arXiv:2305.05065v1 [cs.IR] 8 May 2023

Recommender Systems with Generative Retrieval

Shashank Rajput*†, Nikhil Mehta*‡, Anima Singh‡, Raghunandan Keshavan‡, Trung Vu‡, Lukasz Heldt‡, Lichan Hong‡, Yi Tay‡, Vinh Q. Tran‡, Jonah Samost‡, Maciej Kula‡, Ed H. Chi‡, Maheswaran Sathiamoorthy‡
†University of Wisconsin-Madison, ‡Google
United States

ABSTRACT

Modern recommender systems leverage large-scale retrieval models consisting of two stages: training a dual-encoder model to embed queries and candidates in the same space, followed by an Approximate Nearest Neighbor (ANN) search to select top candidates given a query's embedding. In this paper, we propose a new single-stage paradigm: a generative retrieval model which autoregressively decodes the identifiers for the target candidates in one phase. To do this, instead of assigning randomly generated atomic IDs to each item, we generate Semantic IDs: a semantically meaningful tuple of codewords for each item that serves as its unique identifier. We use a hierarchical method called RQ-VAE to generate these codewords. Once we have the Semantic IDs for all the items, a Transformer based sequence-to-sequence model is trained to predict the Semantic ID of the next item. Since this model predicts the tuple of codewords identifying the next item directly in an autoregressive manner, it can be considered a generative retrieval model. We show that our recommender system trained in this new paradigm improves the results achieved by current SOTA models on the Amazon dataset. Moreover, we demonstrate that the sequence-to-sequence model coupled with hierarchical Semantic IDs offers better generalization and hence improves retrieval of cold-start items for recommendations.

KEYWORDS

Recommender Systems, Generative Retrieval, Vector Quantization

1 INTRODUCTION

Recommender systems help users discover content of interest and are ubiquitous in various recommendation domains such as video [3, 9, 48], app [2], product [6, 8], and music [18, 19]. Modern recommender systems adopt a retrieve-and-rank strategy, where a set of viable candidates are selected in the retrieval stage, which are then ranked using a ranker model. Since the ranker model works only on the candidates it receives, it is desired that the retrieval stage emits highly relevant candidates.

There are standard and well-established methods for building retrieval models. Matrix factorization [19] learns query and candidate embeddings in the same space. In order to better capture the non-linearities in the data, dual-encoder architectures [41] (i.e., one tower for the query and another for the candidate) employing inner-product to embed the query and candidate embeddings in the same space have become popular in recent years. To use these models during inference, an index that stores the embeddings for all items is created using the candidate tower. For a given query, its embedding is calculated using the query tower, and then an Approximate Nearest Neighbors (ANN) algorithm is used to choose the nearest candidates. On the other hand, sequential recommenders [6, 11, 17, 23, 31, 46, 49], which explicitly take into account the order of user-item interactions, are also recently popular. They also typically use softmax in the output layer when training and resort to using ANN during inference.

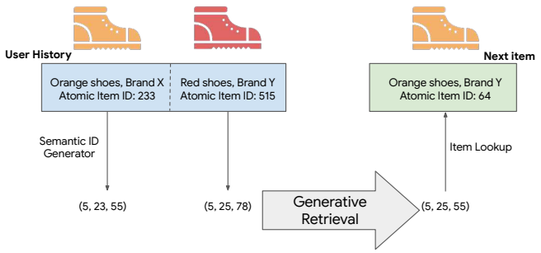


Figure 1: High-level overview of *Transformer Index for GEnerative Recommenders* (TIGER) framework. TIGER proposes representing an item as a tuple of discrete semantic tokens (referred to as Semantic ID), which allows framing the sequential recommendation task as a generative task such that the Semantic ID of the next item is directly predicted using a sequence-to-sequence model.

We propose a new paradigm of building generative retrieval models for sequential recommenders. Instead of traditional query-candidate matching approaches, our method is uses an end-to-end generative model that predicts the candidate IDs directly - dispensing the need for any discrete, non-differentiable inner-product search system or index altogether. Using autoregressive decoding [26, 27] and beam search [32, 40], we can retrieve several viable candidates. In this case, we can interpret the Transformer [36] memory (parameters) as an end-to-end recommendation index, reminiscent of Tay et al. [34]. As such, we name our proposed method TIGER, short for *Transformer Index for GEnerative Recommenders*. A high-level overview of TIGER is shown in Figure 1.

Critically, TIGER is characterized by a new approach to represent each item by a novel "Semantic ID": a sequence of tokens based on the content information about the item (such as its text description). Concretely, given an item's text description, we can use pre-trained text encoders (e.g., SentenceT5 [25]) to generate dense content embeddings. A quantization scheme can then be applied over the embeddings to form a small set of tokens/codewords (integers). We refer to this ordered tuple of codewords as the Semantic ID of the item. We derive inspiration for this idea from human language, where we have words for concepts and string together words to

★ denotes equal contribution.
Correspondence to rajput.shashank11@gmail.com, nikhilmehta@google.com, nlogn@google.com.

采用生成式检索的推荐系统

Shashank Rajput*†, Nikhil Mehta*‡, Anima Singh‡, Raghunandan Keshavan‡, Trung Vu‡, Lukasz Heldt‡, Lichan Hong‡, Yi Tay‡, Vinh Q. Tran‡, Jonah Samost‡, Maciej Kula‡, Ed H. Chi‡, Maheswaran Sathiamoorthy‡
†University of Wisconsin-Madison, ‡Google
United States

摘要

现代推荐系统利用由两个阶段组成的大规模检索模型：首先训练一个双编码器模型，将查询和候选项嵌入同一空间，随后执行近似最近邻（ANN）搜索，根据查询的嵌入选择排名靠前的候选项。在本文中，我们提出一种新的单阶段范式：生成式检索模型在一个阶段中以自回归方式解码目标候选项的标识符。为此，我们不再为每个物品分配随机生成的原子 ID，而是生成语义 ID：由一组具有语义含义的码字构成的元组，作为每个物品的唯一标识符。我们使用一种名为 RQ-VAE 的分层方法生成这些码字。在获得所有物品的语义 ID 后，训练一个基于 Transformer 的序列到序列模型来预测下一个物品的语义 ID。由于该模型以自回归方式直接预测标识下一个物品的码字元组，因此可将其视为生成式检索模型。我们表明，在这一新范式下训练的推荐系统改进了当前 SOTA 模型在 Amazon 数据集上取得的结果。此外，我们证明，序列到序列模型与分层语义 ID 相结合能够实现更好的泛化，从而改善推荐中冷启动物品的检索效果。

关键词

推荐系统、生成式检索、向量量化

1 引言

推荐系统帮助用户发现感兴趣的内容，并广泛应用于视频 [3, 9, 48]、应用 [2]、商品 [6, 8] 和音乐 [18, 19] 等各种推荐领域。现代推荐系统采用"检索后排序"策略：在检索阶段选出一组可行的候选项，随后使用排序模型对其进行排序。由于排序模型只能处理它接收到的候选项，因此希望检索阶段能够输出高度相关的候选项。

构建检索模型已有标准且成熟的方法。矩阵分解 [19] 在同一空间中学习查询嵌入和候选项嵌入。为了更好地捕捉数据中的非线性，近年来采用内积的双编码器架构 [41]（即一个塔用于查询，另一个塔用于候选项）已广受欢迎，它们将查询和候选项嵌入到同一空间。为了在推理期间使用这些模型，需要创建一个索引，以存储

候选塔为所有物品生成的嵌入。对于给定的查询，使用查询塔计算其嵌入，然后采用近似最近邻（ANN）算法选出最近的候选项。另一方面，近年来，显式考虑用户—物品交互顺序的序列推荐器 [6, 11, 17, 23, 31, 46, 49] 也很流行。它们在训练时通常也在输出层使用 softmax，并在推理时转而使用 ANN。

我们提出了一种为序列推荐器构建生成式检索模型的新范式。不同于传统的查询—候选匹配方法，我们的方法采用端到端生成模型直接预测候选 ID，从而完全不再需要任何离散、不可微的内积搜索系统或索引。利用自回归解码 [26, 27] 和束搜索 [32, 40]，我们可以检索出若干可行的候选项。在这种情况下，我们可以将 Transformer [36] 的记忆（参数）解释为一个端到端推荐索引，这与 Tay 等人 [34] 的工作相似。因此，我们将所提出的方法命名为 TIGER，即"生成式推荐器的 Transformer 索引"的简称。图 1 展示了 TIGER 的高层概览。

关键在于，TIGER 的特点是采用一种新方法，用新颖的"语义 ID"表示每个物品：即基于该物品内容信息（例如文本描述）的词元序列。具体而言，给定物品的文本描述，我们可以使用预训练文本编码器（例如 SentenceT5 [25]）生成稠密内容嵌入。随后，可以对这些嵌入应用量化方案，形成一小组词元/码字（整数）。我们将这个有序码字元组称为该物品的语义 ID。这一想法的灵感来自人类语言：我们用词语表示概念，并将词语串联起来以

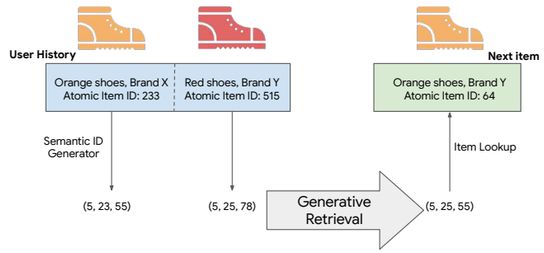
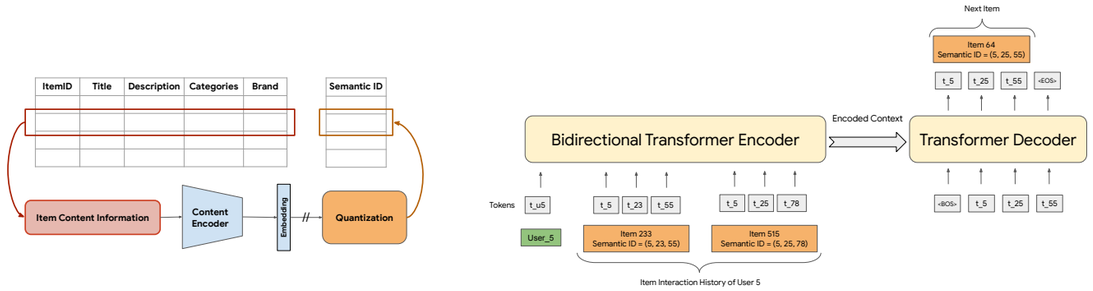


图 1：生成式推荐器的 Transformer 索引（TIGER）框架的高层概览。TIGER 提出将物品表示为离散语义词元的元组（称为语义 ID），从而可以将序列推荐任务构建为生成任务，使序列到序列模型能够直接预测下一个物品的语义 ID。

★表示贡献相同。通信联系邮箱为 rajput.shashank11@gmail.com、nikhilmehta@google.com、nlogn@google.com。



(a) Semantic ID generation for items using quantization over content embeddings derived from content information.

(b) The proposed Transformer based Encoder-Decoder setup for building the sequence-to-sequence model used for generative retrieval.

Figure 2: A high-level overview of the modeling approach used in TIGER.

convey complex ideas. Similarly, we want to develop a language for IDs to represent items. This way, we can represent billions of items using a sequence of limited sets of words (tokens), in contrast to randomly generated atomic IDs requiring one ID per item.

Notably, similar ideas have been adopted when generating images from text (e.g., Parti [43]), where images are represented as tokens using ViT-VQGAN [42]. One crucial difference is that in image generation, a few incorrect tokens lead to a small error or noise in the image. In contrast, a single incorrect token here would mean that the recommender system predicts a different or non-existent item.

There are many benefits to using these Semantic IDs. The number of users and items in modern recommender systems can be in the billions. Thus the embedding tables of ANN-based models can become prohibitively large when using 1:1 mapping between atomic IDs and embedding vectors. Not only does this require a huge memory and storage footprint, but it also leads to imbalance when training the embeddings of the items, where popular items will be over-sampled as compared to infrequent ones [41]. As such, it is common practice to maintain dedicated vocabulary for more popular items and randomly hash remaining items in a fixed set of buckets[16]. However, the hashing scheme leads to random collisions. As new items are introduced in real-world recommendation products, the random collisions could aggravate the cold-start problem. In contrast, our method requires maintaining an embedding table for only a small set of tokens. Furthermore, the collisions in our case are semantically meaningful, which helps mitigate the cold-start problem.

We summarize the main contributions of this work below:

(1) We propose TIGER, a novel generative retrieval-based recommender model, which first assigns unique Semantic IDs to each item, and then trains a retrieval model to predict the Semantic ID of the next item the user will engage with. This provides an alternative to the high-dimensional nearest neighbor search-based or softmax-based recommender systems.
(2) We show that TIGER outperforms existing SOTA recommender systems' recall and NDCG metrics on multiple datasets.
(3) We find that this new paradigm of generative retrieval leads to two extra capabilities in sequential recommender systems: 1. Being able to recommend new and infrequent items, thus improving cold-start problems, and 2. Being able to generate diverse recommendations using a tunable parameter.

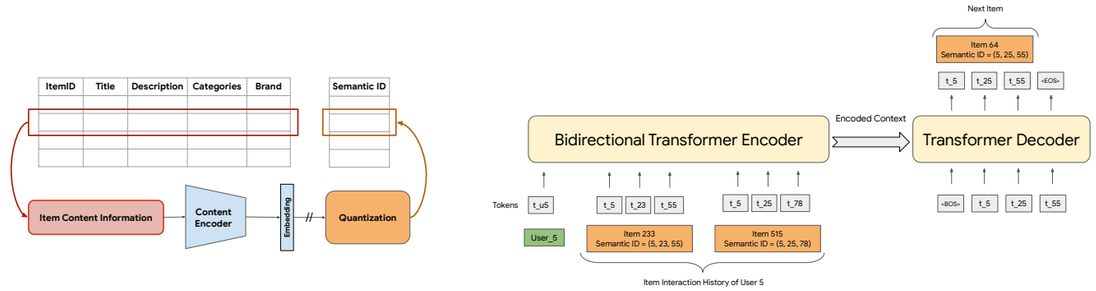
Paper Overview. In Section 2, we provide a brief literature survey of the techniques used in recommender systems, generative retrieval, and the particular Semantic ID generation techniques we use in this paper. In Section 3, we explain our proposed framework, outline the various techniques we use for Semantic ID generation, and provide the results of our experiments in Section 4. We provide further discussions about our work along with some insights in Section 5 and conclude the paper in Section 6.

2 RELATED WORK

Sequential Recommenders. Using deep sequential models in recommender systems has developed into a rich literature. GRU4REC [11] was the first to use GRU based RNNs for sequential recommendations. Li *et al.* [23] proposed NARM (Neural Attentive Session-based Recommendation), where they used the attention mechanism along with a GRU layer to track both the long term and the current intent of the user. AttRec [46] proposed by Zhang *et al.* used the self-attention mechanism to model the user's intent in the current session, and added personalization by modeling user-item affinity separately using metric learning. Concurrently, Kang *et al.* proposed SASRec [17], which used self-attention similar to decoder-only transformer models.

Inspired by the success of masked language modeling in language tasks, BERT4Rec [31] and Transformers4Rec [6] utilize transformer models with masking strategies for sequential recommendation tasks. S^3-Rec [49] goes beyond just masking by pre-training on four self-supervised tasks to improve data representation. In a concurrent work to us, Hou *et al.* propose VQ-Rec [12], which generates "codes" (analogous to Semantic IDs that we use) using content information to represent items. Their focus is on building transferable recommender systems, and do not use the codes in a generative manner for retrieval. While they use product quantization [15] to generate the codes, we use RQ-VAE to generate the Semantic ID.

All of the models described above learn a separate high-dimensional embedding for each item and then perform an ANN or Maximum Inner Product Search (MIPS) to predict the next item. In contrast, our proposed technique, TIGER, uses Generative Retrieval to directly predict the Semantic ID of the next item. P5 [8] and M6 [4] fine-tune pre-trained large language models, to get *multi-task* recommender



(a) 通过对由内容信息得到的内容嵌入进行量化，生成物品的语义 ID。

(b) 所提出的基于 Transformer 的编码器—解码器设置，用于构建生成式检索所使用的序列到序列模型。

图 2：TIGER 中所用建模方法的高层概览。

表达复杂思想。类似地，我们希望开发一种用于表示物品的 ID 语言。这样，我们可以用有限词集合（词元）的序列表示数十亿个物品；相比之下，随机生成的原子 ID 要求每个物品拥有一个 ID。

值得注意的是，在根据文本生成图像（例如 Parti [43]）时也采用了类似思想，其中通过 ViT-VQGAN [42] 将图像表示为词元。一个关键区别是，在图像生成中，少量错误词元只会导致图像中出现较小的误差或噪声。相比之下，此处只要一个词元错误，就意味着推荐系统预测了另一个物品或一个不存在的物品。

使用这些语义 ID 有许多好处。现代推荐系统中的用户和物品数量可能达到数十亿。因此，当原子 ID 与嵌入向量采用 1:1 映射时，基于 ANN 的模型的嵌入表可能变得大到难以承受。这不仅需要巨大的内存和存储空间，还会导致物品嵌入的训练失衡，即热门物品相较于低频物品会被过度采样 [41]。因此，常见做法是为较热门的物品维护专用词表，并将其余物品随机哈希到一组固定的桶中[16]。然而，这种哈希方案会导致随机冲突。随着现实世界的推荐产品不断引入新物品，随机冲突可能加剧冷启动问题。相比之下，我们的方法只需为一小组词元维护嵌入表。此外，在我们的方法中，冲突在语义上是有意义的，这有助于缓解冷启动问题。

我们将本工作的主要贡献概括如下：

(1) 我们提出了 TIGER，这是一种新颖的基于生成式检索的推荐模型。该模型首先为每个物品分配唯一的语义 ID，然后训练检索模型来预测用户下一步将与之交互的物品的语义 ID。

    这为基于高维最近邻搜索或基于 softmax 的推荐系统提供了一种替代方案。

(2) 我们表明，在多个数据集上，TIGER 的召回率和 NDCG 指标均优于现有的 SOTA 推荐系统。
(3) 我们发现，生成式检索这一新范式为序列推荐系统带来了两项额外能力：1. 能够推荐新物品和低频物品，从而改善冷启动问题；2. 能够使用可调参数生成多样化推荐。

论文概述。在第 2 节中，我们简要回顾了推荐系统、生成式检索以及本文所采用的特定语义 ID 生成技术的相关文献。在第 3 节中，我们介绍所提出的框架，概述用于生成语义 ID 的各种技术，并在第 4 节给出实验结果。我们在第 5 节进一步讨论我们的工作并提供一些见解，最后在第 6 节对本文作出总结。

2 相关工作

序列推荐器。使用深度序列模型的推荐系统已形成丰富的研究文献。GRU4REC [11] 是首个将基于 GRU 的 RNN 用于序列推荐的工作。Li 等人 [23] 提出了 NARM（神经注意力会话推荐），其中结合注意力机制与 GRU 层来同时跟踪用户的长期意图和当前意图。Zhang 等人提出的 AttRec [46] 使用自注意力机制对用户在当前会话中的意图进行建模，并通过度量学习单独建模用户与物品的亲和度，从而加入个性化。与此同时，Kang 等人提出了 SASRec [17]，它采用了类似于仅解码器 Transformer 模型的自注意力机制。

受掩码语言建模在语言任务中取得成功的启发，BERT4Rec [31] 和 Transformers4Rec [6] 在序列推荐任务中采用带有掩码策略的 Transformer 模型。S^3-Rec [49] 不仅使用掩码，还在四个自监督任务上进行预训练，以改进数据表示。在一项与我们同期开展的工作中，Hou 等人提出了 VQ-Rec [12]，它利用内容信息生成表示物品的"编码"（类似于我们使用的语义 ID）。他们的重点是构建可迁移的推荐系统，并未以生成方式使用这些编码进行检索。他们使用乘积量化 [15] 来生成编码，而我们使用 RQ-VAE 来生成语义 ID。

上述所有模型都为每个物品学习一个独立的高维嵌入，然后执行 ANN 或最大内积搜索（MIPS）来预测下一个物品。相比之下，我们提出的技术 TIGER 使用生成式检索直接预测下一个物品的语义 ID。P5 [8] 和 M6 [4] 对预训练的大语言模型进行微调，以得到多任务推荐

systems that output token-by-token the recommended item. For P5, the model relies on the tokenizer used by the LLM (SentencePiece tokenizer [29]) to generate tokens out of item IDs that are non-semantic in nature. M6, on the other hand, directly outputs the name of the recommended item, token-by-token. We use a more principled way of generating Semantic IDs, based on content information, that is compatible with most existing sequence-to-sequence models and show in Table 3 that our proposed approach yields much better results than using tuples of random codes. Wang et al. [37] train recommender models that use multi-modal item representations generated from multi-modal side-information, thus allowing them to transfer the model to both text-based and image-based recommendation domains. The predicted items in this framework are also computed using a maximum inner product search as the last step. In [44], the authors show that such side-information-based recommender systems achieve the same performance and sometimes outperform atomic ID based recommender systems.

Generative Retrieval. Generative retrieval is a recently developed approach for Document Retrieval in the NLP community, where the task is to return a set of relevant documents from a database. Document retrieval traditionally involved training a 2-tower model which mapped both queries and documents to the same high-dimensional vector space, followed by performing an ANN or MIPS for the query over all the documents to return the closest ones. This technique presents some disadvantages like having a large embedding table [21, 22]. Generative retrieval is a recently proposed technique that aims to fix some of the issues of the traditional approach by producing token by token either the title, name, or the document id string of the document. Cao *et al.* [5] proposed GENRE for entity retrieval, which used a transformer-based architecture to return, token-by-token, the name of the entity referenced to in a given query. Tay *et al.* [34] proposed DSI for document retrieval, which was the first system to assign structured semantic DocIDs to each document. Then given a query, the model autoregressively returned the DocID of the document token-by-token. The DSI work marks a paradigm shift in IR to generative retrieval approaches and is the first successful application of an end-to-end Transformer for retrieval applications. Subsequently, Lee *et al.* [22] show that generative document retrieval is useful even in the multi-hop setting, where a complex query cannot be answered directly by a single document, and hence their model generates intermediate queries, in a chain-of-thought manner, to ultimately generate the output for the complex query. Wang *et al.* [38] supplement the hierarchical k-means clustering based semantic DocIDs of Tay *et al.* [34] by proposing a new decoder architecture that specifically takes into account the prefixes in semantic DocIDs. In CGR [21], the authors propose a way to take advantage of both the bi-encoder technique and the generative retrieval technique, by allowing the decoder, of their encoder-decoder-based model, to learn separate *contextualized* embeddings which store information about documents intrinsically. To the best of our knowledge, we are the first to use generative Semantic IDs created using an auto-encoder (RQ-VAE [20, 45]) for retrieval models.

Vector Quantization. We refer to Vector Quantization as the process of converting a high-dimensional vector into a low-dimensional tuple of codewords. One of the most straightforward techniques uses hierarchical clustering, such as the one used in [34], where clusters created in a particular iteration are further partitioned into sub-clusters in the next iteration. An alternative popular approach is Vector-Quantized Variational AutoEncoder (VQ-VAE), which was introduced in [35] as a way to encode natural images into a sequence of codes. The technique works by first passing the input vector (or image) through an encoder that reduces the dimensionality. The smaller dimensional vector is partitioned and each partition is quantized separately, thus resulting in a sequence of codes: one code per partition. These codes are then used by a decoder to recreate the original vector (or image).

RQ-VAE [20, 45] applies residual quantization to the output of the encoder of VQ-VAE to achieve a lower reconstruction error. We discuss this technique in more detail in Subsection 3.1.

Locality Sensitive Hashing (LSH) [13, 14] is a popular technique used for clustering and approximate nearest neighbor search. The particular version that we use in this paper for clustering is SimHash [1], which uses random hyperplanes to create binary vectors which serve as hashes of the items. Because it has low computational complexity and is scalable [13], we use this as a baseline technique for vector quantization.

3 PROPOSED FRAMEWORK

Our proposed technique consists of two components:

(1) *Semantic ID generation using content features.* This involves mapping the item content features to embedding vectors, which are further quantized into a tuple of semantic codewords. We call this tuple the item's Semantic ID.
(2) *Training a generative recommender system on Semantic IDs.* A Transformer based sequence-to-sequence model is trained on the sequences of semantic IDs corresponding to the items in the user's interaction history to predict the Semantic ID of the next item in the sequence.

3.1 Semantic ID Generation

In this section, we describe the Semantic ID generation process for the items in the recommendation corpus. We assume that each item has associated content features that capture useful semantic information (*e.g.* titles or descriptions or images). Moreover, we assume that we have access to a pre-trained content encoder to generate a semantic embedding $\boldsymbol{x} \in \mathbb{R}^d$. For example, general-purpose pre-trained text encoders such as Sentence-T5 [25] and BERT [7] can be used to convert an item's text description to embeddings. In this work, we use the textual description of items as content features and use Sentence-T5 [25] encoder on this textual description. The semantic embeddings are then quantized to generate a Semantic ID for each item. Figure 2a gives a high-level overview of the process. This approach is similar to [12], where Hou *et al.* generate embeddings with a BERT encoder but the way they quantize the embeddings is different from ours.

We define a Semantic ID to be a tuple of codewords of length m. Each entry of the tuple, that is, each codeword can come from a different codebook. The number of items that the Semantic IDs can represent uniquely is thus equal to the product of the codebook sizes. While different techniques to generate Semantic IDs result in the codes having different semantic properties or guarantees, we

逐词元输出所推荐物品的系统。对于 P5，模型依赖 LLM 使用的分词器（SentencePiece 分词器 [29]），从本质上不含语义的物品 ID 中生成词元。另一方面，M6 直接逐词元输出所推荐物品的名称。我们采用一种更有原则、基于内容信息且与大多数现有序列到序列模型兼容的语义 ID 生成方法，并在表 3 中表明，与使用随机编码元组相比，我们提出的方法取得了好得多的结果。Wang 等人 [37] 训练使用由多模态辅助信息生成的多模态物品表示的推荐模型，从而能够将模型迁移到基于文本和基于图像的推荐领域。在该框架中，预测物品也以最大内积搜索作为最后一步来计算。在 [44] 中，作者表明，这类基于辅助信息的推荐系统取得了与基于原子 ID 的推荐系统相同的性能，有时甚至优于后者。

生成式检索。生成式检索是 NLP 社区最近为文档检索开发的一种方法，其任务是从数据库中返回一组相关文档。传统文档检索需要训练一个双塔模型，将查询和文档都映射到同一个高维向量空间，然后针对查询在所有文档上执行 ANN 或 MIPS，以返回最接近的文档。这种技术存在一些缺点，例如需要大型嵌入表 [21, 22]。生成式检索是最近提出的一种技术，旨在通过逐词元生成文档的标题、名称或文档 ID 字符串，来解决传统方法的一些问题。Cao 等人 [5] 提出了用于实体检索的 GENRE，它使用基于 Transformer 的架构，逐词元返回给定查询所指实体的名称。Tay 等人 [34] 提出了用于文档检索的 DSI，这是第一个为每个文档分配结构化语义 DocID 的系统。随后，在给定查询后，模型以自回归方式逐词元返回文档的 DocID。DSI 工作标志着 IR 向生成式检索方法的范式转变，也是端到端 Transformer 在检索应用中的首次成功应用。随后，Lee 等人 [22] 表明，即使在多跳场景中，生成式文档检索也很有用；在该场景中，复杂查询无法由单个文档直接回答，因此其模型以思维链方式生成中间查询，最终为复杂查询生成输出。Wang 等人 [38] 针对基于分层 k-means 聚类的语义 DocID 提出一种新的解码器架构，专门考虑其中的前缀，以此补充 Tay 等人 [34] 的方法。在 CGR [21] 中，作者提出一种同时利用双编码器技术和生成式检索技术的方法：允许其基于编码器—解码器模型的解码器学习独立的上下文化嵌入，这些嵌入内在地存储文档信息。据我们所知，我们首次将使用自动编码器（RQ-VAE [20, 45]）创建的生成式语义 ID 用于检索模型。

向量量化。我们将向量量化定义为把高维向量转换为低维码字元组的过程。其中一种最直接的技术

采用层次聚类，例如 [34] 中使用的方法；在该方法中，某次迭代创建的聚类会在下一次迭代中进一步划分为子聚类。另一种流行的方法是向量量化变分自编码器（VQ-VAE），它由 [35] 提出，用于将自然图像编码为编码序列。该技术首先让输入向量（或图像）通过编码器以降低维度。然后将较低维向量划分为多个部分，并分别量化每个部分，从而得到一个编码序列：每个部分对应一个编码。随后，解码器使用这些编码重建原始向量（或图像）。

RQ-VAE [20, 45] 对 VQ-VAE 编码器的输出应用残差量化，以获得更低的重建误差。我们将在第 3.1 小节中更详细地讨论该技术。
局部敏感哈希（LSH）[13, 14] 是一种广泛用于聚类和近似最近邻搜索的技术。本文用于聚类的具体版本是 SimHash [1]，它使用随机超平面创建二进制向量，并将其作为物品的哈希值。由于其计算复杂度低且具有可扩展性 [13]，我们将其用作向量量化的基线技术。

3 所提出的框架

我们提出的技术由两个组件组成：

(1) 使用内容特征生成语义 ID。此过程将物品内容特征映射为嵌入向量，再将其量化为语义码字元组。我们将这个元组称为物品的语义 ID。
(2) 在语义 ID 上训练生成式推荐系统。基于 Transformer 的序列到序列模型使用与用户交互历史中物品相对应的语义 ID 序列进行训练，以预测该序列中下一个物品的语义 ID。

3.1 语义 ID 生成

本节介绍如何为推荐语料库中的物品生成语义 ID。我们假设每个物品都有能够捕获有用语义信息的相关内容特征（例如标题、描述或图像）。此外，我们假设可以使用预训练的内容编码器生成语义嵌入 $\boldsymbol{x} \in \mathbb{R}^d$。例如，Sentence-T5 [25] 和 BERT [7] 等通用预训练文本编码器可用于将物品的文本描述转换为嵌入。在本工作中，我们使用物品的文本描述作为内容特征，并在该文本描述上使用 Sentence-T5 [25] 编码器。随后对语义嵌入进行量化，为每个物品生成一个语义 ID。图 2a 给出了该过程的高层概览。该方法与 [12] 类似，Hou 等人使用 BERT 编码器生成嵌入，但其嵌入量化方式与我们不同。

我们将语义 ID 定义为由 m 个码字组成的元组。元组中的每个条目，即每个码字，都可以来自不同的码本。因此，语义 ID 能够唯一表示的物品数量等于各码本大小的乘积。虽然生成语义 ID 的不同技术会使所得编码具有不同的语义性质或保证，但我们

want them to have the following property in general: *Similar items (items with similar content features or whose semantic embeddings are close) should have overlapping codewords.* For example, an item with Semantic ID $(10, 21, 35)$ should be more similar to one with Semantic ID $(10, 21, 40)$, than an item that is represented as $(10, 23, 32)$. Next, we discuss the quantization scheme RQ-VAE, which is used for Semantic ID generation.

3.1.1 RQ-VAE for Semantic IDs. Residual-Quantized Variational AutoEncoder (RQ-VAE) [45] is a multi-stage vector quantizer that applies quantization on residuals at multiple levels to generate a tuple of codewords (aka Semantic IDs). The Autoencoder jointly trains the codebook and the encoder-decoder to reconstruct the input using only Semantic IDs. Figure 3 illustrates the process of generating Semantic IDs through residual quantization.

RQ-VAE first encodes the input $\boldsymbol{x}$ via an encoder $\mathcal{E}$ to learn a latent representation $\boldsymbol{z} := \mathcal{E}(\boldsymbol{x})$. At the zeroth level ($d = 0$), the initial residual is simply defined as $\boldsymbol{r}_0 := \boldsymbol{z}$. At each level d, we have a codebook $C_d := \{\boldsymbol{e}_k\}_{k=1}^{K}$, where K is the codebook size. Then, $\boldsymbol{r}_0$ is quantized by mapping it to the nearest embedding from that level's codebook. The index of the closest embedding $\boldsymbol{e}_{c_d}$ at $d = 0$, i.e., $c_0 = \arg\min_i \|\boldsymbol{r}_0 - \boldsymbol{e}_k\|$, represents the zeroth codeword. For the next level $d = 1$, the residual is defined as $\boldsymbol{r}_1 := \boldsymbol{r}_0 - \boldsymbol{e}_{c_0}$. Then, similar to the zeroth level, the code for the first level is computed by using the codebook for the first level. This process is repeated iteratively m times to get a tuple of m codewords that represent the Semantic ID. This recursive approach approximates the input from a coarse-to-fine granularity. Note that we chose to use a separate codebook of size K for each of the m levels, instead of using a single, m-times larger codebook. This design choice was motivated to avoid collisions between codewords at different granularity since the average norm of residuals decreases with increasing levels.

Once we have the Semantic ID $(c_0, \ldots, c_{m-1})$, the quantized representation of $\boldsymbol{z}$ is computed as $\widehat{\boldsymbol{z}} := \sum_{d=0}^{m-1} \boldsymbol{e}_{c_i}$. This vector, $\widehat{\boldsymbol{z}}$, is passed to the decoder, which tries to recreate the input $\boldsymbol{x}$ using $\widehat{\boldsymbol{z}}$. The loss that we use to train the RQ-VAE is as follows:

$$\mathcal{L}(\boldsymbol{x}) := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{rqvae}}, \text{ where}$$
$$\mathcal{L}_{\text{recon}} := \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|^2 \text{ and}$$
$$\mathcal{L}_{\text{rqvae}} := \sum_{d=0}^{m-1} \|\text{sg}[\boldsymbol{r}_i] - \boldsymbol{e}_{c_i}\|^2 + \beta\|\boldsymbol{r}_i - \text{sg}[\boldsymbol{e}_{c_i}]\|^2.$$

Here $\widehat{\boldsymbol{x}}$ is the output of the decoder, and sg is the stop-gradient operation [35]. Note that this loss jointly trains the encoder, decoder, and the codebook.

As proposed in [45], to prevent RQ-VAE from a codebook collapse, where most of the input gets mapped to only a few codebook vectors, we use k-means clustering-based initialization for the codebook. More specifically, we apply the k-means algorithm on the first training batch and use the centroids as initialization.

3.1.2 Other options for quantization. A simple alternative to generating Semantic IDs is to use Locality Sensitive Hashing (LSH). We perform an ablation study in Section 4.3.2 where we find that RQ-VAE indeed works better than LSH. Another option is to use k-means clustering hierarchically [34], but it loses semantic meaning between different clusters [38]. We also tried VQ-VAE, and while it performs similarly to RQ-VAE for generating the candidates during retrieval, it loses the hierarchical nature of the IDs which confers many new capabilities as discussed in Section 4.4.

3.1.3 Handling Collisions. For retrieval, we would like to avoid collisions and assign unique IDs for items. Depending on the distribution of semantic embeddings, the choice of codebook size, and the length of codewords, collisions may still occur. For the hyperparameters we used (described in Section 4.1.3), we do observe a few items in the dataset with very similar semantic embeddings have the same Semantic ID assigned to them.

To remove the collisions we append an extra token at the end of the Semantic IDs to make them unique. For example, if two items share the Semantic ID $(12, 24, 52)$, we append extra tokens to differentiate between them, hence they are represented as $(12, 24, 52, 0)$ and $(12, 24, 52, 1)$.

3.2 Generative Retrieval with Semantic IDs

We construct item sequences for every user by sorting chronologically the items they have interacted with. Then, given a sequence of the form $(\text{item}_1, \ldots, \text{item}_n)$, the recommender system's task is to predict the next item item_{n+1}. For this, we propose a generative approach that directly predicts the Semantic IDs of items.

Formally, let $(c_{i,0}, \ldots, c_{i,m-1})$ be the m-length Semantic ID for item_i. Then, we convert the item sequence to the sequence $(c_{1,0}, \ldots, c_{1,m-1}, c_{2,0}, \ldots, c_{2,m-1}, \ldots, c_{n,0}, \ldots, c_{n,m-1})$. The sequence-to-sequence model is then trained to predict the Semantic ID of item_{n+1}, which is $(c_{n+1,0}, \ldots, c_{n+1,m-1})$. Hence, this formulation does not need to make any major modifications to existing sequence-to-sequence model architectures to train them for generative recommendations. Once we have the predicted tuple of codewords $(c_{n+1,0}, \ldots, c_{n+1,m-1})$, we simply look up the item to which this Semantic ID corresponds to. There is a possibility that the generated Semantic ID does not match any item in the dataset. However, as we observe in Figure 6, the probability of such an event is very low.

4 EXPERIMENTS

We conduct exhaustive experiments to answer the following research questions (RQs):

- **RQ1**: How our proposed framework (TIGER) performs on the sequential recommendation task compared to the baseline methods?
- **RQ2**: Is the choice of our item representation with Semantic IDs meaningful?
- **RQ3**: What new recommendation capabilities emerge with this new paradigm?

4.1 Experimental Setup

In this section, we describe the datasets, evaluation metrics, and implementation details of the TIGER framework.

4.1.1 Datasets. We evaluate the proposed framework on three public real-world benchmarks from the Amazon Product Reviews dataset [10], containing user reviews and item metadata from May 1996 to July 2014. In particular, we use three categories of the Amazon Product Reviews dataset for the sequential recommendation

一般而言，我们希望它们具备以下性质：相似的物品（内容特征相似或语义嵌入彼此接近的物品）应具有重叠的码字。例如，语义 ID 为 $(10, 21, 35)$的物品应当比表示为 $(10, 21, 40)$的物品更相似，而不是比表示为 $(10, 23, 32)$的物品更相似。接下来，我们讨论用于生成语义 ID 的量化方案 RQ-VAE。

3.1.1 用于语义 ID 的 RQ-VAE。残差量化变分自动编码器（RQ-VAE）[45]是一种多阶段向量量化器，它在多个层级对残差进行量化，以生成码字元组（即语义 ID）。自动编码器联合训练码本和编码器—解码器，以便仅使用语义 ID 重构输入。图 3 展示了通过残差量化生成语义 ID 的过程。

RQ-VAE 首先对输入 $\boldsymbol{x}$进行编码，所用编码器为 $\mathcal{E}$，以学习潜在表示 $\boldsymbol{z} := \mathcal{E}(\boldsymbol{x})$。在第零层($d = 0$）处，初始残差被直接定义为 $\boldsymbol{r}_0 := \boldsymbol{z}$。在每一层 d，都有一个码本 $C_d := \{\boldsymbol{e}_k\}_{k=1}^{K}$，其中 K 为码本大小。随后，通过将$\boldsymbol{r}_0$映射到该层码本中最近的嵌入来对其进行量化。最近嵌入 $\boldsymbol{e}_{c_d}$在 $d = 0$处的索引，即 $c_0 = \arg\min_i \|\boldsymbol{r}_0 - \boldsymbol{e}_k\|$，表示第零个码字。对于下一层 $d = 1$，残差被定义为 $\boldsymbol{r}_1 := \boldsymbol{r}_0 - \boldsymbol{e}_{c_0}$。随后，与第零层类似，使用第一层的码本计算第一层的编码。这个过程迭代重复 m次，得到由 m个码字组成的元组，用以表示语义 ID。这种递归方法以由粗到细的粒度逼近输入。请注意，我们选择使用大小为 K的独立码本，供 m个层级中的每一层使用，而不是使用一个大 m倍的单一码本。做出这一设计选择是为了避免不同粒度的码字之间发生冲突，因为残差的平均范数会随着层级增加而减小。

得到语义 ID $(c_0, \ldots, c_{m-1})$后，$\boldsymbol{z}$的量化表示计算为 $\widehat{\boldsymbol{z}} := \sum_{d=0}^{m-1} \boldsymbol{e}_{c_i}$。该向量 $\widehat{\boldsymbol{z}}$被传递给解码器，解码器尝试重构输入 $\boldsymbol{x}$，并为此使用 $\widehat{\boldsymbol{z}}$。我们用于训练 RQ-VAE 的损失如下：

$$\mathcal{L}(\boldsymbol{x}) := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{rqvae}}, \text{ where}$$
$$\mathcal{L}_{\text{recon}} := \|\boldsymbol{x} - \widehat{\boldsymbol{x}}\|^2 \text{ and}$$
$$\mathcal{L}_{\text{rqvae}} := \sum_{d=0}^{m-1} \|\text{sg}[\boldsymbol{r}_i] - \boldsymbol{e}_{c_i}\|^2 + \beta\|\boldsymbol{r}_i - \text{sg}[\boldsymbol{e}_{c_i}]\|^2.$$

这里，$\widehat{\boldsymbol{x}}$是解码器的输出，sg 是停止梯度操作 [35]。请注意，该损失会联合训练编码器、解码器和码本。

按照 [45] 中提出的方法，为防止 RQ-VAE 发生码本坍缩，即大多数输入仅被映射到少数几个码本向量，我们采用基于 k-means 聚类的码本初始化方法。更具体地说，我们在第一个训练批次上应用 k-means 算法，并使用聚类中心进行初始化。

3.1.2 其他量化选项。生成语义 ID 的一种简单替代方案是使用局部敏感哈希（LSH）。我们在第 4.3.2 节进行消融研究，结果发现 RQ-VAE 的确优于 LSH。另一种选择是以分层方式使用 k-means 聚类 [34]，但这种方法会丢失不同聚类之间的语义含义 [38]。我们还尝试了 VQ-VAE，尽管它在检索期间生成候选项时的表现与 RQ-VAE 相近，但它会丧失 ID 的层次结构，而这种层次结构能够带来第 4.4 节所讨论的许多新能力。

3.1.3 处理冲突。对于检索，我们希望避免冲突并为物品分配唯一 ID。根据语义嵌入的分布、码本大小的选择以及码字长度，仍可能发生冲突。对于我们使用的超参数（见第 4.1.3 节），我们确实观察到，数据集中少数语义嵌入非常相似的物品被分配了相同的语义 ID。

为消除冲突，我们在语义 ID 末尾附加一个额外词元，使其具有唯一性。例如，如果两个物品共享语义 ID $(12, 24, 52)$，我们会附加额外词元来区分它们，因此它们分别表示为 $(12, 24, 52, 0)$和 $(12, 24, 52, 1)$。

3.2 使用语义 ID 的生成式检索

我们按用户交互物品的时间顺序进行排序，为每位用户构建物品序列。然后，给定形如 (物品$_1$, ..., 物品$_n$)的序列，推荐系统的任务是预测下一个物品，即物品$_{n+1}$。为此，我们提出一种直接预测物品语义 ID 的生成式方法。

形式化地，令 $(c_{i,0}, \ldots, c_{i,m-1})$为长度为 m的语义 ID，用于物品 i。然后，我们将物品序列转换为序列 $(c_{1,0}, \ldots, c_{1,m-1}, c_{2,0}, \ldots, c_{2,m-1}, \ldots, c_{n,0}, \ldots, c_{n,m-1})$。接着训练序列到序列模型来预测物品$_{n+1}$的语义 ID，即 $(c_{n+1,0}, \ldots, c_{n+1,m-1})$。因此，这种建模方式无需对现有序列到序列模型架构进行任何重大修改，即可训练其用于生成式推荐。一旦得到预测的码字元组 $(c_{n+1,0}, \ldots, c_{n+1,m-1})$，我们只需查找该语义 ID 所对应的物品。生成的语义 ID 可能与数据集中的任何物品都不匹配。然而，正如我们在图 6 中观察到的，这种事件发生的概率非常低。

4 实验

我们开展全面的实验，以回答以下研究问题（RQ）：

- **RQ1**：与基线方法相比，我们提出的框架（TIGER）在序列推荐任务上的表现如何？
- **RQ2**：我们选择使用语义 ID 的物品表示是否有意义？
- **RQ3**：这一新范式带来了哪些新的推荐能力？

4.1 实验设置

本节介绍数据集、评估指标以及 TIGER 框架的实现细节。

4.1.1 数据集。我们在 Amazon Product Reviews 数据集 [10] 的三个公开真实世界基准上评估所提出的框架，该数据集包含 1996 年 5 月至 2014 年 7 月的用户评论和物品元数据。具体而言，我们使用 Amazon Product Reviews 数据集中的三个类别进行序列推荐

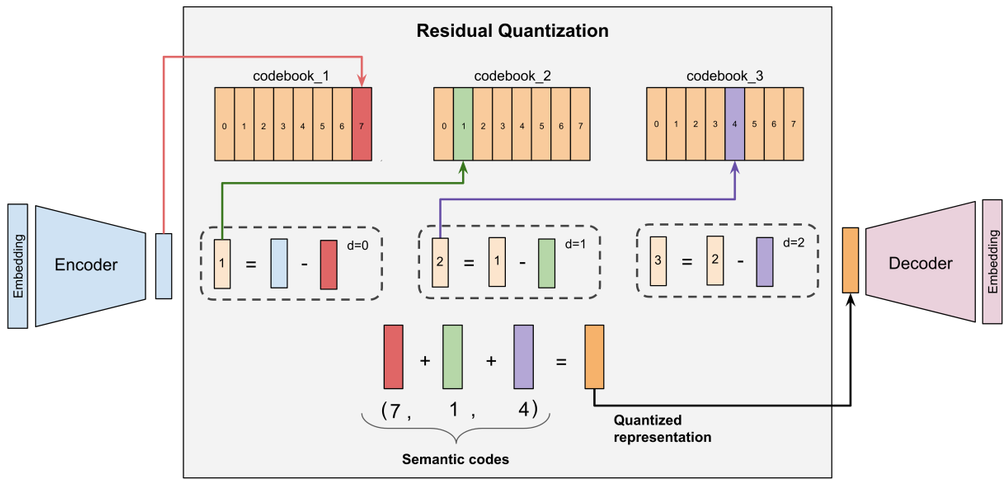


Figure 3: RQ-VAE: In the figure, the vector output by the DNN Encoder, say r_0 (represented by the blue bar), is fed to the quantizer, which works iteratively. First, the closest vector to r_0 is found in the first level codebook. Let this closest vector be e_{c_0} (represented by the red bar). Then, the residual error is computed as $r_1 := r_0 - e_{c_0}$. This is fed into the second level of the quantizer, and the process is repeated: The closest vector to r_1 is found in the second level, say e_{c_1} (represented by the green bar), and then the second level residual error is computed as $r_2 = r_1 - e'_{c_1}$. Then, the process is repeated for a third time on r_2. The semantic codes are computed as the indices of e_{c_0}, e_{c_1}, and e_{c_2} in their respective codebooks. In the example shown in the figure, this results in the code $(7, 1, 2)$.

Table 1: Performance comparison on the sequential recommendation task. The last row depicts the improvement observed with TIGER relative to the best baseline. We use bold and underline to denote the best and the second-best metric.

	Methods	Sports and Outdoors				Beauty				Toys and Games			
		Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
Baselines	P5 [47]	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
	Caser [33]	0.0116	0.0072	0.0194	0.0097	0.0205	0.0131	0.0347	0.0176	0.0166	0.0107	0.0270	0.0141
	HGN [24]	0.0189	0.0120	0.0313	0.0159	0.0325	0.0206	0.0512	0.0266	0.0321	0.0221	0.0497	0.0277
	GRU4Rec [11]	0.0129	0.0086	0.0204	0.0110	0.0164	0.0099	0.0283	0.0137	0.0097	0.0059	0.0176	0.0084
	BERT4Rec [31]	0.0115	0.0075	0.0191	0.0099	0.0203	0.0124	0.0347	0.0170	0.0116	0.0071	0.0203	0.0099
	FDSA [47]	0.0182	0.0122	0.0288	0.0156	0.0267	0.0163	0.0407	0.0208	0.0228	0.0140	0.0381	0.0189
	SASRec [17]	0.0233	0.0154	0.0350	0.0192	0.0387	0.0249	0.0605	0.0318	0.0463	0.0306	0.0675	0.0374
	S^3-Rec [49]	0.0251	0.0161	0.0385	0.0204	0.0387	0.0244	0.0647	0.0327	0.0443	0.0294	0.0700	0.0376
	TIGER [Ours]	**0.0264**	**0.0181**	**0.0400**	**0.0225**	**0.0454**	**0.0321**	**0.0648**	**0.0384**	**0.0521**	**0.0371**	**0.0712**	**0.0432**
		+5.22%	+12.55%	+3.90%	+10.29%	+17.31%	+29.04%	+0.15%	+17.43%	+12.53%	+21.24%	+1.71%	+14.97%

Table 2: Dataset statistics for the three real-world benchmarks.

Dataset	# Users	# Items	Sequence Length	
			Mean	Median
Beauty	22,363	12,101	8.87	6
Sports and Outdoors	35,598	18,357	8.32	6
Toys and Games	19,412	11,924	8.63	6

task: "Beauty", "Sports and Outdoors", and "Toys and Games". Table 2 summarizes the statistics of the datasets. We use users' review history to create item sequences sorted by timestamp and filter out users with less than 5 reviews. Following the standard evaluation protocol [8, 17], we use the leave-one-out strategy for evaluation. For each item sequence, the last item is used for testing, the item before the last is used for validation, and the rest is used for training. During training, we limit the number of items in a user's history to 20.

4.1.2 Evaluation Metrics. We use top-k Recall (Recall@K) and Normalized Discounted Cumulative Gain (NDCG@K) with $K = 5, 10$ to evaluate the recommendation performance.

4.1.3 Implementation Details. Here we discuss the implementation details of the RQ-VAE model and the sequence-to-sequence model. We will release the source code of TIGER upon acceptance.

RQ-VAE Model. As discussed in section 3.1.1, RQ-VAE is used to quantize the semantic embedding of an item. We use the pre-trained Sentence-T5 model to obtain the semantic embedding of each item in the dataset. In particular, we use item's content features such as title, price, brand, and category to construct a sentence, which is then passed to the pre-trained Sentence-T5 model to obtain the item's semantic embedding of 768 dimension.

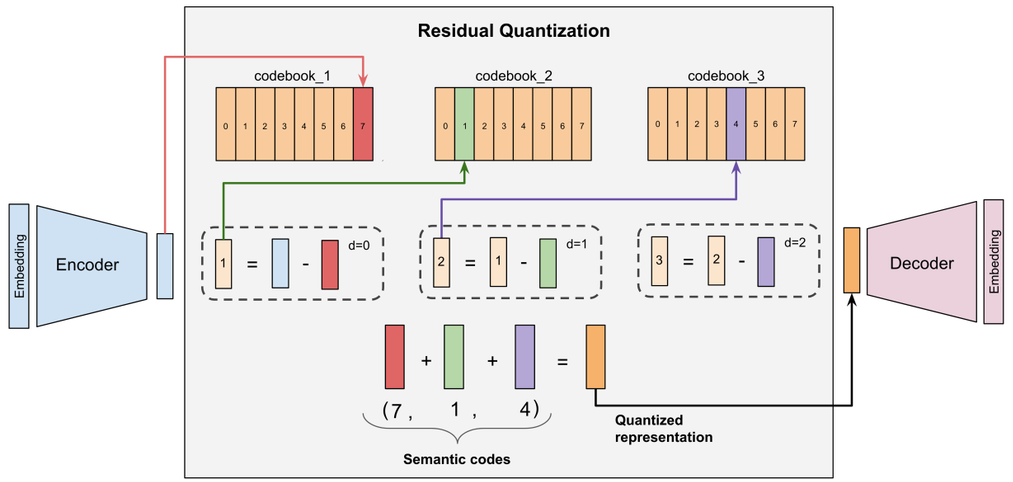


图 3：RQ-VAE：图中，DNN 编码器输出的向量，设为 r_0（以蓝色条表示），被输入量化器，后者以迭代方式工作。首先，在第一级码本中找到与 r_0最接近的向量。设这个最近向量为 e_{c_0}（以红色条表示）。然后，将残差计算为 $r_1 := r_0 - e_{c_0}$。将其输入量化器的第二级，并重复该过程：在第二级中找到与 r_1最接近的向量，设为 e_{c_1}（以绿色条表示），随后将第二级残差计算为 $r_2 = r_1 - e'_{c_1}$。然后，对 r_2第三次重复该过程。语义编码被计算为 e_{c_0}, e_{c_1},和 e_{c_2}在各自

码本中的索引。在图中所示的示例中，最终得到编码 $(7, 1, 2)$。

表 1：序列推荐任务的性能比较。最后一行给出了 **TIGER** 相对于最佳基线的提升。我们使用粗体和下划线分别表示最佳和次佳指标。

	Methods	Sports and Outdoors				Beauty				Toys and Games			
		Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
Baselines	P5 [47]	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
	Caser [33]	0.0116	0.0072	0.0194	0.0097	0.0205	0.0131	0.0347	0.0176	0.0166	0.0107	0.0270	0.0141
	HGN [24]	0.0189	0.0120	0.0313	0.0159	0.0325	0.0206	0.0512	0.0266	0.0321	0.0221	0.0497	0.0277
	GRU4Rec [11]	0.0129	0.0086	0.0204	0.0110	0.0164	0.0099	0.0283	0.0137	0.0097	0.0059	0.0176	0.0084
	BERT4Rec [31]	0.0115	0.0075	0.0191	0.0099	0.0203	0.0124	0.0347	0.0170	0.0116	0.0071	0.0203	0.0099
	FDSA [47]	0.0182	0.0122	0.0288	0.0156	0.0267	0.0163	0.0407	0.0208	0.0228	0.0140	0.0381	0.0189
	SASRec [17]	0.0233	0.0154	0.0350	0.0192	0.0387	0.0249	0.0605	0.0318	0.0463	0.0306	0.0675	0.0374
	S^3-Rec [49]	0.0251	0.0161	0.0385	0.0204	0.0387	0.0244	0.0647	0.0327	0.0443	0.0294	0.0700	0.0376
	TIGER [Ours]	**0.0264**	**0.0181**	**0.0400**	**0.0225**	**0.0454**	**0.0321**	**0.0648**	**0.0384**	**0.0521**	**0.0371**	**0.0712**	**0.0432**
		+5.22%	+12.55%	+3.90%	+10.29%	+17.31%	+29.04%	+0.15%	+17.43%	+12.53%	+21.24%	+1.71%	+14.97%

表 2：三个真实世界基准的数据集统计信息。

Dataset	# Users	# Items	Sequence Length	
			Mean	Median
Beauty	22,363	12,101	8.87	6
Sports and Outdoors	35,598	18,357	8.32	6
Toys and Games	19,412	11,924	8.63	6

任务："Beauty"、"Sports and Outdoors"和"Toys and Games"。表 2 汇总了这些数据集的统计信息。我们使用用户的评论历史创建按时间戳排序的物品序列，并过滤掉评论少于 5 条的用户。按照标准评估协议 [8, 17]，我们采用留一法进行评估。对于每个物品序列，最后一个物品用于测试，倒数第二个物品用于验证，其余物品用于训练。训练期间，我们将用户历史中的物品数量限制为 20。

4.1.2 评估指标。我们使用 top-k 召回率（Recall@K）和归一化折损累计增益（NDCG@K），其中 $K = 5, 10$，来评估推荐性能。

4.1.3 实现细节。这里我们讨论 RQ-VAE 模型和序列到序列模型的实现细节。TIGER 的源代码将在论文被接收后发布。

RQ-VAE 模型。如第 3.1.1 节所述，RQ-VAE 用于量化物品的语义嵌入。我们使用预训练的 Sentence-T5 模型获取数据集中每个物品的语义嵌入。具体而言，我们使用物品的标题、价格、品牌和类别等内容特征构造一个句子，然后将其输入预训练的 Sentence-T5 模型，得到该物品的 768 维语义嵌入。

The RQ-VAE model consists of three components: a DNN encoder that encodes the input semantic embedding into a latent representation, residual quantizer which outputs a quantized representation, and a DNN decoder that decodes the quantized representation back to the semantic input embedding space. The encoder has three intermediate layers of size 512, 256 and 128 with ReLU activation, with a final latent representation dimension of 32. To quantize this representation, three levels of residual quantization is done. For each stage of the quantization, a codebook of cardinality 256 is maintained, where each vector in the codebook has a dimension of 32. When computing the total loss, we use $\beta = 0.25$. The RQ-VAE model is trained for 20k epochs to ensure high codebook usage ($\geq 80\%$). We use Adagrad optimizer with a learning rate of 0.4 and a batch size of 1024. Upon training, we use the learned encoder and the quantization module to generate a 3-tuple Semantic ID for each item. To avoid multiple items being mapped to the same Semantic ID, we add a unique 4^{th} code for items that share the same first three codewords, *i.e.* two items associated with a tuple (7, 1, 4) are assigned (7, 1, 4, 0) and (7, 1, 4, 1) respectively (if there are no collisions, we still assign 0 as the fourth codeword). This results in a unique Semantic ID of length 4 for each item in the recommendation corpus. This is the Semantic ID generation algorithm that we use in TIGER. Notably, we observe less than 40 items sharing the same first three codewords, and hence the cardinality of the 4^{th} codeword is at most 40.

Sequence-to-Sequence Model. We use the open-sourced T5X framework [28] to implement our transformer based encoder-decoder architecture. To allow the model to process the input for the sequential recommendation task, we add semantic codewords to the vocabulary of the sequence-to-sequence model. In particular, we add 1024 (256×4) tokens to the vocabulary. In addition to the semantic codewords, we also add user-specific tokens to the vocabulary. To keep the vocabulary size limited, we only added 2000 tokens for user IDs. We use the Hashing Trick [39] to map the raw user ID to one of the 2000 user ID tokens. We construct the input sequence as the user Id token followed by the sequence of Semantic ID tokens corresponding to a given user's item interaction history. We found that adding user ID to the input, allows the model to personalize the recommended items.

We use 4 layers each for the encoder and decoder models with 6 self-attention heads of dimension 64 in each layer. We used the ReLU activation function for all the layers. The MLP and the input dimension was set as 1024 and 128, respectively. We used a dropout of 0.1. Overall, the model has around 13 million parameters. We train this model for 200k steps for the "Beauty" and "Sports and Outdoors" dataset. Due to the smaller size of the "Toys and Games" dataset, it is trained only for 100k steps. We use a batch size of 256. The learning rate is 0.01 for the first 10k steps and then follows an inverse square root decay schedule.

4.2 Performance on Sequential Recommendation (RQ1)

We first evaluate our model on the sequential recommendation task and compare the performance of TIGER against existing state-of-the-art sequential recommendation models. Next, we briefly describe the recent baselines proposed for the sequential recommendation task and discuss the performance of the all the models.

4.2.1 Baselines. We compare our proposed framework for generative retrieval with several other sequential recommendation methods.

- GRU4Rec [11] is the first RNN-based approach that uses a customized GRU for the sequential recommendation task.
- Caser [33] uses a CNN architecture for capturing high-order Markov Chains by applying horizontal and vertical convolutional operations for sequential recommendation.
- HGN [24]: Hierarchical Gating Network (HGN) captures the long-term and short-term user interests via a new gating architecture.
- SASRec [17]: Self-Attentive Sequential Recommendation (SASRec) uses a causal mask Transformer to model a user's sequential interactions.
- BERT4Rec [31]: BERT4Rec addresses the limitations of unidirectional architectures by using a bi-directional self-attention Transformer for the recommendation task.
- FDSA [47]: Feature-level Deeper Self-Attention Network (FDSA) incorporates item features in addition to the item embeddings as part of the input sequence in the Transformers.
- S^3-Rec [49]: Self-Supervised Learning for Sequential Recommendation (S^3-Rec) proposes pre-training a bi-directional Transformer on self-supervision tasks to improve the sequential recommendation.
- P5 [8]: P5 is a recent method that uses a pretrained Large Language Model (LLM) to unify different recommendation tasks in a single model.

Notably all the baselines, with the exception of P5, learn a high-dimensional vector space using dual encoder, where the user's past item interactions and the candidate items are encoded as a high-dimensional representation and Maximum Inner Product Search (MIPS) is used to retrieve the next candidate item that the user will potentially interact with. In contrast, our *novel* generative retrieval framework directly predicts the item's Semantic ID token-by-token using a sequence-to-sequence model.

4.2.2 Recommendation Performance. We perform an extensive analysis of our proposed TIGER on the sequential recommendation task and compare against several recent baselines. The results for all baselines, except P5, are taken from the publicly accessible results [1] made available by Zhou *et al.* [49]. For P5, we use the source code[2] made available by the authors. However, for a fair comparison, we updated the data pre-processing method to be consistent with the other baselines and our method. We provide further details related to our changes in Appendix A.

The results are shown in Table 1. We observe that TIGER consistently outperforms the existing baselines. We see significant improvement across all the three benchmarks that we considered. In particular, TIGER performs considerably better on the Beauty benchmark compared to the second-best baseline with up to 29% improvement in NDCG@5 compared to SASRec and 17.3% improvement in Recall@5 compared to S^3-Rec. Similarly on the Toys and

[1]https://github.com/aHuiWang/CIKM2020-S3Rec
[2]https://github.com/jeykigung/P5

Rajput and Mehta et al.

RQ-VAE 模型由三个组件构成：将输入语义嵌入编码为潜在表示的 DNN 编码器、输出量化表示的残差量化器，以及将量化表示解码回语义输入嵌入空间的 DNN 解码器。编码器包含三个使用 ReLU 激活函数的中间层，大小分别为 512、256 和 128，最终潜在表示的维度为 32。为了量化该表示，我们进行三个层级的残差量化。在每个量化阶段，都维护一个基数为 256 的码本，其中每个码本向量的维度为 32。计算总损失时，我们使用 $\beta = 0.25$。RQ-VAE 模型训练 20k 个 epoch，以确保较高的码本使用率（($\geq 80\%$）。我们使用 Adagrad 优化器，学习率为 0.4，批大小为 1024。训练完成后，我们使用学得的编码器和量化模块为每个物品生成一个三元组语义 ID。为避免多个物品映射到同一个语义 ID，我们为前三个码字相同的物品添加唯一的4^{th}编码； *i.e.*与元组 (7, 1, 4) 关联的两个物品会分别被分配 (7, 1, 4, 0) 和 (7, 1, 4, 1)（如果不存在冲突，我们仍将 0 指定为第四个码字）。这样，推荐语料库中的每个物品都会获得一个长度为 4 的唯一语义 ID。这就是我们在 TIGER 中使用的语义 ID 生成算法。值得注意的是，我们观察到共享前三个码字的物品少于 40 个，因此4^{th}码字的基数至多为 40。

序列到序列模型。我们使用开源 T5X 框架 [28] 来实现基于 Transformer 的编码器—解码器架构。为了让模型能够处理序列推荐任务的输入，我们将语义码字添加到序列到序列模型的词表中。具体而言，我们向词表中添加 1024（256×4）个 token。除语义码字外，我们还向词表添加用户特定 token。为了限制词表大小，我们仅为用户 ID 添加了 2000 个 token。我们使用哈希技巧 [39] 将原始用户 ID 映射到这 2000 个用户 ID token 之一。我们构造的输入序列以用户 ID token 开头，随后是与给定用户的物品交互历史相对应的语义 ID token 序列。我们发现，在输入中加入用户 ID 能让模型对推荐物品进行个性化。

编码器模型和解码器模型均使用 4 层，每层包含 6 个维度为 64 的自注意力头。所有层均使用 ReLU 激活函数。MLP 维度和输入维度分别设为 1024 和 128。我们使用 0.1 的 dropout。总体而言，该模型约有 1300 万个参数。对于"Beauty"和"Sports and Outdoors"数据集，我们将该模型训练 200k 步。由于"Toys and Games"数据集规模较小，仅训练 100k 步。批大小为 256。前 10k 步的学习率为 0.01，之后采用平方根倒数衰减计划。

4.2 序列推荐的性能（RQ1）

我们首先在序列推荐任务上评估模型，并将 TIGER 的性能与现有最先进的序列推荐模型进行比较。接下来，我们简要介绍近期为序列推荐任务提出的基线，并讨论所有模型的性能。

4.2.1 基线。我们将提出的生成式检索框架与其他几种序列推荐方法进行比较。

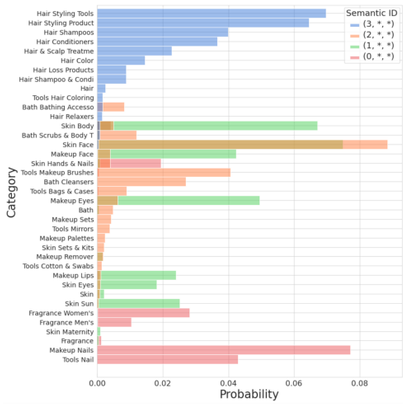
- GRU4Rec [11] 是首个基于 RNN 的方法，它使用定制的 GRU 完成序列推荐任务。
- Caser [33] 使用 CNN 架构，通过应用水平和垂直卷积运算来捕捉高阶马尔可夫链，以进行序列推荐。
- HGN [24]：分层门控网络（HGN）通过一种新的门控架构捕捉用户的长期和短期兴趣。
- SASRec [17]：自注意力序列推荐（SASRec）使用带因果掩码的 Transformer 对用户的序列交互进行建模。
- BERT4Rec [31]：BERT4Rec 通过在推荐任务中使用双向自注意力 Transformer，解决单向架构的局限。
- FDSA [47]：特征级深层自注意力网络（FDSA）除物品嵌入外，还将物品特征作为 Transformer 输入序列的一部分。
- S^3-Rec [49]：序列推荐的自监督学习（S^3-Rec）提出在自监督任务上预训练双向 Transformer，以改进序列推荐。
- P5 [8]：P5 是一种近期方法，它使用预训练的大型语言模型（LLM），在单一模型中统一不同的推荐任务。

值得注意的是，除 P5 外，所有基线都使用双编码器学习高维向量空间，其中用户过去的物品交互和候选物品被编码为高维表示，并使用最大内积搜索（MIPS）检索用户接下来可能交互的候选物品。相比之下，我们的新型生成式检索框架使用序列到序列模型，逐词元直接预测物品的语义 ID。

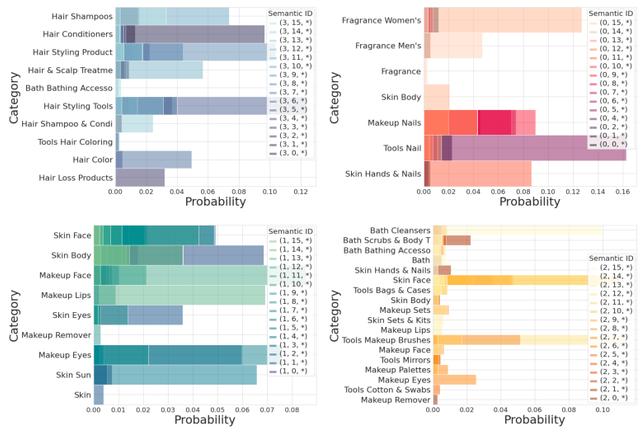
4.2.2 推荐性能。我们对提出的 TIGER 在序列推荐任务上的表现进行广泛分析，并与近期的若干基线进行比较。除 P5 外，所有基线的结果均取自公开结果 [1]，该结果由 Zhou 等人 [49] 提供。对于 P5，我们使用作者提供的源代码[2]。然而，为了公平比较，我们更新了数据预处理方法，使其与其他基线及我们的方法保持一致。附录 A 进一步介绍了与这些更改相关的细节。

结果如表 1 所示。我们观察到，TIGER 始终优于现有基线。在所考察的三个基准上，我们都看到了显著提升。特别是，在 Beauty 基准上，TIGER 的表现明显优于第二好的基线：与 SASRec 相比，NDCG@5 最多提升 29%；与 S^3-Rec 相比，Recall@5 提升 17.3%。同样，在 Toys and

[1]https://github.com/aHuiWang/CIKM2020-S3Rec
[2]https://github.com/jeykigung/P5

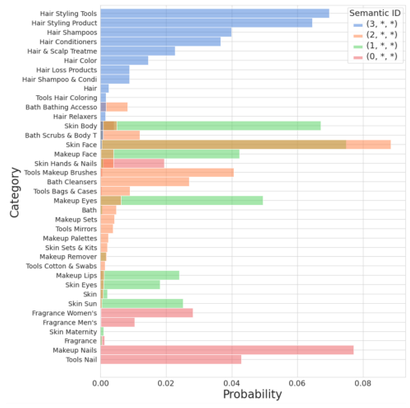


(a) **The ground-truth category distribution for all the items in the dataset colored by the value of the first codeword c_1.**

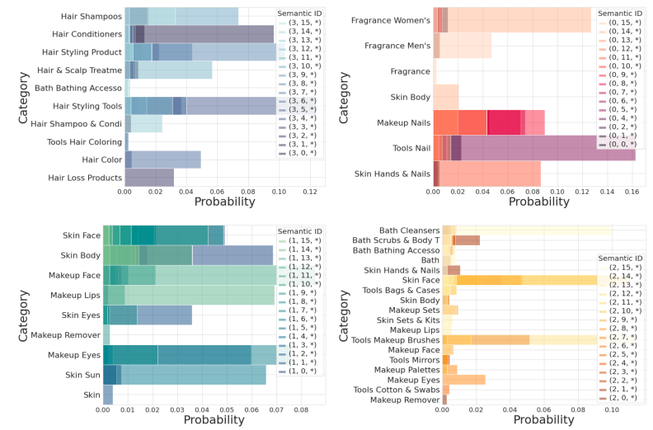


(b) **The ground-truth category distributions for all the items having the Semantic ID as $(c_1, *, *)$, where $c_1 \in \{1, 2, 3, 4\}$. The categories are color-coded based on the second semantic token c_2.**

Figure 4: Qualitative study of RQ-VAE Semantic IDs (c_1, c_2, c_3, c_4) on the Amazon Beauty dataset. We show that the ground-truth categories are distributed across different Semantic tokens. Moreover, the RQVAE semantic IDs form a hierarchy of items, where the first semantic token (c_1) corresponds to coarse-level category, while second/third semantic token (c_2/c_3) correspond to fine-grained categories.



(a) 数据集中所有物品的真实类别分布，按第一个码字 c_1的取值着色。



(b) 语义 **ID** 为 $(c_1, *, *)$的所有物品的真实类别分布，其中 $c_1 \in \{1, 2, 3, 4\}$。类别按第二个语义词元 c_2进行颜色编码。

图 4：**Amazon Beauty** 数据集上 **RQ-VAE** 语义 **ID** (c_1, c_2, c_3, c_4)的定性研究。我们表明，真实类别分布在不同的语义词元中。此外，**RQ-VAE** 语义 **ID** 构成物品的层次结构，其中第一个语义词元 (c_1) 对应粗粒度类别，而第二／第三个语义词元 (c_2/c_3) 对应细粒度类别。

Games dataset, TIGER is 21% and 15% better in NDCG@5 and NDCG@10, respectively.

Games 数据集上，TIGER 在 NDCG@5 和 NDCG@10 上分别提升了 21% 和 15%。

4.3 Item Representation (RQ2)

In this section, we analyze several important characteristics of RQ-VAE Semantic IDs. In particular, we first perform a qualitative analysis to observe the hierarchical nature of Semantic IDs in Section 4.3.1. Next, we evaluate the importance of our design choice of using RQ-VAE for quantization by contrasting the performance with an alternative hashing-based quantization method in section 4.3.2. Finally, we perform an ablation in section 4.3.3 to study the importance of using Semantic IDs by comparing TIGER with a sequence-to-sequence model that uses Random ID for item representation.

4.3.1 Qualitative Analysis. We analyze the RQ-VAE Semantic IDs learned for the Amazon Beauty dataset in Figure 4. For exposition, we set the number of RQ-VAE levels as 3 with a codebook size of 4, 16, and 256 respectively, *i.e.* for a given Semantic ID (c_1, c_2, c_3) of an item, $0 \leq c_1 \leq 3$, $0 \leq c_2 \leq 15$ and $0 \leq c_3 \leq 255$.

In Figure 4a, we annotate each item's category using c_1 to visualize c_1-specific categories in the overall category distribution of the dataset. As shown in Figure 4a, c_1 captures the high-level category of the item. For instance, $c_1 = 3$ contains most of the products related to "Hair". Similarly, majority of items with $c_1 = 1$ are "Makeup" and "Skin" products for face, lips and eyes.

We also visualize the hierarchical nature of RQ-VAE Semantic IDs by fixing c_1 and visualizing the category distribution for all possible values of c_2 in Figure 4b. Once again, we found that the second codeword c_2 can further categorize the high-level semantics captured with c_1 into fine-grained categories.

The hierarchical nature of Semantic IDs learned by RQ-VAE opens a wide-array of new capabilities which are discussed in Section 4.4. As opposed to existing recommendation systems that learn item embeddings based on random atomic IDs, TIGER uses Semantic IDs where semantically similar items have overlapping codewords, which allows the model to effectively share knowledge from semantically similar items in the dataset.

4.3.2 Hashing vs. RQ-VAE for Semantic ID Generation. In this section, we study the importance of RQ-VAE in our framework by comparing RQ-VAE against Locality Sensitive Hashing (LSH) [1, 13, 14] for Semantic ID generation. LSH is a popular hashing technique that can be easily adapted to work for our setting. To generate LSH Semantic IDs, we use h random hyperplanes $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h$ to perform a random projection of the embedding vector $\boldsymbol{x}$ and compute the following binary vector: $(1_{\boldsymbol{w}_1^\top \boldsymbol{x} > 0}, \ldots, 1_{\boldsymbol{w}_h^\top \boldsymbol{x} > 0})$. This vector is converted into an integer code as $c_0 = \sum_{i=1}^{h} 2^{i-1} 1_{\boldsymbol{w}_i^\top \boldsymbol{x} > 0}$. This process is repeated m times using an independent set of random hyperplanes, resulting in m codewords $(c_0, c_1, \ldots, c_{m-1})$, which we refer to as the LSH-based Semantic ID.

In Table 3, we compare the performance of LSH Semantic ID with our proposed RQ-VAE Semantic ID. In this experiment, for LSH Semantic IDs, we used $h = 8$ random hyperplanes and set $m = 4$ to be comparable against RQ-VAE in terms of cardinalities. The parameters for the hyperplanes are randomly sampled from a standard normal distribution, which ensures that the hyperplanes are spherically symmetric. Our results show that RQ-VAE consistently outperforms LSH. This illustrates that learning Semantic IDs via a non-linear, Deep Neural Network (DNN) architecture yields better quantization than using random projections, given the same content-based semantic embedding.

4.3.3 Random ID vs. Semantic ID. We also compare the importance of Semantic IDs in our generative retrieval recommender system. In

4.3 物品表示（RQ2）

本节分析 RQ-VAE 语义 ID 的若干重要特性。具体而言，我们首先在第 4.3.1 节进行定性分析，以观察语义 ID 的层次性质。接下来，我们在第 4.3.2 节将其性能与另一种基于哈希的量化方法进行对比，以评估使用 RQ-VAE 进行量化这一设计选择的重要性。最后，我们在第 4.3.3 节进行消融实验，将 TIGER 与使用随机 ID 表示物品的序列到序列模型进行比较，以研究使用语义 ID 的重要性。

4.3.1 定性分析。我们在图 4 中分析针对 Amazon Beauty 数据集学习到的 RQ-VAE 语义 ID。为便于说明，我们将 RQ-VAE 的层数设为 3，码本大小分别为 4、16 和 256；也就是说，对于物品的给定语义 ID (c_1, c_2, c_3)，$0 \leq c_1 \leq 3$、$0 \leq c_2 \leq 15$且 $0 \leq c_3 \leq 255$。在图 4a 中，我们使用 c_1标注每个物品的类别，从而在数据集的整体类别分布中可视化 c_1所特有的类别。如图 4a 所示，c_1捕获物品的高层类别。例如，$c_1 = 3$ 包含大多数与"Hair"相关的产品。类似地，满足 $c_1 = 1$的物品大多是面部、唇部和眼部的"Makeup"及"Skin"产品。

我们还通过固定 c_1，并针对 c_2的所有可能取值，在图 4b 中可视化类别分布，从而展示 RQ-VAE 语义 ID 的层次性质。我们再次发现，第二个码字 c_2可以进一步将 c_1捕获的高层语义划分为细粒度类别。

RQ-VAE 学习到的语义 ID 的层次性质带来了多种新能力，第 4.4 节将对此进行讨论。现有推荐系统基于随机原子 ID 学习物品嵌入；与之不同，TIGER 使用语义 ID，其中语义相似的物品具有重叠的码字，因此模型能够有效共享数据集中语义相似物品的知识。

4.3.2 用于语义 ID 生成的哈希与 RQ-VAE 对比。本节通过将 RQ-VAE 与局部敏感哈希（LSH）[1, 13, 14] 在语义 ID 生成方面进行比较，研究 RQ-VAE 在我们框架中的重要性。LSH 是一种流行的哈希技术，可以轻松调整以适用于我们的设置。为生成 LSH 语义 ID，我们使用 h个随机超平面 $\boldsymbol{w}_1, \ldots, \boldsymbol{w}_h$对嵌入向量 $\boldsymbol{x}$进行随机投影，并计算以下二进制向量：
$(1_{\boldsymbol{w}_1^\top \boldsymbol{x} > 0}, \ldots, 1_{\boldsymbol{w}_h^\top \boldsymbol{x} > 0})$
. This vector is con-

转换为整数编码 $c_0 = \sum_{i=1}^{h} 2^{i-1} 1_{\boldsymbol{w}_i^\top \boldsymbol{x} > 0}$。使用一组独立的随机超平面将这一过程重复 m次，得到 m个码字 $(c_0, c_1, \ldots, c_{m-1})$，我们将其称为基于 LSH 的语义 ID。

在表 3 中，我们比较了 LSH 语义 ID 与所提出的 RQ-VAE 语义 ID 的性能。在该实验中，对于 LSH 语义 ID，我们使用了 $h = 8$ 个随机超平面，并设置$m = 4$，使其基数与 RQ-VAE 相当。超平面的参数从标准正态分布中随机采样，从而确保超平面具有球面对称性。结果表明，RQ-VAE 始终优于 LSH。这说明，在采用相同的基于内容的语义嵌入时，通过非线性深度神经网络（DNN）架构学习语义 ID，能够得到比随机投影更好的量化结果。

4.3.3 随机 ID 与语义 ID。我们还比较了语义 ID 在生成式检索推荐系统中的重要性。具体而言，

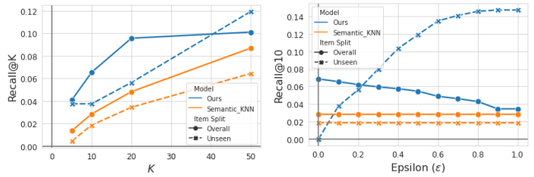
Table 3: Ablation study for different ID generation techniques for generative retrieval. We show that RQ-VAE Semantic IDs perform significantly better compared to hashing-based Semantic IDs and Random IDs.

Methods	Sports and Outdoors				Beauty				Toys and Games			
	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
Random ID	0.007	0.005	0.0116	0.0063	0.0296	0.0205	0.0434	0.0250	0.0362	0.0270	0.0448	0.0298
LSH Semantic ID	0.0215	0.0146	0.0321	0.0180	0.0379	0.0259	0.0533	0.0309	0.0412	0.0299	0.0566	0.0349
RQ-VAE Semantic ID	**0.0264**	**0.0181**	**0.0400**	**0.0225**	**0.0454**	**0.0321**	**0.0648**	**0.0384**	**0.0521**	**0.0371**	**0.0712**	**0.0432**

particular, we compare randomly generated IDs with the Semantic IDs. To generate the Random ID baseline, we assign m random codewords to each item. A Random ID of length m for an item is simply $(c_1, \ldots, c_m)$, where c_i is sampled uniformly at random from $\{1, 2, \ldots, K\}$. We set $m = 4$, and $K = 255$ for the Random ID baseline to make the cardinality similar to RQ-VAE Semantic IDs. A comparison of Random ID against RQ-VAE and LSH Semantic IDs is shown in Table 3. We see that Semantic IDs consistently outperform Random ID baseline, highlighting the importance of leveraging content-based semantic information.

4.4 New Capabilities (RQ3)

We describe two new capabilities that directly follow from our proposed generative retrieval framework, namely cold-start recommendations and recommendation diversity. We refer to these capabilities as "new" since existing sequential recommendation models (See Section 4.2.1) cannot be directly used to satisfy these real-world use cases. We believe these capabilities result from a synergy between RQ-VAE based Semantic IDs and the generative retrieval approach of our framework. We discuss how TIGER is used in these settings in the following sections.



(a) Recall@K vs. K, when $\epsilon = 0.1$. **(b) Recall@10 vs. ϵ.**

Figure 5: Performance in the cold-start retrieval setting.

4.4.1 Cold-Start Recommendation. In this section, we study the cold-start recommendation capability of our proposed framework. Due to the fast-changing nature of the real-world recommendation corpus, new items are constantly introduced. Since newly added items lack user impressions in the training corpus, existing recommendation models that use a random atomic ID for item representation fail to retrieve new items as potential candidates. In contrast, TIGER can easily perform cold-start recommendations in an end-to-end fashion.

For this analysis, we consider the Beauty dataset from Amazon Reviews. To simulate newly added items, we remove 5% of test items from the training data split. We refer to these removed items as *unseen items*. Removing the items from the training split ensures there is no data leakage with respect to the unseen items.

As before, we use Semantic ID of length 4 to represent the items, where the first 3 tokens are generated using RQ-VAE and the 4^{th} token is used to ensure a unique ID exists for all the seen items. We train the RQ-VAE quantizer and the sequence-to-sequence model on the training split. Once trained, we use the RQ-VAE model to generate the Semantic IDs for all the items in the dataset, including any unseen items in the item corpus.

Given a Semantic ID (c_1, c_2, c_3, c_4) predicted by the model, we retrieve the seen item having the same corresponding ID. Note that by definition, each Semantic ID predicted by the model can match at most one item in the training dataset. Additionally, unseen items having the same first three semantic tokens, *i.e.* (c_1, c_2, c_3) are included to the list of retrieved candidates. Finally, when retrieving a set of top-K candidates, we introduce a hyperparameter ϵ which specifies the maximum proportion of unseen items chosen by our framework.

We compare the performance of TIGER with a k-Nearest Neighbors (KNN) approach on the cold-start recommendation setting in Figure 5. For KNN, we use the semantic representation space to perform the nearest-neighbor search. We refer to the KNN-based baseline as Semantic_KNN. Figure 5a shows that our framework with $\epsilon = 0.1$ consistently outperforms Semantic_KNN for all Recall@K metrics. In Figure 5b, we provide a comparison between our method and Semantic_KNN for various values of ϵ. For all settings of $\epsilon \geq 0.1$, our method outperforms the baseline.

4.4.2 Recommendation diversity. While Recall and NDCG are the primary metrics used to evaluate a recommendation system, diversity of predictions is another critical objective of interest. A recommender system with poor diversity can be detrimental to the long-term engagement of users. In this section, we discuss how our generative retrieval framework can be used to predict diverse items. We show that temperature-based sampling during the decoding process can be effectively used to control the diversity of model predictions. While temperature-based sampling can be applied to any existing recommendation model, TIGER allows sampling across various levels of hierarchy owing to the properties of RQ-VAE Semantic IDs. For instance, sampling the first token of the Semantic ID allows retrieving items from coarse-level categories, while sampling a token from second/third token allows sampling items within a category.

We quantitatively measure the diversity of predictions using Entropy@K metric, where the entropy is calculated for the distribution of the ground-truth categories of the top-K items predicted by the model. We report the Entropy@K for various temperature values in Table 5. We observe that temperature-sampling in the decoding stage can be effectively used to increase the diversity in the ground-truth categories of the items. We also perform a qualitative analysis in Table 4.

5 DISCUSSION

Invalid IDs. Since the model decodes the codewords of the target Semantic ID autoregressively, it is possible that the model can

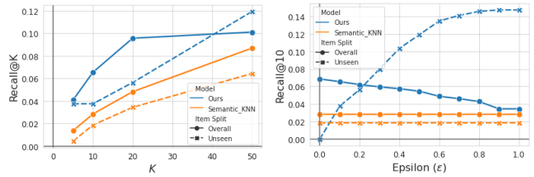
表 3：生成式检索中不同 ID 生成技术的消融研究。我们表明，与基于哈希的语义 ID 和随机 ID 相比，RQ-VAE 语义 ID 的表现显著更好。

Methods	Sports and Outdoors				Beauty				Toys and Games			
	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
Random ID	0.007	0.005	0.0116	0.0063	0.0296	0.0205	0.0434	0.0250	0.0362	0.0270	0.0448	0.0298
LSH Semantic ID	0.0215	0.0146	0.0321	0.0180	0.0379	0.0259	0.0533	0.0309	0.0412	0.0299	0.0566	0.0349
RQ-VAE Semantic ID	**0.0264**	**0.0181**	**0.0400**	**0.0225**	**0.0454**	**0.0321**	**0.0648**	**0.0384**	**0.0521**	**0.0371**	**0.0712**	**0.0432**

我们将随机生成的 ID 与语义 ID 进行比较。为了生成随机 ID 基线，我们为每个物品分配 m个随机码字。一个物品长度为 m的随机 ID 就是 $(c_1, \ldots, c_m)$，其中 c_i从 $\{1, 2, \ldots, K\}$中均匀随机采样。对于随机 ID 基线，我们设置 $m = 4$以及 $K = 255$，使其基数与 RQ-VAE 语义 ID 相近。表 3 展示了随机 ID 与 RQ-VAE 和 LSH 语义 ID 的比较。可以看到，语义 ID 始终优于随机 ID 基线，这凸显了利用基于内容的语义信息的重要性。

4.4 新能力（RQ3）

我们介绍由所提出的生成式检索框架直接产生的两项新能力，即冷启动推荐和推荐多样性。我们将这些能力称为"新"能力，是因为现有序列推荐模型（见第 4.2.1 节）无法直接用于满足这些现实用例。我们认为，这些能力源于基于 RQ-VAE 的语义 ID 与本框架的生成式检索方法之间的协同作用。以下各节将讨论 TIGER 如何用于这些场景。



(a) Recall@K 与 K 的关系，当 $\epsilon = 0.1$时。 **(b) Recall@10 与 ϵ 的关系。**

图 5：冷启动检索场景下的性能。4.4.1

冷启动推荐。本节研究所提出框架的冷启动推荐能力。由于现实世界中的推荐语料库变化迅速，新物品会不断被引入。新加入的物品在训练语料库中缺乏用户曝光，因此使用随机原子 ID 表示物品的现有推荐模型无法将新物品检索为潜在候选项。相比之下，TIGER 能够以端到端方式轻松执行冷启动推荐。

在该分析中，我们采用 Amazon Reviews 的 Beauty 数据集。为了模拟新加入的物品，我们从训练数据划分中移除 5% 的测试物品，并将这些被移除的物品称为未见物品。从训练划分中移除这些物品可确保不存在与未见物品相关的数据泄漏。

与前文相同，我们使用长度为 4 的语义 ID 表示物品，其中前三个 token 使用 RQ-VAE 生成，而第4^{th}
个 token 用于确保所有已见物品都具有唯一 ID。我们在训练划分上训练 RQ-VAE 量化器和序列到序列模型。训练完成后，我们使用 RQ-VAE 模型为数据集中的所有物品生成语义 ID，其中包括物品语料库中的所有未见物品。

给定模型预测的语义 ID (c_1, c_2, c_3, c_4)，我们检索具有相同对应 ID 的已见物品。请注意，根据定义，模型预测的每个语义 ID 最多只能与训练数据集中的一个物品匹配。此外，具有相同前三个语义 token 的未见物品 *i.e.* (c_1, c_2, c_3)也会被加入检索候选项列表。最后，在检索一组 top-K 候选项时，我们引入超参数 ϵ，它指定本框架所选未见物品的最大比例。

在冷启动推荐场景下，我们比较 TIGER 与 k 最近邻（KNN）方法的性能，如
图 5 所示。对于 KNN，我们在语义表示空间中执行最近邻搜索。我们将基于 KNN 的基线称为 Semantic_KNN。图 5a 表明，在所有 Recall@K 指标上，采用 $\epsilon = 0.1$的本框架始终优于 Semantic_KNN。在图 5b, 中，我们针对 ϵ的不同取值比较了本方法与 Semantic_KNN。在 $\epsilon \geq 0.1$的所有设置下，本方法都优于基线。

4.4.2 推荐多样性。虽然召回率和 NDCG 是评估推荐系统所用的主要指标，但预测多样性也是另一个值得关注的关键目标。多样性较差的推荐系统可能会损害用户的长期参与度。本节讨论如何使用生成式检索框架预测多样化物品。我们表明，在解码过程中采用基于温度的采样，可以有效控制模型预测的多样性。虽然基于温度的采样可应用于任何现有推荐模型，但由于 RQ-VAE 语义 ID 的性质，TIGER 可以跨不同层次结构进行采样。例如，对语义 ID 的第一个 token 进行采样，可以检索粗粒度类别中的物品；而对第二个或第三个 token 进行采样，则可以在一个类别内对物品进行采样。

我们使用 Entropy@K 指标定量衡量预测的多样性，其中熵是根据模型所预测 top-K 物品的真实类别分布计算的。表 5 报告了不同温度值下的 Entropy@K。我们观察到，解码阶段的温度采样能够有效提高物品真实类别的多样性。我们还在表 4 中进行了定性分析。

5 讨论

无效 **ID**。由于模型以自回归方式解码目标语义 ID 的码字，因此模型可能会

Table 4: Recommendation diversity with temperature-based decoding using generative retrieval.

Target Category	Most-common Categories for top-10 predicted items	
	T = 1.0	T = 2.0
Hair Styling Products	Hair Styling Products	Hair Styling Products, Hair Styling Tools, Skin Face
Tools Nail	Tools Nail	Tools Nail, Makeup Nails
Makeup Nails	Makeup Nails	Makeup Nails, Skin Hands & Nails, Tools Nail
Skin Eyes	Skin Eyes	Hair Relaxers, Skin Face, Hair Styling Products, Skin Eyes
Makeup Face	Tools Makeup Brushes, Makeup Face	Tools Makeup Brushes, Makeup Face, Skin Face, Makeup Sets, Hair Styling Tools
Hair Loss Products	Hair Loss Products, Skin Face, Skin Body	Skin Face, Hair Loss Products, Hair Shampoos, Hair & Scalp Treatments, Hair Conditioners

Table 5: Recommendation diversity in the model predictions. We evaluate the entropy of the distribution of categories that the model predicts in the Amazon Beauty dataset. A higher entropy corresponds more diverse items predicted by the model.

Temperature	Entropy@10	Entropy@20	Entropy@50
T = 1.0	0.76	1.14	1.70
T = 1.5	1.14	1.52	2.06
T = 2.0	1.38	1.76	2.28

predict an invalid ID (i.e., it may not map to any item in the recommendation dataset). In our experiments, we used semantic IDs of length 4 with each codeword having a cardinality of 256 (i.e., codebook size = 256 for each level). The number of possible IDs spanned by this combination = 256^4, which is approx. 4 trillion. On the other hand, the number of items in the datasets we consider is 10K-20K (See Table 2). Even though the number of valid IDs is only a fraction of all complete ID space, we observe that the model almost always predicts the valid IDs. We visualize the fraction of invalid IDs produced by TIGER as a function of the number of retrieved items K in Figure 6.

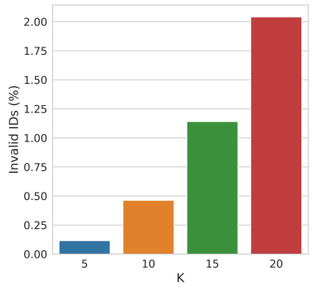


Figure 6: % of invalid IDs when generating Semantic IDs using Beam search for various values of K for the Beauty dataset. As shown, only 2% of the IDs are invalid when retrieving the top-20 items.

Effects of Semantic ID length and codebook size. We tried varying the Semantic ID length and codebook size, such as having an ID consisting of 6 codewords each from a codebook of size 64. We noticed that the recommendation metrics for TIGER were robust to these changes. However, note that the input sequence length increases with longer IDs (i.e., more codewords per ID), which makes the computation more expensive for our transformer-based sequence-to-sequence model.

Inference cost. Despite the remarkable success of our model on the sequential recommendation task, we note that our model is more expensive than ANN-based models during inference (not during training) due to the use of beam search for autoregressive decoding. We emphasize that optimizing the computational efficiency of TIGER was not the main objective of this work. Instead, our work opens up a new area of research: *Recommender Systems based on Generative Retrieval*. As part of future work, we will consider ways to make the model smaller or explore other ways of leveraging Transformer's capabilities, such as building a unified model across multiple datasets and tasks.

6 CONCLUSION

This paper proposed a novel paradigm, called TIGER, to retrieve candidates in recommender systems using a generative model. Underpinning this method is a novel semantic ID representation for items, which uses a hierarchical quantizer (RQ-VAE) on content embeddings to generate tokens that form the semantic IDs. Our method is end-to-end, i.e., a single model can be used to train and serve without creating an index — the transformer memory acts as the index [34]. We note that the cardinality of our embedding table does not grow linearly with the cardinality of item space, which works in our favor compared to systems that need to create large embedding tables during training or generate an index for every single item. Through experiments on three datasets, we show that our model can achieve SOTA results and generalizes well to new and unseen items.

Our work enables many new research directions. For example, it will be interesting to explore how these Semantic IDs can be integrated with LLMs to enable much more powerful conversational recommender models. We will also explore how to improve the Semantic ID representation and how to use it for ranking.

REFERENCES

[1] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.
[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.
[3] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.
[4] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.
[5] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.
[6] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Fifteenth ACM Conference on Recommender Systems*, pages 143–153, 2021.

表 4：生成式检索中采用基于温度解码时的推荐多样性。

Target Category	Most-common Categories for top-10 predicted items	
	T = 1.0	T = 2.0
Hair Styling Products	Hair Styling Products	Hair Styling Products, Hair Styling Tools, Skin Face
Tools Nail	Tools Nail	Tools Nail, Makeup Nails
Makeup Nails	Makeup Nails	Makeup Nails, Skin Hands & Nails, Tools Nail
Skin Eyes	Skin Eyes	Hair Relaxers, Skin Face, Hair Styling Products, Skin Eyes
Makeup Face	Tools Makeup Brushes, Makeup Face	Tools Makeup Brushes, Makeup Face, Skin Face, Makeup Sets, Hair Styling Tools
Hair Loss Products	Hair Loss Products, Skin Face, Skin Body	Skin Face, Hair Loss Products, Hair Shampoos, Hair & Scalp Treatments, Hair Conditioners

表 5：模型预测中的推荐多样性。我们评估模型在 Amazon Beauty 数据集上预测的类别分布的熵。熵越高，模型预测的物品越多样。

Temperature	Entropy@10	Entropy@20	Entropy@50
T = 1.0	0.76	1.14	1.70
T = 1.5	1.14	1.52	2.06
T = 2.0	1.38	1.76	2.28

预测出无效 ID（即该 ID 可能不映射到推荐数据集中的任何物品）。在实验中，我们使用长度为 4 的语义 ID，其中每个码字的基数为 256（即每一级的码本大小 = 256）。这种组合所覆盖的可能 ID 数量为 = 256^4，约为 4 万亿。另一方面，我们考察的数据集中的物品数量为 10K – 20K（见表 2）。尽管有效 ID 的数量仅占完整 ID 空间的一小部分，但我们观察到模型几乎总能预测出有效 ID。我们将 TIGER 产生的无效 ID 比例表示为检索物品数量 K的函数，并在图 6 中进行可视化。

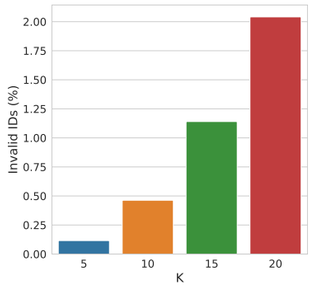


图 6：对于 Beauty 数据集，在 K取不同值时，使用束搜索生成语义 ID 所产生的无效 ID 百分比。如图所示，仅有 2% of the IDs are invalid when retrieving the top-20 items.

语义 ID 长度和码本大小的影响。我们尝试改变语义 **ID** 长度和码本大小，例如让一个 ID 由 6 个码字组成，每个码字均来自大小为 64 的码本。我们注意到，TIGER 的推荐指标对这些变化较为稳健。但需要注意的是，较长的 ID（即每个 ID 包含更多码字）会增加输入序列长度，从而使基于 Transformer 的序列到序列模型的计算成本更高。

推理成本。尽管我们的模型在序列推荐任务上取得了显著成功，但我们注意到，我们的模型在推理期间（而非训练期间）比基于 ANN 的模型成本更高，这是因为自回归解码使用了束搜索。我们强调，优化 TIGER 的计算效率并不是本工作的主要目标。相反，我们的工作开辟了一个新的研究领域：基于生成式检索的推荐系统。作为未来工作的一部分，我们将考虑缩小模型的方法，或探索利用 Transformer 能力的其他方式，例如构建跨多个数据集和任务的统一模型。

6 结论

本文提出了一种名为 TIGER 的新范式，使用生成模型在推荐系统中检索候选项。该方法的基础是一种用于物品的新型语义 ID 表示：它在内容嵌入上使用层次量化器（RQ-VAE）生成构成语义 ID 的词元。我们的方法是端到端的，即单个模型无需创建索引即可用于训练和服务——Transformer 的记忆充当索引 [34]。我们注意到，嵌入表的基数不会随物品空间的基数线性增长；与训练期间需要创建大型嵌入表或为每个物品生成索引的系统相比，这一点对我们有利。通过在三个数据集上开展实验，我们表明该模型能够取得 SOTA 结果，并能很好地泛化到新的、未见过的物品。

我们的工作开启了许多新的研究方向。例如，探索如何将这些语义 ID 与 LLM 集成，以实现能力更强的对话式推荐模型，将会很有意义。我们还将探索如何改进语义 ID 表示，以及如何将其用于排序。

REFERENCES

[1] Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388, 2002.
[2] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, Mustafa Ispir, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st workshop on deep learning for recommender systems*, pages 7–10, 2016.
[3] Paul Covington, Jay Adams, and Emre Sargin. Deep neural networks for youtube recommendations. In *Proceedings of the 10th ACM conference on recommender systems*, pages 191–198, 2016.
[4] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084*, 2022.
[5] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904*, 2020.
[6] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Fifteenth ACM Conference on Recommender Systems*, pages 143–153, 2021.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). *arXiv preprint arXiv:2203.13366*, 2022.

[9] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[10] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[12] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. *arXiv preprint arXiv:2210.12316*, 2022.

[13] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[14] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625, 1997.

[15] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[16] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H Chi. Learning to embed categorical features without embedding tables for recommendation. *arXiv preprint arXiv:2010.10784*, 2020.

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

[18] Dongmoon Kim, Kun-su Kim, Kyo-Hyun Park, Jee-Hyong Lee, and Keon Myung Lee. A music recommendation system with a dynamic k-means clustering algorithm. In *Sixth international conference on machine learning and applications (ICMLA 2007)*, pages 399–403. IEEE, 2007.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[20] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.

[21] Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vlad Karpukhin, Yi Lu, and Minjoon Seo. Contextualized generative retrieval. *arXiv preprint arXiv:2210.02068*, 2022.

[22] Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. Generative retrieval for long sequences. *arXiv preprint arXiv:2204.13596*, 2022.

[23] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[24] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 825–833, 2019.

[25] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[27] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[28] Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022.

[29] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[30] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[31] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[33] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[34] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*, 2022.

[35] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[37] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190*, 2022.

[38] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, et al. A neural corpus indexer for document retrieval. *arXiv preprint arXiv:2206.02743*, 2022.

[39] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1113–1120, 2009.

[40] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[41] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 269–277, 2019.

[42] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022.

[43] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[44] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835*, 2023.

[45] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *CoRR*, abs/2107.03312, 2021.

[46] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018.

[47] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326, 2019.

[48] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 43–51, 2019.

[49] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1893–1902, 2020.

Rajput and Mehta et al.

[7] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[8] Shijie Geng, Shuchang Liu, Zuohui Fu, Yingqiang Ge, and Yongfeng Zhang. Recommendation as language processing (rlp): A unified pretrain, personalized prompt & predict paradigm (p5). *arXiv preprint arXiv:2203.13366*, 2022.

[9] Carlos A Gomez-Uribe and Neil Hunt. The netflix recommender system: Algorithms, business value, and innovation. *ACM Transactions on Management Information Systems (TMIS)*, 6(4):1–19, 2015.

[10] Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pages 507–517, 2016.

[11] Balázs Hidasi, Alexandros Karatzoglou, Linas Baltrunas, and Domonkos Tikk. Session-based recommendations with recurrent neural networks. *arXiv preprint arXiv:1511.06939*, 2015.

[12] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. Learning vector-quantized item representation for transferable sequential recommenders. *arXiv preprint arXiv:2210.12316*, 2022.

[13] Piotr Indyk and Rajeev Motwani. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing*, pages 604–613, 1998.

[14] Piotr Indyk, Rajeev Motwani, Prabhakar Raghavan, and Santosh Vempala. Locality-preserving hashing in multidimensional spaces. In *Proceedings of the twenty-ninth annual ACM symposium on Theory of computing*, pages 618–625, 1997.

[15] Herve Jegou, Matthijs Douze, and Cordelia Schmid. Product quantization for nearest neighbor search. *IEEE transactions on pattern analysis and machine intelligence*, 33(1):117–128, 2010.

[16] Wang-Cheng Kang, Derek Zhiyuan Cheng, Tiansheng Yao, Xinyang Yi, Ting Chen, Lichan Hong, and Ed H Chi. Learning to embed categorical features without embedding tables for recommendation. *arXiv preprint arXiv:2010.10784*, 2020.

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

[18] Dongmoon Kim, Kun-su Kim, Kyo-Hyun Park, Jee-Hyong Lee, and Keon Myung Lee. A music recommendation system with a dynamic k-means clustering algorithm. In *Sixth international conference on machine learning and applications (ICMLA 2007)*, pages 399–403. IEEE, 2007.

[19] Yehuda Koren, Robert Bell, and Chris Volinsky. Matrix factorization techniques for recommender systems. *Computer*, 42(8):30–37, 2009.

[20] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. Autoregressive image generation using residual quantization. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 11523–11532, 2022.

[21] Hyunji Lee, Jaeyoung Kim, Hoyeon Chang, Hanseok Oh, Sohee Yang, Vlad Karpukhin, Yi Lu, and Minjoon Seo. Contextualized generative retrieval. *arXiv preprint arXiv:2210.02068*, 2022.

[22] Hyunji Lee, Sohee Yang, Hanseok Oh, and Minjoon Seo. Generative retrieval for long sequences. *arXiv preprint arXiv:2204.13596*, 2022.

[23] Jing Li, Pengjie Ren, Zhumin Chen, Zhaochun Ren, Tao Lian, and Jun Ma. Neural attentive session-based recommendation. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 1419–1428, 2017.

[24] Chen Ma, Peng Kang, and Xue Liu. Hierarchical gating networks for sequential recommendation. In *Proceedings of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 825–833, 2019.

[25] Jianmo Ni, Gustavo Hernandez Abrego, Noah Constant, Ji Ma, Keith Hall, Daniel Cer, and Yinfei Yang. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 1864–1874, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[26] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, et al. Improving language understanding by generative pre-training. 2018.

[27] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, Peter J Liu, et al. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21(140):1–67, 2020.

[28] Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022.

[29] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*, 2015.

[30] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics.

[31] Fei Sun, Jun Liu, Jian Wu, Changhua Pei, Xiao Lin, Wenwu Ou, and Peng Jiang. Bert4rec: Sequential recommendation with bidirectional encoder representations from transformer. In *Proceedings of the 28th ACM international conference on information and knowledge management*, pages 1441–1450, 2019.

[32] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014.

[33] Jiaxi Tang and Ke Wang. Personalized top-n sequential recommendation via convolutional sequence embedding. In *Proceedings of the eleventh ACM international conference on web search and data mining*, pages 565–573, 2018.

[34] Yi Tay, Vinh Q Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. Transformer memory as a differentiable search index. *arXiv preprint arXiv:2202.06991*, 2022.

[35] Aaron Van Den Oord, Oriol Vinyals, et al. Neural discrete representation learning. *Advances in neural information processing systems*, 30, 2017.

[36] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[37] Jie Wang, Fajie Yuan, Mingyue Cheng, Joemon M Jose, Chenyun Yu, Beibei Kong, Zhijin Wang, Bo Hu, and Zang Li. Transrec: Learning transferable recommendation from mixture-of-modality feedback. *arXiv preprint arXiv:2206.06190*, 2022.

[38] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Hao Sun, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, et al. A neural corpus indexer for document retrieval. *arXiv preprint arXiv:2206.02743*, 2022.

[39] Kilian Weinberger, Anirban Dasgupta, John Langford, Alex Smola, and Josh Attenberg. Feature hashing for large scale multitask learning. In *Proceedings of the 26th annual international conference on machine learning*, pages 1113–1120, 2009.

[40] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

[41] Xinyang Yi, Ji Yang, Lichan Hong, Derek Zhiyuan Cheng, Lukasz Heldt, Aditee Kumthekar, Zhe Zhao, Li Wei, and Ed Chi. Sampling-bias-corrected neural modeling for large corpus item recommendations. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 269–277, 2019.

[42] Jiahui Yu, Xin Li, Jing Yu Koh, Han Zhang, Ruoming Pang, James Qin, Alexander Ku, Yuanzhong Xu, Jason Baldridge, and Yonghui Wu. Vector-quantized image modeling with improved VQGAN. In *International Conference on Learning Representations*, 2022.

[43] Jiahui Yu, Yuanzhong Xu, Jing Yu Koh, Thang Luong, Gunjan Baid, Zirui Wang, Vijay Vasudevan, Alexander Ku, Yinfei Yang, Burcu Karagol Ayan, et al. Scaling autoregressive models for content-rich text-to-image generation. *arXiv preprint arXiv:2206.10789*, 2022.

[44] Zheng Yuan, Fajie Yuan, Yu Song, Youhua Li, Junchen Fu, Fei Yang, Yunzhu Pan, and Yongxin Ni. Where to go next for recommender systems? id-vs. modality-based recommender models revisited. *arXiv preprint arXiv:2303.13835*, 2023.

[45] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. Soundstream: An end-to-end neural audio codec. *CoRR*, abs/2107.03312, 2021.

[46] Shuai Zhang, Yi Tay, Lina Yao, and Aixin Sun. Next item recommendation with self-attention. *arXiv preprint arXiv:1808.06414*, 2018.

[47] Tingting Zhang, Pengpeng Zhao, Yanchi Liu, Victor S Sheng, Jiajie Xu, Deqing Wang, Guanfeng Liu, and Xiaofang Zhou. Feature-level deeper self-attention network for sequential recommendation. In *IJCAI*, pages 4320–4326, 2019.

[48] Zhe Zhao, Lichan Hong, Li Wei, Jilin Chen, Aniruddh Nath, Shawn Andrews, Aditee Kumthekar, Maheswaran Sathiamoorthy, Xinyang Yi, and Ed Chi. Recommending what video to watch next: a multitask ranking system. In *Proceedings of the 13th ACM Conference on Recommender Systems*, pages 43–51, 2019.

[49] Kun Zhou, Hui Wang, Wayne Xin Zhao, Yutao Zhu, Sirui Wang, Fuzheng Zhang, Zhongyuan Wang, and Ji-Rong Wen. S3-rec: Self-supervised learning for sequential recommendation with mutual information maximization. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 1893–1902, 2020.

A MODIFICATIONS TO THE P5 DATA PREPROCESSING

The P5 source code [3] pre-processes the Amazon dataset to first create sessions for each user, containing chronologically ordered list of items that the user reviewed. After creating these sessions, the original item IDs from the dataset are remapped to integers $1, 2, 3, \ldots$ [4]. Hence, the first item in the first session gets an id of '1', the second item, if not seen before, gets an id of '2', and so on. Notably, this pre-processing scheme is applied before creating training and testing splits. This results in the creation of a sequential dataset where many sequences are of the form $a, a+1, a+2, \ldots$. Given that P5 uses Sentence Piece tokenizer [30] (See Section 4.1 in [8]), the test and train items in a user session may share the sub-word and can lead to information leakage during inference.

To resolve the leakage issue, instead of assigning sequentially increasing integer ids to items, we assigned random integer IDs, and then created splits for training and evaluation. The rest of the code for P5 was kept identical to the source code provided in the paper. The results for this dataset are reported in Table 6 as the row 'P5'. We also implemented a version of P5 ourselves from scratch, for only sequential recommendation task, and the results for our implementation are depicted as 'P5-ours'. We were also able to verify in our P5 implementation that using consecutive integer sequences for the item IDs helped us get equivalent or better metrics than those reported in P5.

Table 6: Results of P5[8] with changes to prevent consecutive integer sequences of the form described in Appendix A

Methods	Sports and Outdoors				Beauty				Toys and Games			
	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
P5	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
P5-ours	0.0107	0.0076	0.01458	0.0088	0.035	0.025	0.048	0.0298	0.018	0.013	0.0235	0.015

[3] https://github.com/jeykigung/P5
[4] https://github.com/jeykigung/P5/blob/0aaa3fd8366bb6e708c8b70708291f2b0ae90c82/preprocess/data_preprocess_amazon.ipynb

Recommender Systems with Generative Retrieval

A 对 P5 数据预处理的修改

P5 源代码[3]首先对 Amazon 数据集进行预处理，为每个用户创建会话，其中包含该用户所评论物品的按时间顺序排列的列表。创建这些会话后，数据集中的原始物品 ID 被重新映射为整数$1, 2, 3, \ldots$ [4]。因此，第一个会话中的第一个物品获得 ID"1"；若第二个物品此前未出现，则获得 ID"2"；依此类推。值得注意的是，这一预处理方案在创建训练划分和测试划分之前应用。由此创建的序列数据集中，许多序列的形式为 $a, a+1, a+2, \ldots$由于 P5 使用 Sentence Piece 分词器 [30]（见 [8] 的第 4.1 节），一个用户会话中的测试物品和训练物品可能共享子词，从而在推理期间导致信息泄漏。

为解决泄漏问题，我们不再为物品分配依次递增的整数 ID，而是分配随机整数 ID，然后创建训练划分和评估划分。P5 的其余代码与论文所提供的源代码保持一致。该数据集上的结果在表 6 中以"P5"一行报告。我们还从头自行实现了一个仅用于序列推荐任务的 P5 版本，其结果以"P5-ours"表示。我们也在自己的 P5 实现中验证了，为物品 ID 使用连续整数序列，能够获得与 P5 所报告指标相当或更好的指标。

表 6：P5[8] 在进行修改以避免附录 A 所述形式的连续整数序列后得到的结果

Methods	Sports and Outdoors				Beauty				Toys and Games			
	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10	Recall@5	NDCG@5	Recall@10	NDCG@10
P5	0.0061	0.0041	0.0095	0.0052	0.0163	0.0107	0.0254	0.0136	0.0070	0.0050	0.0121	0.0066
P5-ours	0.0107	0.0076	0.01458	0.0088	0.035	0.025	0.048	0.0298	0.018	0.013	0.0235	0.015

[3] https://github.com/jeykigung/P5
[4] https://github.com/jeykigung/P5/blob/0aaa3fd8366bb6e708c8b70708291f2b0ae90c82/preprocess/data_preprocess_amazon.ipynb