Contrastive Quantization based Semantic Code for Generative Recommendation

Mengqun Jin
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China
jinmq22@mails.tsinghua.edu.cn

Zexuan Qiu
The Chinese University of Hong Kong
Hong Kong, China
qzexuan@link.cuhk.edu.hk

Jieming Zhu
Huawei Noah's Ark Lab
Shenzhen, China
jiemingzhu@ieee.org

Zhenhua Dong
Huawei Noah's Ark Lab
Shenzhen, China
dongzhenhua@huawei.com

Xiu Li*
Shenzhen International Graduate School, Tsinghua University
Shenzhen, China
li.xiu@sz.tsinghua.edu.cn

ABSTRACT

With the success of large language models, generative retrieval has emerged as a new retrieval technique for recommendation. It can be divided into two stages: the first stage involves constructing discrete Codes (i.e., codes), and the second stage involves decoding the code sequentially via the transformer architecture. Current methods often construct item semantic codes by reconstructing based quantization on item textual representation, but they fail to capture item discrepancy that is essential in modeling item relationships in recommendation sytems. In this paper, we propose to construct the code representation of items by simultaneously considering both item relationships and semantic information. Specifically, we employ a pre-trained language model to extract item's textual description and translate it into item's embedding. Then we propose to enhance the encoder-decoder based RQVAE model with contrastive objectives to learn item code. To be specific, we employ the embeddings generated by the decoder from the samples themselves as positive instances and those from other samples as negative instances. Thus we effectively enhance the item discrepancy across all items, better preserving the item neighbourhood. Finally, we train and test semantic code with with generative retrieval on a sequential recommendation model. Our experiments demonstrate that our method improves NDCG@5 with 43.76% on the MIND dataset and Recall@10 with 80.95% on the Office dataset compared to the previous baselines.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Retrieval models and ranking**.

*Xiu Li is the corresponding author.

CIKM '23, 978-1-4503-XXXX-X/18/06

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/XXXXXXX.XXXXXXX

KEYWORDS

Generative recommendation; Semantic code; Contrastive quantization

ACM Reference Format:
Mengqun Jin, Zexuan Qiu, Jieming Zhu, Zhenhua Dong, and Xiu Li. 2023. Contrastive Quantization based Semantic Code for Generative Recommendation. In *Proceedings of THE 1ST WORKSHOP ON RECOMMENDATION WITH GENERATIVE MODELS on the 32nd ACM International Conference on Information and Knowledge Management, October 21-25, 2023, Birmingham UK (CIKM '23).* , 5 pages. https://doi.org/XXXXXXX.XXXXXXX

1 INTRODUCTION

Modern recommender systems employ a retrieve-and-rank strategy, wherein a set of viable candidates are chosen during the retrieval phase and subsequently ranked using a ranking model. Since the ranking model only operates on the candidates it receives, it is essential for the retrieval phase to generate highly relevant candidates. During the retrieval stage, existing methods typically rely on traditional vector search methods, which involve complex optimization processes. Generative retrieval, on the other hand, is an emerging technique that directly generates candidate codes, eliminating the need for any discrete, non-differentiable inner-product search system or index. Utilizing autoregressive decoding and beam search, multiple viable candidates can be retrieved. In this context, we can perceive the Transformer's storage (i.e., parameters) as an end-to-end recommendation index.

In the field of recommendation systems, the current methods of generative retrieval primarily use the textual descriptions of items to construct discrete codes by using VQ-VAE [17], RQ-VAE [12, 13] or hierarchical Kmeans [16]. Similarly, for document generative retrieval methods [4, 16, 20], textual descriptions are also predominantly used to construct discrete codes. Although using such semantic codes can effectively capture semantic information, it overlooks the relationships among items, which are crucial for recommendation modeling.

In this paper, we argue that the construction of item codes in recommendation systems should not solely rely on the semantic information embedded in their textual descriptions. Instead, we propose that the relationships between items, in conjunction with semantic information, can collectively influence the quality of item

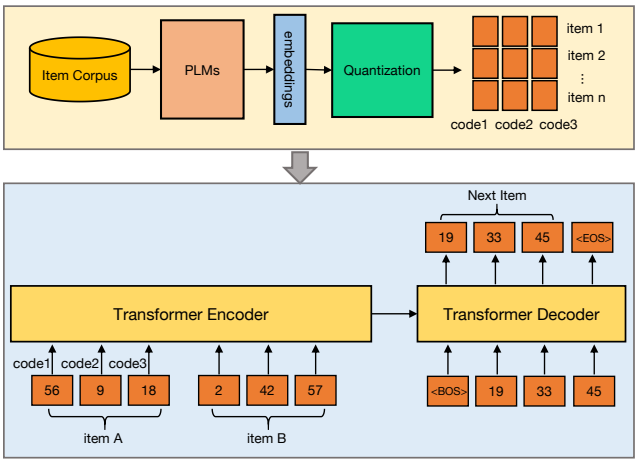

Figure 1: An Overview of Contrastive Quantization based Semantic Code Framework for Generative Recommendation.

codes. To address this, we optimized our code generation method by introducing contrastive learning [14]. In this approach, we consider embeddings generated from the decoder as positive samples, while other generated samples serve as negative samples. Our goal is to make the embeddings as similar as possible to their respective samples while increasing the dissimilarity with other samples. This allows for a more significant differentiation between the content and category of different items. By constructing positive and negative samples, we gain a holistic view of the item repository, transcending the limitations of individual item representations. Moreover, our pretraining method accommodates both small and large-scale item repository, ensuring effective handling of sequences of varying lengths.

The contributions of our work are as follow:

- In the current generative retrieval paradigm, heavy reliance on semantic information limits the effectiveness. We identify that existing code training doesn't consider the relationships between items. To address this, we introduce contrastive learning to create a code pretraining approach that captures relationships. This provide a framework for a versatile pretraining corpus for generative retrieval.
- We present a comprehensive framework for generative retrieval, accommodating both large and small pretraining corpora. We conduct experiments in sequence recommendation on sequences of varying lengths, where our approach demonstrates promising performance on two real-world datasets.

2 RELATED WORK

Generative Retrieval. Generative Retrieval is a recently introduced technique aimed at retrieving a set of relevant documents from a database. It seeks to overcome the limitations of traditional document retrieval methods by generating tokens one by one, including document titles, names, or document ID strings. Traditional document retrieval typically involves training a dual-tower model that maps both queries and documents to the same high-dimensional vector space. Subsequently, it queries all documents to return the closest ones, often necessitating large embedding tables. GENRE [2], utilizing a Transformer architecture, is applied to entity retrieval, generating entity names referenced in a given query one token at a time. On the other hand, DSI [16] is the pioneering system that assigns structured semantic DocIDs to each document, which is used in document retrieval. When presented with a query, the model autonomously generates document DocIDs one by one in an autoregressive manner.

Item Indexing in Recommendations. Models like P5 [5] and M6 [1] from the LLM4Rec series leverage knowledge from the training corpus to address various multi-class recommendation system tasks. [7] further conducts in-depth research on item indexing for its use in recommendations, exploring various categories of codes. VQRec [6] employs the OPQ [8] method to generate product codes, and it performs well in cross-domain downstream tasks. During the pretraining process in the corpus, multiple domains are mixed for code construction. Contrastive learning is employed, with mixed-domain and semi-synthetic data selected as negative samples. TIGER [12] uses RQVAE to generate semantic codes, which are then utilized in retrieval models. [13] conduct validations on sequence recommendation for videos, demonstrating the effectiveness and low storage requirements of semantic codes. However, the relationship between semantic codes is not considered. In our work, we build a more global perspective code training scheme.

3 METHODS

Our method consists of two steps:

(1) Generating pretrained item codes from an information-rich corpus: Firstly, we map content features to embedding vectors with Pretrained Language Models(PLMs). These embedding vectors are then input into a quantization model to train codes that possess both semantic information and relationships.

(2) Utilizing the pretrained retrievable codebook in downstream Tasks: In sequence recommendation tasks, we transform original items into codes. These codes are then input into a seq2seq model for training, enabling the prediction of the code for the next item.

3.1 Construction of Semantic Codes Based on Contrastive Quantization

As depicted in Figure 1, we employ general-purpose pre-trained text encoders such as Sentence-T5 and BERT to convert the textual descriptions of items into embeddings. Subsequently, the RQVAE model is utilized to transform these embeddings into codes. The Residual-Quantized Variational AutoEncoder (RQ-VAE) is a multi-stage vector quantizer that quantizes residuals at various stages to yield a tuple of codes. This autoencoder trains the encoder-decoder and codebook at the same time to reconstruct input data simply using Semantic Codes. Figure 2 (a) illustrates how residual quantization is used to produce codes.

3.1.1 RQVAE for Semantic Codes Construction. A latent representation, denoted as $z := E(x)$, is initially generated by encoding the input x through an encoder E in the RQ-VAE model. At the beginning stage ($d = 0$), the initial residual is simply defined as $r_0 := z$. We establish a codebook $C_d := e_k{}_{k=1}^{K}$, where K represents the codebook size. To quantize r_0, we select the closest embedding

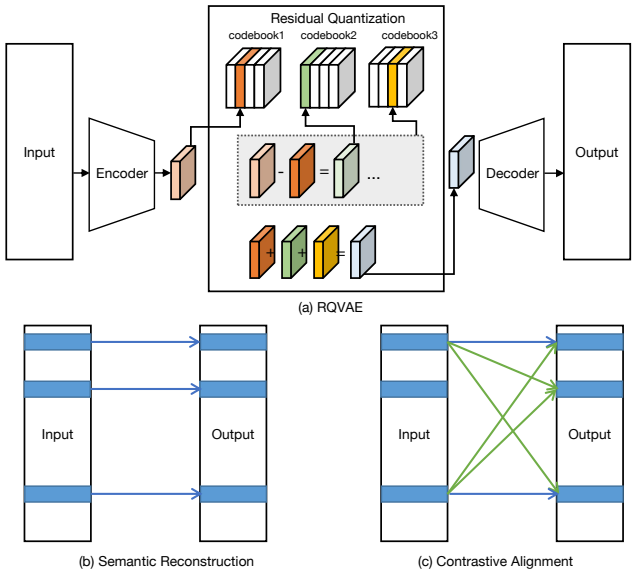

Figure 2: The RQVAE model, the semantic reconstruction and our contrastive quantization.

from the codebook corresponding to that level. The zeroth codeword is denoted as $c_0 = \text{argmin}_i \|r_0 - e_k\|$, which represents the index of the nearest embedding at $d = 0$, i.e., e_{c_0}. For level $d = 1$, the residual is defined as $r_1 := r_0 - e_{c_0}$. Subsequently, we employ the codebook of the first level to compute the code, following the same procedure used for the zeroth level.

To create semantic code, we repeat this process m times, resulting in a tuple of m codewords. This recursive approach approximates the input with increasing granularity. In contrast to using a single codebook magnified m times, we employ distinct codebooks of size K for each of the m levels. This design decision takes into consideration the decreasing average residual norm as the levels increase, thereby avoiding codeword conflicts among different granularities.

After obtaining the semantic code $(c_0, \ldots, c_{m-1})$, we sum up the quantized representation of the selected z values and create $\hat{z}$ as $\hat{z} := \sum_{d=0}^{m-1} e_{c_i}$. This resulting vector, $\hat{z}$, is then fed into the decoder, whose goal is to reconstruct the input using $\hat{z}$. The loss function employed for training the RQ-VAE model comprises two main components:

$$\mathcal{L}_{se}(x) := \mathcal{L}_{\text{recon}} + \mathcal{L}_{\text{rqvae}}, \text{ where}$$
$$\mathcal{L}_{\text{recon}} := \|x - \hat{x}\|^2 \text{ and}$$
$$\mathcal{L}_{\text{rqvae}} := \sum_{d=0}^{m-1} \|\text{sg}[r_i] - e_{c_i}\|^2 + \beta \|r_i - \text{sg}[e_{c_i}]\|^2.$$

Here, $\hat{x}$ represents the decoder's output, and sg denotes the stop-gradient operation [18]. Importantly, this loss function simultaneously trains the encoder, decoder, and codebook.

To prevent codebook collapse in RQVAE, we adopt an initialization approach [22] based on k-means clustering. In the first training batch, k-means algorithm is applied for 100 iterations and the obtained centroids are employed as our initialization strategy.

3.1.2 Contrastive Quantization for Item Relationships. Contrastive learning aims to map the student's representation $\hat{x_0}$ close to the teacher's representation x_0, while the negative samples' representations $\{\hat{x_i}\}_{i=1}^{K}$ far apart from x_0.

To achieve this, we use the following InfoNCE loss [11, 14] for model training:

$$\mathcal{L}_{co}(x) := \alpha \mathcal{L}_{\text{cl}} + \mathcal{L}_{\text{rqvae}}, \text{ where}$$
$$\mathcal{L}_{\text{cl}} := -\log \frac{\exp\left(\langle x_0, \hat{x_0} \rangle / \tau\right)}{\sum_{j=0}^{K} \exp\left(\langle x_0, \hat{x_j} \rangle / \tau\right)}.$$

where $\langle \cdot, \cdot \rangle$ denotes the cosine similarity between two feature vectors. τ represents the temperature parameter that regulates the level of concentration. Figure 2 (b) illustrates the conventional training approach for RQVAE, which focuses solely on the semantic alignment of individual items. In contrast, in Figure 2(c), contrastive learning is executed as a $(K + 1)$-way classification task, wherein its objective is to maximize the lower bound of mutual information between x_0 and $\hat{x_0}$. With the assistance of contrastive learning, items delve deeper into learning their own characteristics while distinguishing themselves from other items. Thus, we have pretrained a code corpus that considers global item relationships.

3.2 Generative Retrieval Based on Codes

We organize interacted item sequences in chronological order in the format $(item_1, ..., item_n)$. As our objective is to forecast the subsequent item $item_{n+1}$, we tokenize each item into codes $C = (c_0, \ldots, c_{m-1})$ and obtain a sequence of $s = (c_{1,0}, \ldots, c_{1,m-1}, c_{2,0}, \ldots, c_{2,m-1}, \ldots, c_{n,0}, \ldots, c_{n,m-1})$. The seq2seq model is then trained to predict $(c_{n+1,0}, \ldots, c_{n+1,m-1})$, the code of $item_{n+1}$. Finally, we query the item-code table to retrieve the corresponding items for the code table.

4 EXPERIMENTS AND ANALYSIS

4.1 Experimental Settings

We conduct extensive experiments on real-world datasets to answer the following questions.

- **Q1:** Is our proposed method superior to state-of-the-art models?
- **Q2:** How sensitive is our model to hyper-parameters?

Datasets and Tasks. We conducted experiments on two real-world datasets: MIND [21], Amazon's Office Product domain (Office). These datasets cover a wide range of item proportions, from 10,000 to 37,000 items. MIND is a comprehensive benchmark dataset for news recommendations, while Amazon's datasets comprise user comments and product descriptions. Details of our preprocessed datasets are summarized in Table 1, including user interaction sequence lengths (Seq), average (mean), and median (medium) sequence lengths. To manage the scale of MIND, we followed the approach outlined in [4], retaining interactions from at least 15 users and capping sequence lengths at 70. For Amazon datasets, we filtered users who interacted with no more than 5 items. To explore how the number of items impacts our pretrained model, we split the Office dataset into Office (S) and Office (L), featuring small and large item subsets of 14,000 and 37,000 items, respectively. We extracted three descriptions from the MIND dataset: type, subtype,

and title, and for the Amazon datasets, we selected type, brand, title, and category. These descriptions served as inputs for Sentence-T5 [10] PLM, producing 768-dimensional semantic embeddings for each item.

Table 1: Descriptive Statistics for experimental datasets

Datasets	User	Item	Seq	Mean	Medium
MIND	29207	12251	[15,70]	25.06	22
Office(S)	2868	14618	[10,20]	13.21	12
Office(L)	16696	37347	[5,50]	8.38	12

Parameter Setting. The RQ-VAE [9, 22] model comprises three core components: a DNN encoder, a residual quantizer, and a DNN decoder. The encoder has three intermediate layers with dimensions of 512, 256, and 128, using ReLU activation functions, and ends with a 96-dimensional latent representation. A unique 3-tuple semantic code is generated for each item, with shared parameters across our three-level codebooks, ensuring consistency. The code cardinality is fixed at 64. We set hyperparameters α to 0.1, β to 0.25 and τ to 0.1 for optimization. The Adam optimizer is used with a learning rate of 0.0001 and a batch size of 256. Importantly, our method ensures that no items share the same code, allowing seamless integration of the pretrained code into downstream recommendation models without manual bit adjustments to avoid conflicts. To initially provide a pretrained model suitable for downstream tasks, we experimented with a seq2seq transformer model [3, 15, 19]. We utilized a batch size of 512 and set the learning rate to 0.001 for training.

Evaluation Metrics. There are two critical aspects to validate our pretrained model, to assess the stability and effectiveness of RQ-VAE training and the superiority of our code in downstream tasks like sequence recommendation. We use two evaluation metrics for RQ-VAE: cosine similarity and $Top-K$ precision. Cosine similarity measures similarity between input and output embeddings for individual samples. $Top-K$ precision quantifies the Euclidean distance between the input of each item and the output of all items and selects items with the K closest ones. Regarding the second aspect, we utilize Recall and Normalized Discounted Cumulative Gain (NDCG). We follow the standard leave-one-out evaluation protocol, reserving the last item for testing, the item before that for validation, and using the rest for training.

4.2 Performance Comparison and Analysis

Result 1: Performance Comparison (for Q1) To validate our contrastive quantization based method, we conducted various comparative experiments, including using $\mathcal{L}_{\text{se}}$ as a baseline, employing only $\mathcal{L}_{\text{co}}$, and combining $\mathcal{L}_{\text{se}}$ with $\mathcal{L}_{\text{co}}$. Table 2 reveals that, for the MIND dataset, our contrastive learning method outperforms the other two, with significant improvements in NDCG and Recall, approximately 38% and 36%, respectively. We also compared our semantic code construction with random hashing methods, confirming its benefits for model inference. Table 3 displays results on Office (S) and Office (L). For shorter sequences and fewer samples, $\mathcal{L}_{\text{co}} + \mathcal{L}_{\text{se}}$ slightly outperforms $\mathcal{L}_{\text{co}}$, while for longer sequences and more samples, $\mathcal{L}_{\text{co}}$ excels. This highlights the versatility of our contrastive quantization based approach for pretraining large-scale item models, with MSE acting as a lower bound for RQVAE training.

Table 2: Experiment results on MIND dataset.

		@5	@10	@20	@40
Random	NDCG	0.0201	0.0265	0.0327	0.0390
	Recall	0.0319	0.0519	0.0766	0.1075
$\mathcal{L}_{se}$	NDCG	0.0363	0.0474	0.0594	0.0727
	Recall	0.0560	0.0905	0.1384	0.2031
$\mathcal{L}_{co}$	NDCG	**0.0522**	**0.0663**	**0.0817**	**0.0975**
	Recall	**0.0803**	**0.1241**	**0.1855**	**0.2625**
$\mathcal{L}_{se} + \mathcal{L}_{co}$	NDCG	0.0444	0.0574	0.0710	0.0865
	Recall	0.0677	0.1081	0.1621	0.2376
Impr.	NDCG	**43.76%**	**39.90%**	**37.52%**	**34.10%**
	Recall	**43.34%**	**37.21%**	**34.05%**	**29.24%**

Table 3: Experiment results on Office dataset.

	Metrics	Office(S)		Office(L)	
		@10	@20	@10	@20
$\mathcal{L}_{se}$	NDCG	0.0024	0.0032	0.0041	0.0053
	Recall	0.0037	0.0068	0.0075	0.0123
$\mathcal{L}_{co}$	NDCG	0.0034	0.0043	**0.0047**	**0.0059**
	Recall	0.0060	0.0094	**0.0079**	**0.0125**
$\mathcal{L}_{se} + \mathcal{L}_{co}$	NDCG	**0.0035**	**0.0042**	0.0042	0.0052
	Recall	**0.0066**	**0.0096**	0.0079	0.0119
Impr.	NDCG	**43.80%**	**31.06%**	**15.11%**	**10.53%**
	Recall	**80.95%**	**41.03%**	**5.38%**	**1.46%**

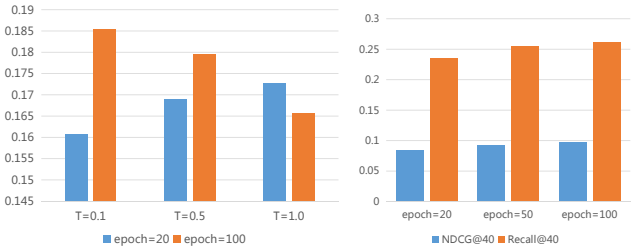

Figure 3: Analysis on τ and training epochs.

Result 3: Sensitivity Analysis of Temperature τ and training epochs (for Q2)

To analyse how temperature τ affects RQVAE training, we use MIND dataset with T = [0.1, 0.5, and 1.0]. In this case, RQ-VAE was trained for 20 and 100 epochs, respectively, with he best results of Recall@40 when T=1.0 and T=0.1 in Figure 3. This shows that the training of RQVAE fluctuates with τ within a certain range. Furthermore, when we fix T=0.1, we conduct training for [20, 50, 100] epochs. As evident from the Figure 3, with increasing training epochs, the loss steadily decreases, NDCG and Recall consistently improve, indicating that RQ-VAE captures richer semantic information.

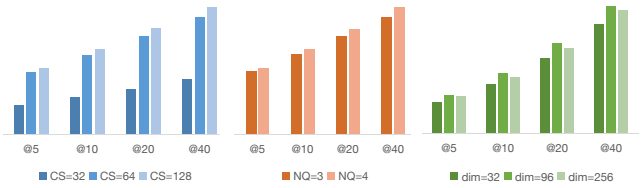

Figure 4: Analysis on codebook size, number of codebooks, dim of embeddings.

Result 4: Sensitivity Analysis of Codebook (for Q2)

Key parameters influencing RQVAE's effectiveness encompass codebook size, codebook quantity, and embedding dimension. As illustrated in Figure 4, as the codebook space expands, NDCG exhibits a consistent upward trend. This implies that there is a larger available space to represent each item, leading to more precise codebook allocation. Moreover, with the increase in embedding dimension, item representational capabilities are enhanced, resulting in improvements across various metrics.

5 CONCLUSION

Existing generative retrieval model predominantly reliant on LLMs for code generation, tend to be overly tied to semantic content, overlooking the broader context of product information and user behavior. In this work, we construct codes with a more comprehensive view for generative retrieval. We introduced contrastive learning into the code generation process. By creating positive and negative samples, we optimized our model's understanding of the global product landscape. Experimental results indicate an enhancement in the performance of our code repository across downstream tasks. In the future, we anticipate that further advancements in code generation techniques, potentially incorporating richer contextual information and user behavior analysis, will continue to improve the efficacy of generative retrieval systems in catering to evolving user needs.

REFERENCES

[1] Zeyu Cui, Jianxin Ma, Chang Zhou, Jingren Zhou, and Hongxia Yang. 2022. M6-rec: Generative pretrained language models are open-ended recommender systems. *arXiv preprint arXiv:2205.08084* (2022).

[2] Nicola De Cao, Gautier Izacard, Sebastian Riedel, and Fabio Petroni. 2020. Autoregressive entity retrieval. *arXiv preprint arXiv:2010.00904* (2020).

[3] Gabriel de Souza Pereira Moreira, Sara Rabhi, Jeong Min Lee, Ronay Ak, and Even Oldridge. 2021. Transformers4rec: Bridging the gap between nlp and sequential/session-based recommendation. In *Proceedings of the 15th ACM Conference on Recommender Systems*. 143–153.

[4] Chao Feng, Wuchao Li, Defu Lian, Zheng Liu, and Enhong Chen. 2022. Recommender Forest for Efficient Retrieval. *Advances in Neural Information Processing Systems* 35 (2022), 38912–38924.

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

[6] Yupeng Hou, Zhankui He, Julian McAuley, and Wayne Xin Zhao. 2023. Learning Vector-Quantized Item Representation for Transferable Sequential Recommenders. In *TheWebConf*.

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

[8] Herve Jégou, Matthijs Douze, and Cordelia Schmid. 2011. Product Quantization for Nearest Neighbor Search. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 33, 1 (2011), 117–128. https://doi.org/10.1109/TPAMI.2010.57

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

[10] Jianmo Ni, Gustavo Hernández Ábrego, Noah Constant, Ji Ma, Keith B Hall, Daniel Cer, and Yinfei Yang. 2021. Sentence-t5: Scalable sentence encoders from pre-trained text-to-text models. *arXiv preprint arXiv:2108.08877* (2021).

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

[12] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan Hulikal Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Tran, Jonah Samost, et al. 2024. Recommender systems with generative retrieval. *Advances in Neural Information Processing Systems* 36 (2024).

[13] Anima Singh, Trung Vu, Raghunandan Keshavan, Nikhil Mehta, Xinyang Yi, Lichan Hong, Lukasz Heldt, Li Wei, Ed Chi, and Maheswaran Sathiamoorthy. 2023. Better Generalization with Semantic IDs: A case study in Ranking for Recommendations. *arXiv preprint arXiv:2306.08121* (2023).

[14] Siqi Sun, Zhe Gan, Yu Cheng, Yuwei Fang, Shuohang Wang, and Jingjing Liu. 2020. Contrastive distillation on intermediate representations for language model compression. *arXiv preprint arXiv:2009.14167* (2020).

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

[16] Yi Tay, Vinh Tran, Mostafa Dehghani, Jianmo Ni, Dara Bahri, Harsh Mehta, Zhen Qin, Kai Hui, Zhe Zhao, Jai Gupta, et al. 2022. Transformer memory as a differentiable search index. *Advances in Neural Information Processing Systems* 35 (2022), 21831–21843.

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

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

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

[20] Yujing Wang, Yingyan Hou, Haonan Wang, Ziming Miao, Shibin Wu, Qi Chen, Yuqing Xia, Chengmin Chi, Guoshuai Zhao, Zheng Liu, et al. 2022. A neural corpus indexer for document retrieval. *Advances in Neural Information Processing Systems* 35 (2022), 25600–25614.

[21] Fangzhao Wu, Ying Qiao, Jiun-Hung Chen, Chuhan Wu, Tao Qi, Jianxun Lian, Danyang Liu, Xing Xie, Jianfeng Gao, Winnie Wu, et al. 2020. Mind: A large-scale dataset for news recommendation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 3597–3606.

[22] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *IEEE/ACM Transactions on Audio, Speech, and Language Processing* 30 (2021), 495–507.