# HUOZIIME: An On-Device LLM-enhanced Input Method for Deep Personalization

**Baocai Shan** **Yuzhuang Xu** **Wanxiang Che***
Harbin Institute of Technology, Harbin, China
{bcshan,car}@ir.hit.edu.cn

## Abstract

Mobile input method editors (IMEs) are the primary interface for text input, yet they remain constrained to manual typing and struggle to produce personalized text. While lightweight large language models (LLMs) make on-device auxiliary generation feasible, enabling deeply personalized, privacy-preserving, and real-time generative IMEs poses fundamental challenges. To this end, we present HUOZIIME, a personalized on-device IME powered by LLM. We endow HUOZIIME with initial human-like prediction ability by post-training a base LLM on synthesized personalization data. Notably, a hierarchical memory mechanism is designed to continually capture and leverage user-specific input history. Furthermore, we perform systemic optimizations tailored to on-device LLM-based IME deployment, ensuring efficient and responsive operation under mobile constraints. Experiments demonstrate efficient on-device execution and high-fidelity memory-driven personalization. Code and package are available at https://github.com/Shan-HIT/HuoziIME.

*Corresponding author

## 1 Introduction

Input method editors (IMEs) serve as the most fundamental and highest-frequency interface for text entry in mobile human-device interaction. While traditional Chinese IMEs have evolved from N-gram statistical models to neural transliteration tools (Chen et al., 2015; Huang et al., 2018), they remain fundamentally constrained to pinyin-to-character conversion. Consequently, they struggle to model user intent or capture long-term personalized writing patterns.

Recently, large language models (LLMs) demonstrate immense potential in writing assistance (Bubeck et al., 2023; Zhao et al., 2023). Mainstream vendors begin exploring hybrid IME architectures that combine traditional candidate generation with LLM-driven intelligent writing and preset personas. However, such integrations are largely incremental: generative AI is introduced as a cloud-based auxiliary add-on rather than a foundational redesign of the IME pipeline, as shown in Table 1. As a result, existing IMEs remain limited in three critical aspects.


Figure 1: Overview of HuoziIME. We illustrate the end-to-end workflow, from user input and memory retrieval to LLM-based prediction, highlighting the full-process context management of LLM in IME.



*Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 327–335
July 2-7, 2026 

| Vendor | Model | Deployment | Core AI Functions | Personalization | Memory | Privacy & Security |
|---|---|---|---|---|---|---|
| SwiftKey | GPT-4 Turbo | Cloud-first | Writing assist; context modeling; emoji gen | ✗ Weak (static) | ✗ No (stateless) | ⚠ Cloud-dependent; data exposure risk |
| BaiduIME | ERNIE Bot | Cloud-first | Writing assist; emotional chat; rewrite | ✗ Weak (prompt-based) | ✗ No (short-term) | ⚠ Cloud-dependent; offline unstable |
| SogouIME | Hunyuan | Cloud-first | Speech polish; spoken-to-written; translation | ✗ Weak (lexicon-based) | ✗ No (no persistent) | ⚠ Cloud-dependent; privacy tradeoff |
| iFlytekIME | Spark | Hybrid | Persona presets; completion; agent | ⚠ Medium (preset switch) | ✗ No (no persistent) | ⚠ Hybrid inference; cloud auth for advanced |
| WeChatIME | Hunyuan | Cloud-first | Knowledge retrieval; error correction | ✗ Weak (general-purpose) | ✗ No (short-term) | ⚠ Cloud-dependent; interaction transmission |
| DoubaoIME | Doubao | Cloud-first | Speech enhancement; association; prediction | ✗ Weak (no habit learning) | ✗ No (no persistent) | ⚠ Cloud + mic access |
| HUOZIIME (Ours) | IME-specific LLM | On-device | Memory-based completion; persona presets; self-evolving capability | ✓ Strong (evolving + adaptive) | ✓ Yes (L1/L2/L3) | ✓ On-device; auditable |

Table 1: Comparison of representative LLM-enhanced Chinese IMEs across functionality, personalization, memory, and privacy. We briefly list the key features of different IMEs.

The first limitation is that generative LLMs are often loosely coupled with the traditional keystroke-to-candidate pipeline, where models are invoked only for long-form rewriting or full-sentence suggestions. This restricts fine-grained modeling of short-phrase completion and interactive edits. Second, personalization mechanisms are shallow, typically confined to short context windows or static personas, without the ability to accumulate long-term user knowledge or continuously adapt from feedback. Third, cloud-dependent inference introduces unpredictable latency and privacy risks for keystroke-level data (Hard et al., 2018; Zhao and Song, 2024), undermining responsiveness in high-frequency typing scenarios.

To close this gap, we present HUOZIIME, to the best of our knowledge the first memory-augmented, on-device LLM-driven IME. We tightly integrate the LLM into the IME pipeline, enabling it to play a persistent and central role throughout the typing process, as illustrated in Figure 1. Through fine-tuning on high-quality synthesized personalization data, we equip the base LLM with initial persona-aware generation capabilities, enabling flexible identity switching. To support continual on-device evolution, we design a three-level hierarchical memory architecture together with a GRPO-based memory trigger, enabling the model to autonomously incorporate user-specific signals over time. All functions operate efficiently on-device, supported by systemic optimization.Extensive evaluations confirm that our architecture ensures millisecond-level responsiveness and precise, memory-augmented personal writing.

The contributions of this work are 3-fold:

- We present a synthetic personalization data pipeline and fine-tuning a lightweight base LLM into a persona-aware IME backbone, enabling identity controllable and human-like prediction within the typing loop.
- We design a 3-level hierarchical memory architecture, coupled with a GRPO-based trigger, which achieves efficient short-term knowledge recall while supporting long-term user modeling and continual on-device evolution.
- We perform systemic hardware-level optimizations tailored to mobile IME workloads, ensuring low-latency and efficient running of LLM-driven IME under real constraints.

## 2 HUOZIIME Design

HUOZIIME is designed for extremely resource-constrained mobile environments. Our architecture follows four core principles:

- **Seamless integration:** enhance rather than disrupt established keystroke habits.
- **Expressive takeover:** personalized generation with memory-grounded factual augmentation.
- **User-experience first:** co-optimize CPU runtime for millisecond-level responsiveness.
- **Privacy preservation:** keep context and memory on-device rather than online.

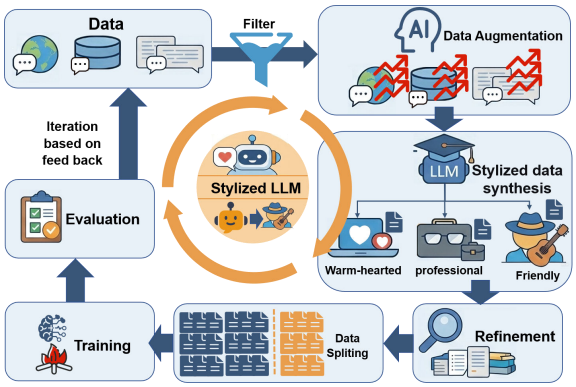



Figure 2: Stylized post-training pipeline. We construct persona-specific conversational data through curation, augmentation, filtering, and iterative refinement.

### 2.1 Real time interactive UI Design

The frontend design of HUOZIIME injects LLM-generated completions into the familiar candidate-based workflow. Figure 1 shows that we use cursor-adjacent *GhostText* with a dual-layer candidate interface. GhostText displays the best short-phrase candidate in real time. A secondary panel displays alternatives. Users accept results by tapping the candidate bubble. This preserves existing interaction habits and minimizes behavioral friction.

### 2.2 Stylized LLM Post-Training

Generic LLMs tend to produce neutral and impersonal completions. Although such responses are correct, they often lack stylistic consistency and identity-specific expression, resulting in mechanical and context-insensitive candidates. In human-centric communication, however, users type text that reflects personal traits and situational tone. Therefore, stylized post-training is necessary to transform a generic LLM into a persona-aware generator that can produce basic personalized continuations within the typing loop.

To this end, we perform stylized post-training over three predefined persona styles, each representing a distinct communicative identity. As shown in Figure 2, we first collect large-scale Internet conversational corpora and augment them with high-quality LLM-synthesized dialogues to enrich scenario diversity. An LLM-assisted filtering and style classification pipeline is then applied to remove low-quality samples and ensure stylistic purity. The curated data are then split into training and test sets. We construct structured training contexts by concatenating personalized system prompt with dialogue history, thereby explicitly generating based on persona. The model is fine-tuned on this stylized corpus and evaluated to assess style fidelity and fluency. Based on evaluation feedback, we iteratively refine data curation and form a closed-loop pipeline that progressively strengthens personality and quality.

### 2.3 Hierarchical Memory Mechanism

The memory mechanism is the core of HUOZIIME, enabling LLM-based completion to incorporate historical context and better reflect the users's own writing style. Inspired by computer memory hierarchy (Packer et al., 2024), HUOZIIME adopts a decoupled L1/L2/L3 memory architecture separating foreground completion from background curation. They are defined and used as follows:

- **L1 High-speed cache layer:** Stores style-specific and per-memory KV blobs. It directly injects precomputed KV states into the foreground path and reduces prefill overhead.
- **L2 Plaintext fact layer:** Serves as an auditable source of user facts. It appends structured memory records in plaintext and maintains a lightweight HNSW vector index (Malkov and Yashunin, 2018) for semantic recall.
- **L3 Parametric weight:** Accumulates interaction trajectories and model decision logs. It automatically produces datasets for subsequent on-device or online fine-tuning.

Since performing memory retrieval at every prediction step incurs substantial computational overhead and is often unnecessary, we introduce a GRPO-based memory trigger to selectively activate retrieval when appropriate. In this trigger, we predefine four task classes: **(A) Direct completion** supports foreground typing. **(B) Memory retrieval** emits queries when facts are needed. **(C1) Memory extraction** converts interaction traces into structured records and send them to memory. **(C2) Invalid-information refusal** rejects noisy or sensitive fragments. After stylized post-training, the retrieval and refusal boundaries remain unclear. We therefore apply GRPO algorithm (Shao et al., 2024) to sharpen these boundaries. The reward function jointly considers format compliance, task correctness, and latency constraints. Detailed GRPO procedures are provided in Appendix A.1.

### 2.4 Efficiency Optimization

To meet millisecond-level latency requirements on heterogeneous mobile CPUs, HUOZIIME adopts

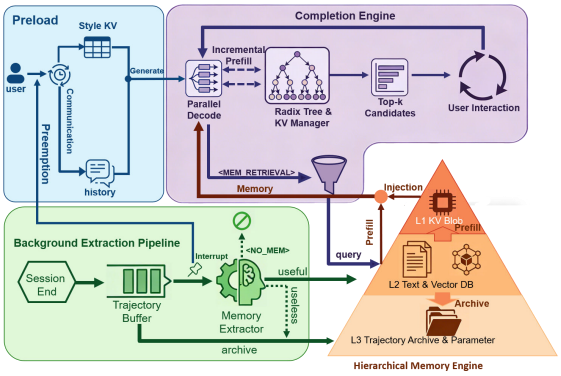



Figure 3: Interaction pipeline between HUOZIIME and HuoziIME-Chat in a daily conversation scenario.

a deeply optimized on-device inference runtime that eliminates redundant computation and maximizes cache reuse. We employ radical quantization and pruning methods to compress the model (Frantar et al., 2022; Xu et al., 2024, 2026, 2025). We also manage the KV cache as a compressed prefix tree (Gusfield, 1997; Zheng et al., 2024), enabling structural sharing across overlapping prefixes caused by typing and edits. Instead of re-prefilling from scratch, the runtime locates the longest common prefix and resumes decoding from the matched node, effectively transforming interactive editing into incremental tree traversal. On big.LITTLE architectures, we further apply thread-affinity scheduling (Gerganov, 2023), dispatching attention and GEMM kernels to high-performance cores, thereby reducing tail latency and stabilizing thermal behavior under bursty workloads.

For memory-augmented generation, we avoid full re-prefill through a lightweight KV-Splice mechanism that injects precomputed memory KV segments into the active decoding path via position-offset mapping. Since RoPE-based models bind KV states to absolute positions (Su et al., 2024), we adopt a position-independent caching strategy inspired by PIC (Gim et al., 2024; Yang et al., 2025), using phase-shift correction (Yang et al., 2025) together with selective tail recomputation (Yao et al., 2025) to preserve causal attention. This enables flexible cached context stitching with minimal recomputation overhead, maintaining fluent continuation while substantially reducing reinjection cost.

## 3 HUOZIIME Implementation

Figure 3 illustrates the end-to-end interaction between HUOZIIME and its companion application *HuoziIME-Chat*. We describe a representative daily conversation scenario to demonstrate how memory-augmented short-phrase completion operates under real mobile constraints. The implementation tightly couples online inference with asynchronous memory evolution, forming a self-updating personalization loop on device.

### 3.1 Online Interaction Loop

**Cross-app context synchronization.** Due to Android sandbox isolation, real-time context sharing across applications requires explicit coordination. We implement lightweight cross-process communication via the Model Context Protocol (MCP) (Anthropic, 2024). When a chat session changes, authorized host applications issue a `SYNC` request, enabling dynamic context updates. If MCP is unavailable, HUOZIIME gracefully degrades to input-only inference without accessing external application context.

Upon receiving a `SYNC` signal, the engine selects the appropriate L1 style cache and formats dialogue history using the post-training template. Incremental prefill with RadixTree-based prefix reuse (§2.4) ensures that only modified suffix segments incur computation, minimizing redundant processing.

**Memory-augmented candidate generation.** During typing, the LLM performs multi-threaded decoding to generate short-phrase candidates. If external factual grounding is required, the model emits the control token `<MEM_RETRIEVAL>`, triggering vector search over the local HNSW index. Retrieved plaintext memories and precomputed KV segments are fused into the active decoding state via KV injection or incremental prefill (§2.4), after which multiple candidate completions are sampled for display.

**Incremental interactive rendering.** The top prediction is rendered as cursor-adjacent GhostText, while alternative candidates appear in the suggestion panel. As the user continues typing or selecting candidates, the runtime reuses prefix states and performs incremental decoding only on newly appended fragments, amortizing the perceived Time-to-First-Candidate latency to near-zero during interaction. After message submission, the updated dialogue context is synchronized and the interaction trace is cached for background processing.

### 3.2 Asynchronous Memory Update

**Memory extraction and structuring.** When the conversation ends or the user switches applications, the system transitions to a background curation

| Pipeline Stage | Success / Total | Rate (%) |
|---|---|---|
| Memory Trigger | 342 / 343 | 99.7 |
| Processing (Normal) | 163 / 169 | 96.4 |
| Processing (Refusal) | 87 / 122 | 71.3 |
| Retrieval@4 | 179 / 200 | 89.5 |
| Grounded Generation | 156 / 179 | 87.2 |

Table 2: Performance of the memory triggering and retrieval effectiveness.

phase. Cached interaction traces are converted into structured records (L3), and the GRPO-trained policy determines whether factual extraction or refusal is appropriate. Candidate facts are rewritten into concise declarative form for persistent storage.

**Tiered persistence with strict preemption.** Validated memories are inserted into the L2 plaintext store, indexed via HNSW, and optionally compiled into reusable L1 KV blobs. To preserve real-time responsiveness, strict foreground preemption is enforced: any incoming `SYNC` request immediately interrupts background execution and restores the online inference path.

Together, the online and asynchronous pipelines enable continual on-device personalization under strict mobile constraints.

## 4 Experiments

We evaluate HUOZIIME on an Android test device equipped with a MediaTek Dimensity 9000 SoC and 12 GB RAM. Our evaluation focuses on memory retrieval effectiveness and on-device inference performance to validate its usability as a real-time mobile application.

### 4.1 Memory Pipeline Effectiveness

We evaluate the memory pipeline across its complete lifecycle, i.e., Triggering, Processing, Retrieval, and Grounded Generation. Specifically, the *Triggering* stage uses a targeted input prefix to prompt the model to initiate a retrieval request. *Processing* handles the frontend user input, either extracting and storing relevant facts into memory or actively disregarding noisy, irrelevant context. The *Retrieval* stage queries the memory database to fetch the most pertinent information. Finally, *Grounded Generation* assesses the model's ability to seamlessly incorporate the retrieved memory into an output that is both factually accurate and stylistically coherent. The test cases are drawn from our pre-synthesized dataset.

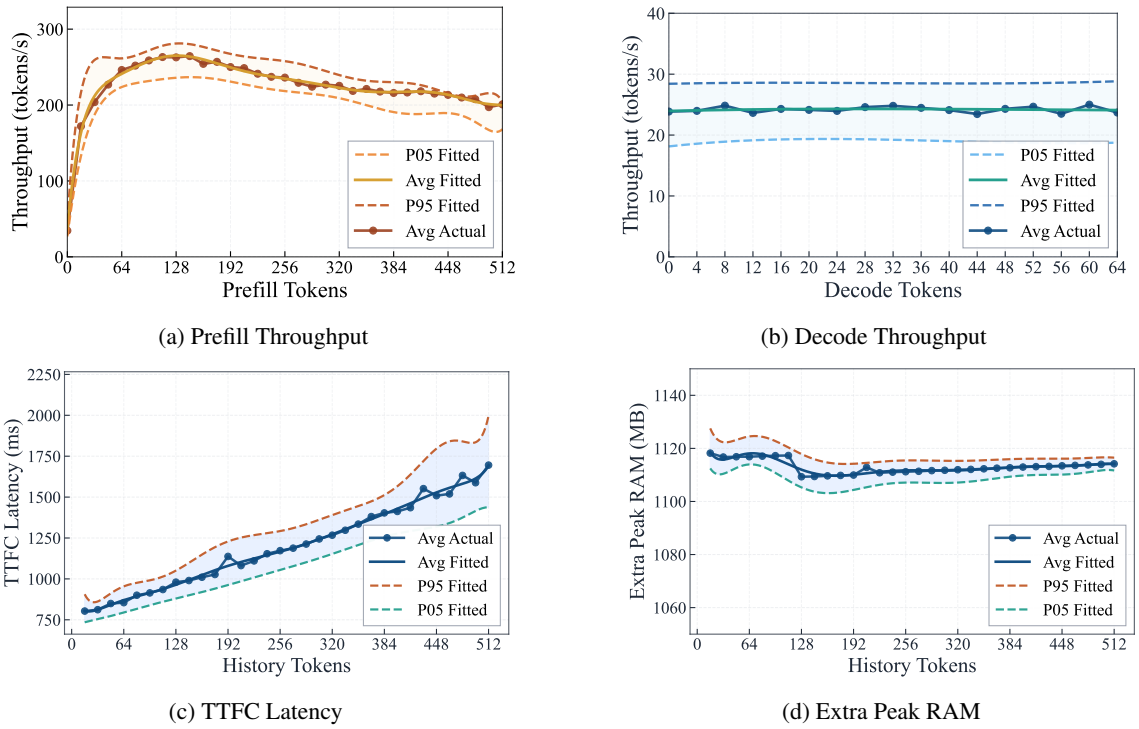
As presented in Table 2, the evaluation demonstrates that, beyond stylized text completion, our model successfully masters memory operations. Delving into the empirical metrics, normal processing successfully extracts factual content into valid, constraint-compliant formats in 96.4% of cases. For the stricter refusal task, the model accurately outputs a `<NO_MEM>` token to discard meaningless context and prevent memory database pollution with a 71.3% success rate. During retrieval, we query a local vector database of 200 memories using the quantized `bge-small-zh-v1.5` model, evaluating exact alignments against gold-truth IDs. By leveraging the Retrieval@4 metric, which directly determines the default four-candidate UI layout of HUOZIIME, we achieve a strong hit rate of 89.5% (179/200). This highlights a core system design advantage: the multi-candidate nature of the IME naturally compensates for the inherent limitations of lightweight, on-device embedding models. Building upon these successfully retrieved cases, manual human inspection confirms an 87.2% (156/179) success rate for grounded generation, demonstrating that HUOZIIME reliably synthesizes retrieved knowledge into stylistically coherent keystroke predictions suitable for real-time user interaction.

### 4.2 On-Device Inference Performance

To ensure a seamless typing experience, we profile the core inference metrics across varying context lengths up to 512 tokens, as shown in Figure 4.

**Throughput & Latency.** Benefiting from our thread-affinity scheduling (§2.4), HUOZIIME achieves a peak prefill throughput of over 260 token/s. During the autoregressive phase, the decode throughput remains remarkably stable at 24–25 tokens/s, which comfortably outpaces standard human reading and typing speeds to ensure fluid *GhostText* rendering. Although the cold-start Time to First Candidate (TTFC) scales linearly from 800 ms to 1700 ms, continuous interactive typing largely avoids this overhead. By leveraging RadixTree-based KV reuse and KV-Splice stitching, HUOZIIME bypass re-prefilling and supports efficient incremental decoding. This amortizes the perceived TTFC to near-zero, preserving the user's cognitive flow.

**Memory Overhead.** To avoid out-of-memory, we quantize the model from 1.34 GB to 485 MB (Q4_0) and bound the L1 KV cache to 24 MB. Fig-




(a) Prefill Throughput

(b) Decode Throughput

(c) TTFC Latency

(d) Extra Peak RAM

Figure 4: On-device inference performance across context lengths up to 512 tokens: (a) prefill throughput, (b) decode throughput, (c) TTFC latency, and (d) extra peak RAM.

ure 4d shows dynamic peak RAM safely stabilizes at $\sim$1.12 GB, ensuring seamless background coexistence with heavy foreground apps.

### 4.3 User Feedback

We invite four participants with NLP backgrounds to engage in free multi-turn conversations using HUOZIIME. Participants report that cross-session recall and context-aware styling noticeably reduce typing friction and improve the overall writing experience, consistent with gains in Keystroke Savings Rate (KSR).

## 5 Related Work

IMEs are initially dominated by transliteration-based methods, with generative paradigms emerging as the field advanced. Based on N-gram models, early IMEs strictly map keystrokes to text sequences. Afterwards, RNN-based IMEs (Chen et al., 2015; Huang et al., 2018; Hard et al., 2018) endow them with certain generative capabilities. However, they inherently lack deep user intent modeling, cross-application context, and true stylistic personalization beyond static lexicons.

Driven by LLMs, the objective shifts from deterministic mapping to next-phrase generation (Zhao et al., 2023; Bubeck et al., 2023; Bai et al., 2023). These systems leverage agentic capabilities, such as memory and tool use, to continuously enhance personalization and generation quality (Packer et al., 2024; Anthropic, 2024). To our knowledge, HUOZIIME is the first generative Chinese IME in our target setting. It addresses the fundamental quality-efficiency-privacy trilemma by deploying a unified post-trained model for both generation and decision-making (Zhao and Song, 2024). Through algorithmic and hardware co-optimization, alongside strictly localized data processing, HUOZIIME provides an exploratory yet robust solution (Hard et al., 2018; Bai et al., 2023).

## 6 Conclusion

We presented HUOZIIME, the first fully on-device, memory-augmented generative Chinese IME. By integrating a persona-adapted LLM, a GRPO-enhanced hierarchical memory, and MCP-based cross-app communication, it transforms static typing into a proactive, personalized experience. Our deeply optimized CPU runtime ensures near-zero latency and a strict small memory footprint. Evaluations confirm HUOZIIME delivers stable on-device deployment and user feedback.

For future work, we will focus on customized pre-training to enforce stronger agentic awareness

and structural formatting. We will also incorporate more advanced edge-side inference optimizations to push the reliability and quality ceilings under extreme hardware constraints.

## Limitations

While HUOZIIME demonstrates the feasibility of an on-device generative IME, several limitations remain. The reasoning capability is inherently bounded by the lightweight model size, leading to occasional agentic instabilities such as over-retrieval and retrieval-token drift under autoregressive sampling. In addition, efficiently and securely acquiring real-time external application context remains challenging due to sandboxing constraints if mobile operating system.

## Ethics Statement

HUOZIIME adopts an on-device deployment paradigm to minimize cloud-side privacy risks and provides transparent local workflows for data collection and deletion. Nevertheless, locally stored user traces may remain vulnerable if a device is compromised, and robust system-level access protection is essential. The system is built upon the Qwen3-0.6B series (Bai et al., 2023) with secondary development of `llama.cpp` and YuyanIME under GPL-3.0. Although the training corpus undergoes automated cleaning and debiasing, residual bias may persist, and user supervision is recommended in practice. Preliminary user feedback is collected from volunteer teammates, with only anonymized identifiers recorded and no personally identifiable information retained.

## Acknowledgments

We gratefully acknowledge the support of the National Natural Science Foundation of China (NSFC) via grant 62236004, 62476073.

## References

Anthropic. 2024. Model Context Protocol (MCP). Official Protocol Specification.

Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, and 29 others. 2023. Qwen technical report. *arXiv preprint*, abs/2309.16609.

Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. 2023. Sparks of artificial general intelligence: Early experiments with GPT-4. *arXiv preprint*, abs/2303.12712.

Shenyuan Chen, Hai Zhao, and Rui Wang. 2015. Neural network language model for Chinese Pinyin input method engine. In *Proceedings of the 29th Pacific Asia Conference on Language, Information and Computation (PACLIC)*, pages 455–461.

Elias Frantar, Saleh Ashkboos, Torsten Hoefler, and Dan Alistarh. 2022. GPTQ: Accurate post-training quantization for generative pre-trained transformers. *arXiv preprint arXiv:2210.17323*.

Georgi Gerganov. 2023. llama.cpp: Port of Meta's LLaMA model in C/C++. GitHub repository.

In Gim, Guojun Chen, Seung seob Lee, Nikhil Sarda, Anurag Khandelwal, and Lin Zhong. 2024. Prompt-Cache: Modular attention reuse for low-latency inference. In *Proceedings of the 7th Conference on Machine Learning and Systems (MLSys)*.

Dan Gusfield. 1997. *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*. Cambridge University Press, Cambridge, UK.

Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. 2018. Federated learning for mobile keyboard prediction. *arXiv preprint*, abs/1811.03604.

Yafang Huang, Zuchao Li, Zhuosheng Zhang, and Hai Zhao. 2018. Moon IME: Neural-based Chinese Pinyin aided input method with customizable association. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 140–145.

Yury A. Malkov and Dmitry A. Yashunin. 2018. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 42(4):824–836.

Charles Packer, Sarah Wooders, Kevin Lin, Vivian Fang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. 2024. MemGPT: Towards LLMs as operating systems. In *Proceedings of the 12th International Conference on Learning Representations (ICLR)*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint*, abs/1707.06347.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. 2024.

DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint*, abs/2402.03300.

Jianlin Su, Yu Lu, Shengfeng Pan, Ahmed Murtadha, Bo Wen, and Yunfeng Liu. 2024. RoFormer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063.

Yuzhuang Xu, Xu Han, Zonghan Yang, Shuo Wang, Qingfu Zhu, Zhiyuan Liu, Weidong Liu, and Wanxiang Che. 2024. Onebit: Towards extremely low-bit large language models. *Advances in Neural Information Processing Systems (NeurIPS)*, pages 66357–66382.

Yuzhuang Xu, Xu Han, Yuanchi Zhang, Yixuan Wang, Yijun Liu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. 2026. CAMERA: Multi-matrix joint compression for MoE models via micro-expert redundancy analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence (AAAI)*, pages 27395–27404.

Yuzhuang Xu, Shiyu Ji, Qingfu Zhu, and Wanxiang Che. 2025. CRVQ: Channel-relaxed vector quantization for extreme compression of LLMs. *Transactions of the Association for Computational Linguistics (TACL)*, 13:1488–1506.

Jingbo Yang, Bairu Hou, Wei Wei, Yujia Bao, and Shiyu Chang. 2025. KVLink: Accelerating large language models via efficient KV cache reuse. *arXiv preprint*, abs/2502.16002.

Jiayi Yao, Hanchen Li, Yuhan Liu, Siddhant Ray, Yihua Cheng, Qizheng Zhang, Kuntai Du, Shan Lu, and Junchen Jiang. 2025. CacheBlend: Fast large language model serving for RAG with cached knowledge fusion. In *Proceedings of the 20th European Conference on Computer Systems (EuroSys)*, pages 1–16.

Guoshenghui Zhao and Eric Song. 2024. Privacy-preserving large language models: Mechanisms, applications, and future directions. *arXiv preprint*, abs/2412.06113.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, Yifan Du, Chen Yang, Yushuo Chen, Zhipeng Chen, Jinhao Jiang, Ruiyang Ren, Yifan Li, Xinyu Tang, Zikang Liu, and 3 others. 2023. A survey of large language models. *arXiv preprint*, abs/2303.18223.

Lianmin Zheng, Liangsheng Yin, Zhiqiang Xie, Chuyue Sun, Jeff Huang, Cody Hao Yu, Shiyi Cao, Christos Kozyrakis, Ion Stoica, Joseph E. Gonzalez, Clark W. Barrett, and Ying Sheng. 2024. SGLang: Efficient execution of structured language model programs. *Advances in Neural Information Processing Systems (NeurIPS)*.

# A Appendix

## A.1 Reward in the GRPO Algorithm

The GRPO algorithm optimizes the trainable policy $\pi_\theta$ by sampling a group of $G$ outputs for each input prompt $x$ from the behavior policy and estimating their relative advantages via in-group reward normalization. The policy is then updated by minimizing a PPO-style clipped surrogate objective (Schulman et al., 2017), regularized by a KL-divergence penalty against the frozen reference policy $\pi_{\text{ref}}$. The core driver of these optimization updates is the rule-based reward score $r(x, y)$, which dictates the advantage estimations for each generation.

The total reward is the sum of a thinking-format term and a task-content term:

$$r(x, y) = r_{\text{think}}(y) + r_{\text{task}}(x, y). \tag{1}$$

where $r_{\text{think}}$ is the thinking-format reward and $r_{\text{task}}$ is the task-content reward. Let $L_{\text{think}}$ be the thought-content length and let $I_{\text{tags}}$ denote tag integrity, we have the following definition:

$$r_{\text{think}}(y) = \begin{cases} +0.2 & \text{if valid think and } 0 < L_{\text{think}} \leq 300, \\ -0.2 & \text{if } L_{\text{think}} > 300 \text{ or no body text}, \\ -0.5 & \text{if } I_{\text{tags}} = \text{Broken}, \\ 0.0 & \text{otherwise.} \end{cases} \tag{2}$$

Task scores are computed by prompt type $T$, i.e., $A, B, C1$ or $C2$. For **normal continuation** ($T = A$), the reward is defined as follows:

$$r_{\text{task}} = \begin{cases} -1.5 & \text{if mem-retrieval tag is emitted,} \\ 1.5 - P_{\text{qual}} & \text{otherwise.} \end{cases} \tag{3}$$

Here $P_{\text{qual}}$ denotes the quality penalty. For **memory retrieval** ($T = B$), the reward is defined as follows:

$$r_{\text{task}} = S_{\text{fmt}} + 0.5 \cdot \mathbb{I}(q \neq \emptyset) - 1.0 \cdot \mathbb{I}(\text{text} \notin \text{tags}), \tag{4}$$

Here $S_{\text{fmt}}$ is the format score and

$$S_{\text{fmt}} \in \{2.5 \text{ (perfect)}, \ 0.3 \text{ (partial)}, \ -1.0 \text{ (wrong)}\}. \tag{5}$$

For **memory extraction** ($T = C1$), the reward is defined as follows:

$$r_{\text{task}} = \begin{cases} -1.5 & \text{if pure NO-MEM is emitted,} \\ -1.0 & \text{if JSON is invalid,} \\ 1.5 + 0.2 \cdot N_{\text{fields}} & \text{if JSON is valid.} \end{cases} \tag{6}$$

The last term adds a richness bonus proportional to $N_{\text{fields}}$. For **extraction refusal** ($T = C2$), the reward is defined as follows:

$$r_{\text{task}} = \begin{cases} 1.5 & \text{if pure NO-MEM is emitted,} \\ -1.0 & \text{if } y \text{ generates JSON,} \\ 0.0 & \text{if noisy NO-MEM appears with extra text.} \end{cases} \tag{7}$$

### A.2 Details of the KV-Splice Algorithm

In Algorithm 1, we present the detailed procedure of the KV-Splice mechanism introduced in Section 2.4. To avoid the high computational overhead of full context re-prefilling when injecting retrieved memory facts ($M$) into the active decoding path, KV-Splice operates by identifying the common prefix ($P$) and suffix ($S$) between the stable base sequence and the memory-augmented sequence. Since RoPE-based models strictly bind KV states to absolute positions, directly inserting new tokens would invalidate the positional encodings of all subsequent tokens. To resolve this, the algorithm calculates a position-offset mapping ($\Delta$) and applies phase-shift correction to the retained suffix KV states. It then selectively recomputes only the newly injected memory tokens and the final tail token to seamlessly preserve causal attention. Furthermore, to ensure inference robustness under extreme mobile constraints, the algorithm incorporates a fallback mechanism that safely reverts to a standard baseline prefill if the resulting KV states are non-consecutive or fail to yield valid candidates.

**Algorithm 1** KV-Splice for Phrase Candidate Generation

**Require:** Stable base sequence $\texttt{seq}_0$, prompt segments $(P, M, S)$, candidate count $K$
**Ensure:** Candidate set $\mathcal{Y}$

$\mathbf{b} \leftarrow \textsc{Tok}(P \| S)$, $\mathbf{f} \leftarrow \textsc{Tok}(P \| M \| S)$
$(\ell_p, \ell_s) \leftarrow \textsc{CommonPrefixSuffix}(\mathbf{b}, \mathbf{f})$
$p_{\text{ins}} \leftarrow \ell_p$
$d \leftarrow |\mathbf{b}| - \ell_p - \ell_s$ ▷ deleted span length in base
$n \leftarrow |\mathbf{f}| - \ell_p - \ell_s$ ▷ inserted span length from memory
$\Delta \leftarrow n - d$
$s_w \leftarrow n_{\text{seq,max}} - 2$, $s_t \leftarrow n_{\text{seq,max}} - 1$
$\textsc{SeqCp}(\texttt{mem}, \texttt{seq}_0, s_w, 0, -1)$
$\textsc{SeqRm}(\texttt{mem}, s_w, p_{\text{ins}}, p_{\text{ins}} + d)$
$\textsc{SeqAdd}(\texttt{mem}, s_w, p_{\text{ins}} + d, -1, \Delta)$
$\textsc{SeqCp}(\texttt{mem}, \texttt{seq}_0, s_t, 0, -1)$
$\textsc{SeqRm}(\texttt{mem}, s_t, p_{\text{ins}}, -1)$ ▷ keep prefix only
**for** $j = 0$ to $n - 1$ **do**
    $x \leftarrow \mathbf{f}[p_{\text{ins}} + j]$
    $\textsc{Decode1}(x, \text{pos} = p_{\text{ins}} + j, \text{seq} = s_t, \text{logits} = 0)$
**end for**
$\textsc{SeqCpOverlay}(\texttt{mem}, s_t, s_w, p_{\text{ins}}, p_{\text{ins}} + n)$
$p_{\text{last}} \leftarrow |\mathbf{f}| - 1$, $x_{\text{last}} \leftarrow \mathbf{f}[p_{\text{last}}]$
$\textsc{SeqRm}(\texttt{mem}, s_w, p_{\text{last}}, p_{\text{last}} + 1)$
$\textsc{Decode1}(x_{\text{last}}, \text{pos} = p_{\text{last}}, \text{seq} = s_w, \text{logits} = 1)$
$\mathcal{Y} \leftarrow \textsc{SampleCandidates}(s_w, K)$
**if** $\mathcal{Y} = \varnothing$ **or** $\neg\textsc{PosConsecutive}(s_w)$ **then**
    $\mathcal{Y} \leftarrow \textsc{Baseline}(P, M, S, K)$
**end if**
**return** $\mathcal{Y}$