Discrete Semantic Tokenization for Deep CTR Prediction

Qijiong Liu*
PolyU, Hong Kong
liu@qijiong.work

Hengchang Hu
National University of Singapore
hengchang.hu@u.nus.edu

Jiahao Wu
PolyU, Hong Kong
jiahao.wu@connect.polyu.hk

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

Min-Yen Kan†
National University of Singapore
kanmy@comp.nus.edu.sg

Xiao-Ming Wu†
PolyU, Hong Kong
xiao-ming.wu@polyu.edu.hk

ABSTRACT

Incorporating item content information into click-through rate (CTR) prediction models remains a challenge, especially with the time and space constraints of industrial scenarios. The content-encoding paradigm, which integrates user and item encoders directly into CTR models, prioritizes space over time. In contrast, the embedding-based paradigm transforms item and user semantics into latent embeddings, subsequently caching them to optimize processing time at the expense of space. In this paper, we introduce a new semantic-token paradigm and propose a discrete semantic tokenization approach, namely UIST, for user and item representation. UIST facilitates swift training and inference while maintaining a conservative memory footprint. Specifically, UIST quantizes dense embedding vectors into discrete tokens with shorter lengths and employs a hierarchical mixture inference module to weigh the contribution of each user–item token pair. Our experimental results on news recommendation showcase the effectiveness and efficiency (about 200-fold space compression) of UIST for CTR prediction.

CCS CONCEPTS

• **Information systems** → **Recommender systems**; **Data mining**.

1 INTRODUCTION

Click-through rate (CTR) prediction models [21] aim to predict the probability of users interacting with items. The real-time demands of online services [8, 9] pose a significant challenge in seamlessly

---

merging such deep CTR models with valuable semantic knowledge, encompassing item content and user history.

Traditional deep CTR models commonly depend on ID-based approaches, incorporating features like item and user IDs, along with other categorical and statistical data [4, 13]. Recognizing the effectiveness of user history [2, 14] in various recommendation scenarios, certain methods [20] explore the incorporation of a *shallow* user encoder into CTR models, a strategy widely adopted in practice. It offers two key advantages over user ID representation in the context of large-scale industrial recommendation settings: 1) improved accuracy facilitated by the sequential features encoded by the user encoder, and 2) reduced memory usage by eliminating the need for a large user ID embedding table.

Meanwhile, the value of item contents, such as texts and images, has been recognized for providing more detailed and nuanced item representations compared to basic item IDs [8, 18]. However, incorporating such item content information into CTR prediction models remains a challenge, especially within the time and space constraints of industrial scenarios. The prevalent use of pretrained models across various domains prompts the integrating of such models as end-to-end item encoders into CTR models, which we term as the **content-encoding paradigm**. Unfortunately, the user encoder requires the behavior sequence as input, necessitating the encoding of each item before fusing the behavior sequence to derive a user representation. This sequential dependency results in unacceptable training and inference inefficiency, impeding its adoption in industrial settings.

To address the efficiency challenge, the **embedding-based paradigm** opts to trade space for time. Some methods [8, 16] utilize pretrained content encoders to convert item semantics into embedding vectors and cache them for subsequent CTR models. Furthermore, some methods [8, 15] explore the pretraining of user encoders, transforming user sequences into cached user embeddings. This approach effectively decouples item and user encoders via offline computing, leading to a substantial acceleration in both training and inference time. However, the use of pretrained models introduces a significant memory bottleneck; simply loading these extracted embeddings for training requires considerable memory.

Recent work involves extracting item content features and condensing them into semantic IDs [6, 11], which efficiently capture the semantic representation of an item while maintaining its hierarchical structure. Building on this concept, we introduce a **semantic-token paradigm** and present **UIST**, a **U**ser–**I**tem **S**emantic **T**okenization approach that converts user sequences and item content into discrete user and item tokens, respectively. In contrast to the embedding-based paradigm, our approach greatly reduces space

Qijiong Liu, Hengchang Hu, Jiahao Wu, Jieming Zhu, Min-Yen Kan, & Xiao-Ming Wu

Figure 1: Our UIST framework for CTR prediction: featuring semantics-based item (d), and user (h) tokenizers, and a hierarchical mixture inference module.

Table 1: Encoding paradigms for CTR prediction comparison. User and item representations correspond to the approaches in Fig. 1. $V = 30,522$, $N = 100M$, and $M = 100M$ represent the vocabulary size of natural language, items, and users, respectively. $K = 4$ denotes the number of tokens used to represent an item or user; $d = 256$ and $D = 768$ denote the embedding dimensionality of the model and pretrained encoders. We use IST to represent single-layered semantic tokenization.

Paradigm	Item Repr.	User Repr.	Efficiency	Examples	Item Memory	User Memory	Instantiation
Content-encoding	(b)	(f)	✕	-	V*d	0	29.81M
ID-based	(a)	(f)	✓	DIN [20]	N*d	0	95.36G
	(a)	(e)	✓✓	DCN [13]	N*d	M*d	190.72G
Embedding-based	(c)	(f)	✓	NP [8]	N*D	0	286.10G
	(c)	(g)	✓✓	PREC [8]	N*D	M*D	572.20G
Semantic-token	(d)	(f)	✓	IST (Ours)	N*K+K*d	0	1.49G
	(d)	(h)	✓✓	UIST (Ours)	N*K+K*d	M*K+K*d	2.98G

consumption while maintaining time efficiency, offering a substantial advantage for large-scale industrial recommender systems.

In Table 1, we present a detailed efficiency and memory consumption comparison of such paradigms used for CTR prediction. Specifically, in the "Instantiation" column, we report the memory usage with vocabulary, user and item numbers set to reasonable settings for industrial applications. Compared to the embedding-based paradigm, our UIST achieves a remarkable ∼200-fold (572.20G/2.98G) space compression, utilizing only four tokens.

Moreover, we devise a hierarchical mixture inference module to enhance the integration of hierarchical item and user tokens. This module dynamically adjusts the significance of various levels of granularity for user–item interactions.

2 USER–ITEM SEMANTIC TOKENIZATION

To balance time (model training and inference) and space (memory usage) efficiency, we introduce a user–item semantic tokenization framework, UIST, following the semantic-token paradigm. UIST comprises of three modules, illustrated in Figure 1: two semantic tokenizers and one hierarchical mixture inference module. The semantic tokenizers initially transform dense and high-dimensional item and user embeddings into discrete tokens. Subsequently, the hyper controller, a hierarchical mixture inference (HMI) module,

evaluates each user–item token pair using the base CTR modules and autonomously learns weights for each pair with supervision of the click labels.

2.1 Discrete Semantic Tokenization

Initially proposed for generative retrieval [6, 11], semantic tokenization has primarily been applied to summarize features on the item side. In this context, we introduce a unified pipeline that extends tokenization to both item and user sides, and we uniformly refer to both item content and user behavior as "sequence". It is important to note that tokenization of items precedes that of users, and during user tokenization, each item in user sequence is initialized with its corresponding representation extracted from the item tokenizer.

Stage 1: Semantic Representation. Traditional approaches often leverage pretrained language models like SentenceBERT to extract text-based content features on the item side by concatenating item attributes into a single sequence. However, this method is not directly applicable to the user side. Instead, we employ an autoencoder network for sequence representation learning.

Specifically, given a sequence **s** with length L, we initially transform it into an embedding sequence $\mathbf{E}^0$ through an embedding layer. Subsequently, a transformer [12] encoder with H layers learns the contextual knowledge of the sequence, expressed as:

$\mathbf{E}^h = \text{ENC}^h(\mathbf{E}^{h-1})$, where $h = 1, 2, \ldots, H$. Next, an additive attention module [1] merges the sequences into a unified representation, defined by: $\mathbf{z} = \text{ATTN}(\mathbf{E}^H)$. Following this, a transformer decoder reconstructs the original sequence. Comprising multiple decoder layers with causal attention, it integrates information from the sequence representation for autoregressive generation: $\mathbf{D}^h = \text{DEC}^h(\mathbf{D}^{h-1}, \mathbf{z})$, where $h = 1, 2, \ldots, H$, and the reconstruction loss is given by:

$$L_{\text{sr}} = -\frac{1}{L} \sum_{i=1}^{L} \log\left(G\left(\mathbf{D}_i^H\right)_{\mathbf{s}[i]} \right), \tag{1}$$

in which G represents the multi-layer perceptron classifier, and $\mathbf{s}[i]$ is the ground truth label for the output embedding $\mathbf{D}_i^H$.

Stage 2: Discrete Tokenization. The above enables random access to sequence embedding $\mathbf{z}$ for arbitrary user histories and item contents. Subsequently, we employ the residual quantization technique, RQ-VAE [19], to discretize the dense sequence representation into concise tokens. RQ-VAE is designed under an encoder-quantizer-decoder framework. From a macro perspective, the encoder maps the sequence embedding $\mathbf{z}$ to a latent vector by $\mathbf{v}$. The residual quantizer learns codes for the latent vector and sums up the code vectors ($\bar{\mathbf{v}}$) to approximate the latent vector $\mathbf{v}$. Finally, the decoder reconstruct the embedding $\bar{\mathbf{z}}$ from $\bar{\mathbf{v}}$.

Specifically, the residual quantizer operates iteratively as follows: **i.** For each layer k (where k ranges from 1 to K), it maps the current vector $\mathbf{v}^k$ to an index i^k by finding the nearest vector in the k-th layer's codebook, minimizing a distance function g (like Euclidean or Manhattan distance) between $\mathbf{v}^k$ and each code vector $\mathbf{C}_j^k$ in the codebook. **ii.** The next vector $\mathbf{v}^{k+1}$ is computed by subtracting the chosen code vector $\mathbf{C}_{i^k}^k$ from $\mathbf{v}^k$, thereby focusing on the residual (the part of the vector not yet quantized) for the next layer. **iii.** The process begins with $\mathbf{v}^1 = \mathbf{v}$, and iteratively refines the representation through the layers.

Formally, the above operations can be denoted as:

$$f^k : \mathbf{v}^k \to i^k, \quad \text{for } k \in 1, 2, \ldots, K, \tag{2}$$

$$\text{where } i^k = \underset{j=1,\ldots,C}{\arg\min}\, g\left(\mathbf{v}^k, \mathbf{C}_j^k\right), \tag{3}$$

$$\mathbf{v}^{k+1} = \mathbf{v}^k - \mathbf{C}_{i^k}^k, \text{ and } \mathbf{v}^1 = \mathbf{v}, \tag{4}$$

where C is the codebook size for each layer.

Therefore, $(i^1, \ldots, i^K)$ is the tokenization result and the original vector $\mathbf{v}$ can be approximated by:

$$\mathbf{v} \approx \bar{\mathbf{v}} = \sum_{k}^{K} \mathbf{C}_{i^k}^k. \tag{5}$$

The overall loss function L_{et} is calculated as:

$$L_{\text{et}} = ||\mathbf{z} - \bar{\mathbf{z}}||^2 + \sum_{k}^{K} \left(|sg[\mathbf{v}^k] - \mathbf{C}_{i^k}^k|_2^2 + \beta|\mathbf{v}^k - sg[\mathbf{C}_{i^k}^k]|_2^2 \right), \tag{6}$$

where sg represents the stop gradient mechanism, and where the first term is the embedding reconstruction loss and the second term is the quantization loss. β is the commitment cost that controls the influence of the vector movement.

2.2 Hierarchical Mixture Inference (HMI)

Following the semantic tokenization process, we obtain item and user tokens for each item $\mathbf{t}$ and user $\mathbf{u}$, represented as $(\mathbf{t}^1, \ldots, \mathbf{t}^K)$ and $(\mathbf{u}^1, \ldots, \mathbf{u}^K)$, respectively. For simplicity, we use the same number of layers K during tokenization. Due to the nature of the residual quantization, these tokens are organized hierarchically, with the lower indices carrying primary component information. To effectively utilize user–item pairs at different levels, our hierarchical mixture inference module analyzes the contribution of each pair in the click-through rate prediction task.

Specifically, we transform one-hot tokens into dense embeddings through the user and item token embedding layers, denoted as $\mathbf{E}_{\mathbf{u}} \in \mathbb{R}^{K \times d}$ and $\mathbf{E}_{\mathbf{t}} \in \mathbb{R}^{K \times d}$. We then construct coarse-to-fine item and user embeddings based on the hierarchical tokens, defined as:

$$\bar{\mathbf{e}}_{\mathbf{x}}^i = \sum_{j}^{i} \mathbf{e}_{\mathbf{x}}^k, \quad i = 1, 2, \ldots, K, \text{ and } \mathbf{x} \in \{\mathbf{u}, \mathbf{t}\}, \tag{7}$$

where $\mathbf{e}_{\mathbf{x}}^i$ is the i-th vector in $\mathbf{E}_{\mathbf{x}}^i$. For each user–item pair $(\bar{\mathbf{e}}_{\mathbf{u}}^i, \bar{\mathbf{e}}_{\mathbf{t}}^j)$, we use a deep CTR model M, such as DCN [13], to predict click scores. Finally, a linear layer is employed to automatically weigh these scores. This yields the final click probability, formulated as:

$$p = \text{Linear}\left(M\left(\bar{\mathbf{e}}_{\mathbf{u}}^i, \bar{\mathbf{e}}_{\mathbf{t}}^j\right) \right). \tag{8}$$

As is standard, we use binary cross-entropy loss to train the recommendation task, calculated by:

$$L_{\text{rec}} = -l \times \log(p) + (1 - l) \log(1 - p), \tag{9}$$

where $l \in \{0, 1\}$ is the ground truth click label.

Furthermore, we also develop an **item-only semantic tokenization (IST)**, tailored for the item side. By virtue of the shorter item tokens compared to the original item content, IST enhances both time and space efficiency in comparison to the content-encoding paradigm. Nonetheless, when contrasted with UIST, IST is a slower alternative due to its retention of the user encoder.

3 EXPERIMENTS

We conduct offline experiments on a real-world news recommendation dataset, MIND [17], containing over 65K items and 94K users. We evaluate the effectiveness of our proposed UIST against three modern deep CTR models: DCN [13], DeepFM [4], and FinalMLP [10]. We follow common practice [8] to evaluate the recommendation effectiveness with AUC [3] and nDCG [5]. We also measure the inference time (latency) of each baseline for a single sample. During training, we employ the Adam optimizer [7] with a learning rate of 1e-3 for all paradigms. We set the number of transformer layers to 6 for 1) the item encoder (Fig. 1b), 2) user encoder (Fig. 1f), and 3) encoder–decoder used in semantic tokenization. For fair comparison, $\mathbf{z}$ in Section 2.1 serves as the pretrained embeddings for the embedding-based paradigms. During semantic tokenization, we set the residual depth to 4 and the codebook size to 64. We will release the code and data for reproducible research[1].

Table 2 compares the various paradigms across three deep CTR models, averaged over five independent runs. We make three key

---

[1] https://github.com/Jyonn/SemanticTokenizer

Table 2: Performance comparison among different paradigms. We use red and green background to represent inefficient and efficient memory usage or inference latency, respectively.

Paradigm	Repr.	Memory		DCN			DeepFM			FinalMLP		
		Item	User	AUC	NDCG@5	Latency	AUC	NDCG@5	Latency	AUC	NDCG@5	Latency
Content-based	(b)/(f)	30K × 256	0	-	-	66ms	-	-	65ms	-	-	68ms
ID-based	(a)/(f)	65K × 256	0	0.5762	0.2808	12ms	0.5665	0.2720	12ms	0.5777	0.2834	12ms
	(a)/(e)	65K × 256	94K × 256	0.5886	0.2862	3ms	0.5749	0.2781	3ms	0.5840	0.2833	3ms
Embedding-based	(c)/(f)	65K × 768	0	0.6301	0.3220	13ms	0.6195	0.3148	13ms	0.6259	0.3193	13ms
	(c)/(g)	65K × 768	94K × 768	0.6324	0.3247	3ms	0.6219	0.3161	3ms	0.6237	0.3174	3ms
Semantic-based	(d)/(f) IST	65K × 4	0	0.6297	0.3224	33ms	0.6183	0.3132	34ms	0.6241	0.3180	34ms
	(d)/(h) UIST	65K × 4	94K × 4	0.6122	0.3085	3ms	0.6077	0.3028	3ms	0.6093	0.3050	3ms

Table 3: Various aggregation mechanisms for user–item tokens. "Add" indicates the addition of item and user token embeddings to create a unified item and user representation. "Layer" signifies that only tokens from the same layer are input into the base CTR models. HMI represents our hierarchical mixture inference module.

	DCN		DeepFM	
	AUC	NDCG@5	AUC	NDCG@5
Add	0.5795	0.2836	0.5723	0.2774
Layer	0.6044	0.2963	0.5980	0.2929
HMI	0.6122	0.3085	0.6077	0.3028

observations. **i.** The content-based paradigm yields latencies exceeding 60ms, intolerable for industrial scenarios. **ii.** The single-layered ID-based and embedding-based approaches – i.e., (a)/(f) and (c)/(f) pairings – exhibit similar latency, as both include a user encoder to model user behavior; however, the performance of the embedding-based approaches are superior due to the use of the content-based item representation. On the other hand, the single-layered semantic-based IST approach is slightly slower because the item tokenization leads to a longer user sequence. **iii.** Our proposed IST and UIST achieve substantial memory compression (approximately 200 times) compared to other paradigms, while maintaining up to 99% accuracy (for IST) and 98% accuracy (for UIST) when compared to the state-of-the-art embedding-based paradigm. These findings, observed from diverse base models, validate the effectiveness and efficiency of our semantic-based approach.

Table 3 examines various aggregation mechanisms for dual tokens, revealing that the simple addition and layer-wise approaches are inferior to our proposed HMI module.

4 CONCLUSION

We introduce a user–item semantic tokenization method, providing a streamlined approach to integrating item content into deep CTR models. Through our experimentation, we demonstrate the significant potential of semantic tokenization, initially proposed for generative retrieval, in boosting recommendation efficiency, particularly in industrial scenarios. Upon reflection, this method also offers new perspectives for applications such as dataset compression. We encourage researchers to further explore its potential.

REFERENCES

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv* (2014).

[2] Qiwei Chen, Huan Zhao, Wei Li, Pipei Huang, and Wenwu Ou. 2019. Behavior sequence transformer for e-commerce recommendation in alibaba. In *1st DLP4Rec*. 1–4.

[3] Tom Fawcett. 2006. An introduction to ROC analysis. *PRL* (2006).

[4] Huifeng Guo, Ruiming Tang, Yunming Ye, Zhenguo Li, and Xiuqiang He. 2017. DeepFM: a factorization-machine based neural network for CTR prediction. *arXiv* (2017).

[5] Kalervo Järvelin and Jaana Kekäläinen. 2002. Cumulated gain-based evaluation of IR techniques. *TOIS* 20, 4 (2002).

[6] Bowen Jin, Hansi Zeng, Guoyin Wang, Xiusi Chen, Tianxin Wei, Ruirui Li, Zhengyang Wang, Zheng Li, Yang Li, Hanqing Lu, et al. 2023. Language Models As Semantic Indexers. *arXiv* (2023).

[7] Diederik P Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. *ICLR* (2015).

[8] Qijiong Liu, Jieming Zhu, Quanyu Dai, and Xiaoming Wu. 2022. Boosting Deep CTR Prediction with a Plug-and-Play Pre-trainer for News Recommendation. In *COLING*. International Committee on Computational Linguistics.

[9] Qijiong Liu, Jieming Zhu, Quanyu Dai, and Xiao-Ming Wu. 2023. Only Encode Once: Making Content-based News Recommender Greener.

[10] Kelong Mao, Jieming Zhu, Liangcai Su, Guohao Cai, Yuru Li, and Zhenhua Dong. 2023. FinalMLP: An Enhanced Two-Stream MLP Model for CTR Prediction. In *AAAI*.

[11] Shashank Rajput, Nikhil Mehta, Anima Singh, Raghunandan H Keshavan, Trung Vu, Lukasz Heldt, Lichan Hong, Yi Tay, Vinh Q Tran, Jonah Samost, et al. 2023. Recommender Systems with Generative Retrieval. *arXiv* (2023).

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *NIPS* 30 (2017).

[13] Ruoxi Wang, Bin Fu, Gang Fu, and Mingliang Wang. 2017. Deep & Cross Network for Ad Click Predictions. In *ADKDD* (Halifax, NS, Canada) *(ADKDD'17)*. ACM, New York, NY, USA, Article 12, 7 pages.

[14] Chuhan Wu, Fangzhao Wu, Suyu Ge, Tao Qi, Yongfeng Huang, and Xing Xie. 2019. Neural News Recommendation with Multi-head Self-attention. In *EMNLP-IJCNLP*.

[15] Chuhan Wu, Fangzhao Wu, Tao Qi, and Yongfeng Huang. 2022. UserBERT: Pre-training User Model with Contrastive Self-supervision. In *SIGIR*. 2087–2092.

[16] Chuhan Wu, Fangzhao Wu, Yang Yu, Tao Qi, Yongfeng Huang, and Qi Liu. 2021. NewsBERT: Distilling pre-trained language model for intelligent news application. *arXiv* (2021).

[17] 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 *ACL*. 3597–3606.

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

[19] Neil Zeghidour, Alejandro Luebs, Ahmed Omran, Jan Skoglund, and Marco Tagliasacchi. 2021. Soundstream: An end-to-end neural audio codec. *TASLP* 30 (2021).

[20] Guorui Zhou, Xiaoqiang Zhu, Chenru Song, Ying Fan, Han Zhu, Xiao Ma, Yanghui Yan, Junqi Jin, Han Li, and Kun Gai. 2018. Deep interest network for click-through rate prediction. In *SIGKDD*. 1059–1068.

[21] Jieming Zhu, Jinyang Liu, Shuai Yang, Qi Zhang, and Xiuqiang He. 2021. Open Benchmarking for Click-Through Rate Prediction. In *CIKM*. 2759–2769.