SIDE: Semantic ID Embedding for effective learning from sequences

Dinesh Ramasamy, Shakti Kumar, Chris Cadonic
Jiaxin Yang, Sohini Roychowdhury, Esam Abdel Rhman, Srihari Reddy
{dineshr,shaktik,ccadonic,jiaxiny,sroychowdhury1,esam,sriharir}@meta.com
Meta Platforms, Inc.

Abstract

Sequence-based recommendations models are driving the state-of-the-art for industrial ad-recommendation systems. Such systems typically deal with user histories or sequence lengths ranging in the order of O(10^3) to O(10^4) events. While adding embeddings at this scale is manageable in pre-trained models, incorporating them into real-time prediction models is challenging due to both storage and inference costs. To address this scaling challenge, we propose a novel approach that leverages vector quantization (VQ) to inject a compact *Semantic ID (SID)* as input to the recommendation models instead of a collection of embeddings. Our method builds on recent works of SIDs by introducing three key innovations: (i) a multi-task VQ-VAE framework, called *VQ fusion* that fuses multiple content embeddings and categorical predictions into a single Semantic ID; (ii) a parameter-free, highly granular SID-to-embedding conversion technique, called *SIDE*, that is validated with two content embedding collections, thereby eliminating the need for a large parameterized lookup table; and (iii) a novel quantization method called *Discrete-PCA* (DPCA) which generalizes and enhances residual quantization techniques. The proposed enhancements when applied to a large-scale industrial ads-recommendation system achieves 2.4× improvement in normalized entropy (NE) gain and 3× reduction in data footprint compared to traditional SID methods.

Keywords

Vector quantization, fusion, encoding, decoder, ads-ranking system

ACM Reference Format:
Dinesh Ramasamy, Shakti Kumar, Chris Cadonic and Jiaxin Yang, Sohini Roychowdhury, Esam Abdel Rhman, Srihari Reddy. 2025. SIDE: Semantic ID Embedding for effective learning from sequences. In *Proceedings of AdKDD 2025 (AdKDD '25, Toronto, Canada).* ACM, New York, NY, USA, 6 pages.

1 Introduction

Industrial ads-ranking systems incorporate diverse signals from multiple sources to accurately predict user-engagement towards specific content. Traditionally, these systems have relied on aggregation based signals, which summarize user behavior and ad-attributes into coarse grained features. However, this approach has limitations, as it fails to capture the nuances of user interactions over time. Recent works such as in [5]- [12] demonstrate the importance of long sequence user behavior for improvement in click-through-rate (CTR) prediction by leveraging O(10^3) to O(10^4) events. In such systems, there is a need to enrich user-engagement signals / events for improved prediction using content engagement signals, which typically are in the form of embeddings. In this work, we address two major challenges while using quantized SIDs to enhance user-ad engagement predictions: (1) How can we perform VQ efficiently when we have a plethora of content signals (embeddings and categorical predictions) as is the case in industrial ads-ranking systems, and (2) How can we structure VQ outputs to utilize them efficiently along with improved ranking performance of downstream models? To address these questions, we propose a novel multi-input vector-quantized variational auto-encoder (VQ-VAE) (called *VQ fusion*), as shown in Figure 1, which addresses the gap in existing SID approaches that require large volumes of data to relearn embeddings for different SID *n*-grams or *tokens*. The proposed *VQ fusion* method consists of an encoder-network that takes different content signals as input and produces a shared latent representation that is quantized and passed through a pseudo-symmetric decoder network; thereby enabling reconstruction of the different content signals. Thus, the proposed structured code-books avoid relearning the ID-to-codeword mappings and significantly reduce the number of parameters and memory required in ads-ranking.

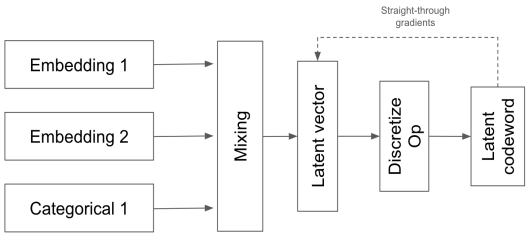

Figure 1: The proposed fusion VQ-VAE setup (encoder portion). An unmixing decoder architecture (symmetric to mixing) is used to train the encoder.

This paper makes three major contributions. First, we propose a novel vector quantization method based on *Residual Quantization* called Discrete PCA (DPCA). Second, we detail a method for fusing multiple embedding and categorical signals into a single Semantic ID which we call VQ-fusion, thus drastically reducing the data storage cost. Third, we introduce a new usage of Finite Scalar

AdKDD '25, Toronto, Canada, 2025

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM

Quantization (FSQ[11]) and the newly proposed DPCA methods to improve ranking outcomes using the so-called SIDE property, which stands for the ability to convert SIDs to latent embeddings in an embedding table free fashion. We demonstrate significant gains in ranking metrics (at least 2.4× gain in normalized entropy) and the total return on investment (at least 7.4× RoI) compared to existing approaches in an industrial ads-ranking setting.

2 Related Work

Scalar quantization [11] traditionally involves quantizing each dimension of a vector independently using a scalar codebook, which can be static (examples are `int4` and `float16` quantization which use 4 and 16 bits per dimension, respectively). A class of compression techniques involves finding a mapping to a lower dimensional subspace so that "most" of the information content in the embedding is preserved and the PCA method identifies the optimal linear mapping when the loss is euclidean. Vector quantization (VQ) started off with the now ubiquitous k-means clustering algorithm that can be extended in two orthogonal ways: (i) Residual k-means clustering which involves clustering the residuals to create a hierarchy of codewords and (ii) Product Quantization [6], which involves using parallel k-means algorithms on a subset of dimensions and is considered as a hybrid between scalar and vector quantization algorithms.

Another clustering technique is called Iterative-Quantization (ITQ, [4]). This involves using a linear transformation to map embeddings into the latent space where they are quantized using a binary $\{-1, 1\}$ codebook. Since the transformations are done in parallel (matrix multiplication), we classify this as a form of product quantization clubbed with auto-encoder methods. A similar auto-encoder with scalar (binary) quantization setup called Lookup-Free Quantizer (LFQ, [2]) was used recently for video generation. Others have proposed Finite Scalar Quantization (FSQ, [11]) which extends LFQ by allowing for more buckets per dimension. One can combine VQ based methods (e.g., k-means) with auto-encoder setup as presented in the paper on RQ-VAE[8] that improves RQ's ability to learn hierarchical structure by adding structured quantizer dropout, as is shown in the proposed work.

Recently there has been lot of interest in extreme scalar quantization for LLMs like 1.58-bit LLM [10]. We draw inspiration from this work and use a ternary $\{-1, 0, 1\}$ scalar codebok as a universal codebook for our proposed DPCA method. In terms of applications of VQ-techniques to ad-ranking systems like those presented in [3], the recent work on using RQ for defining SIDs[1] and the work on generative retrieval [13] stand out. Both these papers use n-grams of RQ codewords as SIDs and utilize these SIDs as categorical features, thereby relying largely on embedding tables. We use this approach and more traditional k-means as baselines in our experiments.

3 Vector Quantization Method

In this section, we define our proposed method *Discrete-PCA*, and expand on the techniques to combine multiple embeddings and categorical signals into one-feature via our proposed VQ-fusion method. Next, we demonstrate our embedding table-free technique called SIDE to convert codewords to embeddings in ads-ranking models. The base concept of residual quantization (RQ) [1] involves the use of D independent codebooks that are stacked on top of each other to compress the residue from the previous layer. The approximation to the original embedding is therefore the sum of the D selected codewords.

3.1 Structured-Quantization method

We propose a new quantization technique by noting that k-means uses a single point for each codeword. This means that the codeword collection has no inherent structure. As a result we need a large number of parameters to represent the codewords (k times d) and we cannot increase the information content measured by $\log k$ significantly (except via product and residual quantization extensions of course). To circumvent this issue we make the codewords structured – to be precise we make groups of L codewords co-linear. Thus, the codebook has the following structure:

$$C = \left\{ s_l \cdot \mathbf{u}_k + \mathbf{b}_k : \forall\, k, l \right\} \tag{1}$$

where $\mathbf{u}_k$ is the codeword group's direction vector (unit vector), $\mathbf{b}_k$ is the reference point for codeword group k and s_l is the signed distance of the selected codeword l from the reference point $\mathbf{b}_k$ along the direction $\mathbf{u}_k$; giving us our *Structured-Quantization* method.

Next, we present the inference logic that is leveraged by the training stage.

3.1.1 Inference procedure. This three-step method identifies the codeword indices k and l given a codebook C and a point $\mathbf{x}$.

First, we first choose the "line" $\hat{k}$ which is closest to the given point $\mathbf{x}$ by measuring the distance of the point from the line corresponding the k-th codeword group and picking the closest line (by choosing $\mathbf{u}_k$):

$$\hat{k} = \arg\min_k \left(\|\mathbf{x} - \mathbf{b}_k\|^2 - |\langle \mathbf{x} - \mathbf{b}_k, \mathbf{u}_k \rangle|^2 \right). \tag{2}$$

Once we identify the closest line, we estimate the optimal signed distance $\hat{s}_{\hat{k}}$ through the following inner product:

$$\hat{s}_{\hat{k}} = \left\langle \mathbf{x} - \mathbf{b}_{\hat{k}}, \mathbf{u}_{\hat{k}} \right\rangle. \tag{3}$$

The estimated projection weight $\hat{s}_{\hat{k}}$ is then quantized using an scalar quantizer $Q(\cdot)$ with L bins to identify the codeword index $\hat{l}_{\hat{k}}$ for the signed distance. This scalar quantizer used is FSQ and is shared across the collection $\{\mathbf{u}_k\}$.

3.1.2 Training procedure. For training this VQ-VAE model, in addition to the reconstruction loss, we add the commitment and codebook losses and use a straight-through estimator to propagate the gradients to the encoder network.

3.2 Towards Discrete-PCA

In this work, we assume that a vector codebook is redundant in the context of residual and/or product quantization based structured quantization; thereby setting the the number of vector codewords to 1. We design residual structured quantization as a form of *generalized PCA* due to its stacked nature which promotes the Matryoshka property[7] of such codebooks. Since the residual depth is difficult to increase due to its serial nature, we propose to use a combination of product and residual quantization to increase the quantizer bit-width. Additionally, we design the scalar codebook (s_l in (1)) by assuming that just three values $\{-1, 0, 1\}$ suffice for the

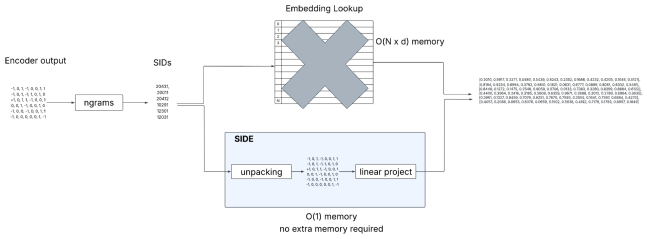

Figure 2: Process of O(Nxd) memory removal using the embedding-free lookup in proposed SIDE. N = hash size of embedding tables and d=embedding dimension. The deterministic unpacking in SIDE ensures O(1) memory requirement.

signed distance. This follows recent research on Large Language Model weights, which suggest that such a ternary codebook can be universal for LLM weights [10]; thereby resulting in quantizer Discrete-PCA. The implementation of the scalar quantizer follows Finite Scalar Quantizer (FSQ) [11] with number of latent dimensions equal to 1 (or the number of product quantization groups when used in conjunction with PQ).

DPCA can be viewed as a form of generalized PCA where the projection weights can only take three values $\{-1, 0, 1\}$ and thus the *component vectors* no longer turn out to be orthogonal. The codebook collection of this method is given by the following expression:

$$C = \sum_d \left(s_d \mathbf{u}_d + \mathbf{b}_d\right), \tag{4}$$

where d refers to the residual depth and s_d can take one of three values in $\{-1, 0, 1\}$. When used alongside product-quantization, we have p-parallel orthogonal codebooks, each with structure resembling (4).

3.3 SIDE: Converting SIDs to embeddings

When codewords are exported, they are typically n-gram-ed to form a SID or a collection of SIDs. For instance, each codeword is of the form $\mathbf{c} \in \{-1, 0, 1\}^n$ and we convert these to Semantic ID (SID) using the following n-gram operation:

$$s = \sum_{k=1}^{n} 3^k (1 + c_k) \tag{5}$$

In the ranking model, this n-gram operation can be undone using floor divide and modulo operations that yields the codeword vectors $\mathbf{c}$ with entries in $\{-1, 0, 1\}$. These latent codewords are called SID Embeddings or *SIDE* and are used directly as embeddings. It is noteworthy that without SIDE, the memory requirement of any quantizer scales as $O(\exp(Cb) \times d)$, where b is the number of bits and d is the dimension of the embedding for some constant C; due to embedding tables used for decoding embeddings. The proposed SIDE approach reduces this to $O(b \times d)$ as shown in Figure 2.

4 VQ Fusion

In this section we describe the ads-ranking systems that utilize a multitude of content signals. These signals can be broadly categorized into two types: embeddings and categorical predictions. While incorporating all content signals into an ads-ranking model can be challenging, adding each signal one by one using SIDs obtained from VQ-methods can lead to increased storage costs and decrease the overall system efficiency. To address this challenge, we propose a novel approach of fusion, called *VQ-fusion* that jointly encodes all available content signals (embeddings, categorical predictions, etc) into a single SID. Our approach leverages the multi-input multi-output auto-encoder setup, where, the latent vector is learned using the following mixing model:

$$\mathbf{h} = f\left(e_1\left(\mathbf{x}_1\right), \ldots, e_n\left(\mathbf{x}_n\right)\right), \tag{6}$$

from the n input signals as $\{\mathbf{x}_k\}$. $e_k(\cdot)$ are the corresponding encoders for each input signal and $f(\cdot)$ is the fusion network. Next, vector quantization is performed $\mathbf{h}$ either using a version of FSQ [11] (with codewords for each dimension corresponding to $\{-1, 0, 1\}$) or our proposed DPCA method in Section 3 to arrive at the codeword $\hat{\mathbf{h}}$. Next, $\hat{\mathbf{h}}$ (via $\mathbf{s}$ demonstrated below) is used to reconstruct the inputs $\{\mathbf{x}_k\}$ using the decoder network. This network is trained end-to-end to minimize the reconstruction loss. To propagate gradients through the non-differentiable VQ operation, $\hat{\mathbf{h}}$ is replaced with its straight-through version $\mathbf{s} = \mathbf{h} - \texttt{stop\_gradient}\left(\mathbf{h} - \hat{\mathbf{h}}\right)$.

Next, we decode the inputs from the straight-through version $\mathbf{s}$ of the latent codeword using the decoder architecture which is symmetric to the encoder as $\hat{\mathbf{x}}_k = g_k\left(r\left(\mathbf{s}\right)\right)$, where $r(\cdot)$ is the shared model and $g_k(\cdot)$ are individual task decoders. By training this joint auto-encoder to minimize weighted reconstruction loss $\sum_k w_k \ell_k\left(\mathbf{x}_k, \hat{\mathbf{x}}_k\right)$ across all content signals, a compact and informative representation can be learned that leverages the underlying relationships between the different signals. This approach leads to reduction in the cost of the semantic ID representation and improvement in the quality of representations across signals.

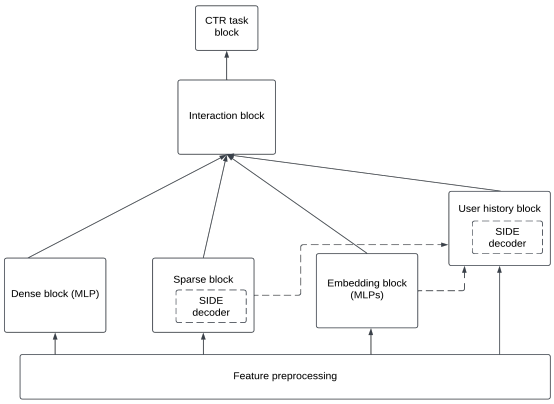

Figure 3: Ranking system overview. The dotted lines represent query vectors for PMA that are typically drawn from Sparse and Embedding blocks.

5 Ads-Ranking system

In this section we focus on the relevant components of the ads-ranking stack [3] and highlight the changes needed to utilize the SIDE property of FSQ and DPCA based SIDs. Large scale deep-learning ranking systems can be broken down into (i) sparse (ii) dense (iii) embedding and (iv) sequence or user history blocks as shown in Figure 3.

Sparse block: Each sparse / categorical / ID feature is mapped to a d-dimensional embedding $\mathbf{s}_k$ using a learnable embedding table $\mathbf{s}_k = S_k\mathbf{x}_k$, where, $\mathbf{x}_k$ is the r-hot representation of the k-th sparse feature and S_k is the associated embedding table for the feature.
Dense block: All dense features are collated and represented by a single vector $\mathbf{v}$. This vector is mapped to multiple d-dimensional representations $\{\mathbf{d}_1, \ldots, \mathbf{d}_r\}$ using learnable functions $D_k(\mathbf{v})$ for k in $\{1, \ldots, r\}$.
Embedding block: Embedding features are handled similar to dense features with each embedding being operated on by a separate function (typically a two-layer multi-layer perceptron) with $r = 1$ outputs to yield the d-dimensional representation for k-th embedding given by $\mathbf{e}_k$.
User history block: On the user-history side, the traditional approach is to convert timestamped sequence of sparse signals into embeddings using an embedding table and then leverage sequence algorithms like the Transformer-Encoder or Pooled-Multihead Attention (PMA)[9]. The user history block is a one-layer PMA module is $U = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$, where, U is a $k \times d$ matrix representing the k user history embeddings. Q is a $k \times d$ matrix derived from a subset of other sparse and embedding features $\{\mathbf{s}_k\}$ and $\{\mathbf{e}_k\}$. The key-matrix in PMA, $K = V\Theta$, where $V = [\mathbf{e}_1, \ldots, \mathbf{e}_l]$ is the $l \times d$ user history embedding matrix. Here l denotes the length of the user sequence and Θ is a $d \times d$ model weight matrix that maps each user history embedding to its corresponding key. We implement SIDs or their n-grams as sparse sequence features (based on prior work in [1]) to derive the user history embeddings $\mathbf{e}_k$.

In the following sections we demonstrate the application of the SIDE property in the user history block and compare the user-ad engagement results to that using SIDs (or their n-grams) directly as sparse sequence features. We convert the SIDs to embeddings $\mathbf{h}_k \in \mathbb{R}^t$ for some t typically smaller than d (as discussed in Section 3.3). Next, we use a d-dimensional projection $\mathbf{h}_k\Omega$ of these SIDE embeddings as features in the user history model, where Ω is a learnable weight matrix of shape $d \times t$.

5.1 Online Training

In this sub-section we present the online training components of the our VQ-fusion model (and correspondingly DPCA model) as a part of the ads-ranking system in production.

5.1.1 Data Collection and Model Training. The data collection process involves logging data from currently deployed upstream content models, which is then routed via logging to train our VQ-fusion encoders. The logged upstream data also serves as the ground-truth for training our auto-encoder. To ensure compliance with data privacy regulations, data is cleaned and filtered through rigorous privacy filters, such as ID masking, prior to any training. The VQ-fusion model converges quickly and needs 1 billion data points for initial offline training. Once the fusion model is trained, its inference is used to generate output features (SID) to train the downstream ranking model with 29 billion data points for the initial offline training.

5.1.2 Recurring Training and Inference. Upstream content models that produce embeddings leverage recurring training through periodic scheduled training jobs. We generate pipelines for our VQ-fusion model and apply them into the existing upstream training jobs. This results in our VQ-fusion models getting trained periodically whenever upstream models are trained. This also addresses any new item embeddings which are learnt in VQ-fusion during this recurring training. The inference output (namely SID) produces IDs that are then logged into downstream training tables and serve as output features for our ads-ranking system.

6 Experiments and Results

In this section, we present the results for reconstruction accuracy of the SID encoding methods RQ-VAE, FSQ and our proposed DPCA in multiple settings: (i) one-embedding at a time (1:1) for two content embeddings and (ii) in the context of VQ-fusion with the same two embeddings. Also, the impact of the proposed SIDE technique in capturing dot-product ordering of the two embedding collections in the 1:1 setting is presented. Finally we analyze the overall ads-ranking performances in Section 6.2.

6.1 Encoder Design

6.1.1 1:1 encoder. We implement two content embeddings from the text-only content model (for converting text → embeddings) and the image-only content model (for converting image → embeddings). For both these embeddings we use 1 billion datapoints to train the 1:1 quantizers (text embedding quantizer and image embedding quantizer using each of RQ, FSQ and DPCA). For fusion setting we jointly train a single quantizer on the combined dataset of text and image embedding using FSQ, DPCA and RQ methods. The cosine reconstruction loss of each method is shown in Table 1.

Setting	Method	Image	△ %age	Text	△ %age
1:1	RQ	0.1995	-	0.3066	-
	DPCA	0.1870	-	**0.2319**	-
	FSQ	**0.1549**	-	0.2435	-
Fusion	RQ	0.2224	11.47%	0.2892	5.67%
	DPCA	**0.1945**	**4.01%**	**0.2365**	**1.98%**
	FSQ	0.21607	39.48%	0.2803	15.11%

Table 1: Cosine reconstruction loss comparison of three 24-bit VQ methods for two 1024-dimensional content embeddings. 1:1 refers to separate training of both the image and text quantizers while the fusion setting refers to using a single quantizer for both embeddings; thereby utilizing half the number of total bits. △ %age refers to the percentage increase in cosine reconstruction loss for fusion model for each method compared to 1:1 encoder model.

6.1.2 Parameter tuning: We tune the 3 major hyper-parameters as the numbers of: scalar quantization buckets, residual quantization layers, and product quantization buckets, to minimize the reconstruction loss of the embeddings. This is performed by standard parameter sweep. Once the reconstruction loss is minimized, the selected parameters are used to test loss values of the derived SIDs in ads-ranking model as sparse features to ensure loss parity or improvements in ads-ranking model performances over raw embeddings.

6.1.3 VQ-Fusion. We train a single quantizer on the combined dataset of image and text embeddings jointly, thereby increasing the compression by a factor of two. In Table 1, Δ%age demonstrates the enhancement of the fusion process over 1:1 quantization method such that for image reconstruction tasks, 1:1 quantizer loss is 0.1995 and fusion quantizer loss is 0.2224 leading to Δ%age = 11.47%. We observe that in a fusion setting (joint training of both image and text quantizer), our proposed DPCA method has the least change in reconstruction loss when compared to FSQ and RQ methods as shown by the least Δ%age in Table 1.

6.1.4 Efficacy of latent codeword representation. As discussed in Section 3.3, for FSQ and DPCA, the n-gram operation is reversed and centered (by subtracting 1 in this case) to convert the Semantic IDs into embeddings. Next, the following steps are performed for the 1:1 encoder: The kNN results (using cosine similarity) are computed for the 1:1 24-bit compression of the two 1024-dimensional content embeddings presented in Table 2. These kNN results are then compared to those obtained from the original raw embeddings' closest 20 kNN results (based on cosine similarity as well). We use Recall@k for different values of k to quantify the correctness of this comparison by using 1000 random seed queries picked from a 200K large embedding corpus.

Corpus	Method	R@20	R@50	R@100
Image	DPCA	0.1467	0.3326	0.5088
	FSQ	**0.2370**	**0.4187**	**0.5998**
Text	DPCA	**0.1703**	**0.3418**	**0.5213**
	FSQ	0.1323	0.2943	0.4848

Table 2: Comparison of DPCA and FSQ methods using Recall at 20, 50 and 100 for closest-20 neighbors as defined by the original embedding to show the efficacy of Semantic ID to latent embedding conversion (SIDE property).

Based on Table 2, we observe that both FSQ and proposed DPCA retain the dot-product ordering information present in the original embedding even with a massive $1024 \times 16 \div 24 \approx 682$ compression ratio. We verify these results qualitatively by inspecting k-NN results like those in Figure 4.

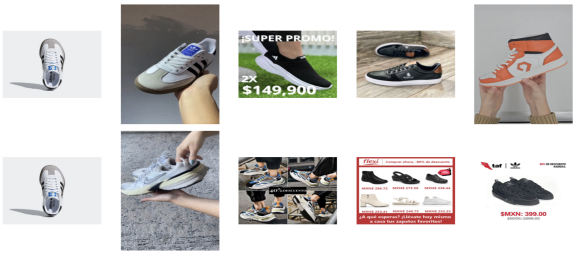

Figure 4: k-NN results with leftmost ad as the query using cosine similarity as the measure. Top: Thumbnails corresponding to original image embedding. Bottom: Thumbnails corresponding to 680x compressed FSQ SIDE representation. All of these ads are related to sneakers.

6.2 Ads-Ranking

In our ads-ranking systems, we predict two main tasks: click (CTR) and the conversion (CVR: eventual purchase) probabilities. In this paper we access the improvement for CTR prediction in terms of Normalized Entropy, which is defined as follows:

$$\text{NE} = \frac{{}^1\!/_N \sum (y_i \log p_i + (1 - y_i) \log(1 - p_i))}{\hat{p} \log \hat{p} + (1 - \hat{p}) \log (1 - \hat{p})}, \tag{7}$$

where, y_i are the labels, p_i are model predictions, $\hat{p} = {}^{\sum y_i}\!/_N$ is the prior probability and N is the number of samples. We utilize $\frac{(NE_{\text{click}} + NE_{\text{conv}})}{2C}$ to compute the overall RoI of the feature, where C is the logging cost of onboarding the feature.

In this experiment, we analyze the impact of adding K-means top-k clusters (k nearest neighbors) and SIDs as sparse-inputs (corresponding to the ad-item to be ranked) in Table 3 for a new content feature. Note that we use the same hash size for both K-means and SIDs. We now demonstrate the NE gains and RoI for a DPCA encoded feature used as an ad-feature in our ranking architecture.

Method	k-means	SID
Incremental Cost	×1	×0.23
Click NE gain	**0.0108%**	0.0085%
Conversion NE gain	0.0037%	**0.0101%**
RoI	×1	×5.57

Table 3: Incremental data and feature cost and normalized entropy gain relative to the baseline for a DPCA encoded ad feature. Baseline is a large-scale production ads-ranking model without the specific ad-features.

In Table 3, the cost C of features is directly proportional to the feature length used. k-means features with length 50 have cost $46.54kW$, while the SID features have length 3 and corresponding cost is $10.84kW$. This leads to a reduction in incremental feature cost by 76.7%. Hence from Table 3 we see that SID improves RoI by 5.57 times over traditional k-means method. This leads to an NE gain for using DPCA as 1.28X or 28%

Next, we present results for adding SID sparse inputs and SIDs recovered as latent embeddings (SIDE) for the user-history block in two settings: (i) 1:1 encoder (in Table 4) and VQ-fusion (in Table 5) for two different types of user history sequences. In Table 4, we

Method	SID	SIDE
Incremental cost	×1	×0.33
Click NE gain	0.0082%	**0.0185%**
Conversion NE gain	0.0086%	**0.0111%**
RoI	×1	×5.33

Table 4: Normalized entropy gain relative to the baseline for a DPCA encoded feature used as user history. Baseline is the large-scale production ads-ranking model.

demonstrate the NE gains for using SID and unpacking it with SIDE for a DPCA encoded feature. The feature cost for SIDE is 1/3rd of SID since we just use a single n-gram from SID to unpack in SIDE.

We observe that SIDE further improves NE gains over SID and enhances the RoI by 5.33× for the DPCA encoded feature on the user-side. Note that the hash size in SID is set to number of scalar quantization raised to power n-gram length, which is 64^3 = max cardinality of the SID feature, as shown in Section 6.2.1.

In Table 5, the gains from SIDE are further enhanced when by using a VQ-fusion SID feature comprising of 6 individual signals encoded into a single feature as. Here, the RoI improves by 7.40× over SID. This leads to an NE gain when using SIDE to be 2.44x or 144%.

Method	SID	SIDE
Incremental cost	×1	×0.33
Click NE gain	0.0100%	**0.0284%**
Conversion NE gain	0.0067%	**0.0124%**
RoI	×1	×7.40

Table 5: Normalized entropy gain for VQ fusion encoded feature comprising 6 individual signals, used as user history. Baseline = 6 different 1:1 encoded DPCA features used as SID.

6.2.1 Encoder hyper-parameterization. From our ads-ranking experiments we demonstrate examples of varying the SID construction hyper-parameters and its impact in NE gains from the ads-ranking model. In the VQ-Fusion experiment, summarizing 6 user-side features in Table 5 and retaining the number of scalar quantization buckets as 64 and the number of residual quantization layers as 9 and increasing the n-gram size in SID, leads to the NE trend in Table 6. Here, we observe that for SID, as more n-gram IDs are

Prefix n-gram length	SID	SIDE
3	0.029%	0.044%
4	0.020%	0.046%

Table 6: Effect of prefix n-gram length on NE, keeping all other Encoder hyper-parameters fixed.

introduced (by making the prefix n-gram longer from 3 to 4) in the ads-ranking model, the NE gains reduce due to higher noise due to increase in hash collisions based on the embedding table size limitations. The maximum hash size for n-gram length 3 is $64^3 = 262,144$ and for length 4 is $64^4 = 16,777,216$. However, our proposed SIDE technique avoids hash collisions while unpacking the 4th n-gram ID leading to NE increase.

We also assess the effect of increasing the number of scalar quantizer codewords on the ads-ranking NE gains. Using 4 scalar quantizer codewords NE gain for SID and SIDE are 0.018% and 0.035% respectively. Extending this analysis to 64 codewords results in NE gains for SID and SIDE as 0.015% and 0.044%, respectively.

6.3 Discussion on Production Impact

In this section we assess the end-user benefits in terms of ads-score (values for both advertisers and users). Ads-score is the sum of the two entities: 1) ads-value or the value to advertisers calculated by adding paced-bids, 2) quality value which measures value to users, considering, positive interactions (e.g., clicks, likes), negative interactions (e.g., hiding an ad), ad-quality and user engagement. We observe overall gains of 0.17% ad-score with the launch bundle involving our proposed method (DPCA, VQ-fusion and SIDE put together) on the end-users.

In terms of other production metrics: our VQ-Fusion models deployed in online inference have very low latency (< 500 ms) and memory consumption (<= 2.5 GB), recurring training has 0.33% QPS regression. The inference model requires 2, 16GB T1 machines, leading to 20× lesser deployment machine costs when compared to traditional quantization models like k-means leading to improved RoI for our proposed system. The latency and throughput effect on feature logging due to the additional VQ-Fusion model is negligible, since real-time feature availability for ads-ranking systems remains less than 60 minute SLA-bounds.

7 Conclusion and Future Work

In this paper, we have presented a novel approach to incorporating high-dimensional content embeddings into sequence-based recommendation models using vector quantization. Our method called VQ-fusion, which leverages a multi-task VQ-VAE fusion framework and an embedding table-free SID to embedding conversion technique (SIDE), has been shown to reduce storage costs while improving ranking outcomes on large-scale ads recommendation systems. The SIDE method can lead to significant improvements in the efficiency for the sequence learning paradigm in modern ad-recommendation systems.

References

[1] Anima Singh et. al. 2024. Better Generalization with Semantic IDs: A Case Study in Ranking for Recommendations. arXiv:2306.08121

[2] Lijun Yu et. al. 2024. Language Model Beats Diffusion – Tokenizer is Key to Visual Generation. arXiv:2310.05737

[3] Maxim Naumov et. al. 2019. Deep Learning Recommendation Model for Personalization and Recommendation Systems. arXiv:1906.00091 [cs.IR] https://arxiv.org/abs/1906.00091

[4] Yunchao Gong, Svetlana Lazebnik, Albert Gordo, and Florent Perronnin. 2013. Iterative Quantization: A Procrustean Approach to Learning Binary Codes for Large-Scale Image Retrieval. *IEEE Trans. Pattern Anal. Mach. Intell.* 35, 12 (dec 2013), 2916–2929. doi:10.1109/TPAMI.2012.193

[5] Ruijie Hou, Zhaoyang Yang, Yu Ming, Hongyu Lu, Zhuobin Zheng, Yu Chen, Qinsong Zeng, and Ming Chen. 2024. Cross-Domain LifeLong Sequential Modeling for Online Click-Through Rate Prediction. *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining* 1, 1 (2024), 5116–5125.

[6] 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. doi:10.1109/TPAMI.2010.57

[7] Aditya Kusupati, Gantavya Bhatt, Aniket Rege, Matthew Wallingford, Aditya Sinha, Vivek Ramanujan, William Howard-Snyder, Kaifeng Chen, Sham Kakade, Prateek Jain, and Ali Farhadi. 2024. Matryoshka Representation Learning. arXiv:2205.13147 [cs.LG] https://arxiv.org/abs/2205.13147

[8] Doyup Lee, Chiheon Kim, Saehoon Kim, Minsu Cho, and Wook-Shin Han. 2022. Autoregressive Image Generation using Residual Quantization. arXiv:2203.01941

[9] Juho Lee, Yoonho Lee, Jungtaek Kim, Adam R. Kosiorek, Seungjin Choi, and Yee Whye Teh. 2018. Set Transformer. *CoRR* abs/1810.00825 (2018). http://arxiv.org/abs/1810.00825

[10] Shuming Ma, Hongyu Wang, Lingxiao Ma, Lei Wang, Wenhui Wang, Shaohan Huang, Li Dong, Ruiping Wang, Jilong Xue, and Furu Wei. 2024. The Era of 1-bit LLMs: All Large Language Models are in 1.58 Bits. arXiv:2402.17764

[11] Fabian Mentzer, David Minnen, Eirikur Agustsson, and Michael Tschannen. 2023. Finite Scalar Quantization: VQ-VAE Made Simple. arXiv:2309.15505

[12] Qi Pi, Guorui Zhou, Yujing Zhang, Zhe Wang, Lejian Ren, Ying Fan, Xiaoqiang Zhu, and Kun Gai. 2020. Search-based user interest modeling with lifelong sequential behavior data for click-through rate prediction. *Proceedings of the 29th ACM International Conference on Information & Knowledge Management* 1, 1 (2020), 2685–2692.

[13] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H. Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q. Tran, Jonah Samost, Maciej Kula, Ed H. Chi, and Maheswaran Sathiamoorthy. 2023. Recommender Systems with Generative Retrieval. arXiv:2305.05065