Improving LLM Personalization Via Monte Carlo Methods

Ishan Khare
Stanford University
`iskhare@stanford.edu`

Anthony Zhan
Stanford University
`azhan9@stanford.edu`

Sabri Eyuboglu
Stanford University
`eyuboglu@stanford.edu`

Abstract

Users of large language models (LLMs) have diverse preferences and needs, requiring personalized responses tailored to their individual contexts. While existing approaches like prompting and fine-tuning have limitations - prompting struggles with complex personalization signals and fine-tuning requires maintaining separate model parameters per user - we propose PERSONACO, a novel Monte Carlo sampling approach. Our method generates multiple responses conditioned on personalization signals and uses self-reflection to select the response that best satisfies the user's preferences. Through empirical evaluation, we demonstrate that PERSONACO outperforms standard prompting baselines while avoiding the computational overhead of maintaining user-specific model parameters.

1 Introduction

Users of large language models (LLMs) [18] exhibit diverse preferences and needs. For example, responses to medical queries like *"What causes COVID?"* should differ between epidemiologists and laypeople, varying in technical depth and terminology. In this work, we study the challenge of personalizing LLMs to effectively serve each user's unique preferences. Specifically, we focus on the task of generating a tailored, high-quality response given a query for the model as well as a set of **personalization signals** (*e.g.* an example of the user's writing style or the phrase *"I have a PhD in epidemiology"*) (Figure 1).

In practice, this personalization task is difficult because of the large number of different (a) users (*e.g.* 200M weekly active users on ChatGPT) and (b) personalization signals. There are two main approaches for personalization commonly used in practice: *prompting* and *fine-tuning*. With prompting, user-specific instructions derived from personalization signals are prepended to the prompt [12, 3, 6, 15]. This approach is easy and efficient to implement because only a single copy of the model needs to be maintained for all users [11]. However, models may struggle to balance all the personalization signals via in-context learning alone [13, 9]. In contrast, fine-tuning provides a more flexible approach to personalization that could be more effective at integrating personalization signals [22, 17, 16, 20]. However, this requires maintaining separate model parameters for each user, which may be prohibitively expensive for large user bases (even with parameter-efficient fine-tuning [17]).

In this work, we explore whether we can improve adherence to personalization signals over simple prompting while avoiding the need for maintaining separate model parameters for each user. We propose a new method, PERSONACO, that leverages Monte Carlo sampling to generate multiple responses conditioned on the personalization signals and then, through self-reflection, selects the response that best satisfies the personalization signals. This approach enables improved adherence to personalization signals while avoiding the need for maintaining separate model parameters for each user.

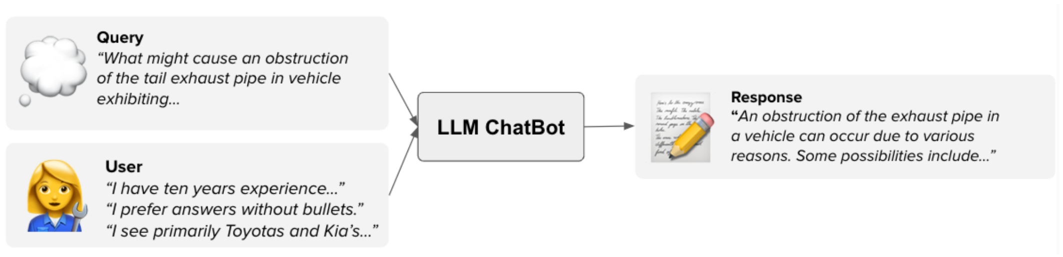
Query
"What might cause an obstruction of the tail exhaust pipe in vehicle exhibiting...
User
"I have ten years experience..."
"I prefer answers without bullets."
"I see primarily Toyotas and Kia's..."
LLM ChatBot
Response
"An obstruction of the exhaust pipe in a vehicle can occur due to various reasons. Some possibilities include..."

Figure 1: **The Language Model Personalization Task.** An example interaction with a large language model chatbot. The user provides both their query about car exhaust pipe obstruction as well as personalization signals indicating their experience level and preferences for response style. The chatbot can then tailor its response appropriately based on these personalization cues.

We find that PERSONACO outperforms the standard prompting baseline in terms of win rate (289 queries for PERSONACO vs. 212 for prompting baseline). We also perform an exploratory analysis and identify subgroups of personalization signals where PERSONACO performs particularly well. Our results highlight the potential of Monte Carlo methods applied to LLM personalization.

2 Preliminaries

2.1 Problem Formulation

We study the problem of personalizing an LLM's response to a user query given a set of personalization signals. Formally, given a query $x \in \mathcal{X}$ and a set of personalization signals $\mathbf{s} \subseteq \mathcal{S}$, the objective is to generate a response $\mathbf{y} \in \mathcal{Y}$ that satisfies both the prompt and personalization signals. We evaluate generators $g : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$ using a preference model $p : \mathcal{Y} \times \mathcal{Y} \to [0, 1]$ that compares pairs of responses based on how well they satisfy the personalization signals. A higher preference score indicates better adherence to the personalization criteria.

2.2 Related Work

Recent approaches to personalization in Large Language Models (LLMs) have primarily focused on two main strategies: prompting and fine-tuning.

Prompting
Prompting involves providing personalization criteria directly to LLMs as part of the input [23]. This method leverages the adaptability of LLMs by guiding their responses through carefully constructed input prompts. However, it has notable limitations. For one, prompting is not always effective when the personalization criteria are complex or nuanced, which can lead to inconsistent or inaccurate model responses. This is particularly problematic in scenarios where highly specific outputs are required [7]. Additionally, prompting primarily relies on textual inputs, rendering it inflexible for multimodal personalization tasks or situations requiring non-textual input modalities. An even more significant drawback arises in contexts where reward models are employed, as prompting alone is insufficient to effectively integrate such models into the personalization pipeline.

Fine-tuning
Fine-tuning explicitly trains LLMs to align with personalized preferences [5, 16]. This approach has demonstrated strong performance in adapting models to specific user needs by tailoring their weights to reflect individual or group preferences. However, it comes with significant challenges. Fine-tuning demands access to extensive datasets that encapsulate the target preferences for each persona, which may not always be available or feasible to collect. Moreover, the process is computationally intensive, often requiring substantial resources to optimize and deploy [10]. In practical applications, such as chatbots or real-time assistants, the need to store separate fine-tuned weights for each user poses scalability challenges, both in terms of storage and maintenance. This limits the applicability of fine-tuning in environments requiring dynamic, large-scale personalization solutions.

Sampling
With the impracticality of web-scale pretraining in academic settings and the exhaustion of non-

synthetic data sources, recent research efforts have shifted to focus on scaling test-time compute [2, 14]. Under this paradigm, LLMs generate a final completion through a series of intermediate generation processes, which can involve Monte Carlo, chain-of-thought, planning, etc. [19]. An appeal of such approaches is that it allows for dynamic compute allocation, i.e. spending more compute on harder or more complex queries; by contrast, under a naive generation approach, both simple and complex queries receive a single left-to-right pass [21].

In one particular study, Brown et al. demonstrate that sampling N times from the same query leads to a predictable increase in "coverage," i.e. best-of-N performance. Crucially, they note that in certain settings (i.e. code), repeated sampling from a simpler model can match or exceed performance from a more complex model while being up to 3x cheaper [2].

It is thus natural to ask whether the same test-time scaling approach using repeated sampling can be adapted to the task of personalization.

3 PERSONACO: A simple monte carlo method for personalization

In this section, we describe PERSONACO, a simple monte carlo method for LLM personalization. PERSONACO works by repeatedly sampling from the model and then prompting the model to reflect on which sample best satisfies the personalization signals.

Inputs

- A user **query** $x \in \mathcal{X}$
- A set of **personalization signals** $\mathbf{s} \in \mathcal{S}$

Generator ($g : \mathcal{X} \times \mathcal{S} \to \mathcal{Y}$)

1. **Construct a prompt.** Construct a prompt z by prepending the user query x with the concatenation of the set of personalization signals $\mathbf{s}$.
2. **Sample.** Draw k independent samples from the model conditioned on the prompt $y_i \sim P_\theta(\cdot|z)$
3. **Reflect.** For each sample y_i and personalization signal s_j, prompt the model to reflect on whether the sample satisfies the personalization signal. This produces a $k \times |\mathbf{s}|$ binary matrix.
4. **Choose.** Choose the sample y_i that maximizes the sum of the responses from the reflect step.

Outputs The generator returns the sample y_i that maximizes the adherence to the personalization signals.

Important hyperparameters of the method include the number of samples k and the exact prompt and template/format used to construct the prompt z. In our experiments, we use $k = 32$ and a temperature of 1.0.

We used the following prompt templates:

Personalization Prompt:
When answering the following query, please consider the following personalizations: {personalizations } Question: {query}. **Reflection Prompt:** Does your answer address the personalization {personalization}? Answer with only a number 0 (for no) or 1 (for yes)

For the simple prompting baseline, we use the personalization prompt above but without the reflection step.

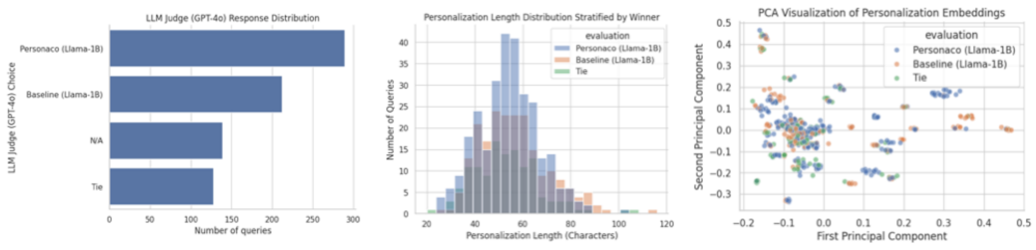

Figure 2: **PERSONACO outperforms prompting baseline.** *(Left)* The distribution of LLM as a judge decisions for the prompting baseline vs PERSONACO. The height of the bar represents the number of query-personalization signal pairs that received a preference decision. *(Middle)* The distribution of personalization signal lengths stratified by LLM as a judge decision. *(Right)* PCA visualization of TF-IDF embeddings of the personalization signals colored by LLM as a judge decision.

4 Results

In this section, we present the results of our experiments comparing generations from PERSONACO, our monte carlo sampling method, and a prompting baseline. Our experiments aim to answer the following questions.

- **Overall Win Rate**: Does PERSONACO produce generations that are better aligned with the personalization signals?
- **Win Rate vs. Personalization Complexity**: As we increase the complexity/length of the personalization signals per prompt, does the win rate of PERSONACO increase?
- **Error Analysis**: Via error analysis, can we identify any types of personalization signals where PERSONACO performs particularly well?

4.1 Experimental Setup

Dataset To simulate chatbot-user interaction, we use the UltraFeedback dataset [4], which consists of 60k queries across a variety of topics (e.g. math, code, general QA). Due to compute and time constraints, instead of working with the entire dataset, we randomly sample a subset of 128 queries.

Personalization Signals For each query, we ask GPT-4o to generate 15 different personalization strings of the form "I prefer when the response uses detailed anecdotes." The prompt that we use for the personalization signals is:

Personalization Prompt: Generate a list of k personalizations for the query below. Output in JSON format and do not include any other text. For example, for the query "Describe the characteristics of Baroque music and its impact on Western music history. , the personalizations might be:

- "Do not use any bullet lists in the response."
- "I can understand complicated language at the level of a PhD in music history."
- "I prefer when the response uses detailed anecdotes. Do not speak in generalities."
- "I prefer answers as concise as possible."
- "Do include citations."

Evaluation The evaluation is setting is a simple binary preference feedback setting — the user (which can be a human or, in our case, an LLM judge) is given two possible completions A and B, and told to select which completion better satisfies the prompt *according to the given personalization criteria*.

There are four possible responses: A, B, "same" (if both satisfy the prompt/criteria to an equal degree), or "n/a" if the personalization criteria doesn't apply to the prompt (*e.g.* if the personalization signal if "I prefer poems that rhyme.", but the query is a simple math problem).

Evaluation Prompt: Below is a query and two responses. You will be asked to evaluate the degree to which each response satisfies a personalization criterion.
Respond only with one of the following (do not include the quotes and do not include any other text):

- 'A' if response A satisfies the criterion better than response B
- 'B' if response B satisfies the criterion better than response A
- 'N/A' if the criterion is not applicable to either response
- 'SAME' if both responses satisfy the criterion to the same degree

Query: {query}
Response 1: {response A}
Response 2: {response B}

Models For generating responses, we use Meta's Llama 3.2 1B as our "generator" model used for both the prompting baseline and PERSONACO [18]. For evaluating responses (LLM as a judge), we use OpenAI's GPT-4o [1].

4.2 Results

Using the experimental setup decribed above, we answer the following questions:

Overall Win Rate: Does PERSONACO produce generations that are better aligned with the personalization signals? Out of 768 query-personalization pairs, we find that GPT-4o evaluator chooses PERSONACO 289 times, the prompting basline 212 times, and tie or not applicable 267 times (see Figure 2 Left). This represents a significant improvement in adherence to the personalization signals.

Win Rate vs. Personalization Complexity: As we increase the complexity/length of the personalization signals per prompt, does the win rate of PERSONACO increase? We do not find a strong relationship between the win rate and the complexity of the personalization signals, as measured by the length of the personalization signal (see Figure 2 Center).

Error Analysis: Via error analysis, can we identify any types of personalization signals where PERSONACO performs particularly well? We perform a simple clustering analysis where we embed each query-personalization pair using TF-IDF and then cluster with k-means. We find that PERSONACO performs particularly well for personalization signals related to the query topic (see Figure 2 Right). The gap between PERSONACO and prompting baseline is the largest for query-personalization pairs with specific grammatical or stylistic preferences (50% win rate for PERSONACO vs. 31.25% for prompting baseline). Also, we found that PERSONACO performs particularly well for query-personalization pairs related to family and leisure (35.42% win rate for PERSONACO vs. 31.25% for prompting baseline). We found no large gap between PERSONACO and prompting baseline for query-personalization pairs in the cluster related to coding (32.05% win rate for PERSONACO vs. 31.25% for prompting baseline). Furthermore, in Figure 2 Right, we provide a PCA visualization of the TF-IDF embeddings for each query-personalization pair colored by the LLM evaluator's choice.

5 Conclusion

We demonstrate that Monte Carlo sampling is an effective way to approach the personalization problem over a simple, prompt-based baseline without requiring additional training and maintenance of seprate models for different users.

5.1 Future Work

While vanilla Monte Carlo offers an efficient, straightforward way to generate repeated samples, more sophisticated methods can guide samples towards areas of high success. One such method is sequential Monte Carlo, which evaluates partial generations and gives more weight to those which seem more promising [8].

In addition, our work addresses immediate preferences which are explicitly provided via prompt. Future work may choose to focus on extracting and learning long-term preferences over multiple interactions with the user, which could be especially attractive for chatbot applications.

6 Ethics and Society Review

We recognize that advancing personalization in large language models (LLMs) through Monte Carlo methods introduces significant ethical challenges that must be addressed to ensure responsible deployment. Below, we expand on key ethical concerns and their implications for society.

6.1 Privacy and Data Security

Personalization inherently relies on collecting and processing user data, which raises critical privacy concerns. While our method does not explicitly require storing user data long-term, the personalization process may involve sensitive information from user inputs. Ensuring robust data security measures is essential to prevent unauthorized access or misuse. Moreover, the transient storage of data during computation must comply with strict privacy regulations such as GDPR or CCPA.

6.2 Bias and Echo Chambers

6.2.1 Bias Amplification

The risk of bias amplification is a persistent issue in LLMs, and personalization can exacerbate this problem. By tailoring outputs to individual preferences, there is a danger of reinforcing pre-existing biases present in training data or user inputs. For instance, if a user's preferences reflect biased perspectives, the model may perpetuate these biases without critical oversight. To mitigate this, future work could incorporate fairness-aware algorithms that actively detect and counteract biased outputs during the personalization process.

6.2.2 Creation of Echo Chambers

Personalized LLMs have the potential to create echo chambers by consistently aligning responses with a user's preferences and beliefs. While this improves user satisfaction in the short term, it may reduce exposure to diverse perspectives and hinder critical thinking over time. This concern is particularly acute in applications like news delivery or educational tools, where balanced viewpoints are essential. To address this, we could incorporate mechanisms that introduce controlled diversity into personalized outputs while still respecting user preferences.

6.3 Environmental Impact

The computational demands of Monte Carlo sampling are significantly higher than simpler prompting methods. This raises concerns about the environmental impact of scaling such approaches across millions of users. The energy consumption associated with running extra inference with models at scale could contribute to carbon emissions unless mitigated through efficient algorithms or renewable energy sources. Exploring sequential Monte Carlo methods or other optimizations could help reduce the computational footprint without compromising personalization quality.

6.4 Accountability and Transparency

As LLMs become more personalized, ensuring accountability for their outputs becomes increasingly complex. Users may attribute harmful or misleading content to the model itself rather than understanding the role their preferences played in shaping the response. To address this, our system should provide transparency regarding how personalization criteria influence outputs. Clear disclaimers or interactive explanations could help users understand the trade-offs involved in personalized responses.

6.5 Ethical Use Cases

Finally, it is critical to consider the broader societal implications of deploying personalized LLMs across various domains. For example, while personalization can enhance user experience in customer service or education, it may be misused in manipulative advertising or political campaigning. Establishing ethical guidelines for permissible use cases will be essential to prevent harm and misuse.

References

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

[2] Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling. *arXiv preprint arXiv:2407.21787*, 2024.

[3] Konstantina Christakopoulou, Alberto Lalama, Cj Adams, Iris Qu, Yifat Amir, Samer Chucri, Pierce Vollucci, Fabio Soldo, Dina Bseiso, Sarah Scodel, et al. Large language models for user interest journeys. *arXiv preprint arXiv:2305.15498*, 2023.

[4] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.

[5] Yuhao Dan, Jie Zhou, Qin Chen, Junfeng Tian, and Liang He. P-tailor: Customizing personality traits for language models via mixture of specialized lora experts. *arXiv preprint arXiv:2406.12548*, 2024.

[6] Wang-Cheng Kang, Jianmo Ni, Nikhil Mehta, Maheswaran Sathiamoorthy, Lichan Hong, Ed Chi, and Derek Zhiyuan Cheng. Do llms understand user preferences? evaluating llms on user rating prediction. *arXiv preprint arXiv:2305.06474*, 2023.

[7] Jaehyung Kim and Yiming Yang. Few-shot personalization of llms with mis-aligned responses. *arXiv preprint arXiv:2406.18678*, 2024.

[8] Alexander K Lew, Tan Zhi-Xuan, Gabriel Grand, and Vikash K Mansinghka. Sequential monte carlo steering of large language models using probabilistic programs. *arXiv preprint arXiv:2306.03081*, 2023.

[9] Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024.

[10] Venkatesh Balavadhani Parthasarathy, Ahtsham Zafar, Aafaq Khan, and Arsalan Shahid. The ultimate guide to fine-tuning llms from basics to breakthroughs: An exhaustive review of technologies, research, best practices, applied research challenges and opportunities. *arXiv preprint arXiv:2408.13296*, 2024.

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

[12] Scott Sanner, Krisztian Balog, Filip Radlinski, Ben Wedin, and Lucas Dixon. Large language models are competitive near cold-start recommenders for language-and item-based preferences. In *Proceedings of the 17th ACM conference on recommender systems*, pages 890–896, 2023.

[13] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou. Large language models can be easily distracted by irrelevant context. In *International Conference on Machine Learning*, pages 31210–31227. PMLR, 2023.

[14] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[15] Chenkai Sun, Ke Yang, Revanth Gangi Reddy, Yi R Fung, Hou Pong Chan, Kevin Small, ChengXiang Zhai, and Heng Ji. Persona-db: Efficient large language model personalization for response prediction with collaborative data refinement. *arXiv preprint arXiv:2402.11060*, 2024.

[16] Zhaoxuan Tan, Zheyuan Liu, and Meng Jiang. Personalized pieces: Efficient personalized large language models through collaborative efforts. In Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, editors, *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 6459–6475, Miami, Florida, USA, November 2024. Association for Computational Linguistics.

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

[18] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.

[19] Evan Wang, Federico Cassano, Catherine Wu, Yunfeng Bai, Will Song, Vaskar Nath, Ziwen Han, Sean Hendryx, Summer Yue, and Hugh Zhang. Planning in natural language improves llm search for code generation. *arXiv preprint arXiv:2409.03733*, 2024.

[20] Fan Yang, Zheng Chen, Ziyan Jiang, Eunah Cho, Xiaojiang Huang, and Yanbin Lu. Palr: Personalization aware llms for recommendation. *arXiv preprint arXiv:2305.07622*, 2023.

[21] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *arXiv preprint arXiv:2305.10601*, 2023.

[22] Bin Yin, Junjie Xie, Yu Qin, Zixiang Ding, Zhichao Feng, Xiang Li, and Wei Lin. Heterogeneous knowledge fusion: A novel approach for personalized recommendation via llm. In *Proceedings of the 17th ACM Conference on Recommender Systems*, pages 599–601, 2023.

[23] Zhehao Zhang, Ryan A Rossi, Branislav Kveton, Yijia Shao, Diyi Yang, Hamed Zamani, Franck Dernoncourt, Joe Barrow, Tong Yu, Sungchul Kim, et al. Personalization of large language models: A survey. *arXiv preprint arXiv:2411.00027*, 2024.