测或者推测解码（Speculative Decoding）[4] 技术来提升生成速度。对于生成式推荐模型而言，多 Token Prediction 同样具有重要意义。由于一个物品通常由多个语义 Token 组成，例如：

$$item_i = (s_i^1, s_i^2, s_i^3) \tag{9.130}$$

传统自回归生成方式需要依次生成：

$$s_i^1 \rightarrow s_i^2 \rightarrow s_i^3 \tag{9.131}$$

而在 Decoder 中引入多 Token 预测头之后，可以通过 $Decoder(h_t)$ 的一次前向计算同时预测多个语义 Token，从而一次性生成完整的语义 ID，显著降低在线推理延迟，提高系统吞吐能力。

进一步地，MTP 还可以与 Beam Search、Speculative Decoding 等加速技术结合。例如，在生成第一个语义 Token 的同时预测后续若干 Token，并利用主模型进行验证，从而减少 Decoder 的串行生成步骤。这类方法本质上是在保持自回归建模能力的同时，通过增加并行度来提升推理效率，因此被认为是生成式推荐模型未来的重要优化方向。

总体来看，无论是基于扩散模型的 DLM，还是基于自回归框架扩展的 MTP，其核心目标都是打破传统 NTP 的严格串行生成方式，从而提高生成效率。如何在生成质量、推理速度以及系统复杂度之间取得平衡，仍然是当前生成式推荐领域的重要研究方向。目前，无论是工业界还是学术界，对于高效生成范式、多 Token 并行预测以及非自回归生成模型等方向仍处于持续探索阶段，尚未形成统一且成熟的解决方案。

## 9.8 生成式推荐讨论

虽然生成式推荐（Generative Recommendation）近年来受到了学术界和工业界的广泛关注，并在推荐系统中取得了一系列令人瞩目的进展，但与此同时，一个值得深入思考的问题是：

笔记 对于推荐系统这样一个复杂的 AI 驱动系统工程而言，One-Model 范式真的能够完全替代传统级联式推荐架构吗？

事实上，类似的讨论也出现在近年来快速发展的具身智能（Embodied AI）领域。随着 VLA（Vision-Language-Action）[2, 12] 等端到端模型的兴起，越来越多的研究尝试利用单一模型同时完成环境感知、语义理解、任务规划以及动作决策等多个环节，希望通过统一建模实现机器人系统的端到端优化。然而，对于具有实际工业模型训练经验的研究者而言，一个模型同时承担多个目标往往意味着需要在优化过程中兼顾多种甚至相互冲突的任务需求。

从优化视角来看，当一个模型需要同时最小化多个目标函数时，不同任务之间往往会产生梯度冲突（Gradient Conflict）。某一目标的优化方向可能会削弱另一目标的性能，从而导致模型在训练过程中面临复杂的 Trade-off 问题，并最终收敛到某种折中的局部最优解。虽然近年来出现了诸如 PCGrad [21]、GradNorm [5] 等多任务优化方法来缓解梯度冲突问题，但在复杂系统工程场景下，多目标协同优化依然是一个开放性研究课题。

推荐系统同样属于典型的复杂多目标优化系统。一个工业级推荐平台不仅需要最大化用户体验相关指标，例如点击率（CTR）、观看时长（Watch Time）、用户留存（Retention）等，同时还需要兼顾内容生态健康、新作者成长、内容多样性，以及广告、直播、电商等商业化业务的发展需求。这些目标之间并非完全一致，甚至在许多情况下存在天然冲突。例如，提高广告曝光比例可能有利于商业收益，却可能损害用户体验；过度追求短期点击率则可能影响内容生态的长期健康发展。

更重要的是，这些复杂的系统目标往往难以被统一表达为单一的损失函数。因此，从工程实践角度来看，推荐系统不仅仅是一个模型优化问题，更是一个涉及业务目标管理、流量治理、生态调控以及风险控制的复杂系统工程问题。

不可否认的是，生成式推荐的发展为推荐系统带来了新的可能性。借助大语言模型（LLM）强大的序列建模能力，生成式推荐已经在候选召回、粗排排序以及用户兴趣建模等场景展现出显著优势，甚至在部分场景下实现了召回与粗排阶段的一体化建模。这种端到端建模方式有效减少了传统推荐链路中的人工特征工程和模块间的信息损失，为推荐系统架构演进提供了新的方向。

然而，从当前工业实践来看，生成式推荐更有可能成为传统推荐架构的重要补充，而非完全替代者。一方面，复杂的多目标优化需求决定了推荐系统仍然需要大量业务规则、流量治理策略以及生态调控机制参与决策

过程；另一方面，工业级推荐系统对于稳定性、可解释性以及容灾能力有着极高要求。传统级联架构中的召回、排序、混排以及后处理等模块天然形成了多层防护机制，即使某一模块出现异常，其影响范围通常也能够被限制在可控范围内。而在 One-Model 架构下，系统决策高度集中于单一模型，一旦模型出现训练异常、数据漂移、服务故障或预测失真等问题，其影响往往会被放大至整个推荐链路。

因此，从目前的发展阶段来看，生成式推荐所代表的端到端建模思想无疑是推荐系统未来的重要发展方向，但其更可能以“生成式能力融入级联架构”的形式逐步落地，而非完全取代传统推荐系统。对于推荐系统初学者、从业工程师以及学术研究者而言，在关注新技术突破的同时，也需要充分认识到工业级推荐系统背后所蕴含的复杂系统工程属性。传统级联架构能够在工业界长期存在并持续演进，其背后不仅是历史积累的结果，更体现了对多目标优化、系统稳定性以及业务可控性等问题的深刻权衡。

## 9.9 本章小结

在大模型技术快速发展的背景下，推荐系统领域也受到生成式建模思想的深刻影响，逐渐形成了以大语言模型为基础的生成式推荐（Generative Recommendation）范式。本章首先介绍了生成式推荐的基本思想，并分析了其与传统级联式推荐系统在建模方式和系统架构上的联系与区别。

随后，我们从系统工程的视角讨论了生成式推荐模块在工业界的几种典型部署方式，包括与传统推荐链路并行的候选生成架构，以及通过流量隔离实现独立推荐的架构。在前一种模式下，生成式推荐模块可以看作是一个打通召回到排序的统一模块；而在后一种模式下，生成式推荐模块则直接承担部分流量的推荐决策任务。

在此基础上，本章围绕生成式推荐系统的三个核心组成部分进行了详细介绍，包括语义 ID 生成（Semantic ID Construction）、生成式推荐模型（Generative Recommendation Model）以及物品 ID 检索（Item Retrieval）。在语义 ID 生成部分，我们介绍了 RQ-VAE、RQ-KMeans 以及 PQ、FSQ、LFQ 等离散 Tokenizer 方法，并分析了实际工程中经常遇到的梯度不可回传、训练不稳定以及码本坍塌等问题。在生成式推荐模型部分，我们介绍了 Transformer 架构以及 Next-Token Prediction（NTP）的训练范式；在物品 ID 检索部分，则讨论了基于 Encoder-Decoder 架构的自回归生成过程，以及通过 Beam Search、Hash 索引、ANN 检索等方式完成从语义 ID 到真实物品 ID 的映射过程。

除了传统的两阶段生成式推荐框架之外，本章还介绍了近年来出现的一阶段生成式推荐研究进展。首先介绍了 DIGER 算法，通过引入 Gumbel 噪声以及可微离散化机制，实现语义 ID 生成模块与生成式推荐模型之间的端到端联合训练；随后介绍了 UniSID 算法，通过统一的多模态大模型框架，联合学习内容 Embedding 与语义 ID，打破传统“Embedding→RQ-VAE→Semantic ID”的两阶段生成流程。这些工作为解决生成式推荐中的目标不一致问题以及信息损失问题提供了新的思路。

接着，我们进一步讨论了生成式推荐在实际落地过程中面临的一些挑战，包括曝光偏差带来的数据循环问题、候选物品聚集度过高的问题，以及自回归生成导致的在线推理耗时过高等问题。针对这些问题，我们分别从探索机制、多样性约束以及生成效率优化等角度进行了分析，并介绍了近年来出现的 Diffusion Language Model（DLM）、Multi-token Prediction（MTP）等加速生成技术，为提升生成式推荐模型的在线服务能力提供了可能。

最后，我们讨论了生成式推荐这种 One Model 推荐范式是否能够完全替代传统级联式推荐系统。生成式推荐通过统一建模的方式展现出了巨大的潜力，但传统推荐系统经过多年工业实践积累，在实时性、可解释性、稳定性以及复杂业务策略支持等方面依然具有显著优势。因此，对于生成式推荐的发展，我们既应积极拥抱新的技术范式，也应以理性和审慎的视角看待其局限性。在可以预见的未来，生成式推荐与传统级联式推荐系统更可能以长期共存、相互融合的方式共同推动推荐系统的发展，而非简单地相互替代。

## 9.10 参考文献

[1] Zachary Ankner et al. “Hydra: Sequentially-dependent draft heads for medusa decoding”. In: *arXiv preprint arXiv:2402.05109* (2024).

[2] Anthony Brohan et al. “Rt-2: Vision-language-action models transfer web knowledge to robotic control, 2023”. In: *URL https://arxiv. org/abs/2307.15818* 1 (2024), p. 2.

[3] Tianle Cai et al. “Medusa: Simple llm inference acceleration framework with multiple decoding heads”. In: *arXiv preprint arXiv:2401.10774* (2024).

[4] Charlie Chen et al. “Accelerating large language model decoding with speculative sampling”. In: *arXiv preprint arXiv:2302.01318* (2023).

[5] Zhao Chen et al. “Gradnorm: Gradient normalization for adaptive loss balancing in deep multitask networks”. In: *International conference on machine learning*. PMLR. 2018, pp. 794–803.

[6] Jiaxin Deng et al. “OneRec: Unifying Retrieve and Rank with Generative Recommender and Iterative Preference Alignment”. In: 2025.

[7] Junchen Fu et al. “Differentiable Semantic ID for Generative Recommendation”. In: *arXiv preprint arXiv:2601.19711* (2026).

[8] Fabian Gloeckle et al. “Better & faster large language models via multi-token prediction”. In: *arXiv preprint arXiv:2404.19737* (2024).

[9] Herve Jegou, Matthijs Douze, and Cordelia Schmid. “Product quantization for nearest neighbor search”. In: *IEEE transactions on pattern analysis and machine intelligence* 33.1 (2010), pp. 117–128.

[10] Jian Jia et al. “From principles to applications: A comprehensive survey of discrete tokenizers in generation, comprehension, recommendation, and information retrieval”. In: *arXiv preprint arXiv:2502.12448* (2025).

[11] Jie Jiang et al. “End-to-End Semantic ID Generation for Generative Advertisement Recommendation”. In: *arXiv preprint arXiv:2602.10445* (2026).

[12] Moo Jin Kim et al. “Openvla: An open-source vision-language-action model”. In: *arXiv preprint arXiv:2406.09246* (2024).

[13] Doyup Lee et al. “Autoregressive image generation using residual quantization”. In: *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 2022, pp. 11523–11532.

[14] Yuhui Li et al. “Eagle: Speculative sampling requires rethinking feature uncertainty”. In: *arXiv preprint arXiv:2401.15077* (2024).

[15] Aixin Liu et al. “Deepseek-v3 technical report”. In: *arXiv preprint arXiv:2412.19437* (2024).

[16] Enze Liu et al. “Generative recommender with end-to-end learnable item tokenization”. In: *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2025, pp. 729–739.

[17] Fabian Mentzer et al. “Finite Scalar Quantization: VQ-VAE Made Simple”. In: *International Conference on Learning Representations*. 2023. DOI: 10.48550/arXiv.2309.15505.

[18] Shen Nie et al. “Large language diffusion models”. In: *Advances in Neural Information Processing Systems* 38 (2026), pp. 50608–50646.

[19] Aaron Van Den Oord, Oriol Vinyals, et al. “Neural discrete representation learning”. In: *Advances in neural information processing systems* 30 (2017).

[20] Lijun Yu et al. “Language model beats diffusion-tokenizer is key to visual generation”. In: *International Conference on Learning Representations*. Vol. 2024. 2024, pp. 765–783.

[21] Tianhe Yu et al. “Gradient surgery for multi-task learning”. In: *Advances in neural information processing systems* 33 (2020), pp. 5824–5836.