

KuaFu: Compressing Long User Behavior into Understanding at Billion Scale

Jiahao Hui*, Lin Zhu*✉, Yishen Hu, Jingdong Shu, Zetai Jiang, Xining Ran, Ben Tan, Yeshou Cai, Gong Chen, Haijie Gu, Jie Jiang

Tencent Inc., Shenzhen, China

{jiahahui, techzhu, eastonhu, breezeshu, zetaijiang, xiningran, bentan, showcai, natchen, jerrickgu, zeus}@tencent.com

Conversational agents, generative recommenders, and personalized advertising systems all rest on the same capability: understanding each user from raw behavior logs. The prevailing industrial practice is *task-specific*: for each profiling task, a task-relevant subsequence is first extracted from the full behavior history, and a dedicated understanding model is then trained on it. In production, however, this pragmatic route runs into two bottlenecks. First, even after filtering, a single-task sequence remains extremely long: content-interest summarization reads several hundred behavior items per user on average and over a thousand at the tail, which already amounts to tens of thousands of tokens once serialized as prompt text. Second, profiles must be refreshed routinely. We recompute profiles for a billion users every week, requiring roughly 100K QPM in aggregate; under a fixed GPU budget, this sets a hard lower bound on the per-GPU throughput that must be sustained. Compression therefore becomes mandatory. Yet truncation or coarse-grained compression can silently distort the resulting profile and introduce four types of hallucination—fabrication, omission, date misattribution, and broken logic—which, absent any way to evaluate the compressed representation itself, surface only as diffuse degradation in downstream business metrics. We present KuaFu, a unified behavior-compression layer whose minimal unit is a single behavior *item*. A two-axis projector compresses each item into 2–4 tokens of width 128–256—about 10× compression along the token axis and 20× along the width axis, cutting the per-item cache from 10 KB to 0.5 KB—combined with fidelity-oriented four-stage training and a layered intermediate evaluation protocol. Across four production profiling tasks, the compressed model matches or exceeds the uncompressed single-task production models on all five headline metrics, increases per-GPU throughput by 37%–350%, and saves 190 GPUs. On public benchmarks, KuaFu is almost uniformly better than existing compressors at the same compression ratio (up to +17.7 EM on out-of-domain MRQA); on RecBench, a 4B model surpasses its 8B counterpart by 1.90 points. KuaFu has been fully deployed on the Tencent advertising and recommendation platform for ten months, with a 1.37% lift in overall GMV.

Keywords: user modeling, context compression, large language models, generative recommendation, industrial deployment

* Equal contribution. ✉ Corresponding author.

1 Introduction

1.1 Background

The growth of digital platforms has made user behavior one of the largest data modalities in production systems. Logs carry both explicit interactions – impressions, clicks, conversions – and implicit signals such as dwell time, scroll depth and session length. At industrial scale the user base reaches a billion, and a single user can accumulate tens of thousands to hundreds of thousands of behavior events over a long horizon. The prevailing industrial practice is *task-wise*: every profiling task (interest summarization, occupation and industry, life stage, ...) first filters a task-relevant subsequence out of the full history and then trains a dedicated understanding model for it. The division of labour is clean and each task can be iterated and shipped independently, which makes this the most pragmatic choice today. It nevertheless faces two production facts – sequences stay extremely long even after filtering, and profiles must be refreshed routinely – both quantified in §1.2.

Large language models change the picture: rather than extrapolating from statistical co-occurrence, an LLM can read the intent behind a behavior sequence. Compression theory supplies the bridge:

the number of bits a model needs to compress a dataset directly reflects how well it understands and predicts that data [3] – compression and understanding are two views of the same objective. Compressing a long behavior sequence into a compact, semantically rich representation before handing it to an LLM is therefore the natural way to reconcile accuracy with efficiency; and a compact, faithful, reusable behavior representation is also what would make the end state – one foundation model that reads the entire history and answers any profiling question on demand – possible at all.

1.2 Three Coupled Problems

Landing this idea in production exposes three coupled problems: a cost wall forces us to compress (P1), compression itself induces hallucination (P2), and the absence of evaluation makes those hallucinations impossible to measure or localize (P3). Figure 1 shows the system that answers them.

P1: the cost wall of long sequences × routine refresh. Serializing history into prompt text inevitably overflows the context window as the sequence grows, and the *filtered*, task-specific sequence is already the reduced case: its tail still runs past a thousand

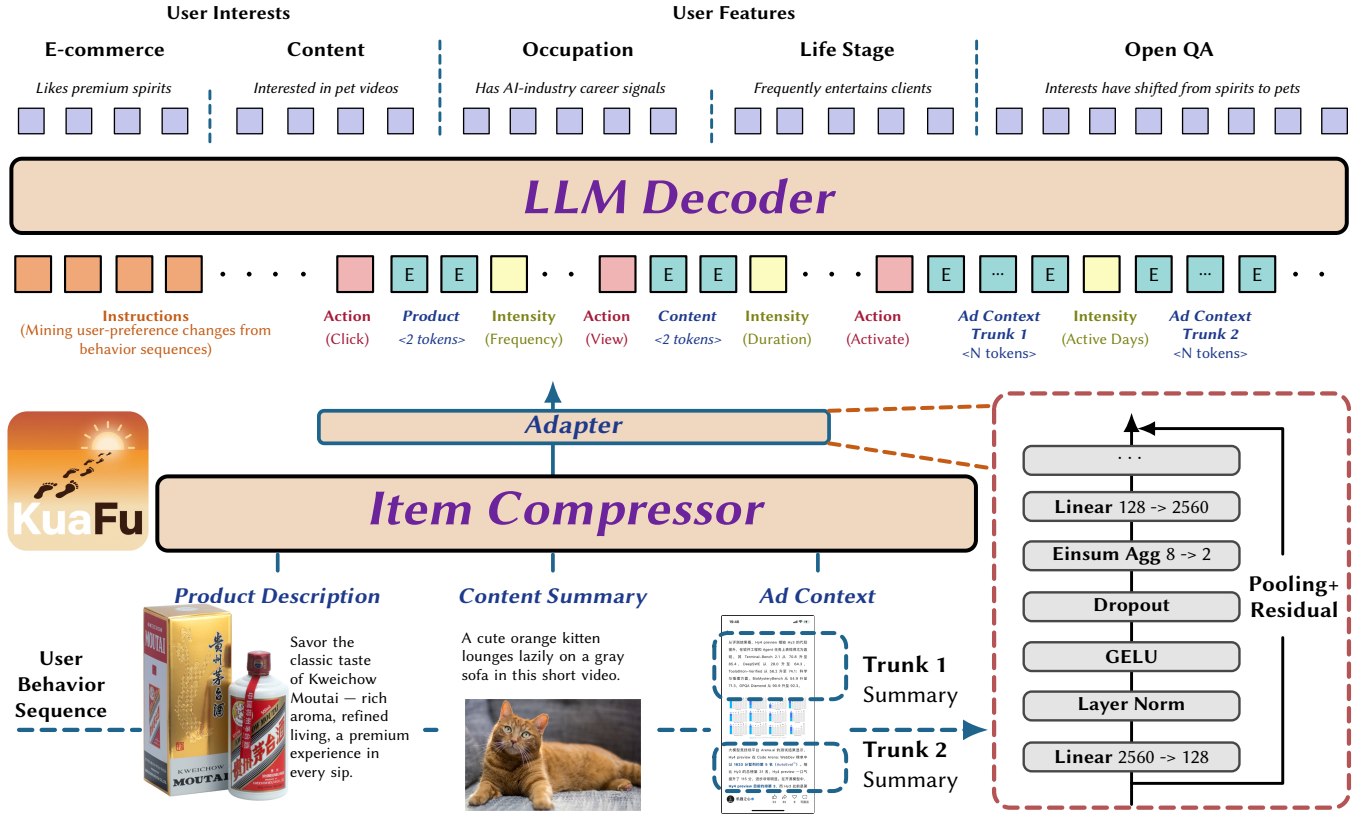


Figure 1: KuaFu in production. Each behavior item – a product page, a piece of content, or a long ad context split into trunks – is encoded once by the item compressor and reduced by the two-axis adapter (structure detailed on the right) to a handful of memory tokens. These tokens are interleaved with action and intensity tokens and with task instructions. A frozen LLM decoder then reads the full strip and emits interest and feature slots. Because the compressed result depends only on the item, it is cached offline and reused across users, tasks, and requests.

items (Table 2), so the untruncated history is out of reach by a wide margin. More binding is the production-scale constraint: profiles are recomputed for a billion users on a weekly cadence, roughly 100K QPM in aggregate, which under a fixed GPU budget becomes a hard lower bound on per-GPU throughput – a bound that direct inference over tens of thousands of raw tokens does not come close to meeting. To amortize its cost, a compressed representation must be cached offline and reused online, which makes the byte size of a single representation critical: storing each item as two 2560-dim vectors (about 10 KB in fp16) is infeasible at full scale, whereas 2×128 (about 0.5 KB) makes the cache viable.

P2: compression-induced hallucination. When an LLM is forced to reason over compressed or truncated behavior descriptions, four hallucination patterns specific to user understanding recur: *fabrication* (inventing interests that never occurred), *omission* (dropping preference signals lost in compression), *date misattribution* (assigning events to the wrong time bucket) and *broken logic* (failing to reason over the residual evidence). Prior work reports that prompt compression almost always raises the hallucination rate, with information loss as the dominant cause [35], matching

our production observations. These patterns concentrate in open-ended generation and, absent ground-truth profiles, are hard to detect.

P3: an unmeasured intermediate stage. Behavior compression and understanding, as an intermediate step, lacks standardized datasets and metrics. Behavior-sequence compression in recommendation is judged end-to-end (CTR, GMV), while generic compression is agnostic to profiling semantics; neither asks whether the compressed representation faithfully preserves user preference. Errors introduced during compression therefore surface only as indirect downstream degradation: one online A/B takes a week, and whether the root cause is the compression artifact, hallucination or decoder capacity cannot be told apart from end-of-month business metrics.

1.3 Contributions

(a) **Architecture: item-level, cacheable, two-axis compression.** Each behavior event is encoded independently, so the embedding is cacheable, reusable across users and tasks, and appendable as new behavior arrives; the cache grows with the item inventory rather than with $\#users \times \#tasks$. To our knowledge this is the first

time behavior text is encoded at item granularity in a billion-user advertising recommendation production environment. A two-axis projector then compresses further along both the token and the width axis (quantified in the abstract).

(b) Training: a fidelity-oriented four-stage recipe. Layered-difficulty data spanning 42 task types; four stages – reconstruction, understanding, co-training of compression and generation, and hallucination-aware RL – inherited progressively from easy to hard; a sequence-length curriculum in the reconstruction stage to secure convergence; and an RL objective with one penalty head per hallucination type plus a completeness bonus, which suppresses hallucination without letting the model degenerate into refusing to answer.

(c) Evaluation and deployment: a layered intermediate protocol and large-scale production validation. From simple classification and complex multi-label attribution to open-domain QA and multi-task transfer, we score the compressed representation directly, shortening the “compression error → downstream degradation” feedback loop from months to days. Results and deployment outcomes are summarized in the abstract and detailed in §4.

2 Related Work

Generic context compression. Hard compression (Selective Context [15], LLMingua [11] and its successors) drops tokens by informativeness and still emits natural language. Soft compression writes the context into a small set of memory slots: GIST [25] compresses short instructions, AutoCompressor [2] recursively produces summary vectors, ICAE [6] writes memory with a LoRA encoder while a frozen decoder generates from it, 500xCompressor [18] pushes the ratio higher, REFRAG [20] uses reinforcement learning to decide which chunks to expand into full tokens for RAG, and the recent SAC [23] and EPL [36] further advance ratio and fidelity; the latter two are our main points of comparison. These methods establish that question answering survives compression, but their object is continuous documents or retrieved passages, and they do not explicitly model timestamped, cross-modal behavior events. Their objectives target reconstruction or downstream QA and do not directly constrain fabrication, missed detection or date misplacement [35].

Item and behavior tokenization in recommendation. Generative recommendation compresses items into discrete or continuous tokens: TIGER [28] derives hierarchical semantic IDs with RQ-VAE, LAMIA [21] learns a set of parallel multi-facet embeddings, Token Factory [1] fuses heterogeneous interaction features, PatchRec [19] groups distant history into session patches, and the Q-Former family [14] (including U²QT [9]) uses a fixed number of queries as the information bottleneck. These representations support next-item prediction, yet evaluation is almost exclusively HR, NDCG and CTR: whether preference and temporal order survive compression cannot be separated out of an end-to-end score, and the same representation transfers poorly to profiling or conversational recommendation, where the behavioral evidence must be explained. GISTBench [5] and UserSumBench [30] begin to assess whether interests are verifiable, but they target uncompressed text

and do not place compression ratio, fidelity and profile quality on one trade-off plane.

Long sequences and LLM efficiency in industrial systems. Industry answers the same question along three directions. *Efficient modeling of long behavior sequences:* ByteDance models 10K-scale ID sequences end-to-end at billion scale on a short-video platform [8], and a collaboration between the Institute of Software, Chinese Academy of Sciences and Meituan brings sparse attention to long-term-behavior CTR prediction [13]; both solve the length problem inside the ranking model and target prediction accuracy, whereas KuaFu performs semantic compression on the LLM input side and targets an interpretable, reusable profile representation. *Efficiency of LLM deployment:* Meituan distills a reasoning LLM into BERT in two stages [32], Alibaba International distills multi-perspective chain-of-thought into latent reasoning [26], Baidu prunes the KV cache online for long contexts [16], and Alibaba together with City University of Hong Kong accelerates decoding for generative recommendation [31] – all trading a large model for a smaller one to save cost, whereas KuaFu retains the understanding capacity of the LLM and compresses its input instead, producing an asset that is cached offline and reused across requests and tasks rather than re-computed per request. *PLM/LLM-driven generative recommendation in production:* Google DeepMind and YouTube adapt a pre-trained language model to industrial generative recommendation [10], and Zhejiang University with Taotian learns long semantic IDs [17]; these understand item-side semantics, and KuaFu is complementary in that it understands the user-side long behavior sequence.

3 Method

3.1 Problem Setting and the Item-Level Paradigm

Let x_i denote the i -th behavior item of a user (one click, one view, ...). The n items of the same user form a chronologically ordered sequence $X = (x_1, \dots, x_n)$. Given a task prompt q , the model produces an answer y . We take a single item as the minimal compression unit and build a multi-item compression paradigm: each x_i is encoded independently into an item embedding z_i , the z_i are concatenated in the temporal order of X , and the result is fed to the decoder together with q . Unlike encoding the whole history into one shared vector, this paradigm satisfies $z_i = f(x_i)$ – the compressed result depends on that item alone. Three consequences follow: (1) the same item’s z_i can be reused inside the sequences of different users; (2) a new behavior x_{n+1} only requires computing z_{n+1} and appending it, with no need to recompress X ; (3) the offline cache grows with the item inventory rather than with the number of users. These three points are the entire source of the system economics reported in §4.9.

3.2 Compressor, Projector and Decoder

Compressor. The compressor E is a causal language model with LoRA and hidden width d . We append m memory tokens to x_i ; these tokens are not part of the ordinary vocabulary and their embeddings come from a separate table, while the words of x_i keep the word embeddings of E . One forward pass over the concatenated sequence yields the last-layer hidden states at the m memory positions,

$$h_i = \text{Enc}_E(x_i \oplus [\text{MEM}]_{1:m}) \in \mathbb{R}^{m \times d}, \quad (1)$$

which depends on x_i only. Each item is encoded on its own, so one embedding corresponds to one cacheable, appendable item.

Projector. h_i still consists of m vectors of width d . The projector compresses along two axes: the token axis reduces the count from m to k (e.g. $8 \rightarrow 2$), and the width axis reduces from d to d' (e.g. $2560 \rightarrow 128$). A Down network reduces the dimension of every vector to d' , a combination matrix $C \in \mathbb{R}^{k \times m}$ fuses the m vectors into k inside that low-dimensional space, and an Up network maps back to width d and is added to a pooled residual,

$$z_i = \text{Up}(C \cdot \text{Down}(h_i)) + \text{Pool}(h_i). \quad (2)$$

The former avoids collapsing raw text into an over-short representation in one step; the latter reduces storage volume. Writing L_i for the raw token count of x_i , the token-axis ratio is L_i/k ; online inference caches the d' -dimensional rather than the d -dimensional representation, giving a width-axis ratio of d/d' .

Decoder and positional encoding. The decoder D does not extend its vocabulary. Every compressed item in the prompt is replaced by its item embedding while the remaining text keeps the word embeddings of D , and generation runs over the concatenated sequence. One-dimensional RoPE is used both inside a single item (including its memory tokens) and over the whole input; the order among items is carried by the concatenation order.

3.3 Four-Stage Training

All four stages serve one objective – factual fidelity of the compressed representation – and are inherited progressively from easy to hard. The three supervised stages share the same forward process (encode each item independently, concatenate in temporal order, feed to the decoder) and differ only in the objective and the trainable set; all of them run a curriculum over the number of items per sample, growing from short to long sequences. Table 1 summarizes which modules are updated.

Compressor pre-training reconstructs the original text from the compressed input: x_i is compressed into z_i and restored by D . Only the LoRA on E and the memory embeddings are updated; the projector is off and D is frozen and used solely to compute the restoration loss. Restoration happens over the whole concatenated compressed sequence, so the decoder must restore the original text item by item from the coexisting z_i . Direct restoration over long sequences does not converge, hence the five-bucket curriculum over item count (1, 1–3, 1–200, 1–500, 1–1200).

Compressor post-training answers questions over the compressed input: the projector is attached and trained together with the LoRA and memory embeddings, while D stays frozen. The decoder must read what it needs from each coexisting z_i and merge them into an understanding of the whole history.

Co-training of compression and generation unfreezes D and jointly updates the LoRA on E , the memory embeddings, the projector and D . In post-training D is frozen, so E can only write for a decoder that never changes; training both ends lets D learn to read compressed item embeddings while E and the projector adjust z_i to what question answering requires. Dimension reduction is annealed with a cosine schedule on the residual, which keeps the compression bottleneck from collapsing under the pressure of the generation loss.

Table 1: Trainable set per stage. C/P/D denote the compressor, the projector and the decoder; post-training covers both the simple and the complex split. The full recipe with sample and token counts is in Appendix A.

Stage	Objective	C	P	D
Pre-training	reconstruct	update	—	freeze
Post-training	compressed QA	update	update	freeze
Co-training	fit both ends	update	update	update
Hallu.-aware RL	de-hallucinate	freeze	freeze	update

3.4 Hallucination-Aware Reinforcement Learning

This stage suppresses hallucination directly on top of the compressed representation: E and the projector are frozen, item embeddings are injected from the offline cache, and only D is updated. For each prompt a group of answers is sampled and judged item by item against the original text by a generative reward model. The reward has two parts: the four error types – fabrication, missed detection, date misattribution, broken logic – are each judged present/absent and penalized with weight w_c increasing in the harm done to the user (fabrication heaviest, missed detection lightest), and completeness is mapped from {complete, partial, missing} to a bonus $b(\cdot)$ that prevents the model from dodging penalties by answering less:

$$r(y) = b(\text{comp}(y)) - \sum_{c=1}^4 w_c \cdot \mathbb{I}[y \text{ hits error } c]. \quad (3)$$

The G answers in a group share one prompt and the advantage is normalized by the group mean. The policy is updated with DAPO [34]: token-level importance ratio, asymmetric clipping, no KL term, and truncated answers excluded from the loss, which mitigates the length collapse of GRPO [29].

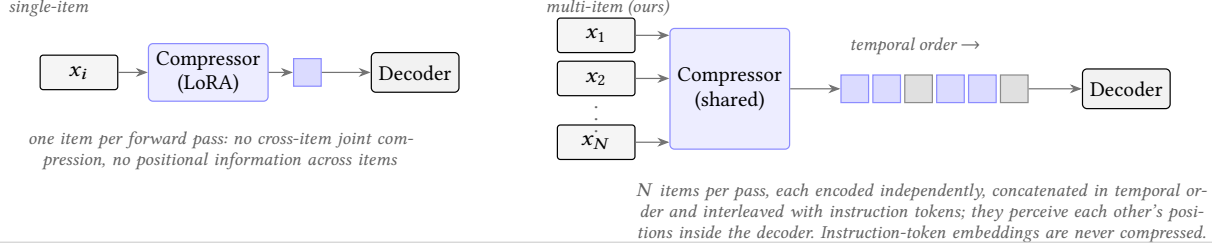
4 Experiments

4.1 Setup

The model follows an LLM-encoder plus LLM-decoder architecture; both the compressor and the downstream decoder are based on Qwen3-4B [33], the comparison against external compression baselines switches the backbone to Llama-3.2-1B [7] to match the control group, and the design has been extended to the multimodal backbone Qwen3-VL-4B-Instruct [27] with a frozen ViT encoder.

Training data come from two sources. Advertising behavior sequences comprise raw user advertising behavior and the interest-summary sequences derived from those long sequences. General-purpose data comprise the Chinese long-text corpus MNBVC [24] (news, fiction, textbooks, legal and medical documents) to shore up generic reconstruction ability, plus general-domain NLP and RAG question answering covering entity recognition, intent recognition, continuation and question answering. After a dedicated judge filters duplicates and low-quality samples, everything is unified into compression–restoration and compression–understanding formats, blended by difficulty and injected stage by stage. Reconstruction samples compress every item (or chunk) in a sample and restore them one by one in temporal order, bucketed by item

(a) Single-item vs. multi-item compression



(b) Two-axis projector

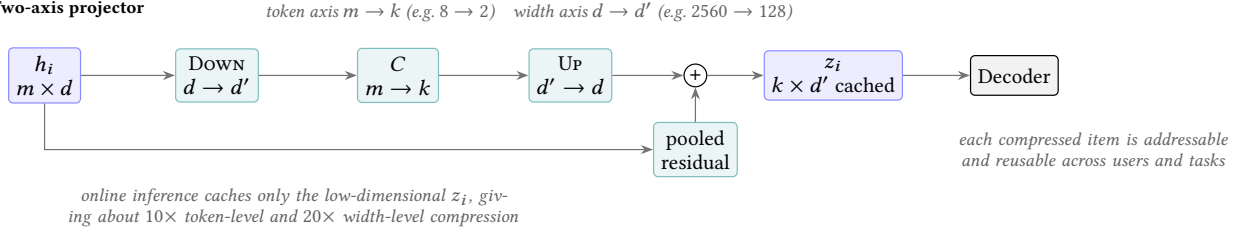


Figure 2: Evolution of the KuaFu training paradigm (a) and the two-axis projector (b). Encoding one item per forward pass learns neither cross-item joint compression nor positional information; encoding N items per pass – each independently, then concatenated in temporal order and interleaved with instruction tokens – lets them perceive each other’s positions inside the decoder. The projector then reduces the token count from m to k and the width from d to d' , adds a pooled residual, and only the low-dimensional z_i is cached online.

count into 1, 1–3, 1–200, 1–500, 1–1200 to form the curriculum; understanding samples are built mainly from advertising behavior sequences (construction details in Appendix A). The final training set spans **42 task types**: 1 sequence-item/paragraph reconstruction, 8 simple behavior-sequence understanding, 7 generic NLP understanding, 22 complex long-sequence behavior understanding and 4 downstream user-understanding tasks; per-stage sample and token counts are given in Appendix A (2.13M/8.06B, 150K/0.80B, 350K/2.30B, 350K/2.30B and 5K/41M respectively).

Baselines fall into three groups. The online baseline is an uncompressed LLM trained with SFT and DPO on each downstream task. The generic behavior compressors are ICAE, LLMLingua-2, 500xCompressor, EPL and SAC. The recommendation group uses LLMs as recommenders, built on GLM, Qwen, Llama and Mistral.

Evaluation is offline and online. Offline we report precision, recall and F1 or accuracy on downstream tasks, plus loss and QPM; online we report GMV and QPM gains. The in-house validation set is split from our own data and focuses on the four downstream tasks. Public data are the six in-domain MRQA [4] subsets (SQuAD, NewsQA, TriviaQA, SearchQA, HotpotQA and NaturalQuestions) and the six out-of-domain ones (BioASQ, DROP, DuoRC, RACE, Relation Extraction and TextbookQA). The recommendation side uses RecBench [22] on MIND, MicroLens, Goodreads, CDs and H&M.

Optimization and serving. Learning rate is about $1e-5$ – $1e-4$ over 3–5 epochs, reinforcement learning uses DAPO, and serving runs on vLLM [12] with an embedding server that caps concurrency by semaphore to keep GPU memory stable.

4.2 Production Profiling Tasks

KuaFu is deployed in the profiling production pipeline of the Tencent advertising and recommendation platform, mining the raw behavior sequences of a billion users every week and emitting structured labels that are written straight into recommendation features. At that cadence throughput is the binding constraint, which is why behavior sequences are compressed at item level. Labels split into two business families. *Interest summarization* targets content interest and e-commerce interest, inducing verifiable interest points from hundreds to thousands of cross-modal behaviors; sequences are longer and the ratio is set to $10\times$. *User profiling* targets occupation/industry and life stage, deciding stable attributes from relatively short behavior; the ratio is $15\times$. Both families share one item-level compressed representation and differ only in the downstream head and the metric.

Table 2 compares the single-task online model (uncompressed) with the multi-task KuaFu that consumes the compressed representation. Interest summarization reports F1; occupation/industry reports industry/occupation accuracy; life stage reports exact-match accuracy. On all five headline metrics the compressed KuaFu is no worse than the online model, with the largest gain on life stage (+2.0). Besides fitting the weekly full refresh of a billion users into the QPM budget, compression also purifies the long sequence to a degree – weakly related, repeated and noisy behaviors are compressed away, so stable interest and attribute signals stand out.

4.3 In-House Benchmark against External Compressors

To test how generic compressors behave on industrial data we evaluate on the validation sets of the four downstream tasks covered

Table 2: Downstream user-understanding tasks in production. Δ is the change in percentage points of KuaFu relative to the single-task online model.

Task	Family	items	ratio	online	KuaFu	Δ
Content interest	summary	400/1200	10×	85.9	86.5	+0.6
E-com. interest	summary	150/200	10×	95.7	96.1	+0.4
Industry	profile	30/100	15×	81.0	82.1	+1.1
Occupation	profile	30/100	15×	86.7	87.6	+0.9
Life stage	profile	16/100	15×	76.0	78.0	+2.0

Table 3: In-house benchmark (percentage scale). All soft baselines are retrained under the aligned protocol described in the text. †: training-free hard compression at 5×, listed as an external reference only.

Method	E-com.	Occup.	Content	Life	Overall	Δ
LLMLingua-2†	49.7	31.1	22.9	22.7	31.6	-31.4
ICAE	50.5	43.1	30.9	21.7	36.6	-26.4
SAC	48.3	50.7	34.3	23.7	39.2	-23.8
500xCompressor	55.7	46.1	32.0	23.9	39.4	-23.6
EPL	53.9	56.3	34.7	24.5	42.4	-20.6
KuaFu	71.7	64.5	64.7	51.1	63.0	—

by KuaFu’s training data. The four soft-compression baselines are not off-the-shelf weights or zero-shot calls but are retrained under conditions aligned with KuaFu: the same user behavior sequences, the same train/validation split (the validation set enters no baseline’s training or model selection), the same supervision from the four downstream tasks, the same input reorganization, a common Llama-3.2-1B backbone and a common compression ratio of 15×; each baseline follows the training objective and procedure of its original paper until convergence, its compressor shares KuaFu’s LoRA parameterization (rank 128, scaling factor 2), and the target language model stays frozen. LLMLingua-2 is a training-free hard-compression method invoked zero-shot from official weights at a ratio of 5×, which does not align with the soft-compression setting; it is listed separately as an external reference and excluded from the aligned protocol. All scores are produced by an LLM judge.

KuaFu leads clearly with 63.0, and its margin over the strongest baseline of each task is 16.0, 8.2, 30.0 and 26.6 points. No baseline exceeds 60 on any of the four tasks, while KuaFu’s weakest task (life stage, 51.1) still tops the best overall baseline average (42.4). With training data, split, supervision, input reorganization, backbone, trainable scope, parameterization and compression ratio all aligned, the lead is attributable to the compression structure itself – taking a behavior item as the compression unit and mapping each item to a group of embeddings, as opposed to squeezing the whole sequence into a few tokens or selecting at token granularity. The four tasks also show a consistent difficulty order: life stage scores lowest for every method, a gradient that does not depend on any single method.

4.4 Generic Compression: MRQA In-Domain and Out-of-Domain

To test generic compression ability we retrain KuaFu on SAC’s training data following its original recipe; all baseline numbers are

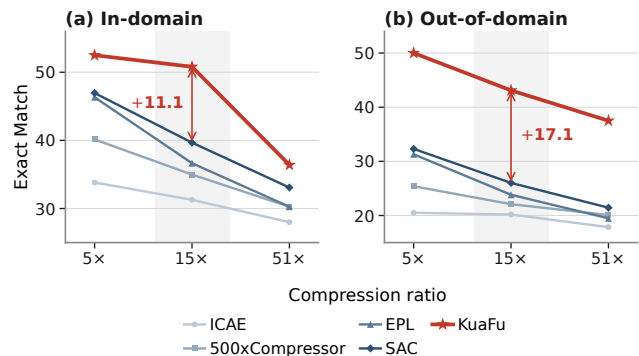


Figure 3: MRQA exact match vs. compression ratio (Llama-3.2-1B). Shaded band: 15×, the ratio used everywhere else. ROUGE-1 F1 and all baseline values: Appendix B.

taken from the SAC paper. In-domain evaluation uses the official test sets of the six MRQA subsets. Out-of-domain evaluation measures zero-shot transfer: it uses exactly the same checkpoint with no domain adaptation, so the two settings are directly comparable.

Figure 3 plots exact match against the compression ratio. KuaFu leads at every ratio in both settings; uncompressed it reaches 70.8 EM in-domain and 64.4 out-of-domain. Two readings matter. At 15× its 50.8 in-domain EM already matches SAC’s 5× level on the same backbone (46.95), i.e. the same usability at three times the compression strength. And out-of-domain the margin narrows only mildly as the ratio grows: at 51× KuaFu still scores above every baseline at 5×, so cross-domain generalization survives aggressive compression. The one exception is ROUGE-1 F1 at 51× in-domain, where KuaFu (45.9) trails SAC (46.37).

4.5 Recommendation Benchmark

We then check on the recommendation side whether memory compression preserves preference signal, following RecBench’s zero-shot pairwise CTR setting on its five datasets and restricting the comparison to that benchmark’s zero-shot block. Without any recommendation-specific training, and with the user-behavior input already compressed into a fixed-length memory representation, KuaFu-ZS attains a five-dataset mean of 0.617, trailing only Mistral-7B and GLM-4-9B and ahead of every other zero-shot model. The same-backbone control is the telling one: KuaFu-ZS exceeds its own backbone Qwen3-4B (0.570, identical weights and protocol) by 4.70 pp, so the gain comes from the compressed memory representation on the input side and not from parameter count; on that basis a 4B model overtakes its 8B sibling Qwen3-8B (0.598) by 1.90 pp. All twenty zero-shot models are plotted in Appendix C.

4.6 Hallucination-Aware Reinforcement Learning

Extending compressed understanding to open-ended question answering over long behavior sequences makes the four hallucination types surface together. We run hallucination-aware DAPO from the co-training checkpoint with the compression side frozen and only the decoder updated; the generative reward model used in training

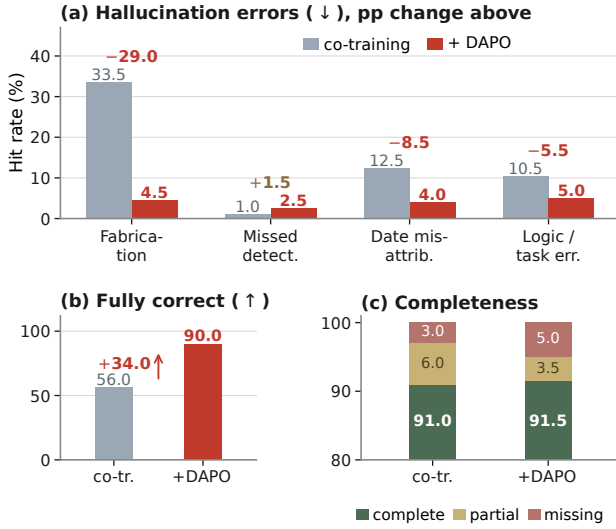


Figure 4: Hallucination-aware RL, scored by a stronger LLM than the training reward model. (a) hit rate per error type (lower better), pp change above each pair; (b) fully correct answers; (c) completeness, shares summing to 100%. All values in percent.

judges each of the four error types and completeness against the uncompressed original text. The offline evaluation set consists of 1000 long-sequence open-ended QA samples covering five instruction families, and every answer is judged by a different, stronger LLM judge so that the training judge and the test judge are not the same model. Error types are weighted by the harm they do to the user, with a separate completeness bonus; the weights are in Appendix E.

Figure 4 shows the effect. The most harmful category, fabrication, collapses by 29.0 pp, date and logic errors fall in step, and the fully-correct rate rises by 34.0 pp. Missed detection and missing answers drift up slightly – exactly what the “heavy penalty on fabrication, light penalty on omission” weighting asks for – while completeness holds, so the model answers in full rather than declining to answer.

4.7 Ablations

We ablate four points of the recipe: on the compressor side, how the curriculum splits the data and whether pre-trained weights must be inherited; on the projector side, module composition along the token axis and the dimension reduction strategy along the width axis; and on the decoder side, how the backbone training strategy affects final task accuracy. The four groups agree: neither a single module nor a naive scheme reaches the optimum, and only a progressive combination attains the lowest loss or the highest reward.

The curriculum group matters most. Without difficulty splitting – training directly on mixed lengths of 1–1200 – the loss stalls at 2.27 and will not fall further even at the largest data size (980K samples); with five increasing buckets by item count the stage-wise losses converge to 0.2, 0.28, 0.16, 0.02 and 0.034. The two settings consume comparable total training tokens (8.06B vs. 8.1B) and differ

only in how difficulty is organized, so *difficulty splitting rather than data volume is the necessary condition for the compressor to converge*. The inheritance group shows that watching the training loss alone is misleading: without inheritance the training loss still falls to 0.7 while the validation loss climbs anomalously to 4.1, whereas with inheritance the two descend together (0.6 and 0.8).

4.8 Online A/B Experiment

Online validation is an A/B experiment with traffic split randomly by unique visitor. Holding the resources used for profile mining identical between the two arms, the control keeps the previous profile features (one year of behavior, refreshed monthly) while the treatment replaces them with KuaFu profile features (two years of behavior, refreshed weekly); the recommendation model, the training samples and all other features are identical, so the difference between arms reflects the profile itself and nothing else. Traffic was ramped 1% → 5% → 20% → 80% → 100% within a four-week observation window; after full rollout a 5% holdout was retained and observed for a further ten months, over which overall GMV rose by 1.37% against that holdout (95% CI [0.71%, 2.03%]).

Because the two arms consume the same end-to-end resources (chiefly GPU and storage), the gain comes from the two kinds of headroom that compression frees up: the affordable behavior horizon stretches from one year to two, letting longer-range interest signals reach the recommendation model; and the refresh cadence moves from monthly to weekly, so interest drift is reflected in online features sooner.

4.9 Production Deployment and Throughput

KuaFu is deployed systematically on the Tencent advertising and recommendation platform, building two-year behavior sequences for a billion users. The pipeline is organized as two routine flows (Figure 5). *Item side, refreshed daily*: incremental items are extracted each day and compressed into memory embeddings by the compressor service, which writes them into the item representation bank; because the compressed result depends only on the item itself, one memory embedding serves as a long-lived, cross-user reusable cache and never needs recomputation per user. *User side, refreshed weekly*: item text inside a billion users’ sequences is replaced by cached memory embeddings, the recomposed sequence enters the KuaFu decoding service, and content interest, e-commerce interest, occupation/industry and life stage are mined and written into the online feature store for the recommendation model to request. Memory embeddings stay in the offline bank and serve only sequence recomposition; the online feature store carries profile results only.

Engineering-wise the compressor and the decoder are deployed on separate clusters: the compressor is an offline batch, per-item cacheable computation, whereas the decoder is a routine, per-user sequence-understanding computation. Their load shapes, batch structures and scaling rhythms differ substantially, so separating them lets each scale independently and keeps decoder-side sequence computation from crowding out item compression throughput. Combined with item-level caching, this makes the offline cache grow with item inventory instead of user count, which is what lets a weekly full recomputation for a billion users finish inside strict latency and cost budgets.

Table 4: Ablation summary. Best value of each group in bold; per-group details are in Appendix D.

Group	Settings compared	Metric
Curriculum (compressor)	mixed 1–1200 (980K) vs. five increasing buckets	train loss 2.27 vs. 0.2, 0.28, 0.16, 0.02, 0.034
Inheritance (compressor SFT)	without vs. with pre-training	init/train/val loss 3.2/0.7/ 4.1 vs. 1.2/0.6/ 0.8
Token axis (projector)	pooling / adapter / adapter + pooled residual	train loss 1.9967 / 1.8026 / 1.4148
Width axis (projector)	naive / +residual / +LR annealing / +continued training	train reward 0.1 / 0.25 / 0.6 / 0.8
Decoder backbone	Base / SFT / DAPO	life stage 0.74 / 0.81 / 0.81; e-commerce 0.806 / 0.823 / 0.826

Throughput gains differ by task: per-GPU QPM for life-stage profiling rises from 592 to 810 (+37%), cutting the required GPUs from 41 to 30; e-commerce interest mining rises from 250 to 800 (+220%), from 90 GPUs to 28; content interest mining rises from 40 to 180 (+350%), from 150 GPUs to 33 – **190 GPUs saved** in total. Per-item byte sizes and the total storage for a 20-billion deduplicated item inventory at each width setting are in Appendix D.

5 Lessons Learned

(i) **The curriculum is a necessary condition for convergence, not an acceleration trick.** Training directly on mixed 1–1200 lengths leaves the loss stuck at 2.27 even at the largest data size of 980K samples; five increasing buckets by item count converge to 0.2/0.28/0.16/0.02/0.034. The convergence bottleneck of long-sequence compression lies in the difficulty structure of the length distribution, not in data volume.

(ii) **The four hallucination types appear in production only as a diffuse decline of downstream metrics.** An online A/B takes a week and root causing takes a month; layered intermediate evaluation shortens that loop to a day, which is also the precondition for iterating on hallucination-aware RL at all.

(iii) **RL rewards should be ordered by harm to the user and must explicitly counteract refusal.** Fabrication carries the heaviest weight (0.45) and missed detection the lightest (0.07), with completeness as a separate bonus. After DAPO fabrication falls from 33.5 to 4.5 and the fully-correct rate rises from 56.0 to 90.0 with completeness held at 91.5; the price is 1.5–2.0 points more missed detection and missing answers, an expected trade-off.

(iv) **Cacheability of the representation decides the economics of the system.** Item-level independent encoding keys the cache by item, so it grows with item inventory rather than $\#users \times \#tasks$; the compressor (offline batch) and the decoder (routine sequence understanding) are deployed on separate clusters and scale independently. A weekly full refresh for a billion users turns on representation design, not on stacking compute.

6 Conclusion and Limitations

To address context overflow and hallucination when ultra-long user behavior sequences are placed in LLM prompts, we presented KuaFu: a two-axis compressed representation layer whose unit is a single behavior item, trained with a layered-difficulty curriculum, compression–generation co-training and hallucination-aware RL to constrain factual fidelity, and probed directly by a layered intermediate evaluation protocol. Offline and online results are in §4; the system has been fully deployed on the Tencent advertising and recommendation platform for ten months.

Limitations and future work: the item compression ratio is currently fixed per task family (10× for interest summarization, 15× for profiling), and adaptation on sparser sequences remains to be refined; overly long items may lose detail at low compression widths; and scaling along data and parameter size has not been validated systematically. Future work targets three-dimensional compression (token / width / temporal position), query-aware compression, scaling studies, and a continual-learning memory system for long-horizon interest evolution.

Statements

Reproducibility. The architecture, the compression operators and all four training objectives are given in §3, including the exact trainable parameter set of each stage and the reward used for reinforcement learning. §4 records the backbones, the composition and scale of every training stage, the data formats and the optimization and serving configuration. Every ablation table reports both the compared settings and their metrics. The training corpus is built from proprietary industrial behavior logs and cannot be released for privacy reasons; the public datasets and recommendation benchmarks used for evaluation are all listed in §4 and publicly available, so the compression and understanding components can be reproduced independently of proprietary data.

Use of generative AI. We used generative AI tools for language editing and polishing of author-written drafts, and as a programming aid for data processing and plotting scripts. They were not used for ideation, experimental design or the derivation of any reported result, and agentic use does not apply. All AI-assisted text was verified by the authors against the underlying experimental records and all AI-assisted code was reviewed and tested.

Ethics. This work analyzes user behavior logs from the Tencent advertising and recommendation platform. All collection and processing follow the platform privacy policy and applicable regulations. Logs are accessed in de-identified form; no raw personally identifiable information is used in training or evaluation and no individual-level data is released. Compressed item embeddings are keyed by item rather than by user, which limits the user-level information retained in the offline cache.

References

- [1] Xilun Chen, Shao-Chuan Wang, Baykal Kakici, Lukasz Heldt, Lichan Hong, Raghu Keshavan, Aniruddh Nath, Li Wei, and Xinyang Yi. 2026. Token Factory: Efficiently Integrating Diverse Signals into Large Recommendation Models. arXiv:2606.19635 [cs.LR]
- [2] Alexis Chevalier, Alexander Wettig, Anirudh Ajith, and Danqi Chen. 2023. Adapting Language Models to Compress Contexts. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 3829–3846. doi:10.18653/v1/2023.emnlp-main.232
- [3] Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew

Item side – refreshed daily (offline batch cluster)

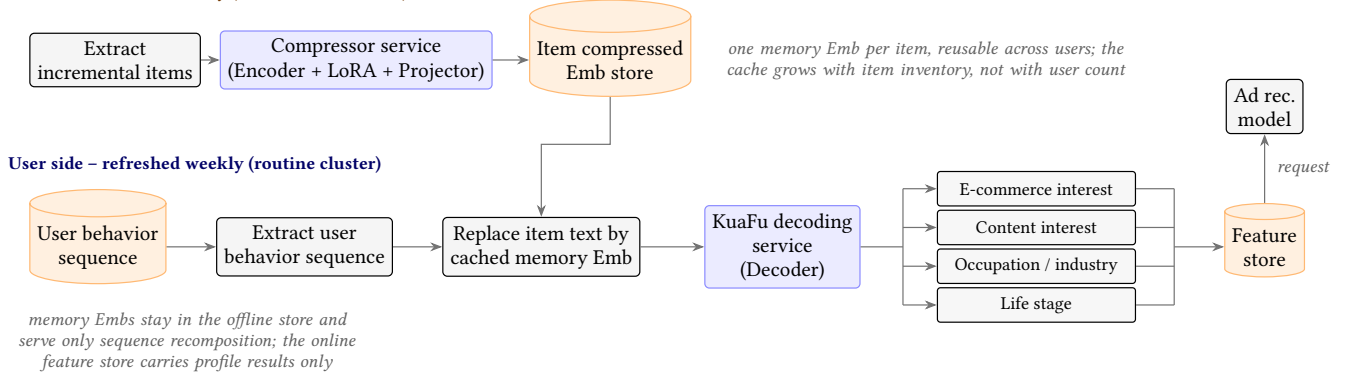


Figure 5: Production deployment of KuaFu: a daily item-side pipeline and a weekly user-side pipeline, served by two independently scaled clusters. The compressor is an offline batch, per-item cacheable computation; the decoder is a routine, per-user sequence-understanding computation. Because the compressed result depends only on the item, the offline cache grows with item inventory rather than with user count.

- Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. 2023. Language Modeling Is Compression. arXiv:2309.10668 [cs.LG]
- [4] Adam Fisch, Alon Talmor, Robin Jia, Minjoon Seo, Eunsol Choi, and Danqi Chen. 2019. MRQA 2019 Shared Task: Evaluating Generalization in Reading Comprehension. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*. Association for Computational Linguistics, Hong Kong, China, 1–13. doi:10.18653/v1/D19-5801
- [5] Iordanis Fostropoulos, Muhammad Rafay Azhar, Abdalaziz Sawwan, Boyu Fang, Yuchen Liu, Jiayi Liu, Hanchao Yu, Qi Guo, Jianyu Wang, Fei Liu, and Xiangjun Fan. 2026. GISTBench: Evaluating LLM User Understanding via Evidence-Based Interest Verification. arXiv:2603.29112 [cs.IR]
- [6] Tao Ge, Jing Hu, Lei Wang, Xun Wang, Si-Qing Chen, and Furu Wei. 2023. In-context Autoencoder for Context Compression in a Large Language Model. arXiv:2307.06945 [cs.CL]
- [7] Aaron Grattafiori et al. 2024. The Llama 3 Herd of Models. arXiv:2407.21783 [cs.AI]
- [8] Lin Guan, Jia-Qi Yang, Zhishan Zhao, Beichuan Zhang, Bo Sun, Xuanyuan Luo, Jinan Ni, Xiaowen Li, Yuhang Qi, Zhifang Fan, Hangyu Wang, Qiwei Chen, Yi Cheng, Feng Zhang, and Xiao Yang. 2026. Make It Long, Keep It Fast: End-to-End 10k-Sequence Modeling at Billion Scale on Douyin. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 7989–7998. doi:10.1145/3774904.3792811
- [9] Chuan He, Yang Chen, Bin Dou, Wuliang Huang, Baokun Wang, Yongchao Liu, Xing Fu, Yu Cheng, Chuntao Hong, Weiqiang Wang, Zhongle Xie, Jiajun Zheng, and Xin-Wei Yao. 2025. FOUNdV2: Learning Unified User Quantized Tokenizers for User Representation. arXiv:2508.00956 [cs.IR]
- [10] Ruining He, Lukasz Heldt, Lichan Hong, Raghunandan Keshavan, Shifan Mao, Nikhil Mehta, Zhengyang Su, Alicia Tsai, Yueqi Wang, Shao-Chuan Wang, Xinyang Yi, Lexi Baugher, Baykal Cakici, Ed Chi, Cristos Goodrow, Ningren Han, He Ma, Romer Rosales, Abby Van Soest, Devansh Tandon, Su-Lin Wu, Weilong Yang, and Yilin Zheng. 2026. PLUM: Adapting Pre-trained Language Models for Industrial-scale Generative Recommendations. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8093–8104. doi:10.1145/3774904.3792802
- [11] Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. LLMingua: Compressing Prompts for Accelerated Inference of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 13358–13376. doi:10.18653/v1/2023.emnlp-main.825
- [12] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient Memory Management for Large Language Model Serving with PagedAttention. In *Proceedings of the 29th Symposium on Operating Systems Principles (SOSP '23)*. Association for Computing Machinery, Koblenz, Germany, 611–626. doi:10.1145/3600006.3613165
- [13] Weijiang Lai, Beihong Jin, Di Zhang, Siru Chen, Jiongyan Zhang, Yuhang Gou, Jian Dong, and Xingxing Wang. 2026. Unleashing the Potential of Sparse Attention on Long-term Behaviors for CTR Prediction. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8041–8050. doi:10.1145/3774904.3792808
- [14] Junnan Li, Dongxu Li, Silvio Savarese, and Steven Hoi. 2023. BLIP-2: Bootstrapping Language-Image Pre-training with Frozen Image Encoders and Large Language Models. arXiv:2301.12597 [cs.CV]
- [15] Yucheng Li, Bo Dong, Frank Guerin, and Chenghua Lin. 2023. Compressing Context to Enhance Inference Efficiency of Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, Singapore, 6342–6353. doi:10.18653/v1/2023.emnlp-main.391
- [16] Yuchen Li, Rui Kong, Xinran Chen, Chengzhe Zhang, Jiamin Chen, Cheng Deng, Xinyu Ma, Haojie Zhang, Tianhao Peng, Hengyi Cai, Shuaiqiang Wang, Jiashu Zhao, Yongqi Zhang, Haoyi Xiong, Jimmy Xiangji Huang, Lei Chen, Jun Wang, and Dawei Yin. 2026. Probe-and-Fetch: Dynamic KV Cache Pruning for Accelerated Long-Context Inference in Web-Scale AI Search. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8127–8137. doi:10.1145/3774904.3792794
- [17] Zhao Li, Fengyang Qi, Chuanyu Xu, Tao Zhang, Chengfu Huo, and Peng Zhang. 2026. LSIG: Long Semantic IDs for Generative Recommendation. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 7779–7788. doi:10.1145/3774904.3792834
- [18] Zongqian Li, Yixuan Su, and Nigel Collier. 2024. 500xCompressor: Generalized Prompt Compression for Large Language Models. arXiv:2408.03094 [cs.CL]
- [19] Jiayi Liao, Ruobing Xie, Sihang Li, Xiang Wang, Xingwu Sun, Zhanhui Kang, and Xiangnan He. 2025. Multi-Grained Patch Training for Efficient LLM-based Recommendation. In *Proceedings of the 48th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '25)*. Association for Computing Machinery, Padua, Italy. doi:10.1145/3726302.3730042
- [20] Xiaoqiang Lin, Aritra Ghosh, Bryan Kian Hsiang Low, Anshumali Shrivastava, and Vijai Mohan. 2025. REFRAG: Rethinking RAG based Decoding. arXiv:2509.01092 [cs.CL]
- [21] Qijiong Liu, Jieming Zhu, Zhaocheng Du, Lu Fan, Zhou Zhao, and Xiao-Ming Wu. 2024. Learning Multi-Aspect Item Palette: A Semantic Tokenization Framework for Generative Recommendation. arXiv:2409.07276 [cs.IR]
- [22] Qijiong Liu, Jieming Zhu, Lu Fan, Kun Wang, Hengchang Hu, Wei Guo, Yong Liu, and Xiao-Ming Wu. 2025. Can LLMs Outshine Conventional Recommenders? A Comparative Evaluation. arXiv:2503.05493 [cs.IR]
- [23] Xin Liu, Runsong Zhao, Pengcheng Huang, Xinyu Liu, Junyi Xiao, Chunyang Xiao, Tong Xiao, Shengxiang Gao, Zhengtao Yu, and Jingbo Zhu. 2025. Autoencoding-Free Context Compression for LLMs via Contextual Semantic Anchors. arXiv:2510.08907 [cs.CL]
- [24] MNBVC Community. 2023. MNBVC: Massive Never-ending BT Vast Chinese Corpus. <https://github.com/esbatmop/MNBVC>. Accessed 2026-09-23.
- [25] Jesse Mu, Xiang Lisa Li, and Noah Goodman. 2023. Learning to Compress Prompts with Gist Tokens. In *Advances in Neural Information Processing Systems 36*. Curran Associates, Inc., New Orleans, LA, USA, 19327–19352. arXiv:2304.08467
- [26] Baopu Qiu, Hao Chen, Yuanrong Wu, Changtong Zan, Chao Wei, Weiru Zhang, and Xiaoyi Zeng. 2026. Thinking Broad, Acting Fast: Latent Reasoning Distillation from Multi-Perspective Chain-of-Thought for E-Commerce Relevance. In *Proceedings of the ACM Web Conference 2026 (WWW '26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8171–8182. doi:10.1145/3774904.3792812

- [27] Qwen Team et al. 2025. Qwen3-VL Technical Report. arXiv:2511.21631 [cs.CV]
- [28] 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 [cs.IR]
- [29] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, et al. 2024. DeepSeekMath: Pushing the Limits of Mathematical Reasoning in Open Language Models. arXiv:2402.03300 [cs.CL]
- [30] Chao Wang, Neo Wu, Lin Ning, Jiaxing Wu, Luyang Liu, Jun Xie, Shawn O’Banion, and Bradley Green. 2024. UserSumBench: A Benchmark Framework for Evaluating User Summarization Approaches. arXiv:2408.16966 [cs.CL]
- [31] Yejing Wang, Shengyu Zhou, Jinyu Lu, Ziwei Liu, Langming Liu, Maolin Wang, Wenlin Zhang, Feng Li, Wenbo Su, Pengjie Wang, Jian Xu, and Xiangyu Zhao. 2026. NEZHA: A Zero-sacrifice and Hyperspeed Decoding Architecture for Generative Recommendations. In *Proceedings of the ACM Web Conference 2026 (WWW ’26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8073–8082. doi:10.1145/3774904.3792797
- [32] Runze Xia, Yupeng Ji, Yuxi Zhou, Haodong Liu, Teng Zhang, and Piji Li. 2026. From Reasoning LLMs to BERT: A Two-Stage Distillation Framework for Search Relevance. In *Proceedings of the ACM Web Conference 2026 (WWW ’26)*. Association for Computing Machinery, Dubai, United Arab Emirates, 8222–8231. doi:10.1145/3774904.3792801
- [33] An Yang et al. 2025. Qwen3 Technical Report. arXiv:2505.09388 [cs.CL]
- [34] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, et al. 2025. DAPO: An Open-Source LLM Reinforcement Learning System at Scale. arXiv:2503.14476 [cs.LG]
- [35] Zheng Zhang, Jinyi Li, Yihuai Lan, Xiang Wang, and Hao Wang. 2025. An Empirical Study on Prompt Compression for Large Language Models. arXiv:2505.00019 [cs.CL]
- [36] Runsong Zhao, Xin Liu, Xinyu Liu, Pengcheng Huang, Chunyang Xiao, Tong Xiao, and Jingbo Zhu. 2024. Position IDs Matter: An Enhanced Position Layout for Efficient Context Compression in Large Language Models. arXiv:2409.14364 [cs.CL]

A Training Recipe and Data Construction

Table 5 gives the full four-stage recipe behind Table 1. C/P/D denote the compressor, the projector and the decoder; “update” and “freeze” refer to the trainable set of that stage.

Understanding samples. Understanding tasks are built mainly from advertising behavior sequences. A large instruction model first retrieves similar items to assist prompt writing; formal samples are then generated in two steps: word frequencies are counted and nominal subjects extracted to pin down the key information that must be preserved, and a normalized question–answer pair is generated accordingly. Simple understanding covers entity recognition, restating items at a given index, parity or contiguous range, counting, and longest-interest-run extraction. Complex understanding covers user profiling and interest mining, item similarity, topic coverage and interest grouping, explanation of behavioral causes, cross-source coordination, and temporal behavior analysis.

B Full MRQA Results

Table 6 reports exact match and ROUGE-1 F1 at every compression ratio, in-domain and out-of-domain, for every baseline.

C Full RecBench Results

Figure 6 places all twenty zero-shot models of RecBench [22] on the accuracy-versus-size plane, together with the caveats that the five-dataset mean alone does not convey.

D Ablation Details and Production Storage

Table 7 collects the per-group numbers behind Table 4 together with the production storage footprint.

E Judging Protocol for Long-Sequence Open QA

Each answer is judged against the uncompressed original text along five independent dimensions; the first four are scored present/absent and the fifth on a three-level scale. In the RL reward the four error types are weighted by the harm they do to the user: fabrication 0.45, date misattribution 0.24, logic/task error 0.24, missed detection 0.07. The judge is instructed to be lenient about minor wording and mild inference, and to answer “present” only for substantive errors.

- **Fabrication.** Introduces an entity, brand, person, place, title, organisation, number, event or direct quotation that is absent from the source; or invents a platform / region label the source never gives.
- **Missed detection.** Asserts “none / not found / no such X” while the source contains at least one relevant entry, including synonymous or near-synonymous phrasings.
- **Date misattribution.** Assigns an event to the wrong date, misquotes a year / month / day, or invents a period label the source never gives.
- **Broken logic.** Factual distortion other than dates – a value reversed in direction, an attribute assigned to the wrong subject, a broken correspondence, self-contradiction, or a claim that conflicts with an explicit fact; or reversed causality and plainly invalid inference.
- **Completeness.** Complete / partial / missing, with respect to the parts of the instruction that the answer actually addresses.

F Difficulty-Increasing Curriculum: Task Families

Needle in a haystack. For a given item in the sequence, find every related item; given a target type, find every item of that type; run entity recognition over all items; summarize all items; find all items at odd positions; find all items at even positions; return the items in positions j through k ; return the items at several randomly specified positions.

Interest distribution and diversity. Given the behavior sequence (a smaller index means a more recent behavior), mine sub-category interests under each main interest category. The output is JSON with a subcategory_interests dictionary, keyed by main category, whose values are lists of sub-categories carrying a keyword, a count and supporting evidence, plus a short reasoning field.

Note on data handling: the evidence field holds de-identified references rather than verbatim behavior text, consistent with the ethics statement in the body.

Table 5: Four-stage training recipe and data statistics. C/P/D denote the compressor, the projector and the decoder; upd. / frz. mark whether the module is updated or frozen in that stage.

Stage	Objective	C	P	D	Task composition	Samples	Tokens
Pre-training	restore text	upd.	—	frz.	behavior seq. + long text, bucketed by item count	2.13M	8.06B
Post-tr. (simple)	compressed QA	upd.	upd.	frz.	8 behavior-understanding + 7 generic NLP tasks	150K	0.80B
Post-tr. (complex)	compressed QA	upd.	upd.	frz.	22 complex long-sequence + 4 profiling tasks	350K	2.30B
Co-training	fit both ends	upd.	upd.	upd.	same composition as post-training	350K	2.30B
Hallu.-aware RL	suppress 4 errors	frz.	frz.	upd.	long-sequence open-ended QA	5K	41M

Table 6: Full MRQA results, EM / ROUGE-1 F1 on a Llama-3.2-1B backbone. The out-of-domain columns use the same checkpoint, zero-shot. Only KuaFu has an entry in the uncompressed (Full) column: the baselines are compression methods and are undefined without compression, so those cells are a property of the protocol rather than missing data.

Method	In-domain				Out-of-domain			
	Full	5×	15×	51×	Full	5×	15×	51×
ICAE [6]	—	33.82 / 47.53	31.28 / 44.5	27.99 / 40.39	—	20.51 / 31.22	20.18 / 30.81	17.85 / 27.98
500xCompressor [18]	—	40.14 / 55.26	34.98 / 49.35	30.26 / 43.19	—	25.4 / 38.46	22.09 / 34.04	20.09 / 30.43
EPL [36]	—	46.33 / 62.9	36.65 / 51.52	30.26 / 43.22	—	31.3 / 46.95	23.83 / 36.74	19.48 / 30.22
SAC [23]	—	46.95 / 63.63	39.67 / 54.95	33.08 / 46.37	—	32.3 / 47.72	26.02 / 39.26	21.44 / 32.24
KuaFu (ours)	70.8 / 79.3	52.5 / 66.1	50.8 / 61.7	36.4 / 45.9	64.4 / 73.7	50.0 / 60.1	43.1 / 54.2	37.5 / 48.2

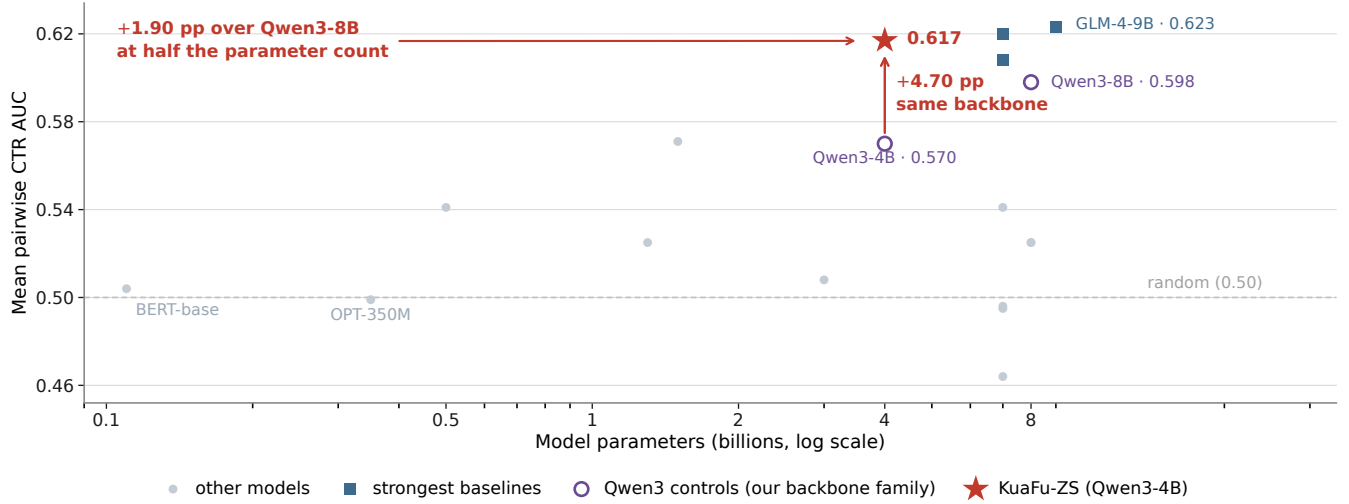
**Figure 6: RecBench zero-shot pairwise CTR, mean AUC over the five datasets, against parameter count. Four notes. (i) GPT-3.5 (0.523) is served through an API and P5-Beauty (0.505) reports no parameter count, so neither is plotted; the two strongest models overall are Mistral-7B (0.6199) and GLM-4-9B (0.6231), both marginally above KuaFu-ZS. (ii) All differences in the body are computed on *unrounded* means – KuaFu-ZS is 0.6166, drawn here as 0.617 – so recomputing them from three-decimal figures differs by 0.03–0.05 pp. (iii) The mean hides one regression: on the CDs subset the compressed representation brings no gain (0.489, below the 0.510 of the Qwen3-4B backbone). (iv) KuaFu-ZS gets no task-specific training; only its user-behavior input is compressed.**

Table 7: Ablation details and production storage. (a) compressor curriculum: both settings consume comparable total tokens, so the difference is the difficulty organization alone. (b) inheriting pre-trained compressor weights before downstream SFT. (c) token-axis composition, width-axis strategy and decoder backbone. (d) storage for a deduplicated inventory of 20 billion items shared across users.

(a) Compressor curriculum					
Setting	Items	Samples	Avg. tok.	Total tok.	Loss
curriculum	1	550K	133	73.15M	0.2
	1-3	400K	267	106.8M	0.28
	1-200	500K	5330	2.6B	0.16
	1-500	600K	7656	4.6B	0.02
	1-1200	83K	8202	680M	0.034
no curriculum	1-1200	980K	8245	8.1B	2.27

(b) Inheritance			
	init. loss	train loss	val. loss
without	3.2	0.7	4.1
with	1.2	0.6	0.8

(d) Production storage				
Compression	20×	10×	2.5×	1×
(token, width)	(2, 128)	(2, 256)	(2, 1024)	(2, 2560)
per item	512 B	1 KB	4 KB	10 KB
total (TiB)	9.3	18.6	74.5	186.2

(c) Projector axes and decoder backbone	
Token axis	train loss
pooling	1.9967
adapter	1.8026
adapter + pooled residual	1.4148
Width axis	train reward
naive	0.1
+ progressive residual	0.25
+ differentiated LR annealing	0.6
+ continued training	0.8
Decoder backbone	Base / SFT / DAPO
life stage	0.74 / 0.81 / 0.81
e-commerce interest	0.806 / 0.823 / 0.826