Reasoning-Enhanced Self-Training for Long-Form Personalized Text Generation

Anonymous ACL submission

Abstract

Personalized text generation requires a unique ability of large language models (LLMs) to learn from context that they often do not encounter during their standard training. One way to encourage LLMs to better use personalized context for generating outputs that better align with the user's expectations is to instruct them to reason over the user's past preferences, background knowledge, or writing style. To achieve this, we propose Reasoning-Enhanced Self-Training for Personalized Text Generation (REST-PG), a framework that trains LLMs to reason over personal data during response generation. REST-PG first generates reasoning paths to train the LLM's reasoning abilities and then employs Expectation-Maximization Reinforced Self-Training to iteratively train the LLM based on its own high-reward outputs. We evaluate REST-PG on the LongLaMP benchmark, consisting of four diverse personalized long-form text generation tasks. Our experiments demonstrate that REST-PG achieves significant improvements over state-of-the-art baselines, with an average relative performance gain of 14.5% on the LongLaMP benchmark.

1 Introduction

Personalizing large language models (LLMs) emerges as a critical topic in natural language processing (Salemi et al., 2024b; Kumar et al., 2024), due to its wide-ranging applications in recommender systems (Hua et al., 2023; Chen, 2023), virtual assistants (Li et al., 2024b; Kocaballi et al., 2019), and content generation (Alhafni et al., 2024). The importance of personalization in such systems stems from the fact that they provide targeted content to their users, which enhances user satisfaction, improves engagement, and increases efficiency.

Augmenting the input context of the LLMs with retrieved personalized context alongside the user prompt has proven effective in tailoring responses to individual users (Salemi et al., 2024b,a). However, defining the notion of relevance, a prerequisite for retrieving personalized context, is challenging (Salemi et al., 2024a). In personalization, a part of the user's context that appears not directly "relevant" to the prompt might be more useful (than a directly relevant one) if it better reflects the user's implicit preferences. For example, a sentence like "*I have two children of age 3 and 4...*" in the user context does not seem directly relevant to the prompt "*Give some suggestions about brands of room heaters.*" However, this knowledge indicates that the user could be concerned about safety for children and therefore would expect the model to consider this in its response of recommending room heaters. Establishing such an "implicit" relevance requires reasoning beyond the words or semantics of the provided personalized user context, just like the user themselves does. We argue that an approach for encouraging an LLM to better use personalized context is also asking it to reason over it prior to generating the final response. For instance, the model may summarize the user's writing style, interests, background knowledge, and preferences before actually responding to the user prompt. However, it is often infeasible or costly to obtain sufficient human reasoning paths to train an LLM for personalized reasoning.

This paper addresses these challenges by introducing Reasoning-Enhanced Self-Training for Personalized Text Generation (REST-PG), a multi-stage framework designed to teach LLMs reasoning over personalized context through reinforced self-training. As an alternative to human reasoning paths, REST-PG uses an LLM to generate the reasoning steps considering the input, expected output, and personalized context. These generated reasoning paths are then used to train the LLM, through supervised fine-tuning, to produce both the reasoning steps and the final response in a single inference path. Nevertheless, we find that supervised

fine-tuning on generated reasoning data alone is insufficient for training the LLMs to produce both the reasoning path and final response, and exploring diverse reasoning paths plays a key role in obtaining effective personalized outputs; we observe a drop in performance compared to an LLM without reasoning. This suggests that the reasoning paths generated by the fine-tuned LLMs may not yet align well with the user's preferences. To address this, we employ Expectation-Maximization Reinforced Self-Training, which optimizes the model to generate reasoning paths that yield better aligned responses—-i.e., responses that achieve higher rewards. In an Expectation (E) step, the LLM generates different reasoning paths and responses for each input. In a Maximization (M) step, the reasoning paths that result in high-reward responses—those with high similarity to the expected output for the user—are then used to train the LLM in subsequent iterations. Through iterative process of expectation maximization, the LLM learns to generate reasoning steps and responses that are more aligned with the user's preferences.

We perform our experiments on the Long-form Language Model Personalization (LongLaMP) benchmark (Kumar et al., 2024), comprising four diverse long-form personalized text generation tasks. Experiments on this benchmark show that REST-PG on average significantly outperforms all state-of-the-art baseline models across all tasks of the LongLaMP benchmark. Specifically, REST-PG improves performance by up to 14.5% compared to supervised fine-tuning (SFT) and by 6.5% compared to self-training without reasoning enhancement. Additionally, our extensive ablation study provides valuable insights into various components of the proposed method about self-training and reasoning in personalizing LLMs.

2 Problem Formulation

This paper addresses personalized text generation, a task that uses user-specific information to tailor responses to individual users. A general LLM M_θ generates a piece of text in response to an input prompt x from a user u, denoted as $\hat{y} = M_\theta(x)$. To personalize an LLM for the user u, we assume each prompt x from the user, with the expected output y, is accompanied by the user profile $P_u = \{d_{(u,i)}\}_{i=1}^{|P_u|}$, consisting of unstructured information pieces about the user u. Accordingly, we assume access to training and evaluation data

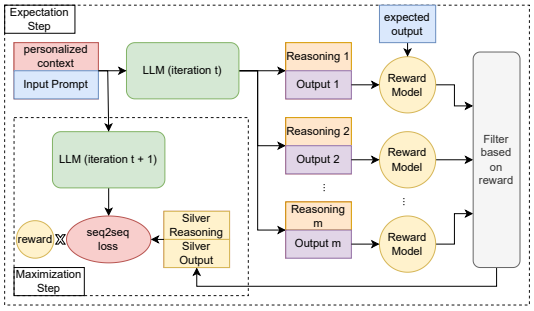

Figure 1: The overview of Reasoning-Enhanced Self-Training for Personalized Text Generation (REST-PG).

$D = \{(x_i, y_i, P_i)\}_{i=1}^{|D|}$ in the above format. Our primary objective is to utilize the personalized information from user profile with LLM M_θ to generate a response $\hat{y} = M_\theta(x, P_u)$ that maximizes the reward $r = \mathcal{R}(x, y, \hat{y})$ generated from a reward function $\mathcal{R}$ given the input prompt x, the expected output y by the user u, and the actually generated response $\hat{y}$. The primary objective of the reward model is to evaluate the similarity between the generated output and the expected personalized output.

3 REST-PG

LLMs have proven effective in learning from their context (Wei et al., 2022; Brown et al., 2020), making the augmentation of their input with personalized context an effective strategy for personalizing their responses (Salemi et al., 2024b; Salemi and Zamani, 2024). However, learning to personalize from context requires a specialized form of context-based learning, as it involves not only understanding task-relevant information but also inferring user-specific preferences. For instance, a sentence in the personalized context that is seemingly irrelevant to the user prompt could indicate implicit preference, like mentioning children could imply prioritizing safety. Teaching LLMs to recognize this nuanced notion of relevance is crucial for improving personalized text generation. One approach to do this is to instruct LLMs to reason over the personalized context by generating a summary of the user's preferences before responding to the prompt. However, collecting training data for this is challenging, as human annotations are costly and often fail to accurately capture the nuances of individual user preferences. To address this, LLMs can be used to generate reasoning paths based on the personalized context, input prompt, and expected output to creating reasoning training data that guides the model from the input to the

output without relying on human annotation. This data can be used to train LLMs to do reasoning during personalized response generation.

While this approach seems effective, the generated reasoning paths are based on the model's implicit understanding of user preferences, which may not always align with the actual preferences. To address this, the LLM can be trained to improve this alignment by optimizing a reward function that evaluates the user's satisfaction by comparing the generated output with the expected output for that user. This alignment pushes the model toward generating reasoning paths that lead to responses more consistent with the user's preferences. This paper focuses on training the LLM to reason over personalized contexts and to generate personalized outputs in a single forward pass.

Figure 1 provides an overview of our optimization approach used in this paper for training an LLM capable of reasoning in a single forward path for personalized text generation. We employ Expectation-Maximization Reinforced Self-Training (Singh et al., 2024) as a preference alignment algorithm to self-train the LLM, enhancing its ability to generate reasoning paths that lead to more effective personalized outputs, according to a reward model that considers the expected personalized output. This enables the model to better leverage user-specific context with improved reasoning ability, ultimately improving the quality of the generated personalized responses.

3.1 Enhancing Personalization by Reasoning

Current state-of-the-art methods for personalizing LLMs augment the input with a personalized context (often retrieved from a personal corpus) (Salemi et al., 2024b,a; Kumar et al., 2024). We argue that effectively utilizing personalized context necessitates a specialized form of context-based learning, as it requires understanding both task-relevant information and user-specific preferences—an aspect that LLMs are rarely exposed to during standard training. One way to encourage LLMs to better utilize personalized context is to instruct them to focus on user-specific elements such as preferences, interests, background knowledge, and writing style that are present in the personalized context. Incorporating these attributes from the personalized context enables the model to generate more aligned, user-specific responses. These attributes can be inferred by the LLM through reasoning over the personalized context, enabling it to interpret the user's preferences, interests, knowledge, and writing style before generating the final personalized response to the user's prompt. This reasoning step helps the model produce more accurate and personalized outputs.

To generate the necessary data for training the LLM to perform such reasoning steps, we introduce a semi-supervised data generation method tailored for this purpose. In this method, for a given input x for user u, the user profile P_u, and the expected output y, we use an LLM[1] to generate a summary of user's preferences, interests, background knowledge, and writing style features tailored to the given input and corresponding output from the user context. The detailed prompt is presented in Figure 7 in Appendix C. This prompt encourages the model to take into account both the expected output and the input, and based on this, generate its interpretation of the user's interests, preferences, and familiarity with various topics from the personalized context as a reasoning path. Additionally, the approach guides the model to infer patterns in the user's preferences across different topics. For instance, if the user writes about a specific topic in a particular style, the model can generalize this pattern, assuming the user might adopt a similar style for other topics as well. Figures 12 and 13 in Appendix E present some examples of the generated reasoning paths. These figures illustrate how the model reasons over the personalized context by analyzing the key aspects of user's preferences.

Finally, to train the LLM to reason over personalized context during output generation, the generated reasoning over personalized context is combined with the expected output using a predefined template, as shown in Figure 8 in Appendix C. This template allows us to train the model to generate this combined output given an input from a specific user accompanied by its personalized context. The model is first asked to generate a summary of the user's preferences and writing style features based on the input and personalized context then generates a response to the input. Here, the combined generated reasoning and expected output are used as the new expected output for the corresponding input in the template. Indeed, the model's task is to generate both the reasoning path, based on the personalized context, and the final response in a single inference pass. This structured approach helps

[1]We utilize Gemma 7B (Gemma-Team, 2024) as the LLM to generate preliminary reasoning data.

Algorithm 1 Reasoning-Enhanced Self-Training for Personalized Text Generation (REST-PG).

1: **Input:** training dataset D, training LLM M_θ, data generation LLM for preference summarization
2: // generating the reasoning data
3: $D_{\text{reasoning}} = \{(x, \text{concat}(reasoning, y), P) | (x, y, P) \sim D : reasoning = \text{LLM}(x, y, P)\}$
4: // SFT on the reasoning dataset
5: $\theta^1 = \arg\max_\theta \mathbb{E}_{(x,y,P)\sim D_{\text{reasoning}}} [\log p_\theta(y|x; P)]$
6: // training the model for T iterations
7: **for** $t = 1$ to T **do**
8: // Expectation step: generating different reasoning paths and outputs to be rewarded
9: $D_t = \{(x, y, P, \hat{y}_j) | (x, y, P) \sim D, \hat{y}_j \sim M_{\theta^t}(x, P) : \mathcal{R}(x, y, \hat{y}_j) \geq \tau\}$
10: // Maximization step: maximizing the probability of the outputs with high reward
11: $\theta^{t+1} = \arg\max_\theta \mathbb{E}_{(x,y,\hat{y},P)\sim D_t} [\mathcal{R}(x, y, \hat{y}) \log p_\theta(\hat{y}|x; P)]$
12: **end for**

the LLM learn to incorporate reasoning over the personalized context as the steps toward generating the final response to the input.

3.2 Reasoning-Enhanced Self-Training

While we can train the model using SFT on the generated reasoning data from Section 3.1 so that it reasons towards generating personalized responses, the reasoning itself is derived from the LLM's interpretation of the user profile, input prompt, and expected output. This reliance on the LLM's implicit understanding introduces potential limitations, as the reasoning path may not fully align with the user's preferences. Moreover, there is no guarantee that the generated reasoning path can consistently improve the final output. There may exist alternative reasoning paths that lead to more effective personalized responses, which are not captured by the initially generated reasoning paths for SFT.

A solution to address this is to employ RL, which allows the model to explore the trajectory space (i.e., reasoning paths) to identify those that lead to personalized outputs with higher rewards. By leveraging exploration, the model can discover reasoning paths that yield higher rewards, corresponding to more desirable personalized outputs. Specifically, we employ Expectation-Maximization Reinforced Self-Training (Singh et al., 2024) as an offline RL algorithm to encourage the model to discover reasoning paths that lead to higher rewards. The algorithm used for this purpose is detailed in Algorithm 1. After performing SFT on the data generated in Section 3.1, we iteratively alternate between the following steps:

Expectation Step: In this step, the optimized parameter set from the previous iteration (i.e., θ^t) is used to collect new trajectories for training the model for the next iteration (i.e., θ^{t+1}). Specifically, for each input $x \in D$, the LLM M_{θ^t} is employed to generate m outputs using a decoding temperature γ. The temperature γ controls the amount of randomness in the generated outputs, which indicates the freedom of the model in the exploration phase of the reinforcement learning algorithm. The generated outputs are then evaluated using the reward model, denoted as $\mathcal{R}(x, y, \hat{y}_j)$. The reward model focuses solely on the final output generated by the model, disregarding the reasoning path itself, and assigns a score to each output. Thus, the reward model only considers the similarity between the generated response and expected output to score the reasoning paths. Finally, the outputs that achieve a reward of τ or higher are considered high quality outputs and are included in the next round of training data, where they act as the expected output for the corresponding inputs. To prevent the model from overfitting on easy examples, we limit the number of outputs retained per input to a maximum of 10 to ensures diverse outputs and avoid overfitting to simpler cases.

Maximization Step: This step uses the dataset generated from the expectation step to optimize the model. In this phase, the outputs that received high rewards are used as the expected outputs for their corresponding inputs. Furthermore, the weight of each output is adjusted according to the reward it receives, as detailed in Algorithm 1 (line 11). Indeed, instead of maximization, a SFT sequence-to-sequence loss (Sutskever et al., 2014) can be minimized to train the LLM,[2] with the loss being

[2]Minimizing seq2seq loss corresponds to maximizing likelihood of generating the ground-truth sequence.

adjusted based on the amount of reward each output receives. The underlying idea is that samples resulting in higher rewards should have a larger impact on the loss. This approach ensures that the model learn more from high-reward examples, helping it generate high-quality, personalized responses.

4 Experiments

4.1 Experimental Setup

Datasets. We adopt the LongLaMP benchmark (Kumar et al., 2024) to conduct our experiments, which consists of four personalized long-form text generation tasks: (1) Personalized Email Completion, (2) Personalized Abstract Generation, (3) Personalized Review Writing, and (4) Personalized Topic Writing. Each example in this dataset represents a separate user, including an input prompt, an expected output, and a user profile containing information about the user (i.e., documents written by the user over time). This setup allows us to evaluate the effectiveness of our approach in generating personalized responses across diverse tasks. More details about the datasets in the LongLaMP benchmark are provided in Appendix B.

Reward Modeling & Evaluation. While the LongLaMP benchmark uses ROUGE metrics (Lin, 2004) for evaluating long-form generated text, previous research shows that term-matching metrics like ROUGE often struggle to capture nuanced text similarities (Zhang et al., 2020), particularly in long-form text generation (Koh et al., 2022; Krishna et al., 2021). Following recent text generation evaluation approaches (Kocmi and Federmann, 2023; Liu et al., 2023b), we use LLMs, in our case Gemma 7B (Gemma-Team, 2024), as the text generation evaluator. We provide the evaluator LLM with the input prompt, the generated output, and the expected output, along with an explaination of the evaluation criteria, as shown in Figure 6 in Appendix A. The LLM then scores the generated personalized response by comparing it to the expected reference output, taking into account the evaluation criteria. These scores range from 1 to 10 in our work, based on the defined criteria. Finally, we normalize this score in range of 0 and 1 by dividing it by 10. The details of the evaluation metric are explained in Appendix A.

Training & Inference Setting. We use Gemma 2B (Gemma-Team, 2024) as the personalized generator LLM. Given that user profiles can contain numerous items, making it impractical to use all of them, we utilize RAG to integrate personalized context (Salemi et al., 2024b). We employ the prompt illustrated in Figure 8 in Appendix C, where we retrieve $k = 5$ items from the user profile using Contriever (Izacard et al., 2022), based on their similarity of the items to the input. Since LLMs have been shown to effectively handle multiple tasks concurrently, we train a single model on all datasets. Following Singh et al. (2024), the models are trained for $T = 3$ iterations, generating $m = 32$ outputs for each input during the expectation step with temperature $\gamma = 0.7$ using Nucleus Sampling (Holtzman et al., 2020), unless otherwise specified. We set the output selection threshold $\tau = 1.0$. At each iteration, we start from a new untrained checkpoint unless otherwise noted. For inference, temperature $\gamma = 0.1$ is used. The details are provided in Appendix B.

4.2 Main Findings

How does training the LLM with REST-PG affect the performance? We trained the LLM using the proposed approach, which incorporates both reasoning-enhancement and self-training. For the baselines, we evaluate LLMs that were: (1) trained using *SFT* with retrieval augmentation (Salemi et al., 2024b; Kumar et al., 2024), (2) trained using *SFT with Reasoning-Enhancement* as described in Section 3.1, and (3) trained exclusively using self-training with *ReST-EM* (Singh et al., 2024). The results, shown in Table 1, indicate that the proposed approach, *REST-PG*, outperforms all baselines across all tasks, with statistically significant improvements in 3 out of 4 tasks. Additionally, the approach shows statistically significant superior performance on average across all tasks. This demonstrates that using reasoning over personalized context, combined with self-training, can significantly enhance the performance of personalized text generation, highlighting the value of incorporating reasoning during personalized generation. The main reason for this improvement is that combining reasoning with self-training enhances the model's ability to effectively use the personalized context and align its reasoning process with the user's preferences. This, in turn, results in more tailored and accurate output for the user.

How does reasoning-enhancement alone affect the performance? We compare the model trained on the reasoning-enhancement data gener-

	Model	LongLaMP-1: Personalized Email Completion	LongLaMP-2: Personalized Abstract Generation	LongLaMP-3: Personalized Review Writing	LongLaMP-4: Personalized Topic Writing	Average (macro)
1	SFT	0.2974	0.4135^{2}	0.6525	0.2270	0.3976
2	SFT w/ Reasoning-Enhancement	0.2834	0.3829	0.6773^{1}	0.2184	0.3905
3	ReST-EM	0.3032	0.4549^{12}	0.6656	0.2859^{12}	0.4274^{12}
	REST-PG	**0.3059**	$\mathbf{0.4845}^{123}$	$\mathbf{0.7077}^{123}$	$\mathbf{0.3238}^{123}$	$\mathbf{0.4554}^{123}$

Table 1: The performance of all methods on the test sets of the LongLaMP benchmark. The superscripts 1, 2, and 3 denote statistically significant improvements compared to the model in the corresponding row using the two-tailed paired t-test ($p < 0.05$). The results on the validation sets are reported in Table 3 in Appendix D.

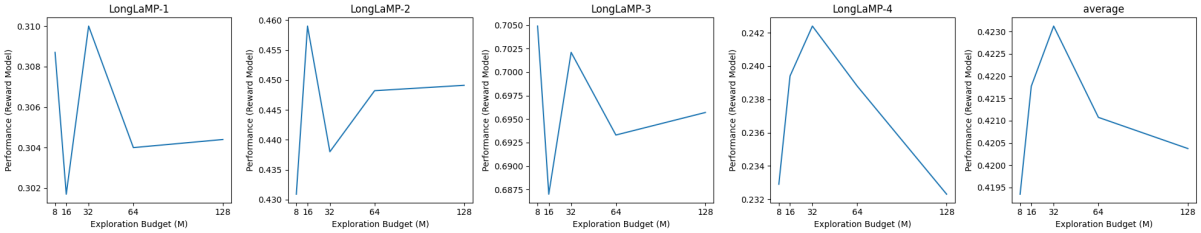

Figure 2: The performance of our approach with different exploration budgets (m) when trained for one iteration on the test set. The same plot on validation sets is depicted in Figure 9 in Appendix D.

ated in Section 3.1 and the SFT model trained on the original inputs and outputs of the LongLaMP dataset. The results of this experiment are reported in Table 1. These indicate that supervised fine-tuning on the generated reasoning-enhancement data from a larger model only statistically significantly improves performance on LongLaMP-3. However, there is a performance drop on the rest of the tasks, with the model performing worse than the SFT on average across all datasets, where on LongLaMP-2 this drop is statistically significant. However, on average, there is no statistically significant difference between this approach and SFT. Note that this approach underperforms compared to both methods that incorporate self-training. This observation suggests that, as discussed in our motivation, training solely on generated reasoning data is suboptimal as there is no alignment between these reasoning paths and the user's preferences for personalized text generation.

How does self-training alone affects the performance? We trained the LLM with *ReST-EM* (Singh et al., 2024), similar to our approach for self-training but without considering reasoning enhancement. This approach operates similarly to ours but does not involve reasoning over the personalized context. The results of this experiment are reported in Table 1 with the model name *ReST-EM*. The results indicate that self-training significantly improves performance on LongLaMP-2 and LongLaMP-4 over both SFT and SFT with Reasoning-Enhancement. Although it improves results on LongLaMP-1 and LongLaMP-3, these improvements are not statistically significant. Moreover, it does not outperform SFT with Reasoning-Enhancement on LongLaMP-3. However, on average, this approach significantly outperforms both baselines. Note that this model is unable to outperform *REST-PG* on any of the tasks, with significant differences in performance observed in 3 out of 4 tasks and in the overall average performance. This observation suggests that self-training is a promising approach for enhancing performance in personalized text generation. However, without explicitly considering the user's implicit preferences or writing style, the improvement on personalized text generation tasks is limited.

How does the exploration budget affect the performance of *REST-PG*? We apply our method using different exploration budgets m during the expectation step, generating 8, 16, 32, 64, and 128 outputs per input and train the LLM for one iteration on them. The results are shown in Figure 2. While different tasks benefit from varying exploration budgets, on average, increasing this exploration budget improves the results up to a certain point before decreasing the performance. This suggests that overly increasing the exploration budget may not be beneficial; as the model generates more examples, the diversity among high-reward examples can negatively impact the model's performance. Therefore, tuning this parameter considerably affects performance.

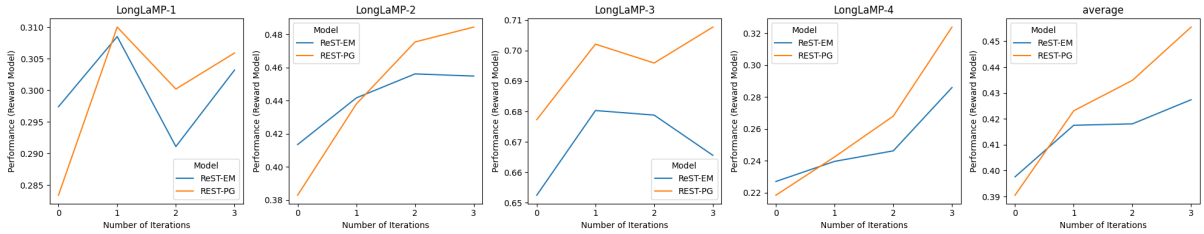

Figure 3: The effect of number of expectation-maximization steps on the performance on the test set. The same plot on validation sets is depicted in Figure 11 in Appendix D.

How does the number of training iterations affect the performance? We vary the number of training iterations for self-training models, *ReST-EM* and *REST-PG*, and evaluate them after each iteration. The results are illustrated in Figure 3. This figure suggests that, on average, increasing the number of iterations leads to improvements in both models. However, the performance gap between the models widens as the number of iterations increases, with *REST-PG* consistently outperforming *ReST-EM*. Additionally, while without any self-training, the *SFT with reasoning-enhancement* performs worse than the *SFT* on 3 out of 4 tasks, after just one iteration of self-training, *REST-PG* surpasses *ReST-EM* in all tasks. This show that while both benefit from more iterations, improvements are more pronounced for *REST-PG*, as additional iterations allow the model to discover more effective reasoning paths, further enhancing its performance.

Is it better to start from a base checkpoint or continue training from the SFT? We train two models using the proposed approach: one starting from the base checkpoint and the other from the SFT checkpoint, which was trained with the data generated in Section 3.1. We plotted the relative performance of these two models after training using our approach in Figure 4. This figure demonstrates that the model starting from the SFT checkpoint underperforms compared to the model starting from the base checkpoint across all tasks, achieving only 96% of the performance of the latter on average. This suggests that starting from a new base model in each iteration is more effective. We believe this is because starting from a base checkpoint allows the model to learn reasoning paths more freely, without being constrained by patterns learned during previous training iteration.

4.3 Case Study

To compare the generated outputs using our approach, we provide two categories of examples.

Improvements in the final generated response. Figure 15 in Appendix E shows an output generated by *REST-PG* and *ReST-EM* for a prompt from the personalized abstract generation dataset. *REST-PG* provides a more precise description of the proposed method and correctly predicts the evaluation dataset, ImageNet, while *ReST-EM* produces a hallucinated and incorrect prediction. This example highlights that *REST-PG* better utilizes the user's personalized context to generate more accurate and personalized response. In this case, *REST-PG*'s correct prediction was guided by the author's previous experiments on the ImageNet dataset.

Improvements in reasoning path toward generating the final response. Figure 14 in Appendix E shows an example of personalized output generated by *REST-PG* and *SFT with Reasoning-Enhancement* for a prompt from the personalized review writing dataset. Here, *SFT with Reasoning-Enhancement* introduced some hallucinated names in the reasoning, which were carried over into the final output. In contrast, *REST-PG* successfully avoided this issue by recognizing that adding inaccurate details negatively affects the reward model's evaluation. Notably, *REST-PG* inferred that the user "*values well-developed characters and relationships*" and incorporated this into the review, aligning closely with the expected output.

5 Related Work

Personalization is an important topic with use cases in search, recommendation, and text generation (Fowler et al., 2015; Xue et al., 2009; Naumov et al., 2019; Salemi et al., 2024b). Salemi et al. (2024b) introduced a Retrieval-Augmented Generation (RAG)-based method for personalizing LLMs and the LaMP benchmark for evaluating short-form personalized text generation. Kumar et al. (2024) extended this by introducing the LongLaMP benchmark for long-form personalized text generation. Another direction has focused on designing person-

alized writing assistants (Li et al., 2023a; Mysore et al., 2023; Lu et al., 2024) and agents (Zhang et al., 2024b). Efforts to personalize LLMs include training retrieval models based on feedback for text generation (Salemi et al., 2024a), optimizing LLMs with personalized feedback (Jang et al., 2023), and automatic personalized prompt generation (Li et al., 2024a). Recent studies have explored parameter-efficient fine-tuning (Tan et al., 2024) and their integration with RAG (Salemi and Zamani, 2024). This paper differs itself by focusing on training LLMs to effectively leverage personalized context and incorporate reasoning into output generation.

Reasoning-Enhancement in LLMs is the model's ability to think step-by-step, also known as chain-of-thoughts (CoT), before responding to prompts. This improves performance of LLMs in complex tasks such as mathematical, logical, and commonsense reasoning (Wei et al., 2024; Liu et al., 2023a; Yin et al., 2024). Additionally, smaller LLMs can acquire this ability through distillation from larger models (Li et al., 2023b). Reasoning-enhancement has not been studied for personalization due to difficulty of understanding user's implicit intent and collecting data to train LLMs for this ability. This paper focuses on training LLMs to achieve this using RL. Concurrently, OpenAI released O1 (OpenAI, 2024), incorporating reasoning into response generation, focusing on math and logical problems.

Self-Training is a new paradigm in which LLMs generate the training data for themselves (Amini et al., 2024). Here, the LLM generates outputs for given inputs, and those that are of high quality, assessed by a reward function, are used to train the model further (Singh et al., 2024; Zelikman et al., 2022). Singh et al. (2024) employ expectation maximization with RL to optimize the model on self-generated outputs, focusing on math and code generation. Similarly, Zelikman et al. (2022) use CoT prompting to generate answers for math and commonsense problems, utilizing only those that lead to correct answers for training the model. Extensions to both approaches includes improved rewarding mechanism (Zhang et al., 2024a) and generating per token rationals. Our work differs from prior studies in key aspects. Previous work focuses on math reasoning and code generation, where multiple-choice or clearly defined correct answers are present. Conversely, free-form personalized generation lacks a definitive correct or incorrect answer because an output might be desirable for one user but not for the others. Additionally, our approach extends the work of Singh et al. (2024) by incorporating reasoning into response generation, allowing for more personalized responses.

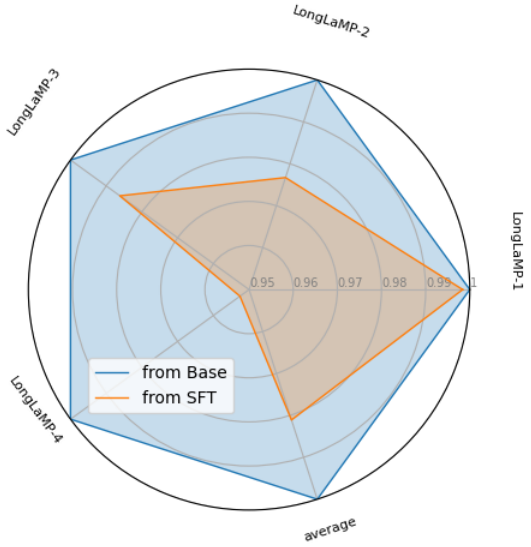

Figure 4: The relative performance of our model trained from the base checkpoint and the SFT checkpoint for one iteration on the test set. The same plot on validation sets is depicted in Figure 10 in Appendix D.

6 Conclusions

This paper proposes REST-PG, a multi-stage framework designed to train LLMs to reason over personalized contexts during response generation. The framework begins by instructing the LLM to generate a reasoning path, based on the input, expected output, and personalized context, outlining how the final output should be derived. This reasoning paths are then used to train the LLM to generate both the reasoning steps and the response in a single inference path, instilling a preliminary reasoning ability in the LLM. Following this, we apply expectation-maximization reinforced self-training to iteratively align the model's reasoning with the user's preferences based on a reward function that evaluates the similarity between the generated response and the expected output for the user. Our results on the LongLaMP benchmark show that our approach significantly outperforms supervised fine-tuning, achieving 14.5% improvement, and it outperforms self-training without reasoning by 6.5% in personalized text generation. Additionally, we conduct a detailed ablation study which provides insights into various aspects of our proposed method.

Limitations

This work has limitations concerning both evaluation and latency of the proposed approaches.

Evaluation of Long-Form Personalized Text Generation. Evaluating personalization in text generation presents inherent challenges, as the ideal judge for the outputs would be the individual who created the inputs (Salemi et al., 2024b). Unfortunately, accessing these original users for existing datasets is often unfeasible. Furthermore, human evaluation remains difficult, as it's not guaranteed that annotators can accurately assess whether the output meets the original prompt writer's expectations. Additionally, as highlighted in previous studies, evaluating long-form text generation is a complex and active area of research in the natural language processing community (Koh et al., 2022; Krishna et al., 2021; Belz and Reiter, 2006). In this paper, we combine these two challenging concepts, which further complicates the evaluation process.

To the best of our knowledge, there is currently no widely accepted metric for evaluating generated personalized outputs. Traditional metrics, such as ROUGE (Lin, 2004) and BLEU (Papineni et al., 2002), which rely on term matching, have proven inadequate for assessing long-form text generation (Koh et al., 2022; Krishna et al., 2021; Belz and Reiter, 2006). Recent efforts in the community have shifted toward utilizing LLMs as evaluators (Li et al., 2024c). Given that we have access to the expected output for each user, we follow the same approach and employ LLMs to assess the similarity between the generated output and the expected output for that specific user. While this evaluation method is not perfect, it represents the most effective approach available within the constraints.

Latency of Reasoning During Response Generation. While incorporating reasoning over personalized context in this paper leads to substantial improvements in the quality of the final generated output, it also introduces a trade-off: an increase in the overall output length. This extended length, when processed by a standard transformer-based LLM, results in a rise in decoding time. This study, however, does not address or attempt to optimize this increased decoding overhead by reasoning-enhancement. While the current focus is on enhancing output quality and personalization, future research could explore strategies to mitigate these computational costs.

Effect of the LLM Family and Size. One limitation of this work is that we conduct our experiments using only the Gemma family of open-source models at the 2B parameter scale. While evaluating the proposed method on additional backbone LLMs of varying sizes could offer further insights into its generalizability, it is very costly and time-consuming to perfrom. Additionally, the primary objective of this paper is to demonstrate the effectiveness of the proposed approach in incorporating reasoning in personalized generation, independent of the specific LLM backbone. Thus, although broader model evaluations could enhance the scope of our findings, this remains a limitation rather than a fundamental shortcoming of this paper.

Comparison with the Reasoning and Thinking LLMs. Recent models designed for reasoning and problem-solving, such as GPT-O1, have demonstrated strong performance on mathematical and logical tasks. In this work, we do not include GPT-O1 in our comparisons. While we acknowledge that large-scale commercial models like GPT-O1 could provide valuable insights into this task, including them as baselines would lead to an unfair comparison. The models we evaluate are significantly smaller (around 2 billion parameters) and fully open-source. Comparing across such a large gap in model scale makes it difficult to isolate the effectiveness of our proposed method. Moreover, since GPT-O1 is a closed-source system, it is unclear what kind of reasoning-specific training it may have received, which further complicates any fair assessment. At the time we conducted our experiments, no open-source models explicitly optimized for reasoning were available. That said, we believe results from models like GPT-O1 could help contextualize our findings, but are not essential to demonstrate the effectiveness of REST-PG.

References

Bashar Alhafni, Vivek Kulkarni, Dhruv Kumar, and Vipul Raheja. 2024. Personalized text generation with fine-grained linguistic control. In *Proceedings of the 1st Workshop on Personalization of Generative AI Systems (PERSONALIZE 2024)*, pages 88–101, St. Julians, Malta. Association for Computational Linguistics.

Massih-Reza Amini, Vasilii Feofanov, Loic Pauletto, Lies Hadjadj, Emilie Devijver, and Yury Maximov. 2024. Self-training: A survey. *Preprint*, arXiv:2202.12040.

Anja Belz and Ehud Reiter. 2006. Comparing automatic and human evaluation of NLG systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 313–320, Trento, Italy. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Junyi Chen. 2023. A survey on large language models for personalized and explainable recommendations. *Preprint*, arXiv:2311.12338.

Andrew Fowler, Kurt Partridge, Ciprian Chelba, Xiaojun Bi, Tom Ouyang, and Shumin Zhai. 2015. Effects of language modeling and its personalization on touchscreen typing performance. In *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems*, CHI '15, page 649–658, New York, NY, USA. Association for Computing Machinery.

Gemma-Team. 2024. Gemma: Open models based on gemini research and technology. *Preprint*, arXiv:2403.08295.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Wenyue Hua, Lei Li, Shuyuan Xu, Li Chen, and Yongfeng Zhang. 2023. Tutorial on large language models for recommendation. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 1281–1283.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Transactions on Machine Learning Research*.

Joel Jang, Seungone Kim, Bill Yuchen Lin, Yizhong Wang, Jack Hessel, Luke Zettlemoyer, Hannaneh Hajishirzi, Yejin Choi, and Prithviraj Ammanabrolu. 2023. Personalized soups: Personalized large language model alignment via post-hoc parameter merging. *Preprint*, arXiv:2310.11564.

Norm Jouppi, George Kurian, Sheng Li, Peter Ma, Rahul Nagarajan, Lifeng Nai, Nishant Patil, Suvinay Subramanian, Andy Swing, Brian Towles, Clifford Young, Xiang Zhou, Zongwei Zhou, and David A Patterson. 2023. Tpu v4: An optically reconfigurable supercomputer for machine learning with hardware support for embeddings. In *Proceedings of the 50th Annual International Symposium on Computer Architecture*, ISCA '23, New York, NY, USA. Association for Computing Machinery.

Ahmet Baki Kocaballi, Shlomo Berkovsky, Juan C Quiroz, Liliana Laranjo, Huong Ly Tong, Dana Rezazadegan, Agustina Briatore, and Enrico Coiera. 2019. The personalization of conversational agents in health care: Systematic review. *J Med Internet Res*, 21(11):e15360.

Tom Kocmi and Christian Federmann. 2023. Large language models are state-of-the-art evaluators of translation quality. *Preprint*, arXiv:2302.14520.

Huan Yee Koh, Jiaxin Ju, He Zhang, Ming Liu, and Shirui Pan. 2022. How far are we from robust long abstractive summarization? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2682–2698, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Kalpesh Krishna, Aurko Roy, and Mohit Iyyer. 2021. Hurdles to progress in long-form question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4940–4957, Online. Association for Computational Linguistics.

Ishita Kumar, Snigdha Viswanathan, Sushrita Yerra, Alireza Salemi, Ryan A. Rossi, Franck Dernoncourt, Hanieh Deilamsalehy, Xiang Chen, Ruiyi Zhang, Shubham Agarwal, Nedim Lipka, and Hamed Zamani. 2024. Longlamp: A benchmark for personalized long-form text generation. *Preprint*, arXiv:2407.11016.

Cheng Li, Mingyang Zhang, Qiaozhu Mei, Weize Kong, and Michael Bendersky. 2024a. Learning to rewrite prompts for personalized text generation. In *Proceedings of the ACM on Web Conference 2024*, WWW '24. ACM.

Cheng Li, Mingyang Zhang, Qiaozhu Mei, Yaqing Wang, Spurthi Amba Hombaiah, Yi Liang, and Michael Bendersky. 2023a. Teach llms to personalize – an approach inspired by writing education. *Preprint*, arXiv:2308.07968.

Hao Li, Chenghao Yang, An Zhang, Yang Deng, Xiang Wang, and Tat-Seng Chua. 2024b. Hello again! llm-powered personalized agent for long-term dialogue. *Preprint*, arXiv:2406.05925.

Liunian Harold Li, Jack Hessel, Youngjae Yu, Xiang Ren, Kai-Wei Chang, and Yejin Choi. 2023b. Symbolic chain-of-thought distillation: Small models can also "think" step-by-step. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages

2665–2679, Toronto, Canada. Association for Computational Linguistics.

Zhen Li, Xiaohan Xu, Tao Shen, Can Xu, Jia-Chen Gu, Yuxuan Lai, Chongyang Tao, and Shuai Ma. 2024c. Leveraging large language models for nlg evaluation: Advances and challenges. *Preprint*, arXiv:2401.07103.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Hanmeng Liu, Zhiyang Teng, Leyang Cui, Chaoli Zhang, Qiji Zhou, and Yue Zhang. 2023a. Logicot: Logical chain-of-thought instruction tuning. In *The 2023 Conference on Empirical Methods in Natural Language Processing*.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023b. G-eval: NLG evaluation using gpt-4 with better human alignment. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 2511–2522, Singapore. Association for Computational Linguistics.

Zhuoran Lu, Sheshera Mysore, Tara Safavi, Jennifer Neville, Longqi Yang, and Mengting Wan. 2024. Corporate communication companion (ccc): An llm-empowered writing assistant for workplace social media. *Preprint*, arXiv:2405.04656.

Sheshera Mysore, Zhuoran Lu, Mengting Wan, Longqi Yang, Steve Menezes, Tina Baghaee, Emmanuel Barajas Gonzalez, Jennifer Neville, and Tara Safavi. 2023. Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. *Preprint*, arXiv:2311.09180.

Maxim Naumov, Dheevatsa Mudigere, Hao-Jun Michael Shi, Jianyu Huang, Narayanan Sundaraman, Jongsoo Park, Xiaodong Wang, Udit Gupta, Carole-Jean Wu, Alisson G. Azzolini, Dmytro Dzhulgakov, Andrey Mallevich, Ilia Cherniavskii, Yinghai Lu, Raghuraman Krishnamoorthi, Ansha Yu, Volodymyr Kondratenko, Stephanie Pereira, Xianjie Chen, Wenlin Chen, Vijay Rao, Bill Jia, Liang Xiong, and Misha Smelyanskiy. 2019. Deep learning recommendation model for personalization and recommendation systems. *Preprint*, arXiv:1906.00091.

OpenAI. 2024. Openai o1 system card.

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 on Association for Computational Linguistics*, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Alireza Salemi, Surya Kallumadi, and Hamed Zamani. 2024a. Optimization methods for personalizing large language models through retrieval augmentation. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 752–762, New York, NY, USA. Association for Computing Machinery.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2024b. LaMP: When large language models meet personalization. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 7370–7392, Bangkok, Thailand. Association for Computational Linguistics.

Alireza Salemi and Hamed Zamani. 2024. Comparing retrieval-augmentation and parameter-efficient fine-tuning for privacy-preserving personalization of large language models. *Preprint*, arXiv:2409.09510.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4603–4611. PMLR.

Avi Singh, John D Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron T Parisi, Abhishek Kumar, Alexander A Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Fathy Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura A Culp, Lechao Xiao, Maxwell Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. 2024. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*. Expert Certification.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. 2024. Personalized pieces: Efficient personalized large language models through collaborative efforts. *Preprint*, arXiv:2406.10471.

Yaqing Wang, Jiepu Jiang, Mingyang Zhang, Cheng Li, Yi Liang, Qiaozhu Mei, and Michael Bendersky. 2023. Automated evaluation of personalized text generation using large language models. *Preprint*, arXiv:2310.11593.

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yo-

gatama, Maarten Bosma, Denny Zhou, Donald Metzler, Ed H. Chi, Tatsunori Hashimoto, Oriol Vinyals, Percy Liang, Jeff Dean, and William Fedus. 2022. Emergent abilities of large language models. *Transactions on Machine Learning Research*. Survey Certification.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2024. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, NIPS '22, Red Hook, NY, USA. Curran Associates Inc.

Gui-Rong Xue, Jie Han, Yong Yu, and Qiang Yang. 2009. User language model for collaborative personalized search. *ACM Trans. Inf. Syst.*, 27(2).

Han Yin, Jianxing Yu, Miaopei Lin, and Shiqi Wang. 2024. Answering spatial commonsense questions based on chain-of-thought reasoning with adaptive complexity. In *Web and Big Data*, pages 186–200, Singapore. Springer Nature Singapore.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah Goodman. 2022. STar: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems*.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Rest-mcts*: Llm self-training via process reward guided tree search. *Preprint*, arXiv:2406.03816.

Kai Zhang, Yangyang Kang, Fubang Zhao, and Xiaozhong Liu. 2024b. LLM-based medical assistant personalization with short- and long-term memory coordination. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 2386–2398, Mexico City, Mexico. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

A Large Language Model Evaluator & Human Evaluation

Although the LongLaMP benchmark (Kumar et al., 2024) primarily relies on ROUGE (Lin, 2004) to assess the quality of long-form text generation, prior studies suggest that lexical overlap metrics often fail to capture semantic similarities (Zhang et al., 2020), especially in long-form generation tasks (Koh et al., 2022; Krishna et al., 2021; Belz and Reiter, 2006). Following the approach proposed by Liu et al. (2023b), we employ an instruction-tuned LLM, Gemma (Gemma-Team, 2024), with 7 billion parameters as our text similarity evaluator. Since this LLM is trained on large instruction tuning datasets, if provided with a well-defined evaluation instruction, it can serve as effective judges for text similarity tasks (Li et al., 2024c).

Following Kocmi and Federmann (2023), to evaluate the generated outputs, we feed the evaluator LLM with the input prompt, the generated text, and the reference output, accompanied by a prompt that explains the evaluation criteria (as depicted in Figure 6). In this prompt, the criteria that determine whether the generated output receives the defined score are clearly outlined. After feeding the model with the prompt (containing the input, expected output, and generated output), the LLM evaluator generates the score for the generated output by comparing it to the reference expected output, considering the conditions defined by the criteria in the prompt. This score is in the rage of 1 to 10. To normalize the score and ensure it falls within the range of 0 to 1, the selected score is divided by 10 (i.e., the maximum score that the LLM evaluator can assign to an output). This normalized score reflects the model's assessment of the generated output based on the predefined criteria. To validate whether the LLM evaluator model can accurately assess the quality of generated texts, we design two experiments.

In the first experiment, we conducted a human evaluation to validate the LLM evaluator. Annotators were presented with 100 pairs of generated texts from the models discussed in this paper. For each pair, the annotators were asked to select the text that best reflected the expected output given the input. The pairs were selected such that there was a score difference of at least 0.5 between the two texts, as determined by our LLM evaluator model. The results of the human evaluation indicate that our metric aligns with human judgment in 73% of the cases. Additionally, the metric shows a correlation of 0.46 with human judgment, suggesting that the LLM evaluator model generally agrees with human assessments. Note that previous studies on designing automatic metrics for personalized text generation have highlighted that such approaches may struggle to achieve very high agreement with human evaluations. This is because personalized text generation is inherently subjective, and only the individual who wrote the input can fully assess whether the generated output meets their expectations or preferences (Wang et al., 2023). Since access to these specific annotators is not possible

for existing datasets, this type of evaluation may not provide a completely reliable measure of the quality of personalized text generation.

To further evaluate the LLM evaluator, we designed an experiment in which the model trained on the LongLaMP benchmark using supervised fine-tuning (as detailed in Section 4) is tested with personalized contexts that are randomly assigned to inputs at varying rates. Specifically, we randomly replaced S percent of the personalized contexts with those from other users, while keeping the input prompt and expected output unchanged. This experiment aims to determine whether the LLM evaluator can detect changes in the personalized context based on the generated text and its comparison with the expected output. The results of this experiment are shown in Figure 5. The figure illustrates that as the rate of random sampling increases, the LLM evaluator linearly assigns lower scores to the texts generated by the same model. This suggests that the LLM evaluator is linearly sensitive to discrepancies in the generated text context from unmatched personalized context with the expected output for the given input.

Therefore, considering both experiments, we believe and are convinced that the LLM evaluator used in this paper is capable of evaluating the quality of generated personalized text when a personalized expected output is provided. These findings demonstrate that the LLM evaluator can effectively align with human judgments and is sensitive to changes in personalized context, supporting its utility for assessing personalized text generation.

B Detailed Experiments Setup

This section outlines the detailed configuration of the experiments conducted in this paper.

Datasets & Tasks. In this paper, we utilize the LongLaMP benchmark (Kumar et al., 2024), publicly accessible benchmark for personalized text generation, to conduct our experiments, which consists of four personalized long-form text generation tasks:

1. Personalized Email Completion: Given an input email, the task is to generate a personalized continuation based on the user's writing style and preferences.

2. Personalized Abstract Generation: This task involves generating personalized abstracts for technical documents or articles given the title and some keywords, reflecting the user's writing patterns and focus areas.

3. Personalized Review Writing: The model generates personalized product reviews that reflect the user's preferences, given the description of the product and the score that is assigned to the product by the user.

4. Personalized Topic Writing: For a post summary on a topic, the task is to generate a personalized long-form full post that reflects the user's writing style, preferences, and opinion on topic.

Each example in the dataset represents a distinct user and includes, an input prompt relevant to the task, an expected output tailored to that specific user, and a user profile containing historical data, such as previously generated texts, to capture the user's writing habits and preferences. We utilize the user-based setting of the LongLaMP benchmark to perform our experiments. The statistics of the datasets are reported in Table 2.

Training Setup. We utilize the Gemma model (Gemma-Team, 2024) with 2 billion parameters as the LLM. To incorporate personalized context, we follow the retrieval-augmented generation approach for personalized text generation, as described in Salemi et al. (2024b), with the prompt shown in Figure 8 in Appendix C. We employ multi-task learning to train a single model across all tasks in the LongLaMP benchmark, allowing the model to generalize and perform well on diverse personalized text generation tasks. We retrieve $k = 5$ items from the user profile using Contriever (Izacard et al., 2022). Following Singh et al. (2024), the models are trained over $T = 3$ iterations, generating $m = 32$ outputs per input during the expectation step, with a decoding temperature of $\gamma = 0.7$ using Nucleus Sampling (Holtzman et al., 2020), unless otherwise specified. We set the output selection threshold to $\tau = 1.0$, and at each iteration, the training begins from a new, untrained checkpoint unless otherwise stated.

For each iteration of training, we use the Adafactor optimizer (Shazeer and Stern, 2018) with a learning rate of 5×10^{-6} and a linear learning rate decay of 0.1, along with 250 warmup steps, for a maximum of 10,000 training steps. The batch size is set to 64, and we apply a weight decay of

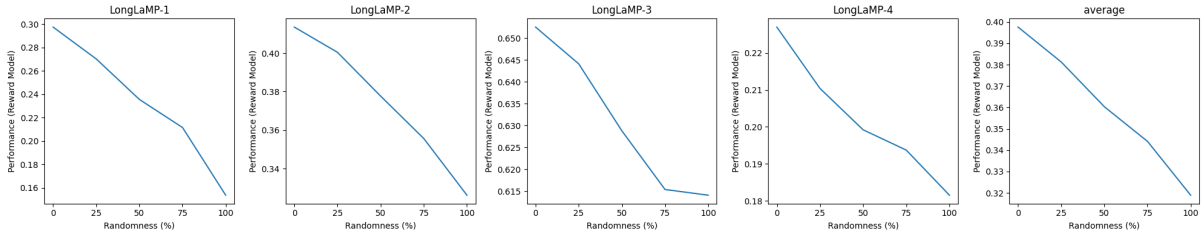

Figure 5: The affect of randomly shuffling profiles on the reward model's scores.

You are a helpful assistant. Please act as an impartial judge and evaluate the quality of the response provided by an AI assistant to the user instruction displayed below. Based on the scoring criteria, given the instruction, please provide a score of the AI assistant's answer compared to the ground truth. Be as objective as possible.

[Scoring Criteria]:

Score 1: The answer is completely unrelated to the reference.
Score 3: The answer has minor relevance but does not align with the reference.
Score 5: The answer has moderate relevance but contains inaccuracies.
Score 7: The answer aligns with the reference but has minor omissions.
Score 10: The answer is completely accurate and aligns perfectly with the reference.

[Instruction]: [the input prompt]
[Ground truth]: [expected output]
[AI assistant's Answer]: [generated output]

[Score]:

Figure 6: The prompt used for reward model to evaluate the generated text based on the input, the reference output, and the provided criteria.

0.01. We also utilize a gradient cliping of 1.0 for optimization. The input length is limited to a maximum of 5,120 tokens, and the output is capped at 1,536 tokens. The experiments are conducted on 64 TPU-v4 (Jouppi et al., 2023) cores, each with 32GB of memory, for a maximum duration of 1 day. All reported results are based on a single run.

Inference Setup. During inference, we limit the input to a maximum of 5,120 tokens and the output to 1536 tokens, where we use nucleus sampling (Holtzman et al., 2020) with a sampling temperature of $\gamma = 0.1$ to produce more deterministic outputs from the LLM. For evaluation, models are assessed using full precision on the entire test dataset. However, during checkpoint validation in the training phase, we randomly sample 1,024 examples from the validation set to evaluate the model and choose the best checkpoint every 1000 steps. Inference is conducted on the same infrastructure and resources used during the training setup.

C Overview of Prompts and Templates

We utilize an instruction-tuned Gemma (Gemma-Team, 2024) LLM with 7 billion parameters to generate the initial reasoning over personalized context data. These reasoning data is used to train the model to develop a preliminary reasoning ability over personalized user context. The prompt used to generate such data is shown in Figure 7. This prompt encourages the model to consider both the final output and the input prompt, and based on this, generate a summary of user's preferences, interests, background knowledge, and writing style features from the user's personalized context that reflects their interests, preferences, and familiarity with various topics. Additionally, the prompt encourages the model to make reasonable inferences about the user's preferences on different topics. For instance, if a user writes in a specific writing style on a particular topic, the model may infer that the user is likely to use a similar approach for other related topics as well.

Finally, to train the LLM to perform reasoning over personalized context during output generation, the generated reasoning data is combined with the expected output using a predefined template, as shown in Figure 8. This template enables the model to generate personalized responses by incorporating reasoning based on the user's preferences. The model is fed with an input consisting of the user's prompt and personalized context. The model is first tasked with generating a summary of user's preferences and writing style features based on the input, which is then followed by generating the final response to the prompt. The combined output—— both the reasoning path and the final response—— serves as the expected output in this template. Essentially, the model is trained to generate both the reasoning steps and the final response in a single inference pass.

D Results on the Validation Sets

This section reports the results of the experiments performed in Section 4 on the validation set of the datasets in the LongLaMP benchmark (Kumar et al., 2024). To accelerate the training phase, we randomly selected at most 1,024 samples from each dataset and evaluated the checkpoints on those samples. Therefore, the results presented here are not

Your role:
You are a professional writing assistant whose task is to summarize the writing style of a user from the profile, which is past documents written by that user. The extracted writing style summary should contain the unique features of users writing style and preferences from the proile that are similar to the expected output.

You task:

Your task is to summarize the user writing style from the profile considering the expected output. From the profile, you may infer the user's interests, preference, familiarity on various topics, etc. While inferring the user's interests, you can make reasonable guesses, e.g. people who are interested in topic A are also likely to be interested in topic B or if they write a sentence in a specific writing style on topic A it is likely they write it with the same style on topic B. As a concrete example, if a user writes "I am interested in action movies" in its past document, this is relevant to "I like to go to cinema" in the expected output. Another example would be if a person prefers specific words or phrases in their writing or using a specific grammar. You can also mention such words that they often use in your summary.

Your input:
- profile: the past documents written by the same person that are separated with | symbol.
- subject: the subject for the expected output
- expected output: the expected output written by the same person as the past documents.

Your output:

a list of bullet points and explanations describing writing style of the user. Also, make sure that you only talk about information from the profile while considering the expected output in writing style summarization. You cannot directly copy or mention anything about the expected output. The expected output is only used to determine the writing style of the user and how profile can affect the expected output.

Examples

profile:
[documents from the user u's profile concatenated with "|" symbol]

subject:
[input prompt x for user u]

expected output:
[expected output y for user u]

Your output:

Figure 7: The prompt used to generate summary of user's preferences, interests, background knowledge, and writing style features as a reasoning method over the personalized context.

Input

The following context is written by a specific user. Please use the following context to generate a personalized response to the instruction. your response should follow the same pattern in terms of preferences and writing style in the provided context. You should first summarize the writing style of the user based on the provided context. Then, you should use the summarized writing style to generate a response to the instruction.

instruction: [input prompt x for user u]

context: [documents from the user u's profile concatenated with "|" symbol]

answer:

output

To summarize the writing style of the user, we can consider the following aspects:

[generated profile writing style summary from Gemma 7B]

Now, considering the style summary, we can generate the final answer: [expected output y for user u]

Figure 8: The input output template used for training the model with reasoning-enhancement data.

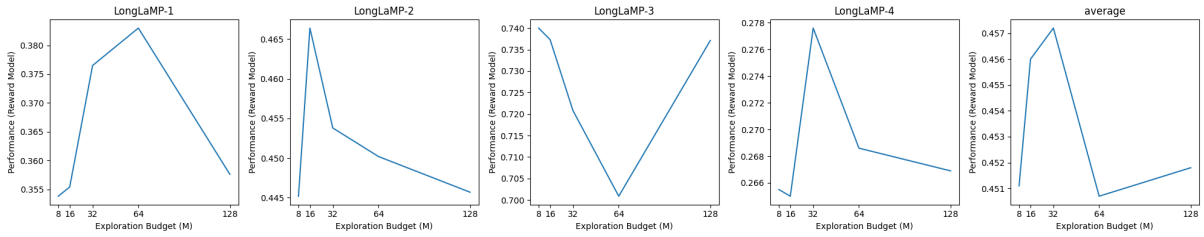

Figure 9: The performance of our approach with different exploration budgets (m) when trained for one iteration on the validation set. In order to speed up the experiments, a maximum of 1,024 samples from each task randomly was selected, instead of evaluating on the full validation set.

Task	#train	#validation	#test	Input Length	Output Length	Profile Size
LongLaMP-1: Personalized Email Completion	3286	958	823	46.45 ± 21.45	92.59 ± 60.68	85.65 ± 51.67
LongLaMP-2: Personalized Abstract Generation	13693	4560	4560	33.82 ± 5.71	144.28 ± 68.40	120.30 ± 118.81
LongLaMP-3: Personalized Review Writing	14745	1826	1822	119.39 ± 73.06	304.54 ± 228.61	34.39 ± 57.31
LongLaMP-4: Personalized Topic Writing	11442	2452	2453	28.36 ± 36.08	263.03 ± 243.34	50.39 ± 2898.60

Table 2: The statistics of the datasets in the LongLaMP benchmark on user-based setting.

	Model	LongLaMP-1: Personalized Email Completion	LongLaMP-2: Personalized Abstract Generation	LongLaMP-3: Personalized Review Writing	LongLaMP-4: Personalized Topic Writing	Average (macro)
1	SFT	0.3672	0.4046	0.6455	0.2293	0.4116
2	SFT w/ Reasoning-Enhancement	0.3426	0.3824	0.7181	0.2495	0.4231
3	ReST-EM	0.3711	0.4550	0.6664	0.2853	0.4444
	REST-PG	**0.3800**	**0.4827**	**0.7197**	**0.3561**	**0.4846**

Table 3: The performance of all methods on the validation sets of the LongLaMP benchmark. In order to speed up the experiments, a maximum of 1,024 samples from each task randomly was selected, instead of evaluating on the full validation set.

	Model	Metric	LongLaMP-1: Personalized Email Completion	LongLaMP-2: Personalized Abstract Generation	LongLaMP-3: Personalized Review Writing	LongLaMP-4: Personalized Topic Writing
1	SFT	ROUGE-1	46.4	43.1	41.3	29.0
		ROUGE-L	41.4	27.5	18.0	15.0
		BLEU	41.6	15.5	7.2	7.0
2	SFT w/ Reasoning-Enhancement	ROUGE-1	44.5	42.1	34.5	28.4
		ROUGE-L	39.7	26.2	16.9	14.1
		BLEU	37.2	14.6	5.2	4.3
3	ReST-EM	ROUGE-1	41.5	43.6	32.4	26.5
		ROUGE-L	37.5	28.3	17.8	14.8
		BLEU	36.7	16.3	6.5	6.3
	REST-PG	ROUGE-1	41.5	43.9	31.3	26.1
		ROUGE-L	37.0	28.4	16.9	14.0
		BLEU	33.3	16.5	4.4	4.1

Table 4: The performance of all methods on the test sets of the LongLaMP benchmark, using traditional term-matching metrics. However, as noted in prior work on evaluating long-form text generation (Koh et al., 2022; Krishna et al., 2021; Belz and Reiter, 2006), standard automatic metrics are not reliable indicators of quality in this setting. For the sake of transparency, we report these metrics, but we do not rely on them for our core evaluation.

based on the entire validation set of the datasets.

The results of baselines and the proposed approaches on the dev set are reported in Table 3. The results in this table suggest that SFT with reasoning-enhancement, unlike on the test set, was able to help the model outperform the SFT model without reasoning. Additionally, we observe that self-training without reasoning using ReST-EM outperforms the SFT baselines, similar to the results on the test set. Finally, REST-PG outperforms all the baselines across all tasks in the LongLaMP benchmark, consistent with the test set results.

The results of the experiment on varying the exploration budget in the expectation step of self-training on the dev set are shown in Figure 9. Similar to the test set, the results indicate that while different tasks may benefit from different budgets, on average, generating 32 outputs leads to the best performance. This again emphasizes the importance of hyper-parameter tuning for this approach.

The results of the experiments on varying the number of training iterations are reported in Figure 11. This figure suggests that, similar to the test set, increasing the number of iterations leads to improved performance for both ReST-EM and REST-PG. The gap between their performance grows as iterations increase, showing that REST-PG benefits more from additional iterations. Note that after just one iteration, REST-PG outperforms ReST-EM on all datasets, even on those that performed worse

with reasoning-enhancement before self-training.

Finally, the results of experiments on starting from a new base checkpoint or continuing training from the previous checkpoint are reported in Figure 10. Similar to the test set, the results show that, on average, starting from a fresh base checkpoint performs better than continuing training from the previous checkpoint. This finding reinforces the idea that initializing from a fresh checkpoint leads to improved performance compared to fine-tuning from previously trained models.

E Case Study & Output Examples

This section presents samples of the outputs generated at various stages of our approach.

Generated reasoning path using Gemma 7B given input, output, and personalized context. As explained in Section 3.1, we utilize the Gemma 7B model to generate reasoning over personalized context by considering the personalized context, input prompt, and expected output. Figures 12 and 13 showcase two examples of such reasoning outputs. These generated reasoning summaries are subsequently used to train a smaller model, enabling it to develop preliminary reasoning abilities during the generation of responses.

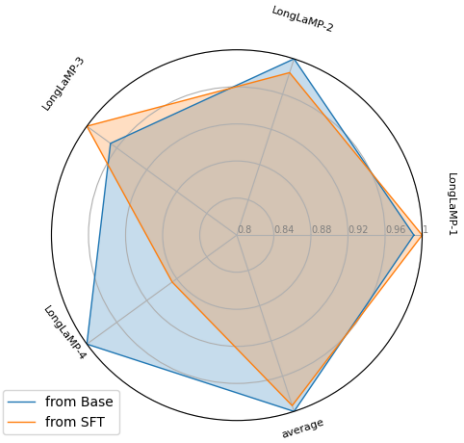

Figure 10: The relative performance of our model trained from the base checkpoint and the SFT checkpoint for one iteration on the validation set. In order to speed up the experiments, a maximum of 1,024 samples from each task randomly was selected, instead of evaluating on the full validation set.

Improvements in the final generated response. Figure 15 provides an example of personalized output generated by REST-PG and ReST-EM for a prompt from the personalized abstract generation dataset. The REST-PG model delivers a more accurate description of the proposed method and correctly predicts the evaluation dataset, ImageNet, while the ReST-EM model hallucinates and provides an incorrect guess. This example illustrates that REST-PG more effectively leverages the user's past history to generate more accurate and personalized text. In this case, the author's previous experiments on the ImageNet dataset helped the model make the correct prediction.

Improvements in reasoning path toward generating the final response. Figure 14 shows an example of outputs generated by REST-PG and SFT with Reasoning-Enhancement for a given prompt from the personalized review writing dataset. In this case, the SFT with Reasoning-Enhancement model hallucinated some names in the reasoning path and incorporated them into the final generated output. In contrast, REST-PG effectively avoided such reasoning, as it recognizes that introducing inaccurate details negatively impacts the reward model's assessment. Additionally, an interesting observation is that REST-PG inferred that the user *"values well-developed characters and relationships"* and reflected this in the review text, aligning with the expected output.

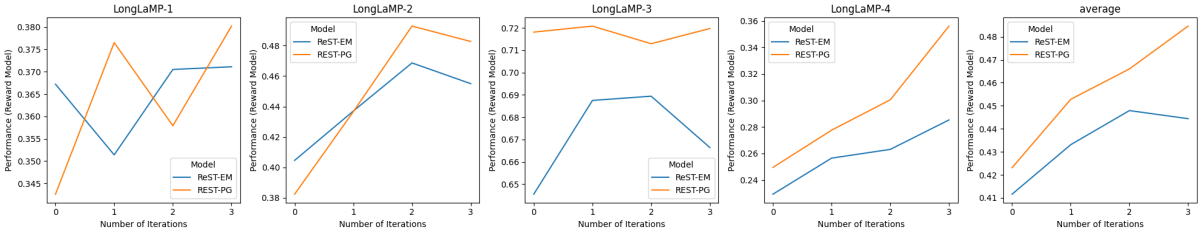

Figure 11: The affect of number of expectation-maximization steps on the performance on the validation set. In order to speed up the experiments, a maximum of 1,024 samples from each task randomly was selected, instead of evaluating on the full validation set.

Personalized Context

Shared information can benefit an agent, allowing others to aid it in its goals. However, such information can also harm, for example when malicious agents are aware of these goals, and can then thereby subvert the goal-maker's plans. In this paper we describe a decision process framework allowing an agent to decide what information it should reveal to its neighbours within a communication network in order to maximise its utility. We assume that these neighbours can pass information onto others within the network. The inferences made by agents receiving the messages can have a positive or negative impact on the information providing agent, and our decision process seeks to assess how a message should be modified in order to be most beneficial to the information producer. Our decision process is based on the provider's subjective beliefs about others in the system, and therefore makes extensive use of the notion of trust with regards to the likelihood that a message will be passed on by the receiver, and the likelihood that an agent will use the information against the provider. Our core contributions are therefore the construction of a model of information propagation; the description of the agent's decision procedure; and an analysis of some of its properties. | In this paper we describe a decision process framework allowing an agent to decide what information it should reveal to its neighbours within a communication graph in order to maximise its utility. We assume that these neighbours can pass information onto others within the graph, and that the communicating agent gains and loses utility based on the information which can be inferred by specific agents following the original communicative act. To this end, we construct an initial model of information propagation and describe an optimal decision procedure for the agent. | The emergence of large scale, distributed, sensor-enabled, machine-to-machine pervasive applications necessitates engaging with providers of information on demand to collect the information, of varying quality levels, to be used to infer about the state of the world and decide actions in response. In these highly fluid operational environments, involving information providers and consumers of various degrees of trust and intentions, information transformation, such as obfuscation, is used to manage the inferences that could be made to protect providers from misuses of the information they share, while still providing benefits to their information consumers. In this paper, we develop the initial principles for relating to inference management and the role that trust and obfuscation plays in it within the context of this emerging breed of applications. We start by extending the definitions of trust and obfuscation into this emerging application space. We, then, highlight their role as we move from the tightly-coupled to loosely-coupled sensory-inference systems and describe how quality, value and risk of information relate in collaborative and adversarial systems. Next, we discuss quality distortion illustrated through a human activity recognition sensory system. We then present a system architecture to support an inference firewall capability in a publish/subscribe system for sensory information and conclude with a discussion and closing remarks. | This work develops alternatives to the classical subjective logic deduction operator. Given antecedent and consequent propositions, the new operators form opinions of the consequent that match the variance of the consequent posterior distribution given opinions on the antecedent and the conditional rules connecting the antecedent with the consequent. As a result, the uncertainty of the consequent actually map to the spread for the probability projection of the opinion. Monte Carlo simulations demonstrate this connection for the new operators. Finally, the work uses Monte Carlo simulations to evaluate the quality of fusing opinions from multiple agents before and after deduction. | In this paper, we investigate how packet delays and losses affect the quality of target tracking. Specifically, we use Bayesian information of the posterior distribution of target locations to quantify the quality of target tracking and investigate how network quality and measurement quality affect the value of Bayesian information. We show that improving measurement quality provides diminishing gain on tracking quality, while the gain from improving network quality does not diminish. We obtain the condition under which a user obtains information gain on the target location from a tracking process. We further use Bayesian information as the metric for the gateway to select the sensor for taking measurements and determine the measurement time to control the tracking quality.

Input Prompt

Generate an abstract for the title "Reasoning about the Impacts of Information Sharing." using the following items: decision process, communication graph, utility maximization, trust, information propagation.

Expected Final Output

In this paper we describe a decision process framework allowing an agent to decide what information it should reveal to its neighbours within a communication graph in order to maximise its utility. We assume that these neighbours can pass information onto others within the graph. The inferences made by agents receiving the messages can have a positive or negative impact on the information providing agent, and our decision process seeks to identify how a message should be modified in order to be most beneficial to the information producer. Our decision process is based on the provider's subjective beliefs about others in the system, and therefore makes extensive use of the notion of trust. Our core contributions are therefore the construction of a model of information propagation; the description of the agent's decision procedure; and an analysis of some of its properties.

Generated Reasoning from LLM (Gemma 7b)

Writing Style Summary:

- **Formal and technical:** The user's writing exhibits a formal and technical tone, utilizing complex vocabulary and sentence structures.
- **Mathematical and theoretical:** The user frequently employs mathematical concepts, theories, and frameworks in their writing.
- **Analytical and argumentative:** The user presents analytical arguments and conclusions based on their analysis of complex concepts.
- **Trust and information propagation:** The user's writing emphasizes the importance of trust and information propagation in various contexts, such as communication networks and information sharing.
- **Subjective beliefs:** The user's decision-making processes are influenced by their subjective beliefs and assumptions about the behavior of others.

Figure 12: The generated profile summary with Gemma 7B on personalized abstract generation task.

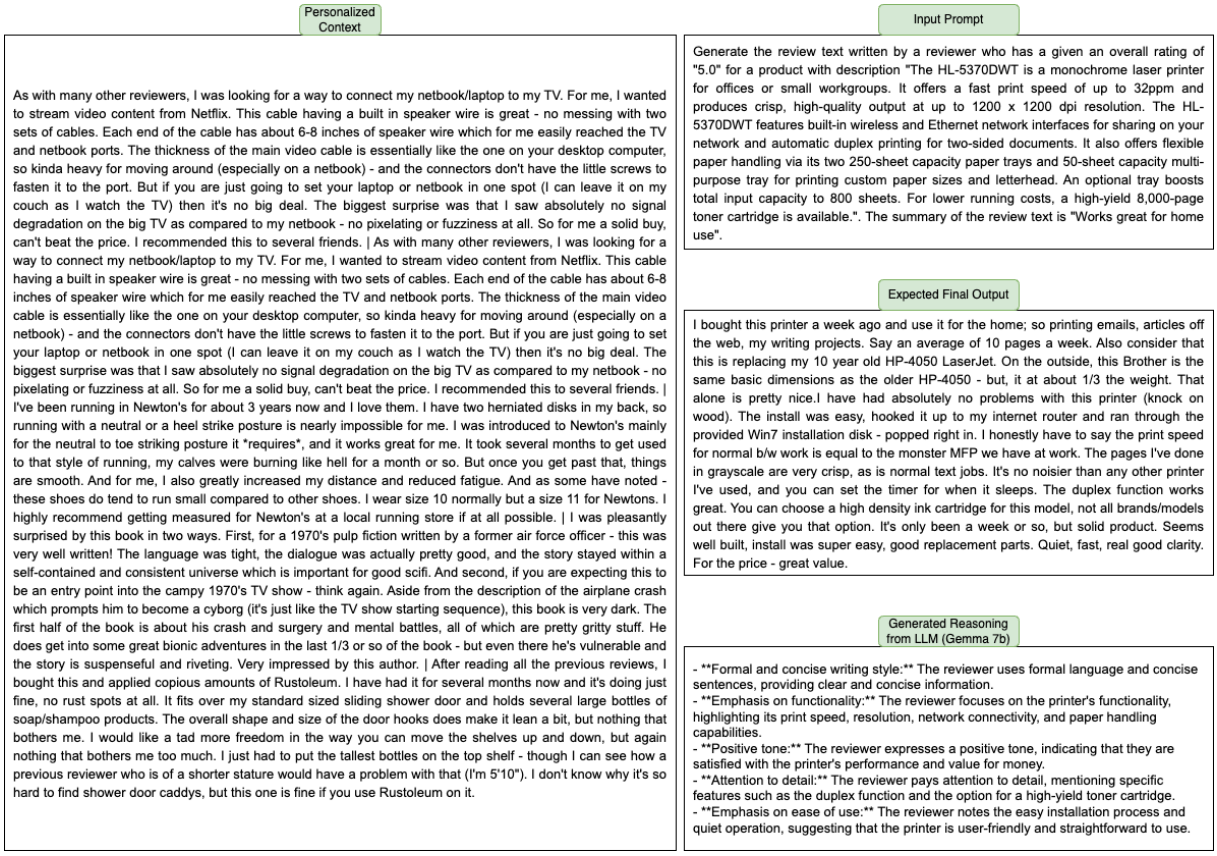

Figure 13: The generated profile summary with Gemma 7B on personalized review writing task.

Figure 14: The comparison between answer generated with REST-PG and SFT with Reasoning-Enhancement on an example from personalized review writing task.

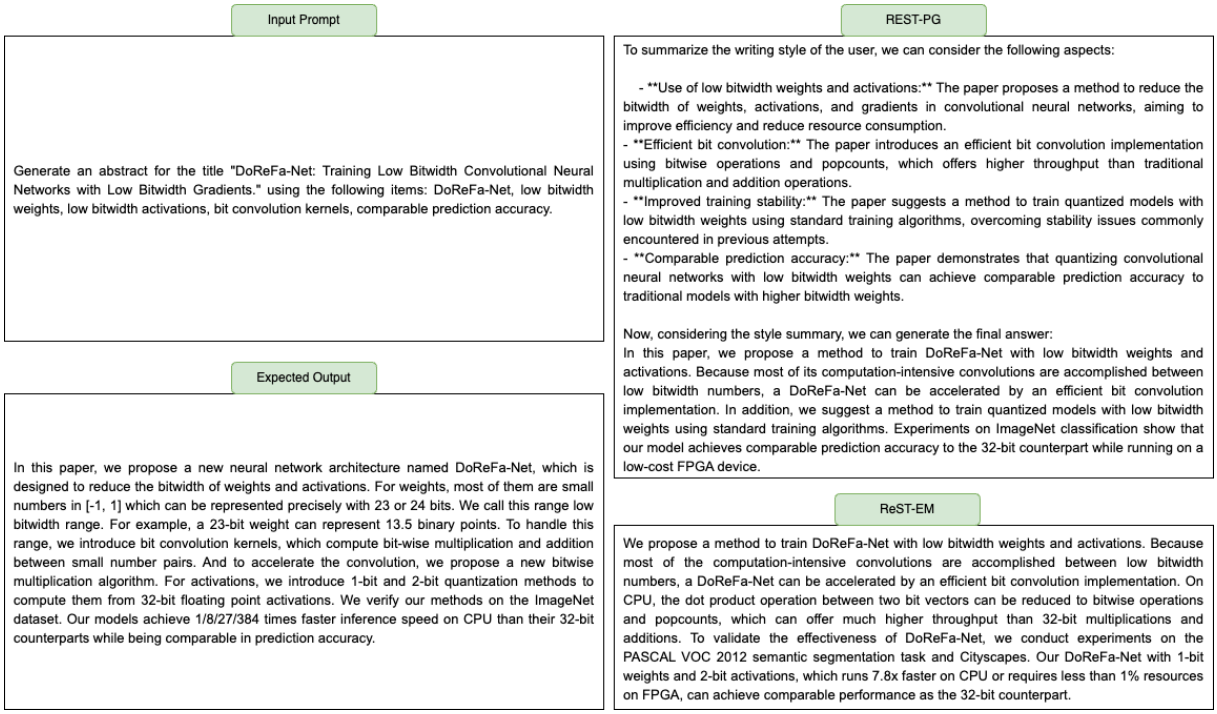

Figure 15: The comparison between answer generated with REST-PG and ReST-EM on an example from personalized abstract generation task.