Personalized Question Answering with User Profile Generation and Compression

Hang Su, Yun Yang, Tianyang Liu, Xin Liu, Peng Pu, Xuesong Lu*
East China Normal University, Shanghai, China
{hsu_ete, smally, tianyangliu}@stu.ecnu.edu.cn,
{xliu, ppu, xslu}@dase.ecnu.edu.cn

Abstract

Large language models (LLMs) offer a novel and convenient avenue for humans to acquire knowledge. However, LLMs are prone to providing "midguy" answers regardless of users' knowledge background, thereby failing to meet each user's personalized needs. To tackle the problem, we propose to generate personalized answers with LLMs based on users' past question-answering records. We dynamically generate and update a user's domain and global profiles as the user asks questions, and use the latest profile as the context to generate the answer for a newly-asked question. To save tokens, we propose to compress the domain profile into a set of keywords and use the keywords to prompt LLMs. We theoretically analyze the effectiveness of the compression strategy. Experimental results show that our method can generate more personalized answers than comparative methods. The code and dataset are available at https://github.com/DaSESmartEdu/PQA.

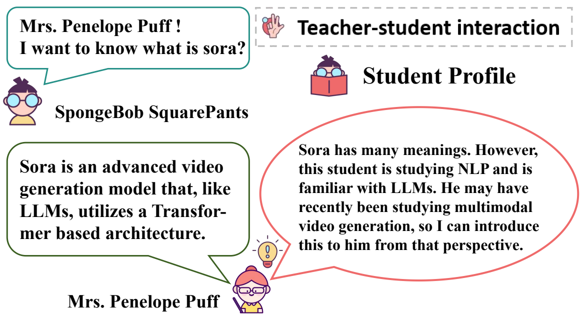

Figure 1: A motivation example. When the student asks "What is Sora?", the teacher gives a personalized answer based on the student's knowledge background.

1 Introduction

The rapid advancement of large language models (LLMs) (Wu et al., 2023; Achiam et al., 2023; Touvron et al., 2023; Dubey et al., 2024; Yang et al., 2024) and the maturation of associated technologies such as RAG (Qi et al., 2024; Wang et al., 2024; Kim and Lee, 2024) and Agent (Zong et al., 2024; Zhao et al., 2024a; Kim et al., 2024) have established LLMs as a pivotal avenue of knowledge acquisition. However, existing QA systems based on LLMs lack personalization, or simply retrieve the QA history to facilitate multi-round dialogue. Under such circumstances, LLMs often fail to generate responses that match users' background knowledge.

In this work, we investigate the task of *personalized question answering with LLMs*, which attempts to respond to a user's question based on her background knowledge, thereby generating a tailored answer. Existing studies have discussed different approaches to adapting LLMs to various personalized tasks, among which two types of approaches are generally used. The first type of approaches fine-tunes the LLMs (Dai et al., 2024; Liu et al., 2024; Tan et al., 2024; Zhuang et al., 2024) using users' personal knowledge, which is resource-intensive and impractical to fine-tune for each user. The second type of approaches introduces users' historical information into prompts to obtain personalized responses from LLMs (Baek et al., 2024; Zhong et al., 2024; Richardson et al., 2023). However, users' knowledge in a specific domain is hard to retrieve from fragmented information, thereby resulting in incorrect prompts.

To tackle the above challenges, we are inspired by daily question-answering between students and teachers. As illustrated in Figure 1, when a student asks a question, the teacher assesses the student's domain knowledge pertaining to the question and gives an answer that is particularly tailored for the student. Motivated by this, we maintain domain and global profiles for each user by leveraging the user's historical QA records, which represent the user's background knowledge. When the user asks a new question, we retrieve from the profiles the

* Corresponding author.

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 4744–4763
November 4-9, 2025 ©2025 Association for Computational Linguistics

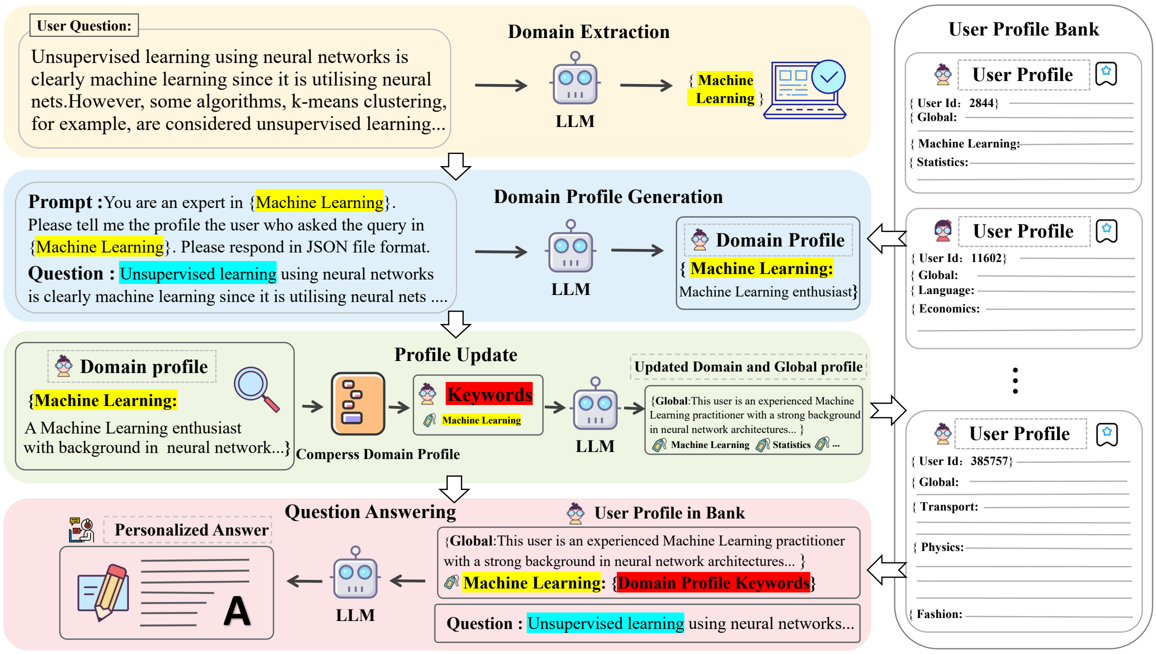

Figure 2: Overview of our method.

relevant background knowledge as the context to make an LLM generate a personalized answer. Particularly, the challenge is to maintain a precise and concise profile, so that the retrieved background knowledge does not consume too many additional tokens. Inspired by the recent studies on prompt compression (Jiang et al., 2023, 2024; Pan et al., 2024), we propose to compress the domain profile into a set of keywords and use the compressed profile to guide personalized question-answering. We theoretically analyze the effectiveness of the compression strategy. We conduct experiments on three QA datasets to show that our proposed method outperforms comparative methods in terms of personalization.

Our contribution can be summarized as follows:

- We propose to leverage historical QA records and maintain user profiles for personalized question-answering. When a user asks a new question, the profiles are retrieved as context to prompt an LLM to generate the answer, thereby achieving personalization.
- We propose to compress the domain profile into keywords and use the keywords as the context for prompting. The strategy saves lots of tokens, thereby making our method efficient. We provide a theoretical analysis of the effectiveness of the compression strategy.
- We conduct extensive experiments to demonstrate the performance of our method. The results show that our method outperforms comparative methods. We also provide deeper analysis through ablation study, hyperparameter tuning and case study.

2 The Methodology

Figure 2 shows the overview of our proposed method, which consists of four steps, namely, domain extraction (Section 2.1), domain profile generation (Section 2.1), profile update and question answering (Section 2.2). In addition, we optimize the process with domain profile compression (Section 2.3). We also theoretically analyze why the compressed profile can replace the domain profile and maintain the effectiveness of the methodology (Section 2.4).

2.1 Domain Extraction and Domain Profile Generation

For each new question asked by a user, we prompt an LLM to extract the domain of the question, such as machine learning, philosophy and economics. We do not restrict the category of domains and purely let the LLM decide the domain. Consequently, the formulation can be expressed as:

$$d = \mathcal{M}(Q; \theta), \tag{1}$$

Where Q denotes the input question, d represents the predicted domain of Q, $\mathcal{M}$ denotes the LLM and θ denotes the model parameters.

Then, we ask the LLM to generate a **domain profile** of the user pertaining to d based on the question, which represents the user's background knowledge about domain d:

$$p_d = \mathcal{M}(Q, d; \theta), \tag{2}$$

where p_d denotes the domain profile of the user pertaining to domain d. Note that we do not include the answer for domain profile generation based on two considerations. First, the answer is usually very long, which may cause performance degradation. Second, there might be noisy information in the answer, which would introduce interfering information in the profile. Due to the page limit, we show the prompts corresponding to Equation 1 and Equation 2 in Figure A1 and Figure A2, respectively, in the appendix. Particularly, we ask the LLM to give the reason for the response to improve the quality of the generated domain profile. The resulting domain profile is a domain name paired with the user's profile pertaining to the domain.

2.2 Profile Update and Question Answering

Once p_d is generated, we ask the LLM to combine it with the old p_d and obtain the updated p_d of the user. The prompt is shown in Figure A3. Note that if p_d is the first profile of domain d of the user, the old p_d is an empty string.

To enable the LLM to answer the question based on the user's overall knowledge, we maintain a **global profile** p_g of the user, which is a more concise description of the user's expertise across multiple domains. We iteratively update p_g every time a p_d is updated. The step can be formally expressed as:

$$p_g = \mathcal{M}(p_d, p_g; \theta), \tag{3}$$

where the p_g on the right is the old global profile of the user. Note that p_g is an empty string initially. The prompt for global profile update is shown in Figure A4 in the appendix.

Finally, we ask the LLM to answer the question based on both the domain profile p_d and the global profile p_g of the user. This strategy enables the LLM to generate the answer based not only on the user's background knowledge of domain d, but also on the user's overall knowledge. Formally, the answer A to the input question Q is generated as:

$$A = \mathcal{M}(Q, p_d, p_g; \theta), \tag{4}$$

The prompt is depicted in Figure A5 in the appendix.

2.3 Domain Profile Compression

The above process consumes a lot of tokens when updating the global profile (Equation 3) and answering the question (Equation 4), as they use both the domain profile and the global profile as input. Although personalization may be achieved by this method, the large token consumption makes it unaffordable for real-world deployment. To save tokens, we are inspired by prompt compression (Pan et al., 2024) and train a normal model to compress the domain profile into a set of keywords. Then, we use the keywords instead of the domain profile to update the global profile and answer the question.

Dataset Preparation. We construct a dataset to train the compression model, where each data point is a pair of a domain profile and a set of corresponding keywords. First, we generate the domain profiles. To include as many domains as possible, we choose StackExchange (Stackexchange, 2014) which contains 172 knowledge domains such as economy, history and biology. For each domain, we randomly select 100 questions or select all the questions if it has fewer than 100. This results in 11,655 questions. After removing questions containing hyperlinks, we employ Qwen2.5-72B (Yang et al., 2024) to generate a domain profile for each question in both English and Chinese, using the prompt in Figure A2. We discard the responses failing to follow the profile generation instruction and obtain 7,840 domain profiles.

Second, we compress each domain profile into a set of keywords. We follow the method in (Pan et al., 2024) and use Qwen2.5-72B to generate the keywords. The compression prompt is depicted in Figure A6 of Appendix C. After obtaining the keywords, we label each word in the domain profile to indicate whether it is a keyword. Sometimes Qwen2.5-72B generates the keywords that are not in the domain profile. To ensure the data quality, we calculate the **Variation Rate (VR)** score for each pair of the domain profile and the keywords, which measures the proportion of words in the keyword set that are absent in the domain profile:

$$VR = \frac{|ks - (p_d \cap ks)|}{|ks|}, \tag{5}$$

where ks denotes the keyword set and $|\cdot|$ denotes the cardinality of a set. The lower the VR score, the higher the quality of the data pair. Therefore, we select the 90% data pairs with the lowest VR scores, which include 6,732 pairs. We refer to as **StackExchange-Cmpres** the constructed dataset.

Model Training. We utilize XLM-RoBERT (Conneau et al., 2020) as the feature encoder f_ϕ of the compression model, on top of which we add a linear classification layer. Given a domain profile containing N words $p_d = \{x_i\}_{i=1}^N$, the compression process can be formulated as:

$$H = f_\phi(p_d), \tag{6}$$

$$\hat{Y} = softmax(WH + B), \tag{7}$$

where $H = \{h_i\}_{i=1}^N$ denotes the feature vectors, $\hat{Y} \in \mathbb{R}^{2\times N}$ denotes the probability distribution of labels $\{preserve, discard\}$ for each word in p_d, and $\{\phi, W, B\}$ represent all the trainable parameters. Denoted by $Y = \{y_i\}_{i=1}^N$ the true labels for the words in p_d, we employ cross entropy loss as the loss function:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N} \text{CrossEntropy}(y_i, \hat{y}_i). \tag{8}$$

2.4 Theoretical Analysis

In this section, we theoretically analyze the effect of the compression strategy. We borrow the idea of self-information (Shannon, 1948) to represent the information carried by each word in the domain profile p_d, that is, the information of x_i in $p_d = \{x_i\}_{i=1}^N$ can be formulated as:

$$I(x_i) = -\log p(x_i), \tag{9}$$

where $p(x_i)$ is the probability of x_i. In natural language, it has been found that the word frequency follows Zipf's law (Piantadosi, 2014), which can be formulated as:

$$p(x_i) \propto rank(x_i)^{-\alpha}, \tag{10}$$

where $rank(x_i)$ denotes the frequency rank of x_i and α is often empirically set to 1. Without loss of generality, we assume that $\{x_i\}_{i=1}^{N'}$ represent the N' unique words in p_d and their frequencies follow Zipf's law, the information of x_i can be proportionally represented as:

$$I(x_i) = -\log rank(x_i)^{-1} = \log rank(x_i), \tag{11}$$

Since the normalization constant does not affect the equation results, we omit it. Again, without loss of generality, we assume that $\{x_i\}_{i=1}^{N'}$ are already ordered by frequency rank, that is, $rank(x_i) = i$. we can obtain the total information of p_d as:

$$I(p_d) = \sum_{i=1}^{N'} p(x_i)I(x_i) = \sum_{i=1}^{N'} \frac{\log i}{i}. \tag{12}$$

In Appendix B.1, we prove that $\sum_{i=1}^{N'} \frac{\log i}{i} \approx \frac{(\log N')^2}{2}$ when N' is sufficiently large.

Suppose that our compression model is perfect and always discards the words with the lowest information first, the information loss ϵ after compression can be approximately computed as:

$$\epsilon = \sum_{i=1}^{n} p(x_i)I(x_i) = \sum_{i=1}^{n} \frac{\log i}{i} \approx \frac{(\log n)^2}{2}, \tag{13}$$

where $\{x_i\}_{i=1}^n$ are discarded by the compression model and $1 \le n \le N'$. The value of n depends on the compression rate of the compression model. As we assume that our model always discards the words with the lowest information first, denoted by τ the compression rate, we can compute it as:

$$\tau = \frac{\sum_{i=1}^n \frac{1}{i}}{\sum_{i=1}^{N'} \frac{1}{i}}. \tag{14}$$

In Appendix B.2, we prove that $\sum_{i=1}^{N'} \frac{1}{i} \approx \log N' + 1/2$ when N' is sufficiently large. As such we can approximately obtain:

$$\tau = \frac{\log n + 1/2}{\log N' + 1/2}, \tag{15}$$

$$\log n = \tau(\log N' + 1/2) - 1/2 = C\tau - 1/2, \tag{16}$$

where $C = \log N' + 1/2$ is a constant. Combining Equation 13 and Equation 16, we can obtain:

$$\epsilon \approx \frac{(C\tau - 1/2)^2}{2} = O(\tau^2). \tag{17}$$

Since $0 \le \tau \le 1$, the information loss ϵ increases less than linear as the compression rate τ increases. In other words, we can retain most information in p_d when τ is not large, thereby providing sufficient information with the keywords for question answering. In Section 4.4, we empirically show how the performance of question-answering varies as τ increases.

3 Experimental Setup

3.1 Datasets

Currently, there is no readily available dataset for personalized question-answering. Hence, we construct three datasets, namely, Wildchat-PerQA, StackExchange-PerQA, and CS101-PerQA, which are collected from Wildchat (Zhao et al., 2024b), StackExchange (Stackexchange, 2014) and a learning platform in our university, respectively. Each

Table 1: Summary statistics of the datasets.

Dataset	#Users	#QA Records
Wildchat-PerQA	100	3,410
StackExchange-PerQA	379	3,500
CS101-PerQA	108	671

dataset contains a group of users and their QA records. The summary statistics of the three datasets are shown in Table 1. Due to page limit, we describe the details of the datasets in Section H.

3.2 Comparative Methods

We compare our method with eight baselines. The base LLMs used for QA are Qwen2.5-7B (Yang et al., 2024) and Llama3.1-8B (Dubey et al., 2024). We implement the comparative methods on both LLMs, which are listed as follows:

- **DirectQA**. The method directly uses a question as input to obtain the answer from LLMs.
- **CoTQA**. The method employs the Chain-of-Thought (CoT) approach to answer questions. The CoT prompt is shown in Figure A7 of Appendix D.
- **RAGQA**. The method encodes users' QA history into vector representations, and retrieves the most semantically similar one as context to answer a question.
- **ProfileQA**. The method follows the process described in Section 2.1 and 2.2, i.e., without domain profile compression.
- **CmpreProfileQA$_{\textbf{BERT}}$**. The method compresses the domain profile using BERT-Base and XLM-RoBERTa trained in (Hu et al., 2023), which are originally trained for compressing meeting content. We refer to the two variants as CmpreProfileQA$^{B}_{BERT}$ and CmpreProfileQA$^{X}_{BERT}$, respectively.
- **CmpresProfileQA$_{\textbf{LLM}}$**. The method compresses the domain profile using an LLM. Following (Jiang et al., 2023), we use Phi-2-2.7B (Javaheripi et al., 2023) and Llama2-7B (Touvron et al., 2023). We refer to the two variants as CmpresProfileQA$^{P}_{LLM}$ and CmpresProfileQA$^{L}_{LLM}$, respectively.
- **CmpresProfileQA$_{\textbf{Tuned}}$**. This is our complete method, which compresses the domain profile using XLM-RoBERTa that is tuned on StackExchange-Cmpres.

3.3 Evaluation Metrics

We employ traditional evaluation metrics such as **BLEU** (Papineni et al., 2002) and **ROUGE** (Lin and Och, 2004). However, they measure the similarity between generated and reference answers, making them unsuitable for assessing personalization. As such, we propose alternative metrics. From an educational perspective, a personalized answer should place unfamiliar concepts within the user's existing cognitive framework. In other words, the content of the answer should have a certain intersection with the questions asked by the user before. Formally, we use the **Jaccard Index** (Jaccard) and **Inclusion Coefficient** (IC) to calculate the intersection degree. Denoted by $\{a_0, a_1 ... a_n\}$ the words in the generated answer and by $\{q_0, q_1 ... q_m\}$ the words in all the questions asked by a user before, the two metrics are calculated as:

$$Jaccard = \frac{|\{a_0, a_1 ... a_n\} \cap \{q_0, q_1 ... q_m\}|}{|\{a_0, a_1 ... a_n\} \cup \{q_0, q_1 ... q_m\}|}, \tag{18}$$

$$IC = \frac{|\{a_0, a_1 ... a_n\} \cap \{q_0, q_1 ... q_m\}|}{|\{q_0, q_1 ... q_m\}|}. \tag{19}$$

For both metrics, the higher the value, the better the performance in terms of personalization.

Furthermore, inspired by (Baek et al., 2024), we propose three human evaluation metrics to comprehensively evaluate answers' quality: (1) **Relatedness** (Rel.)-whether the answer is related to the user's profile and questions; (2) **Correctness** (Corr.)-whether the answer is correct to the question; (3) **Comprehensiveness** (Comp.)-whether the answer is comprehensive to the question. For each metric, we employ a 3-point scale representing agree (2), neutral (1), and disagree (0) to assess the answers. Among them, Relatedness is semantically similar to Jaccard and IC, and can measure whether a generated answer is personalized. We invite five graduate students of the computer science major to evaluate the generated answers based on the metrics. We also instruct GPT-4 to evaluate the answers using the human evaluation metrics.

3.4 Implementation Details

We deploy Qwen2.5-7B (Yang et al., 2024) and Llama3.1-8B (Dubey et al., 2024) using vLLM[1]

[1] https://github.com/vllm-project/vllm

Table 2: The results of automated evaluation using BLEU and ROUGE on Stackexchange-PerQA (%).

Base LLMs	Methods	Evaluation Metrics				
		BLEU1	BLEU2	ROUGE-1	ROUGE-2	ROUGE-L
Llama3.1-8B	DirectQA	32.09	17.60	51.27	18.29	23.66
	CoTQA	32.01	17.51	51.03	17.99	23.11
	RAGQA	31.87	17.66	50.95	17.88	23.22
	ProfileQA	31.63	16.93	50.50	17.73	23.17
	CmpreProfileQA$^{B}_{BERT}$	31.99	17.67	51.42	17.88	23.56
	CmpreProfileQA$^{X}_{BERT}$	31.37	16.85	50.33	17.63	22.79
	CmpresProfileQA$^{P}_{LLM}$	31.66	17.01	50.47	17.78	23.02
	CmpresProfileQA$^{L}_{LLM}$	31.86	17.22	50.74	17.71	23.23
	CmpresProfileQA$_{Tuned}$	**32.35**	**17.77**	**51.56**	**18.71**	**24.12**
Qwen2.5-7B	DirectQA	29.41	15.24	46.53	15.33	20.96
	CoTQA	28.99	14.22	46.33	14.98	20.12
	RAGQA	28.79	15.11	45.99	14.92	19.99
	ProfileQA	28.82	14.49	45.60	14.62	19.96
	CmpreProfileQA$^{B}_{BERT}$	28.76	14.26	45.82	14.59	19.38
	CmpreProfileQA$^{X}_{BERT}$	28.77	14.44	45.64	14.68	20.04
	CmpresProfileQA$^{P}_{LLM}$	28.71	14.05	44.34	14.72	19.19
	CmpresProfileQA$^{L}_{LLM}$	28.52	14.91	44.46	14.00	20.21
	CmpresProfileQA$_{Tuned}$	**30.13**	**15.72**	**46.93**	**15.47**	**21.01**

and LangChain[2] on four NVIDIA A800 80GB GPU. We set the temperature to 0.0 and top$_p$ to 0.9 for both LLMs. The XLM-RoBERTa used for compressing profiles is trained with a learning rate of 1e-5, a batch size of 10, and 10 epochs. At inference, we set the hyperparameter τ of the compression model to 0.6.

4 Experimental Results

4.1 Results of Automated Evaluation

Among the three datasets, only Stackexchange-PerQA has the human answers to the questions, which can be used as a reference. As such, we compare the methods using BLEU and ROUGE on Stackexchange-PerQA. We employ the highest-voted answer to each question as the reference. Table 2 shows the results, where CmpresProfileQA$_{Tuned}$ has the best performance. However, we observe that all methods perform poorly on the metrics because such metrics measure the percentage of word intersection between the generated and reference answers. As such, they are not suitable for measuring the personalization of the generated answers.

As discussed in Section 3.3, we employ Jaccard and IC as proxies of personalization. Table 3 shows the results. CmpreProfileQA$^{B}_{BERT}$ and CmpreProfileQA$^{X}_{BERT}$ have no result on CS101-PerQA because they are trained on English datasets. For each column, we bold the best result and under-

[2] https://github.com/langchain-ai/langchain

Table 3: The results of automated evaluation using Jaccard and IC (%).

Base LLMs	Methods	Dataset					
		Wildchat-PerQA		StackExchange-PerQA		CS101-PerQA	
		Jaccard	IC	Jaccard	IC	Jaccard	IC
Llama3.1-8B	DirectQA	6.64 ± 0.05	8.25 ± 0.03	41.15 ± 0.29	63.76 ± 0.16	17.96 ± 0.04	38.30 ± 0.14
	CoTQA	6.08 ± 0.04	8.11 ± 0.11	40.09 ± 0.13	62.59 ± 0.11	17.56 ± 0.14	38.41 ± 0.15
	RAGQA	6.45 ± 0.08	8.20 ± 0.15	40.99 ± 0.19	63.55 ± 0.18	17.94 ± 0.09	38.21 ± 0.14
	ProfileQA	6.81 ± 0.07	8.74 ± 0.13	$\underline{43.97 \pm 0.02}$	$\underline{70.37 \pm 0.05}$	$\underline{19.10 \pm 0.19}$	$\underline{43.14 \pm 0.19}$
	CmpreProfileQA$^{B}_{BERT}$	6.80 ± 0.03	8.63 ± 0.08	43.65 ± 0.10	69.48 ± 0.24	——	——
	CmpreProfileQA$^{X}_{BERT}$	6.81 ± 0.05	8.64 ± 0.14	43.83 ± 0.22	70.02 ± 0.31	——	——
	CmpresProfileQA$^{P}_{LLM}$	$\underline{6.90 \pm 0.13}$	8.79 ± 0.04	43.36 ± 0.21	69.12 ± 0.27	18.39 ± 0.10	42.48 ± 0.41
	CmpresProfileQA$^{L}_{LLM}$	6.88 ± 0.08	$\underline{8.81 \pm 0.12}$	43.46 ± 0.15	69.55 ± 0.41	18.41 ± 0.04	42.21 ± 0.11
	CmpresProfileQA$_{Tuned}$	$\mathbf{6.99 \pm 0.09}$	$\mathbf{8.98 \pm 0.04}$	$\mathbf{44.50 \pm 0.16}$	$\mathbf{70.90 \pm 0.15}$	$\mathbf{19.30 \pm 0.53}$	$\mathbf{43.35 \pm 0.29}$
Qwen2.5-7B	DirectQA	6.55 ± 0.12	8.61 ± 0.06	40.49 ± 0.16	64.72 ± 0.13	15.68 ± 0.15	42.77 ± 0.14
	CoTQA	5.91 ± 0.10	7.72 ± 0.08	41.10 ± 0.15	64.77 ± 0.12	14.41 ± 0.12	40.56 ± 0.41
	RAGQA	6.41 ± 0.18	8.23 ± 0.11	40.59 ± 0.16	63.54 ± 0.14	15.49 ± 0.09	41.67 ± 0.33
	ProfileQA	$\underline{7.24 \pm 0.10}$	9.78 ± 0.14	42.99 ± 0.27	73.07 ± 0.80	$\underline{16.35 \pm 0.17}$	46.54 ± 0.56
	CmpreProfileQA$^{B}_{BERT}$	7.06 ± 0.08	9.58 ± 0.06	$\underline{43.15 \pm 0.05}$	$\underline{73.27 \pm 0.17}$	——	——
	CmpreProfileQA$^{X}_{BERT}$	7.23 ± 0.05	$\underline{9.79 \pm 0.04}$	42.79 ± 0.26	72.09 ± 0.96	——	——
	CmpresProfileQA$^{P}_{LLM}$	7.15 ± 0.13	9.61 ± 0.09	42.74 ± 0.03	72.42 ± 0.09	16.06 ± 0.26	$\underline{46.63 \pm 0.48}$
	CmpresProfileQA$^{L}_{LLM}$	7.23 ± 0.16	9.75 ± 0.08	42.70 ± 0.10	72.46 ± 0.65	16.18 ± 0.13	46.23 ± 0.61
	CmpresProfileQA$_{Tuned}$	$\mathbf{7.41 \pm 0.16}$	$\mathbf{10.11 \pm 0.07}$	$\mathbf{43.57 \pm 0.07}$	$\mathbf{74.13 \pm 0.21}$	$\mathbf{16.57 \pm 0.11}$	$\mathbf{46.92 \pm 0.45}$

Table 4: The results of human evaluation by volunteers.

Base LLMs	Methods	Dataset								
		Wildchat-PerQA			StackExchange-PerQA			CS101-PerQA		
		Rel.	Corr.	Comp.	Rel.	Corr.	Comp.	Rel.	Corr.	Comp.
Llama3.1-8B	DirectQA	1.52	1.72	1.76	1.50	1.72	1.76	1.52	1.82	1.86
	ProfileQA	1.72	1.78	1.84	1.76	1.78	1.84	1.80	1.88	1.84
	CmpresProfileQA$_{Tuned}$	**1.76**	**1.78**	**1.86**	**1.78**	**1.80**	**1.84**	**1.88**	**1.90**	**1.86**
Qwen2.5-7B	DirectQA	1.58	1.72	1.84	1.58	1.82	1.84	1.62	1.80	1.84
	ProfileQA	1.76	1.78	1.84	1.78	1.78	1.84	1.88	1.80	1.84
	CmpresProfileQA$_{Tuned}$	**1.78**	**1.78**	**1.86**	**1.80**	**1.82**	**1.86**	**1.92**	**1.84**	**1.88**

Table 5: The results of ablation study (%).

Base LLMs	Methods	Dataset					
		Wildchat-PerQA		StackExchange-PerQA		CS101-PerQA	
		Jaccard	IC	Jaccard	IC	Jaccard	IC
Llama3.1-8B	CmpresProfileQA$^{a}_{Tuned}$	6.82	8.73	43.78	69.94	18.81	42.45
	CmpresProfileQA$^{d}_{Tuned}$	6.82	8.73	43.20	68.87	18.42	41.94
	CmpresProfileQA$^{g}_{Tuned}$	6.78	8.67	43.33	69.08	18.41	42.13
	CmpresProfileQA$_{Tuned}$	**6.99**	**8.98**	**44.50**	**70.90**	**19.30**	**43.35**
Qwen2.5-7B	CmpresProfileQA$^{a}_{Tuned}$	7.15	9.85	42.87	72.73	15.80	46.61
	CmpresProfileQA$^{gomain}_{Tuned}$	7.15	9.65	42.87	72.74	15.80	46.61
	CmpresProfileQA$^{g}_{Tuned}$	7.03	9.46	42.68	72.46	15.55	45.87
	CmpresProfileQA$_{Tuned}$	**7.41**	**10.11**	**43.57**	**74.13**	**16.57**	**46.92**

line the second best. We observe that for both base LLMs, our method CmpresProfileQA$_{Tuned}$ performs the best on the three datasets for both metrics. Compared to ProfileQA, which is the variant of our method that does not compress the domain profile, CmpresProfileQA$_{Tuned}$ not only saves tokens but also performs much better. This indicates that removing unimportant content from the domain profile actually improves the representativeness of the profile, thereby eventually promoting the performance. Comparing the compression strategies, the four methods of CmpreProfileQA$_{BERT}$ and CmpresProfileQA$_{LLM}$ perform similarly to or worse than ProfileQA, whereas our proposed method CmpresProfileQA$_{Tuned}$ performs constantly better than ProfileQA. This shows that our compression model can extract profile keywords more precisely.

4.2 Results of Human Evaluation and LLM Evaluation

For human evaluation, we focus on the comparison between CmpresProfileQA$_{Tuned}$ and the second best method ProfileQA, and use DirectQA as the baseline. Table 4 reports the results of human evaluation by volunteers. We observe that for both base LLMs CmpresProfileQA$_{Tuned}$ shows the best performance on all datasets for all metrics, i.e., the volunteers think that CmpresProfileQA$_{Tuned}$ generates more personalized, correct and comprehensive answers than the comparative methods do. The results of LLM evaluation by GPT-4 using the same metrics confirm the observations, which are shown in Table A1 due to page limit.

4.3 Ablation Study

We compare CmpresProfileQA$_{Tuned}$ with three ablation methods. First, we introduce the answer into the context for domain profile generation and refer to the resulting method as CmpresProfileQA$^{a}_{Tuned}$. Second, we only use the domain profile or only the global profile in the prompt for question answering. We refer to the two models as CmpresProfileQA$^{d}_{Tuned}$ and CmpresProfileQA$^{g}_{Tuned}$, respectively.

Table 5 shows the results. We observe that all ablation methods have decreased performance on the three datasets for both metrics, indicating the effectiveness of our specific designs in profile generation and answer prompting.

4.4 The Impacts of τ

We vary the compression rate τ in $\{0, 0.2, 0.4, 0.6, 0.8, 1\}$ and see how it affects the performance of question-answering. When $\tau = 0$, the domain profile is not compressed. When $\tau = 1$, the domain profile is discarded and only the global profile is used for answering questions. We show the results on StackExchange-PerQA in Figure 3 and put those on the other two datasets in Appendix F due to page limit. We observe that Jaccard and IC both gradually increase with τ when $\tau \leq 0.6$ and suddenly decrease when $\tau \geq 0.6$. The observation verifies the analysis in Section 2.4, i.e., the keywords provide sufficient information when τ is not large, thereby maintaining the performance of question-answering. The increased performance for $\tau \leq 0.6$ may indicate that the compression model not only preserves the key information in the domain profile, but also discards the irrelevant words which may negatively impact the performance. Figure A8 and A9 show similar results.

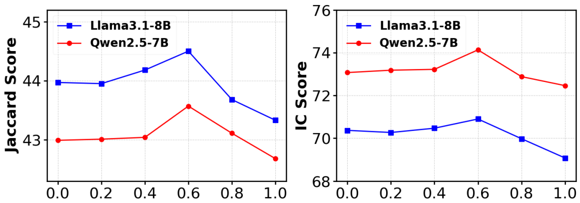

Figure 3: The impacts of τ (StackExchange-PerQA).

4.5 Efficiency Analysis

We show the average execution time per inference (in seconds) of the comparative methods in Table 6. We use vLLM on two NVIDIA A800 80GB GPUs. The LLM used as the compression model is deployed on an additional NVIDIA A800 80GB GPU. We observe that the personalized approaches consume approximately twice the time of the non-personalized approaches. The increase in inference time is justified as a tradeoff for achieving personalization. After reviewing the operational logs, we find that the prefix cache hit rate is approximately 30%, which saves a lot of time for the multi-step inference process. When multiple users ask questions in parallel, it can further leverage the KV cache of vLLM to save more time.

4.6 Case Study

We have deployed CmpresProfileQA$_{Tuned}$ using Qwen2.5-7B in a learning platform[3] of our university for computer general courses. Figure 4 shows the user interface. For a question raised by a student, we simultaneously display the original answer of Qwen2.5-7B (left) and the answer of CmpresProfileQA$_{Tuned}$ (right). In the figure, the student's question is "**what is uv pip?**". Qwen2.5-7B has no background knowledge of the student and provides an answer pertaining to the UV pipette. In contrast, CmpresProfileQA$_{Tuned}$ knows that the student is learning Python programming from his profile and gives the personalized answer. Due to page limits, we put the complete answers and more examples in the Appendix G.

5 Related Work

Personalized Generation with LLMs. Salemi et al. (2023) propose retrieval-augmented methods to personalize LLMs' output based on user profiles. Zhong et al. (2024) use the user's history as a memory component for large language models, achieving personalization through the integration of long-term and short-term memory. Baek et al. (2024) leverage user history to generate personalized search suggestions. Ren et al. (2024) combine representation learning with LLMs to model

[3] https://www.shuishan.net.cn/

Table 6: The results of average execution time per inference (in seconds).

Base LLMs	Methods	Dataset		
		Wildchat-PerQA	StackExchange-PerQA	CS101-PerQA
Llama3.1-8B	DirectQA	4.83	3.42	4.47
	CoTQA	4.94	3.56	4.57
	RAGQA	5.04	3.99	5.12
	ProfileQA	8.95	7.37	8.29
	CmpreProfileQA$^B_{BERT}$	8.83	7.39	8.31
	CmpreProfileQA$^X_{BERT}$	8.92	7.43	8.29
	CmpresProfileQA$^P_{LLM}$	10.98	9.73	11.30
	CmpresProfileQA$^L_{LLM}$	11.04	9.85	11.34
	CmpresProfileQA$_{Tuned}$	8.94	7.42	8.30
Qwen2.5-7B	DirectQA	5.60	4.01	3.69
	CoTQA	5.72	4.13	3.88
	RAGQA	5.79	4.22	3.99
	ProfileQA	10.01	8.36	6.49
	CmpreProfileQA$^B_{BERT}$	10.00	8.32	6.55
	CmpreProfileQA$^X_{BERT}$	10.12	8.46	6.52
	CmpresProfileQA$^P_{LLM}$	12.06	11.45	9.90
	CmpresProfileQA$^L_{LLM}$	12.12	11.49	9.97
	CmpresProfileQA$_{Tuned}$	10.04	8.41	6.52

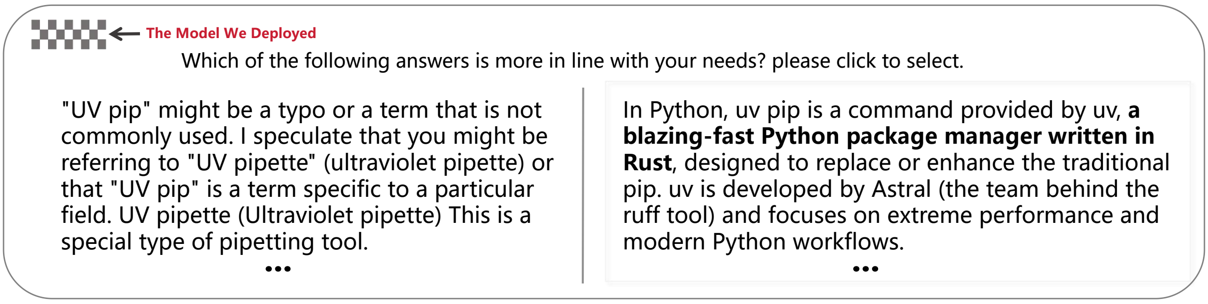

Figure 4: A case study of the deployed CmpresProfileQA$_{Tuned}$ using Qwen2.5-7B.

user preference for recommendation. Li et al. (2024) dynamically modify prompts for personalized text generation by summarizing and rewriting user-specific information. Tan et al. (2024) use user-specific behavioral patterns to adapt LLMs to individuals. No previous study has yet explored personalized question-answering by leveraging historical QA records. As such, the above methods are not suitable for comparison in this study.

Prompt Compression. Prompt compression can be categorized into task-aware compression and task-agnostic compression. Task-aware compression tailors the context to the downstream task. Kim et al. (2022) introduce a token pruning method in the forward pass. Jiang et al. (2024) propose a question-aware compression strategy by estimating token information entropy. Xu et al. (2024) train summarization models for task-specific compression. Task-agnostic compression aims at developing general approaches. Jiang et al. (2023) employ token-level perplexity to filter out insignificant tokens. Pan et al. (2024) finetune a compression model to predict keywords for broader applicability. Inspired by this, we compress domain profiles by predicting keywords.

6 Conclusion

This study aims to generate personalized answers that match users' individual background knowledge. We extract and compress user profiles based on the content of questions asked by users in the past, and use the compressed profiles to prompt LLMs for personalized question answering. We theoretically analyze why the compression strategy is effective. Experimental results show that our method outperforms all comparative methods. In the future, we plan to leverage other user generated data to enhance personalized question-answering, and design more semantic evaluation metrics.

Acknowledgement

This work is supported by the grants from National Natural Science Foundation of China (Grant No. 62277017 and 62137001).

Limitations

We identify two main limitations in the current study. First, the keywords of a user profile are labeled by LLMs. As such, the ability of LLMs to identify keywords determines the capability of the profile compression model, which may in turn affect the performance of personalized QA. In the future, we plan to develop and compare different methods for keyword labeling, and evaluate the impacts on the performance. Second, we propose two new metrics from an educational perspective to evaluate the personalization of the answers. However, one single dimension may not be sufficient to evaluate personalization. In the future, we plan to investigate more theories in knowledge acquisition, and develop more comprehensive evaluation metrics.

Ethics Statement

The main ethical issue may arise from the use of the questions and answers generated by human users and LLMs, which may disclose personal information or contain hallucination and toxic information. If the user profile is built on top of such information, it may lead to privacy issues or prompt LLMs to generate new hallucinations and toxic responses. As such, we manually remove from the datasets all personal information such as usernames, IDs and addresses, as well as hallucination and toxic information.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Jinheon Baek, Nirupama Chandrasekaran, Silviu Cucerzan, Allen Herring, and Sujay Kumar Jauhar. 2024. Knowledge-augmented large language models for personalized contextual query suggestion. In *Proceedings of the ACM on Web Conference 2024*, pages 3355–3366.

Alexis Conneau, Kartikay Khandelwal, Naman Goyal, Vishrav Chaudhary, Guillaume Wenzek, Francisco Guzmán, Edouard Grave, Myle Ott, Luke Zettlemoyer, and Veselin Stoyanov. 2020. Unsupervised cross-lingual representation learning at scale. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8440–8451, Online. Association for Computational Linguistics.

Zhenlong Dai, Chang Yao, WenKang Han, Yuanying Yuanying, Zhipeng Gao, and Jingyuan Chen. 2024. Mpcoder: Multi-user personalized code generator with explicit and implicit style representation learning. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3765–3780.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Daya Guo, Qihao Zhu, Dejian Yang, Zhenda Xie, Kai Dong, Wentao Zhang, Guanting Chen, Xiao Bi, Yu Wu, YK Li, et al. 2024. Deepseek-coder: When the large language model meets programming– the rise of code intelligence. *arXiv preprint arXiv:2401.14196*.

Yebowen Hu, Timothy Ganter, Hanieh Deilamsalehy, Franck Dernoncourt, Hassan Foroosh, and Fei Liu. 2023. Meetingbank: A benchmark dataset for meeting summarization. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16409–16423.

Mojan Javaheripi, Sébastien Bubeck, Marah Abdin, Jyoti Aneja, Sebastien Bubeck, Caio César Teodoro Mendes, Weizhu Chen, Allie Del Giorno, Ronen Eldan, Sivakanth Gopi, et al. 2023. Phi-2: The surprising power of small language models. *Microsoft Research Blog*.

Huiqiang Jiang, Qianhui Wu, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2023. Llmlingua: Compressing prompts for accelerated inference of large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 13358–13376.

Huiqiang Jiang, Qianhui Wu, Xufang Luo, Dongsheng Li, Chin-Yew Lin, Yuqing Yang, and Lili Qiu. 2024. LongLLMLingua: Accelerating and enhancing LLMs in long context scenarios via prompt compression. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1658–1677, Bangkok, Thailand. Association for Computational Linguistics.

Kiseung Kim and Jay-Yoon Lee. 2024. RE-RAG: Improving open-domain QA performance and interpretability with relevance estimator in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 22149–22161, Miami, Florida, USA. Association for Computational Linguistics.

Minsoo Kim, Jongyoon Kim, Jihyuk Kim, and Seung-won Hwang. 2024. QuBE: Question-based belief enhancement for agentic LLM reasoning. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 21403–21423, Miami, Florida, USA. Association for Computational Linguistics.

Sehoon Kim, Sheng Shen, David Thorsley, Amir Gholami, Woosuk Kwon, Joseph Hassoun, and Kurt Keutzer. 2022. Learned token pruning for transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 784–794.

Cheng Li, Mingyang Zhang, Qiaozhu Mei, Weize Kong, and Michael Bendersky. 2024. Learning to rewrite prompts for personalized text generation. In *Proceedings of the ACM on Web Conference 2024*, pages 3367–3378.

Chin-Yew Lin and Franz Josef Och. 2004. Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd annual meeting of the association for computational linguistics (ACL-04)*, pages 605–612.

Jiongnan Liu, Yutao Zhu, Shuting Wang, Xiaochi Wei, Erxue Min, Yu Lu, Shuaiqiang Wang, Dawei Yin, and Zhicheng Dou. 2024. Llms+ persona-plug= personalized llms. *arXiv preprint arXiv:2409.11901*.

Zhuoshi Pan, Qianhui Wu, Huiqiang Jiang, Menglin Xia, Xufang Luo, Jue Zhang, Qingwei Lin, Victor Rühle, Yuqing Yang, Chin-Yew Lin, H. Vicky Zhao, Lili Qiu, and Dongmei Zhang. 2024. LLMLingua-2: Data distillation for efficient and faithful task-agnostic prompt compression. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 963–981. Association for Computational Linguistics.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the*

40th annual meeting of the Association for Computational Linguistics, pages 311–318.

Steven T Piantadosi. 2014. Zipf's word frequency law in natural language: A critical review and future directions. *Psychonomic bulletin & review*, 21:1112–1130.

Jirui Qi, Gabriele Sarti, Raquel Fernández, and Arianna Bisazza. 2024. Model internals-based answer attribution for trustworthy retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6037–6053, Miami, Florida, USA. Association for Computational Linguistics.

Xubin Ren, Wei Wei, Lianghao Xia, Lixin Su, Suqi Cheng, Junfeng Wang, Dawei Yin, and Chao Huang. 2024. Representation learning with large language models for recommendation. In *Proceedings of the ACM on Web Conference 2024*, pages 3464–3475.

Chris Richardson, Yao Zhang, Kellen Gillespie, Sudipta Kar, Arshdeep Singh, Zeynab Raeesy, Omar Zia Khan, and Abhinav Sethy. 2023. Integrating summarization and retrieval for enhanced personalization via large language models. *arXiv preprint arXiv:2310.20081*.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. Lamp: When large language models meet personalization. *arXiv preprint arXiv:2304.11406*.

Claude E Shannon. 1948. A mathematical theory of communication. *The Bell system technical journal*, 27(3):379–423.

Stackexchange. 2014. Stack exchange data dump. https://archive.org/details/stackexchange. Accessed: [Insert Access Date Here].

Zhaoxuan Tan, Qingkai Zeng, Yijun Tian, Zheyuan Liu, Bing Yin, and Meng Jiang. 2024. Democratizing large language models via personalized parameter-efficient fine-tuning. *arXiv preprint arXiv:2402.04401*.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Xiaohua Wang, Zhenghua Wang, Xuan Gao, Feiran Zhang, Yixin Wu, Zhibo Xu, Tianyuan Shi, Zhengyuan Wang, Shizheng Li, Qi Qian, Ruicheng Yin, Changze Lv, Xiaoqing Zheng, and Xuanjing Huang. 2024. Searching for best practices in retrieval-augmented generation. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 17716–17736, Miami, Florida, USA. Association for Computational Linguistics.

Tianyu Wu, Shizhu He, Jingping Liu, Siqi Sun, Kang Liu, Qing-Long Han, and Yang Tang. 2023. A brief overview of chatgpt: The history, status quo and potential future development. *IEEE/CAA Journal of Automatica Sinica*, 10(5):1122–1136.

Fangyuan Xu, Weijia Shi, and Eunsol Choi. 2024. Recomp: Improving retrieval-augmented lms with context compression and selective augmentation. In *The Twelfth International Conference on Learning Representations*.

An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115*.

Jun Zhao, Can Zu, Xu Hao, Yi Lu, Wei He, Yiwen Ding, Tao Gui, Qi Zhang, and Xuanjing Huang. 2024a. LONGAGENT: Achieving question answering for 128k-token-long documents through multi-agent collaboration. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 16310–16324, Miami, Florida, USA. Association for Computational Linguistics.

Wenting Zhao, Xiang Ren, Jack Hessel, Claire Cardie, Yejin Choi, and Yuntian Deng. 2024b. Wildchat: 1m chatgpt interaction logs in the wild. In *The Twelfth International Conference on Learning Representations*.

Wanjun Zhong, Lianghong Guo, Qiqi Gao, He Ye, and Yanlin Wang. 2024. Memorybank: Enhancing large language models with long-term memory. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 19724–19731.

Yuchen Zhuang, Haotian Sun, Yue Yu, Rushi Qiang, Qifan Wang, Chao Zhang, and Bo Dai. 2024. HYDRA: Model factorization framework for black-box LLM personalization. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Chang Zong, Yuchen Yan, Weiming Lu, Jian Shao, Yongfeng Huang, Heng Chang, and Yueting Zhuang. 2024. Triad: A framework leveraging a multi-role LLM-based agent to solve knowledge base question answering. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 1698–1710, Miami, Florida, USA. Association for Computational Linguistics.

A The Prompts Used in Our Method

Figure A1 shows the prompt used for domain extraction. Figure A2 shows the prompt used for domain profile generation. Figure A3 shows the prompt used for domain profile update. Figure A4 shows the prompt used for global profile update. Figure A5 shows the prompt used for answer generation.

> Please tell me which domain of the following question is related to.
> Question: {Question}
> Requirements:
> 1. Please provide your decision in JSON format, following this structure:
> {
> "domain": "A summarization of which domain this query is related to" (if you are unable to summarize it, please set this value to "None"),
> "reasoning": "briefly explain your reasoning for the summarization"
> }
> 2. Please ensure 'domain' is a single word.
> 3. The "reasoning" has no word limits.
> 4. Do not provide any other text outside the JSON string.

Figure A1: Domain extraction.

> You are an expert in {domain}. Please tell me the profile the user who asked the question in {domain}.
> Question: {Question}
> Requirements:
> 1. Please provide your decision in JSON format, following this structure:
> {
> "profile": "A Summary of the user's profile in {domain}" (starting with "This user"),
> "reasoning": "briefly explain your reasoning for the summarization"
> }
> 2. Please ensure that the "profile" is accurate.
> 3. Do not provide any other text outside the JSON string.

Figure A2: Domain profile generation.

> Please generate a new domain profile based on the user's old domain profile and the current domain profile.
> old domain profile: {profile content}
> current domain profile: {profile content}
> Requirements:
> 1. Please provide your decision in JSON format, following this structure:
> {
> "profile": "A Summary of the user's profile in this {domain}" (starting with "This user"),
> "reasoning": "briefly explain your reasoning for the summarization"
> }
> 2. Please ensure that the "profile" is accurate.
> 3. Do not provide any other text outside the JSON string.

Figure A3: Domain profile update.

> Please generate a new global profile based on the user's domain profile and the old global profile. A global profile is the profile of the user in various domains.
> current domain profile: {profile content}
> old global profile: {profile content}
> Requirements:
> 1. Please provide your decision in JSON format, following this structure:
> {
> "profile": "A Summary of the user's profile in various domains" (starting with "This user"),
> "reasoning": "briefly explain your reasoning for the summarization"
> }
> 2. Please ensure that the "profile" is accurate.
> 3. Do not provide any other text outside the JSON string.

Figure A4: Global profile update.

> You are an expert in {domain}. Please answer the Question below. Please make sure your answer fits the user's profile.
> user domain profile: {domain profile}
> user global profile: {global profile}
> Question: {Question}

Figure A5: Answer generation.

B Theoretical Analysis

B.1 Proof one

We prove the asymptotic approximation:

$$\sum_{i=1}^{N'} \frac{\log i}{i} \approx \frac{(\log N')^2}{2} \quad \text{for large } N'. \tag{20}$$

Proof:

For sufficiently large N', we can approximate the sum by an integral:

$$\sum_{i=1}^{N'} \frac{\log i}{i} \approx \int_1^{N'} \frac{\log x}{x}\, dx. \tag{21}$$

To evaluate the integral, we use the substitution $u = \log x$, which gives $du = \frac{1}{x}dx$. The integral becomes:

$$\int_1^{N'} \frac{\log x}{x}\, dx = \int_0^{\log N'} u\, du = \left.\frac{u^2}{2}\right|_0^{\log N'} = \frac{(\log N')^2}{2}. \tag{22}$$

The error terms from approximating the sum by the integral become negligible as $N' \to \infty$. Therefore, we have:

$$\sum_{i=1}^{N'} \frac{\log i}{i} \approx \frac{(\log N')^2}{2} \quad \text{as } N' \to \infty. \tag{23}$$

Q.E.D.

B.2 Proof two

We prove the asymptotic approximation:

$$\sum_{i=1}^{N'} \frac{1}{i} \approx \log N' + 1/2. \tag{24}$$

Proof:

The Euler-Maclaurin formula provides a precise relationship between summation and integration. For $f(x) = \frac{1}{x}$, its first derivative is $f'(x) = -\frac{1}{x^2}$, the second derivative is $f''(x) = \frac{2}{x^3}$, and so on. Applying the Euler-Maclaurin formula (taking the first few terms):

$$\begin{aligned}\sum_{i=1}^{N'} f(i) = \int_1^{N'} f(x)\, dx &+ \frac{f(1)+f(N')}{2} \\ &+ \frac{f'(N')-f'(1)}{12} \\ &+ \cdots + R_k,\end{aligned} \tag{25}$$

where R_k is the remainder term. For $f(x) = \frac{1}{x}$, substituting yields:

$$\begin{aligned}H_{N'} = \log N' &+ \frac{1}{2}\left(1 + \frac{1}{N'}\right) \\ &- \frac{1}{12}\left(-\frac{1}{N'^2} + 1\right) + \cdots.\end{aligned} \tag{26}$$

After Neglecting higher-order small terms (as $N' \to \infty$, $\frac{1}{N'^2} \to 0$), we obtain:

$$\begin{aligned}H_{N'} = \log N' &+ \frac{1}{2} + \frac{1}{2N'} \\ &- \frac{1}{12N'^2} + O\left(\frac{1}{N'^4}\right).\end{aligned} \tag{27}$$

When N' is sufficiently large, the partial sum of the harmonic series asymptotically approaches:

$$\sum_{i=1}^{N'} \frac{1}{i} = \log N' + \frac{1}{2} + o(1). \tag{28}$$

Q.E.D.

C The Instruction to Construct StackExchange-Cmpres

Figure A6 shows the prompt we use with Qwen2.5-75B to generate the dataset for training the compression model.

You are an expert on compressing text into short expressions. I need you to comply with some conditions below:
1. Retain the words that demonstrate the user's level of knowledge.
2. ONLY remove unimportant words. e.g. 'user', 'the', 'are'.
3. Maintain the original order of the words.
4. Use the original words, e.g. 'asking'->'ask' is INCORRECT, 'current'->'now' is INCORRECT.
5. Do not use abbreviations or emojis, e.g. 'without'->'w/o' is INCORRECT, 'as soon as possible'->'ASAP' is INCORRECT.
6. Do not add new words or symbols. This is very important.
For example, 'dedicate 3 hours to each chapter'->'3 hours/chapter' is INCORRECT because you add a new symbol '/'. Just compress it into '3 hours each chapter'. '30 eggs plus 20 eggs equals 50 eggs'->'30+20=50' is also INCORRECT becuase you add new symbols + and =. Just compress it into '30 plus 20 equals 50'.
Compress the original text aggressively by removing words only. Compress the original text as short as you can, while retaining as much information as possible.
If you understand already, please compress the following text:
{text to compress}
The compressed text is:

Figure A6: Domain Profile Compression.

D The CoT Prompt of CoTQA

Figure A7 shows the prompt we use in the CoTQA method.

E Results of LLM Evaluation

Table A1 sbows the results of LLM evaluation by GPT4. We observe that for both base LLMs CmpresProfileQA$_{Tuned}$ shows the best performance on all datasets for all metrics, which coincides with the results in Table 4.

Table A1: The results of LLM evaluation by GPT-4.

Dataset		Wildchat-PerQA			StackExchange-PerQA			CS101-PerQA		
Base LLMs	Methods	Rel.	Corr.	Comp.	Rel.	Corr.	Comp.	Rel.	Corr.	Comp.
Llama3.1-8B	DirectQA	1.81	1.60	1.83	1.91	1.83	1.95	1.83	1.84	1.83
	ProfileQA	1.84	1.72	1.85	1.98	1.83	1.95	1.87	1.86	1.83
	CmpresProfileQA$_{Tuned}$	**1.90**	**1.88**	**1.90**	**1.99**	**1.85**	**1.97**	**1.89**	**1.87**	**1.86**
Qwen2.5-7B	DirectQA	1.70	1.70	1.80	1.91	1.86	1.90	1.80	1.83	1.96
	ProfileQA	1.76	1.81	1.83	1.92	1.86	1.92	1.89	1.82	1.95
	CmpresProfileQA$_{Tuned}$	**1.82**	**1.82**	**1.87**	**1.94**	**1.87**	**1.97**	**1.90**	**1.85**	**1.98**

Let's solve the following problem step by step. Please provide a personalized answer based on the user's knowledge background according to the steps below:
Step 1: Identify the domain of the user's question.
Step 2: Assess the user's level of expertise in that domain based on the question.
Step 3: Provide a personalized answer tailored to the user's knowledge background.
Question: {Question}

Figure A7: The CoT Prompt of CoTQA.

F More Results on the Impacts of τ

Figure A8 and Figure A9 depict the impacts of τ on the question-answering performance for the Wildchat-PerQA and CS101-PerQA datasets. We observe similar results to Figure 3, where τ = 0.6 achieves the best performance.

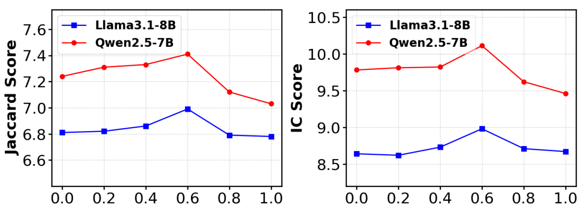

Figure A8: The impacts of τ (WildChat-PerQA).

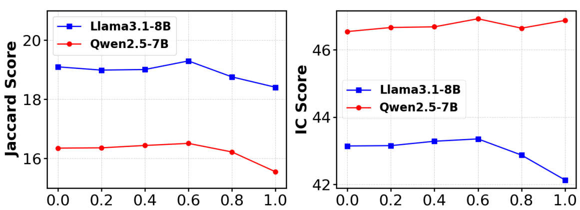

Figure A9: The impacts of τ (CS101-PerQA).

G Additional Case Study

Figure A10 and Figure A11 show the answers to the question "What is UV pip?". Since the student is novice in programming, the UV pipette concept falls outside her knowledge background. The answer in Figure A11 fits more the student's existing expertise. Figure A12 and Figure A13 show the answers to the question "Methods for Iterating Through Arrays". The student has no prior knowledge of JavaScript. As such the answer using C in Figure A13 is more suitable. Figure A14 and Figure A15 show the answers to the question "What is Entropy?". The answer in Figure A15 is more fundamental and therefore fits more the student's background as a beginner in computer science.

H Dataset Construction and the License

Wildchat-PerQA. This dataset constitutes a subset of the "Wildchat" dataset provided by the AllenAI lab (Zhao et al., 2024b). The WildChat dataset comprises 650,000 user interactions with both GPT-3.5-turbo and GPT-4, spanning diverse domains including programming, physics, chemistry, history, and other fields. We perform relevance matching based on users' request headers, language preferences, and regional settings to compile individual question sets for each user, while simultaneously removing toxic content from the sets. Then, we randomly select 100 users with their 3,410 QA records to construct Wildchat-PerQA, where each user asks at least 10 questions.

StackExchange-PerQA. This dataset consists of questions that are asked by the users on Stack Exchange. To cover various disciplines, we randomly sample 100 users in the Machine Learning, Chinese learning, and English learning domains, respectively, and include all users in the psychology domain. The resulting dataset contains 379 users and 3,500 QA records, where each user asks at least 2 questions.

CS101-PerQA. To verify the generalizability of our method in non-English language, we construct CS101-PerQA, which is a Chinese dataset collected from an introductory computer science course of our university in China. The course de-

"UV pip" might be a typo or a term that is not commonly used. I speculate that you might be referring to **"UV pipette"** (ultraviolet pipette) or that "UV pip" is a term specific to a particular field.

1. **UV pipette (Ultraviolet pipette)**
   - This is a special type of pipetting tool, usually used in laboratories to handle chemicals or biological samples that are sensitive to light. Ultraviolet light can be used for sterilization or detection of certain substances. For example, in fluorescence microscopy experiments, a UV pipette might be used to observe fluorescently labeled molecules.
   - For instance, in some biochemistry experiments, researchers may need to operate solutions containing fluorescent dyes under ultraviolet light to ensure the accuracy of the experiment.
2. **Other possible meanings**
   - If "UV pip" is a term from a specific field (such as software, engineering, medicine, etc.), its exact meaning might need more context to be determined accurately. For example, in software development, "pip" is a package management tool for Python, but "UV pip" is not a common combination. It might be the name of a specific project or tool.

Figure A10: Question: What is uv pip? (the answer by Qwen2.5-7B)

ploys DeepSeek-Coder-33B-Instruct (Guo et al., 2024) to answer students' daily questions about the course. The dataset consists of student questions collected from May to July 2024 that received responses marked as 'useful' by the students. In total, CS101-PerQA consists of 108 users and 671 QA records, where each user asks at least 2 questions.

The WildChat-PerQA dataset is derived from WildChat(Zhao et al., 2024b) and is licensed under the ODC-BY agreement. The StackExchange-PerQA is derived from StackExchange and is licensed under the CC BY-SA 4.0. We construct the CS101-PerQA dataset, which will be made available under the CC BY-SA 4.0 license.

`uv pip` is a command provided by `uv`, a blazing-fast Python package manager written in Rust, designed to replace or enhance the traditional `pip`. `uv` is developed by Astral (the team behind the `ruff` tool) and focuses on extreme performance and modern Python workflows.

Purpose of uv pip

1. **High-performance package installation**
   - `uv pip` is significantly faster than traditional `pip` (reportedly 10-100x), especially in dependency resolution and package downloads.
   - Built on Rust, it optimizes dependency resolution and network requests.

2. **Full compatibility with pip commands**
   - Supports all common `pip` commands, such as:

```
uv pip install pandas      # Install a package
uv pip uninstall numpy     # Uninstall a package
uv pip freeze > requirements.txt  # Export dependencies
```

3. **Modern alternative to pip**
   - Can be used as a drop-in replacement for `pip` without changing user habits.
   - Particularly useful for quickly setting up environments or handling large dependency trees (e.g., AI/ML projects).

4. **Replaces pip-tools functionality**
   - `uv pip compile` generates precise dependency lock files (similar to `pip-tools`' `requirements.txt`) but much faster.

Installing uv

```
# Install via pip (requires Python 3.8+)
pip install uv

# Use directly (no virtual environment activation needed)
uv pip install package
```

Why choose uv pip?

- **Speed**: Much faster than `pip` and `pip-tools`, ideal for frequent environment setups/updates.
- **Lightweight**: Rust-based, with no Python environment overhead.
- **Gradual adoption**: Can incrementally replace existing `pip` workflows without forced migration.

Example Comparison

```
# Traditional pip
pip install -r requirements.txt

# Using uv pip (same command, faster execution)
uv pip install -r requirements.txt
```

Notes

- uv is still under active development but stable for most use cases.
- If edge-case issues arise (e.g., certain special packages or private repositories), you can fall back to traditional `pip`.

If you need an ultra-fast package management experience, `uv pip` is a tool worth trying!

Figure A11: Question: What is uv pip? (the answer by CmpresProfileQA$_{Tuned}$)

Methods for Iterating Through Arrays
Iterating through arrays is a fundamental operation in programming. Below are several common methods for traversing arrays:

1. **Using a for loop**

```
const arr = [1, 2, 3, 4, 5];

for (let i = 0; i < arr.length; i++) {
console.log(arr[i]);
}
```

2. **Using a for...of loop (ES6+)**

```
const arr = [1, 2, 3, 4, 5];

for (const item of arr) {
console.log(item);
}
```

3. **Using the forEach method**

```
const arr = [1, 2, 3, 4, 5];

arr.forEach(item => {
console.log(item);
});
```

4. **Using the map method (does not modify the original array)**

```
const arr = [1, 2, 3, 4, 5];

arr.map(item => {
console.log(item);
return item;
});
```

Important Notes

1. If you need the index, the forEach and map methods can also provide it:

```
arr.forEach((item, index) => {
     console.log(`Index ${index}: Value ${item}`);});
```

2. If you need to terminate the loop early, you can use the break statement in for or while loops.
3. In performance-critical scenarios, traditional for loops are usually the fastest.
4. Modern JavaScript also provides other iteration methods such as filter, reduce, etc. Choose the appropriate method based on your requirements.

The choice of method depends on your specific needs, code readability requirements, and whether you need to break out of the loop early.

Figure A12: Question: Methods for Iterating Through Arrays (the answer by Qwen2.5-7B)

Methods for Iterating Through Arrays In C, there are several ways to traverse (iterate through) an array:

1. **Using a for loop with index:**

```
int arr[5] = {1, 2, 3, 4, 5};
int length = sizeof(arr) / sizeof(arr[0]);

for (int i = 0; i < length; i++) {
    printf("%d ", arr[i]);
}
```

2. **Using a while loop with index:**

```
int i = 0;
while (i < length) {
    printf("%d ", arr[i]);
    i++;
}
```

3. **Using a pointer:**

```
int *ptr = arr;
for (int i = 0; i < length; i++) {
    printf("%d ", *ptr);
    ptr++;
}
```

4. **Using pointer arithmetic:**

```
for (int *ptr = arr; ptr < arr + length; ptr++) {
    printf("%d ", *ptr);
}
```

Important notes:

1. Always ensure you don't exceed the array bounds to avoid undefined behavior.
2. The sizeof trick (sizeof(arr)/sizeof(arr[0])) only works for arrays, not pointers.
3. For character arrays (strings), you can also iterate until the null terminator '\0' is encountered.

Would you like me to explain any of these methods in more detail?

Figure A13: Question: Methods for Iterating Through Arrays (the answer by CmpresProfileQA$_{Tuned}$)

What is Entropy? Entropy is a fundamental concept in physics and information theory, describing the degree of disorder or uncertainty in a system. Its meaning varies across disciplines:

1. **Thermodynamic Entropy (Clausius Entropy):**
    - Measures the dispersal of energy and its ability to do useful work.
    - Governed by the Second Law of Thermodynamics: Entropy in an isolated system never decreases.
    - Example: Melting ice increases entropy as molecules become more disordered.
    - Formula: $\Delta S = \frac{Q_{\text{rev}}}{T}$ (where Q_{rev} is reversible heat transfer, T is temperature).
2. **Statistical Mechanics (Boltzmann Entropy):**
    - Links entropy to the number of microscopic states (Ω) a system can occupy.
    - Formula: $S = k_B \ln \Omega$ (k_B is Boltzmann's constant).
    - Example: A gas has higher entropy than a solid because its molecules have more possible arrangements.
3. **Information Theory (Shannon Entropy):**
    - Quantifies uncertainty or information content in a random variable.
    - Formula: $H(X) = -\sum p(x) \log p(x)$ ($p(x)$ is the probability of event x).
    - Example: A fair coin toss has maximum entropy (1 bit); a biased coin has lower entropy.
4. **Entropy in Biology:**
    - Living organisms maintain low local entropy (e.g., ordered cells) by increasing environmental entropy.

Key Intuitions:

- High entropy = Disorder (e.g., mixed gases, randomized data).
- Low entropy = Order (e.g., crystals, predictable signals).

Entropy fundamentally measures the multiplicity of possible states—whether in energy distribution, information encoding, or physical configurations.

Figure A14: Question: What is Entropy? (the answer by Qwen2.5-7B)

What is Entropy? In information theory, **entropy** is a core concept introduced by Claude Shannon in 1948 to measure the uncertainty or randomness of information. It quantifies the unpredictability of a random variable or the "surprise" inherent in its possible outcomes.

1. **Core Definition:**
   - For a discrete random variable X with possible values $\{x_1, x_2, \ldots, x_n\}$ and probability distribution $P(X = x_i) = p_i$, the entropy $H(X)$ is defined as:

   $$H(X) = -\sum_{i=1}^{n} p_i \log_2 p_i$$

   (Typically, base-2 logarithm gives units in **bits**; natural logarithm uses **nats**.)

2. **Intuitive Interpretation:**
   - **Uncertainty:** Higher entropy means greater unpredictability. For example, a fair coin (50% heads) has higher entropy than a biased coin (e.g., 90% heads).
   - **Information Content:** Entropy represents the minimum average number of bits needed to encode outcomes. A fair coin requires 1 bit (0 or 1), while biased coins may need fewer (due to predictable outcomes).

3. **Key Properties:**
   - **Non-negativity:** $H(X) \geq 0$, with equality iff X is deterministic.
   - **Maximum Entropy:** For n outcomes, entropy is maximized when all probabilities are equal ($p_i = 1/n$), yielding $H(X) = \log_2 n$.
   - **Additivity:** For independent X and Y, $H(X, Y) = H(X) + H(Y)$.

4. **Examples:**
   - **Binary Variable (e.g., coin toss):** If the probability of heads is p, entropy is $H = -p \log_2 p - (1 - p) \log_2(1 - p)$. For $p = 0.5$, $H = 1$ bit (maximum uncertainty); for $p = 0$ or 1, $H = 0$.
   - **Weather Forecast:** A 100% chance of rain has $H = 0$ (no uncertainty); a 50% chance has $H = 1$ bit.

5. **Extensions:**
   - **Joint Entropy:** Uncertainty of multiple variables.
   - **Conditional Entropy:** Remaining uncertainty of one variable given another.
   - **Mutual Information:** Shared information between variables.

Applications: Entropy underpins data compression (e.g., ZIP files), cryptography, and machine learning (e.g., decision tree splitting). **Key Insight:** It measures the "cost of unpredictability" in information, whether in encoding, transmission, or analysis.

Figure A15: Question: What is Entropy? (the answer by CmpresProfileQA$_{Tuned}$)