Response-Aware User Memory Selection for LLM Personalization

Jillian Fisher [1] **Jennifer Neville** [2] **Chan Young Park** [2]

Abstract

A common approach to personalization in large language models (LLMs) is to incorporate a subset of the user memory into the prompt at inference time to guide the model's generation. Existing methods select these subsets primarily using similarity between user memory items and input queries, ignoring how features actually affect the model's response distribution. We propose **R**esponse-**U**tility optimization for **M**emory **S**election (RUMS), a novel method that selects user memory items by measuring the mutual information between a subset of memory and the model's outputs, identifying items that reduce response uncertainty and sharpen predictions beyond semantic similarity. We demonstrate that this information-theoretic foundation enables more principled user memory selection that aligns more closely with human selection compared to state-of-the-art methods, and models $400\times$ larger. Additionally, we show that memory items selected using RUMS result in better response quality compared to existing approaches, while having up to 95% reduction in computational cost.

1. Introduction

As large language models (LLMs) become ubiquitous in daily interactions, personalization has emerged as critical for user satisfaction. To meet this demand, current approaches to personalization generally fall into two categories: training-based methods and inference-time methods. Training-based approaches, such as low-rank preference models (Bose et al., 2025) or parameter-efficient fine-tuning modules (Tan et al., 2024), adapt the model parameters for individual users. Although these methods can be effective for small populations, they become computationally prohibitive when scaled to many users. As a result, inference-time personalization, such as including user-specific context in the input prompt, is often the preferred solution.

However, inference-time methods raise a central question: **What information, if any, should be included to guide the model?** A naïve strategy is to include all stored user memories in the prompt. However, this method often reduces performance, as irrelevant or distracting information may overwhelm the model and produce responses that are less accurate or sound unnatural (Li et al., 2023; Rajeev et al., 2025). Moreover, this strategy becomes increasingly impractical as the memory size grows. Therefore, existing methods instead attempt to select and retrieve relevant subsets of the user memory by measuring the relationship between a *memory item and the input query*. In other words, these methods try to pick out only the user memory items that seem most related to the user's current request. This is often done by heuristic retrieval (Lù, 2024), surface-level semantic similarity (Lei et al., 2023), or prompting LLMs (Handa et al., 2024). While intuitive, **we hypothesize that measuring how memory items affect output (i.e. response) distributions directly, rather than similarity to input queries, provides a more reliable signal for personalization**. Selecting user memory items based on semantic similarity to the input may biases selection toward redundant, surface-aligned items and away from non-obvious but important ones, leading to both over-selection and under-selection relative to true utility. Such shortcomings can introduce noisy items that degrades response quality while increasing latency, cost, and privacy risks (Hagström et al., 2025).

To test this hypothesis, we introduce Response-Utility for Memory Selection (RUMS), which comprises of two key components: a novel utility function (*RUMS-Utility*) that quantifies the informativeness of user memory subsets, and a lightweight model (*RUMS-Model*) that efficiently approximates this utility computation at inference time.

RUMS-Utility utilizes an information-theoretic framework to directly measure how much each subset of user memories reduces uncertainty in the model's response distribution. Specifically, it computes the conditional mutual information between candidate memory subsets and the model's output distribution given the input query. We provide a theoretical foundation for this objective, proving that under the existence of a latent variable representing user intent,

[1]Department of Computer Science and Engineering, University of Washington, Seattle, WA, United States of America [2]Microsoft Research, Redmond, WA, United States of America. Correspondence to: Jillian Fisher <jrfish@uw.edu>.

Pluralistic Alignment Workshop @ ICML 2026, Seoul, South Korea. Copyright 2026 by the author(s).

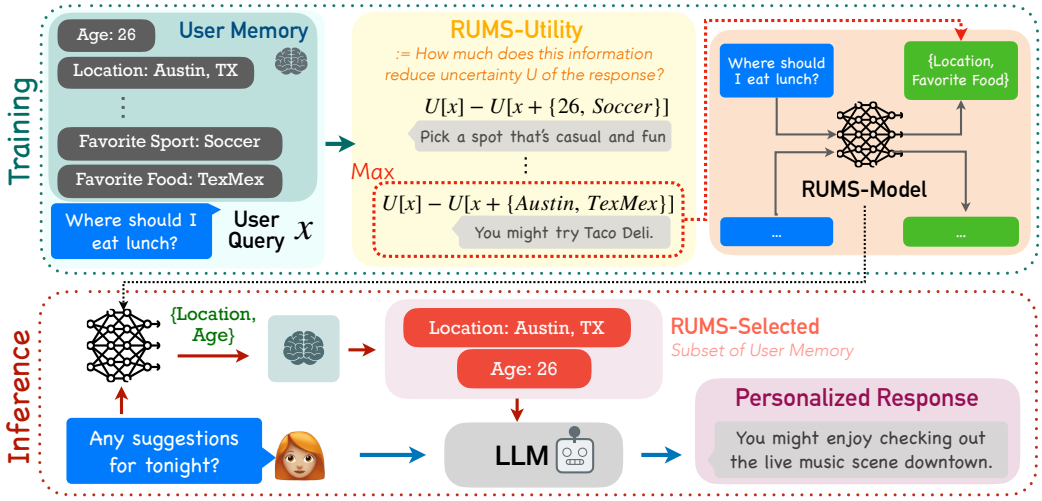


Figure 1. Overview of RUMS. RUMS-Utility quantifies how much memory subsets reduce response uncertainty (training phase), while RUMS-Models efficiently selects informative items at inference.

minimizing predictive entropy is approximately equivalent to maximizing expected user utility. While this guarantee assumes the model's distribution aligns with human preferences, our empirical results confirm that modern LLMs are sufficiently calibrated for this entropy signal to serve as a high-fidelity proxy for true utility.

However, computing *RUMS-Utility* for all queries is computationally expensive, particularly with large user profiles. Therefore, we use the utility function during an offline training phase to generate high-quality training data. We then train *RUMS-Models*, a lightweight encoder-classifiers based on DeBERTa (He et al., 2021), to directly predict memory items that are most informative given a query. This learned approach preserves the principled selection quality of the utility function while enabling efficient deployment.

Through extensive empirical testing, the experimental results validate our hypothesis: RUMS enables more principled memory item selection that is more closely aligned to human selection compared to state-of-the-art (SOTA) memory selection methods and models $400\times$ its size. We also show that the items selected using RUMS result in better response quality than other methods, while lowering the cost of personalization overall.

Our work offers novel conceptual, theoretical, and empirical contributions to personalization:

- We propose RUMS, a lightweight, response-based method for user memory selection that directly accounts for the impact of memory items on model outputs.
- We establish a principled utility function, RUMS-Utility, and provide a theoretical proof linking entropy reduction to expected utility maximization. We further validate the necessary assumption of model-human alignment through empirical results, showing that RUMS-Utility can effectively guide the training of a lightweight RUMS-Model.
- We show that RUMS (i) reliably identifies when personalization is beneficial, aligning closely with human judgments, (ii) selects memory subsets that align more strongly with human annotations than SOTA retrieval and prompting baselines, and (iii) improves generation quality over SOTA baselines and $400\times$ larger models on synthetic and real-world datasets, while adapting to underlying model capabilities and costing up to 95% less than other methods.

2. Related Works

User Memory for LLM Personalization Memory augmentation enables personalization in LLMs by retrieving subsets of user memory at inference time to guide responses. Simple approaches prepend user context to prompts, such as profile summaries (Richardson et al., 2023; Pan et al., 2025) or past interactions (Kang et al., 2023; Liu et al., 2023). However, as the amount of user memory grows, these methods quickly become intractable. Due to this, there has been an increased trend in selecting a *subset* of the user memory to prepend. For example, Chen (2025) retrieves relevant historical items via semantic similarity for recommendations, while LaMP (Salemi et al., 2024b) investigates personalized sentence retrieval. PEARL (Mysore et al., 2024) and ROPG (Salemi et al., 2024a) train retrieval components with user signals to improve personalization. For a broader overview, see Liu et al. (2025). However, these methods rely on the similarity between user input and memory item, without considering their impact on the model's downstream response.

Selective Personalization A related line of reserach is selective conditioning or abstention which studies how models can ignore user memory or abstain from predictions when inputs are uninformative. Geifman & El-Yaniv (2017) introduce selective classification for neural networks, and recent work extends this to language models, proposing mecha-

nisms to ignore input features or context that may degrade performance (Hagström et al., 2025). In memory selection, this corresponds to sometimes choosing no subset when personalization is unnecessary. Prior work typically relies on global thresholds (Li, 2023) or static policies (Joren et al., 2023), whereas our method learns a sequential, adaptive policy that dynamically selects which items to condition on based on the input and context.

Information-theoretic Uses in NLP Information-theoretic ideas have long informed NLP (Brown et al., 1992), spanning multimodal learning (Almudévar et al., 2025), information control (Ravfogel et al., 2020), interpretability (Chen et al., 2018), and prompt design (Zhang et al., 2022). In feature selection, EDDI (Ma et al., 2019) estimated conditional mutual information for individual features, while later work extended this to dynamic or sequential selection (Covert et al., 2023; Gadgil et al., 2024). Recent methods also use differences in next-token likelihood with and without context to guide generation through distribution editing (He et al., 2025). While much of this work has focused on domains such as image classification and fraud detection, the closest parallel to personalization arises in recommendation systems using Bayesian active learning to select a model query to ask the user, with the goal of improving recommendation quality (Handa et al., 2024). We draw on these principles to select memory items that maximize information gain for personalized responses.

3. Method

We aim to develop a method that selects a subset of user memory based on the goal of maximizing the utility of the model's responses.

Problem Formulation Consider user u with query x (e.g. "Where should I eat lunch?"). We let θ denote an LLM and $Y = (Y_1, ..., Y_T)$ denote a random variable representing an output sequence distributed according to $p_\theta(Y|x)$. We measure the quality of a specific realized response y by its utility to the user, $\mathrm{U}_u(y)$.

Also, we consider a set of memory items $\mathcal{M}$ (e.g. Location), in which user u has specific values m_u (e.g. Location: Austin, TX). Our aim is to select the memory subset $S \subseteq \mathcal{M}$ with instantiated values $m_{u,S} = \{m_u^{(a)} : a \in S\}$, which when given to the model with the input x yields the highest response utility. Formally,

$$S^*(x, u) = \arg\max_{S \subseteq \mathcal{M}} \mathbb{E}_{y \sim p_\theta(\cdot|x, m_{u,S})}\big[\mathrm{U}_u(y)\big]. \quad (1)$$

For brevity, unless otherwise specified, expectations are taken w.r.t. the model's predictive distribution, $\mathbb{E}[\cdot] = \mathbb{E}_{y \sim p_\theta(\cdot|x, m_{u,S})}[\cdot]$. Also, we assume x is fixed throughout.

However, we note that U_u is unobserved and cannot be evaluated prior to generation, so there is a need for a proxy utility function.

3.1. RUMS-Utility

In order to formulate a surrogate utility function, we frame memory selection for personalization as an information acquisition problem: given a large space of user memory, the goal is to select the items that most improve our response to an input. Given this framing, we propose a novel utility function based on information gain from memory subsets, serving as an entropy-based surrogate for $\mathrm{U}_u(y)$. Formally,

$$\mathrm{U}_{RUMS}(S, u) = H_\theta\left(Y|x\right) - H_\theta\left(Y|x, m_{u,S}\right) \quad (2)$$

where $H_\theta(\cdot)$ denotes Shannon entropy under the model distribution (defined in Section B.1). Given that only the second term in Equation (2) is dependent on the subset selected S, we can simplify the objective and state that RUMS chooses the optimal subsets S^* such as,

$$S^*(x, u) = \arg\min_{S \subseteq \mathcal{M}} H_\theta(Y \mid x, m_{u,S})$$

The intuition is that a memory subset S is valuable to the extent it reduces uncertainty in the model's predictions. For instance, for "where should I eat lunch?", without user information, the model spreads probability across countless restaurants, yielding high entropy. However, adding even a single relevant fact, such as "Location: Austin, TX" or "Dietary preference: Vegan", sharply concentrates probability on relevant options. Therefore, the utility of the memory subset S is measured directly by reduction in the entropy.

3.2. Theoretical Foundation

We formalize conditions under which reducing predictive uncertainty leads to higher expected user utility. While the theoretical foundation rely on a latent-variable abstraction, we validate empirically (Section 4) that minimizing predictive entropy aligns with improved user-perceived utility.

Assumption 1 (Existence of an Optimal Latent Variable)**.** For each user u and query x, there exists a latent random variable $Z_u \sim p(Z_u \mid x)$ representing the user-specific latent target that determines what constitutes an optimal response. User utility depends on alignment between the generated response and this latent target.

Formally, where ℓ is a strictly proper scoring rule, we assume

$$\mathbb{E}\big[\mathrm{U}_u(y)\big] = -\mathbb{E}_{z_u \sim p(Z_u|x)}\mathbb{E}_{y \sim p_\theta(\cdot|x, m_{u,S})}[\ell(y, z_u)].$$

Conceptually, Z_u is an abstract latent variable capturing the maximal predictive information needed for an optimal response.[1]

[1]While this offers a formal link between predictive distributions and decision quality (Gneiting & Raftery, 2007), it assumes static user preferences. In practice, preferences can be non-stationary and shaped by model outputs, so a fixed target may miss the co-evolutionary dynamics of long-term interactions.

Assumption 2 (Approximate Sufficiency of Latent Variable)**.** Conditioned on the query x and latent target Z_u, the memory values $m_{u,S}$ do not provide substantial additional information about the output Y:

$$p_\theta(Y \mid x, Z_u, m_{u,S}) \approx p_\theta(Y \mid x, Z_u).$$

This assumption treats Z_u as an informational bottleneck, assuming memory items primarily inform the response by clarifying latent intent. However, this may overlook scenarios where the raw data, such as specific formatting, provides utility not captured by an abstract intent. Therefore, we adopt approximate rather than exact sufficiency to account for real-world cases where memory might contain extraneous details beyond the latent preference (Tishby et al., 1999; Alemi et al., 2017).

Theorem 3.1 (Optimality of RUMS)**.** *Under Assumptions 1 and 2, for any two memory subsets $S, S' \subseteq \mathcal{M}$,*

$$H_\theta(Y \mid x, m_{u,S}) \leq H_\theta(Y \mid x, m_{u,S'}) \quad \Rightarrow \quad \mathbb{E}_{y\sim p_\theta(\cdot|x,m_{u,S})}[U_u(y)] \geq \mathbb{E}_{y\sim p_\theta(\cdot|x,m_{u,S'})}[U_u(y)].$$

Theorem 3.1 formalizes the intuition that reducing predictive entropy improves expected utility, with approximation arising from the latent-variable abstraction in Assumption 2.

Proof Sketch. Predictive entropy decomposes into uncertainty about the latent target Z_u and residual uncertainty about y. Under Assumption 2, selected memory mainly reduces uncertainty about Z_u, and because strictly proper scoring rules (Assumption 1) reward matching the conditional distribution, this reduction directly improves expected utility. Full derivation is provided in Appendix C.

Interpretation and Limitations Theorem 3.1 provides a principled justification for selecting memory subsets that reduce predictive entropy. However, in practice, *this surrogate objective is only meaningful if the model's predictive distribution $p_\theta(y|x, m_{u,S})$ is aligned with human judgments*, that is, responses the model assigns higher probability to under relevant memory subsets tend to be more useful to humans. While Theorem 3.1 provides a principled foundation, *empirical evaluation (Section 4) is necessary* to verify that reductions in entropy translate into improvements in actual user-perceived utility.

3.3. Practical Estimation

Although RUMS-Utility is theoretically grounded, directly computing it for LLM-generated sequences is intractable, as the space of possible outputs y grows exponentially with the sequence length T. Therefore, we employ practical approximations via token-level decomposition and sampling.

Token-Level Decomposition To achieve tractability, we leverage the chain rule of probability to decompose the sequence-level entropy into a manageable sum of conditional token-level entropies:

$$H_\theta(Y \mid x, m_{u,S}) = \mathbb{E}\Big[\sum_{t=1}^{T} H\big(p_\theta(y_t \mid y_{<t}, x, m_{u,S})\big)\Big].$$

Here, the expectation is taken w.r.t. $p_\theta(y_t \mid y_{<t}, x, m_{u,S})$ the model's conditional distribution over the next token. This decomposition reduces the problem to evaluating token-level entropies, which is computationally manageable for long sequences (see proof of this composition in Section D).

Monte Carlo Approximation The outer expectation over token prefixes remains intractable, so we approximate it using Monte Carlo sampling. We generate N sequences $\{y^i\}_{i=1}^N$ from $p_\theta(\cdot \mid x, m_{u,S})$ and compute

$$\hat{H}_\theta(Y \mid x, m_{u,S}) = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{T_i}\sum_{t=1}^{T_i} H\big(p_\theta(y_t^i \mid y_{<t}^i, x, m_{u,S})\big),$$

where T_i is the length of the i-th sequence. We normalize by sequence length to estimate per-token entropy, which allows comparison across variable-length generations.

This approximation converges to the true per-token sequence entropy as $N \to \infty$ under i.i.d. sampling from $p_\theta(\cdot \mid x, m_{u,S})$ with finite expectation. In practice, however, factors such as constrained decoding or approximate sampling distributions can introduce bias. Despite these limitations, we find $N = 5$ samples provides a good balance between accuracy and efficiency. For more empirical analysis, see Section A.1.

The final estimated utility is then,

$$\widehat{\mathrm{U}}_{\mathrm{RUMS}}(S, u) = \hat{H}_\theta(Y \mid x) - \hat{H}_\theta(Y \mid x, m_{u,S}). \quad (3)$$

Then, our final objective selects the memory subset that maximizes this utility:

$$S^*(x, u) = \arg\max_{S\subseteq M} \widehat{U}_{\mathrm{RUMS}}(S, u). \quad (4)$$

$\widehat{\mathrm{U}}_{\mathrm{RUMS}}(S, u)$ serves as a practical surrogate for user utility and retains the theoretical connection to expected utility established in Theorem 3.1.

3.4. RUMS-Models: Learning to Approximate Utility

While RUMS provides a principled approach to memory selection, explicitly computing entropy reductions over all possible subsets at inference time is prohibitive, particularly for users with large memory profiles. To enable efficient deployment, we use RUMS offline to generate training data and train a lightweight model to directly predict memory selections at inference.

Training Data Generation Our training pipeline uses RUMS to generate (user query, optimal memory subset) pairs. For each query x, we pair it with a diverse set of users $\{u_1, \ldots, u_K\}$, each with memory items m_u. When profiles are large ($|m_u| \geq 50$), we first use an LLM to filter to k candidate items. We then compute $\widehat{\mathrm{U}}(S, u)$ from Equation (4) for candidate subsets S, selecting the highest-utility subset $S^*(x, u) = \arg\max_S \widehat{\mathrm{U}}(S, u)$ if it exceeds threshold τ, or the empty set otherwise. This yields K training examples $\{(x, u_k, S^*(x, u_k))\}_{k=1}^K$ per query. Pairing each query with diverse profiles encourages the model to learn genuinely informative item selection rather than spurious correlations. The threshold τ, tuned on validation data, enables automatic abstention when personalization would not help.

Model Architecture and Training We train RUMS-Models as lightweight classifiers to approximate utility computation at inference. For dynamic memory schemas, where item types vary across users, a binary classifier (RUMS-Binary) $f(x, m_i) \to \{0, 1\}$ evaluates each memory item $\{m_i\}_{i=1}^n$ independently, handling arbitrary structures but requiring n forward passes. For static schemas with d fixed item types, a multiclass classifier (RUMS-Multi) $f(x) \to \{0, 1\}^d$ outputs all selections in a single pass. The resulting RUMS-Models match the inference speed of simple similarity-based retrieval while retaining principled selection quality, making it practical for large-scale deployment.

4. Results

In this section, we evaluate RUMS by defining three outputs: *RUMS-Utility*, the utility score from Equation (3); *RUMS-Subset*, the selected memory subset from Equation (4); and *RUMS-Models*, models trained on RUMS-generated data.

We evaluate our approach along three dimensions:

- H1: RUMS-Utility provides a reliable signal of human preference for distinguishing between inputs that do and do not benefit from personalization.
- H2: The subsets selected, RUMS-Subset, choose memory subsets more aligned with human preference than SOTA baselines and larger models. Furthermore, the utility signal is strong enough that models trained on them (RUMS-Models) also outperform baselines and larger models.
- H3: Memory subsets selected with RUMS-Models improve downstream response generation compared to state-of-the-art baselines and larger models.

4.1. Experimental Setup

Across all experiments (H1–H3), we consider $n = 50$ static user memory items (e.g., *location*, *age*, *name*)[2], and assumes that all values are given for each user (see Section F.3 for list of items). The task is to select up to 5 items that are most relevant for answering a given user input x.

Datasets. To test both instances that would and would not benefit from personalization, we use a diverse mixture of possible user inputs. We evaluate across 3 datasets:

- **PersonaFeedback** (Tao et al., 2025): A synthetic dataset of LLM-generated user inputs using randomly sampled user memory, ensuring each input benefits from personalization. We translated the original Chinese data using a translation model (Tiedemann & Thottingal, 2020). Example: *"Do you have any music for the trip?"*
- **FreebaseQA** (Jiang et al., 2019): A dataset of trivia-style questions and answers. Example: *"What type of snake is a sidewinder?"*
- **WildChat** (Zhao et al., 2024): Real-world ChatGPT interactions (English only, non-toxic, and request-based). We use only the first human-written input. Examples: *"im hungry"; "who is Mohamed Gamal Mohamed Maniieh"*

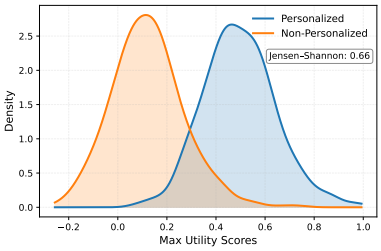
By nature of the data, we classify PersonaFeedback as personalized inputs and FreebaseQA as non-personalized inputs, and refer to these together as the "Synthetic" dataset. For WildChat, two annotators labeled each input as personalized or non-personalized (agreement required). We denote these subsets as WildChat-Personal and WildChat-NonPersonal, which form the "Real-World" dataset.

User Profiles. We used GPT-4 (OpenAI et al., 2024) to generate 50 common user memory items and n=100 consistent user profiles (avoiding contradictions like "vegan" with "favorite food = steak") for training and testing. See Section F.3 for prompts, distributions of memory item values and generation details.

RUMS-Model Training To create training data with RUMS, we used Llama-3.1-8B (AI, 2024b) to compute entropies. Following the procedure in Section 3.4, GPT-4 was first prompted to reduce the candidate memory items from 50 to at most 10 for each query before computing the utility scores. The utility scores were estimated with $n = 5$ Monte Carlo samples. Lastly, a threshold of $\tau = 0.29$, determined on a held-out validation set,was applied to choose between the subset with maximum utility or the empty set (i.e., no personalization). We use $n = 2.5$K from PersonaFeedback and 2K from FreebaseQA for training. Since both datasets are also used for evaluation, we rely on held-out splits to avoid overlap and interpret results as in-domain performance. To assess generalization, we also evaluate on WildChat, a real-world dataset from a different distribution.

We train two RUMS-Model variants: The first, RUMS-Binary, is a binary classifier trained on individual (query, memory value) pairs. Because each memory value is treated separately and negative labels dominate, this model tends

[2]We generated 100 candidate memory items, but items beyond the top 50 were increasingly obscure and less real-world attributes.



(a) Distribution of maximum utility scores for personalized (PersonaFeedback) and non-personalized (FreebaseQA) inputs.

(b) Recall (accuracy on personalized) and specificity (accuracy on non-personalized) comparison across datasets and methods. **Bolded** is the highest value in each category.

Dataset / Metric	RUMS-Utility	GPT4-Few	GPT4-Zero	Semantic Similarity
Synthetic (n=20) / Recall	94.4	**100.0**	33.3	**100.0**
Synthetic (n=20) / Specificity	**94.4**	50.0	50.0	0.0
Real-World (n=31) / Recall	**92.3**	76.9	53.9	7.7
Real-World (n=31) / Specificity	96.0	**100.0**	**100.0**	**100.0**

Figure 2. Results for H1 analysis.

Table 1. Precision, recall, and F1 scores of memory items selected compared to human gold label annotation across datasets and methods.

Dataset	**Metric**	RUMS-Subset	RUMS-Binary	RUMS-Multi	Semantic Sim.	GPT4-Zero	GPT4-Few
Synthetic	Precision	**0.61**	0.60	0.23	0.09	0.26	0.53
	Recall	0.70	**0.89**	0.39	0.23	0.42	0.65
	F1	0.70	**0.78**	0.31	0.14	0.35	0.64
Real World	Precision	**0.48**	0.32	0.30	0.17	0.39	0.42
	Recall	0.36	**0.60**	0.38	0.27	0.52	0.39
	F1	**0.53**	0.45	0.35	0.20	0.43	0.39

to under-select, making it better suited for noise-sensitive settings. The second, RUMS-Multi, is a multiclass classifier trained on denser labels constructed by merging the selected subsets from the $u = 5$ user profiles per query. This means, all memory items chosen by any of the profiles were included. As a result, the model tends to over-select, which is advantageous for more robust models that can filter noise. Both are based on DeBERTa-v3-large (He et al., 2021). Full details are in Section F.5.

Baselines Current approaches to memory selection primarily rely on two strategies: semantic similarity-based retrieval and LLM prompting. To comprehensively evaluate against existing methods, we employ baselines representing both approaches: (1) **Semantic Similarity**: We compute cosine similarity between input and memory item embeddings using Sentence Transformer (Wang et al., 2020), representing the dominant paradigm of surface-level similarity matching in current retrieval systems, (2) **GPT-4 Prompting**: We use direct prompting of GPT-4 (*zero-shot* and *few-shot* - up to 4 in-context examples) as a representative LLM-based approach and upper bound on model capability, given its 400× larger size compared to RUMS-Models. For H3, which evaluates downstream generation quality, we thus additionally include established retrieval methods and naive baselines to ensure comprehensive coverage: (3) **SOTA Retrieval Methods**: We evaluate *BM25* (Lù, 2024) and *ReContriever* (Lei et al., 2023) as representative SOTA retrieval approaches for memory selection, (4) **Naive Baselines**: We include *None* (no items), *All* (50 items), and *Random* (5 items) to establish performance bounds and test robustness to different selection strategies.

4.2. H1: Does RUMS-Utility Distinguish Personalized from Non-Personalized Inputs?

We hypothesize that entropy reduction when conditioning on user memory reflects its importance. Thus, inputs that do not benefit from personalization should yield low maximum utility scores (i.e., the maximum utility score across all subsets of candidate memory items for a specific user), while personalized inputs should yield high ones. To test this, we compare the maximum utility scores on a known non-personalized (FreebaseQA) and a known personalized (PersonaFeedback) datasets. We sampled 500 inputs per dataset and computed the maximum utility score using Equation (3) for each input. The resulting distributions were compared using divergence metrics and hypothesis testing.

Utility scores separate personalized from non-personalized inputs. In Figure 2a we show that personalized inputs exhibit much higher utility scores (mean = 0.50, std = 0.12) than non-personalized ones (mean = 0.12, std = 0.09). The Jensen–Shannon divergence between distributions is high at 0.66, and a two-sample Kolmogorov–Smirnov test yields a statistic of 0.82 ($p < .001$), strongly rejecting the null hypothesis of equal distributions. These results indicate that maximum utility scores reliably separate personalized from non-personalized inputs. For further exploration on hyperparameter selection and it's effects on the robustness of these results see Section A.1.

4.2.1. H1′: Do Utility Scores Align with Human Judgments of Personalization?

We next test whether personalization decisions made by the utility score thresholding aligns with human judgment.

Table 2. Average win rate of our method against baselines on Synthetic (left) and Real World (right) datasets. Cells in green mark cases where our method outperforms the baseline, and cells in red mark cases where the baseline does better. Values with magnitude above 0.1 are **bolded** to highlight strong differences. Results are reported for two response models (LLaMA 70B Instruct and GPT-4) and two RUMS variants: RUMS-Binary (fewer items) and RUMS-Multi (more items).

	Synthetic						Real World					
	LLaMA 70B			GPT-4			LLaMA 70B			GPT-4		
	Utility	Binary	Multi	Utility	Binary	Multi	Utility	Binary	Multi	Utility	Binary	Multi
None	**0.11**	**0.25**	**-0.13**	**0.20**	**0.26**	**0.36**	-0.05	0.05	0.03	0.10	**0.20**	**0.17**
All	**0.18**	**0.16**	**-0.19**	**0.20**	**-0.16**	0.02	0.03	-0.04	-0.10	**-0.14**	**-0.14**	-0.06
Random	**0.44**	**0.34**	**0.13**	0.10	0.06	**0.19**	**0.25**	**0.11**	**0.13**	0.05	0.10	0.04
Semantic Sim.	**0.43**	**0.27**	0.09	**-0.12**	-0.03	0.05	**0.16**	-0.03	0.00	-0.06	-0.02	0.00
BM25	0.03	**0.19**	**-0.13**	**0.18**	**0.20**	**0.32**	0.06	0.01	0.05	0.10	**0.18**	**0.13**
ReContriever	**0.31**	**0.27**	-0.03	-0.05	-0.04	0.03	**0.22**	0.06	0.01	0.07	-0.05	**0.13**
GPT4-Few	0.07	**0.18**	**-0.16**	-0.07	-0.09	-0.01	0.01	0.00	-0.08	-0.08	0.04	**0.11**

We randomly sampled $n = 25$ inputs from each of PersonaFeedback, FreebaseQA, WildChat-NonPersonal, and WildChat-Personal. Four human annotators labeled whether each input benefits from personalization, which yielded an agreement between the parties of Cohen's $\kappa = 0.59$; we retained only inputs with unanimous agreement ($n = 64$).

In order to get a robust signal using RUMS, we used $n = 10$ Monte Carlo samples and $u = 10$ randomly selected user-profiles. We used the average maximum utility score over the profiles as RUMS-Utility. We then tuned a threshold on 20% validation data. We compared RUMS-Utility against GPT-4 (prompted to determine personalization necessity) and semantic similarity baselines with a threshold tuned on the validation set. The performance was measured by recall (accuracy in identifying personalized inputs) and specificity (accuracy in identifying non-personalized inputs). If both humans and the model selected no items, the sample was counted correct for both metrics.

RUMS-Utility correlate more strongly with human judgments. As shown in Table 2b, RUMS-Utility outperforms GPT-4 and semantic similarity baselines. Compared to GPT-4 FewShot, it achieves +7% recall and +15% specificity overall. Gains are strongest on synthetic data, with competitive results on real-world. Other baselines show imbalanced behavior (e.g., GPT-4 ZeroShot often avoids personalization, giving high specificity but low recall). In contrast, utility scores perform well on both metrics, reflecting their ability to capture each item's true effect on generation.

4.3. H2: Does RUMS-Subset Match Human Judgment?

Building on H1, we now ask whether our method identifies which specific memory items help personalization. We randomly selected $n = 15$ samples from each of PersonaFeedback, FreebaseQA, WildChat-NonPersonal, and WildChat-Personal for a total of $n = 60$. As before, four annotators selected any (or none) of 50 possible items they judged useful, yielding 41% majority ($k \geq 3$) agreement.

While moderate, this level of agreement is expected given the subjective nature of the task. We therefore considered an item as selected by humans if a majority agreed, resulting in an average of 4.19 items selected per user query for inputs where at least one item was chosen.

We evaluated (i) direct selection from utility scores RUMS-Subset (up to 5) and (ii) RUMS-Models, the trained models described in Section 4. We compare RUMS-Subset and RUMS-Models against GPT-4 (prompted to select up to five relevant memory items, both zero- and few-shot) and semantic similarity (selecting up to five highest-scoring items with cosine similarity > 0). Performance was measured with precision, recall, and F1 against human labels.

RUMS-Subset aligns more closely with human annotation. As shown in Table 1, the two methods based on RUMS achieve on average 20% higher precision, 31% higher recall, and 9% higher F1 compared to the baselines across both synthetic and real-world datasets. Semantic similarity performs worst, reflecting reliance on surface overlap rather than downstream utility. GPT-4 shows strong recall but substantially lower precision, indicating that it tends to over-select. In contrast, the results show that the raw utility scores, RUMS-Subset, themselves provide strong signal for identifying useful memory items, as RUMS-Models achieves similar F1, demonstrating that small models with modest amounts of data can learn effective memory selection from utility scores. For analysis of the approximation gap between RUMS-Utility and RUMS-Models, see Section A.3; qualitative examples appear in Section A.6.

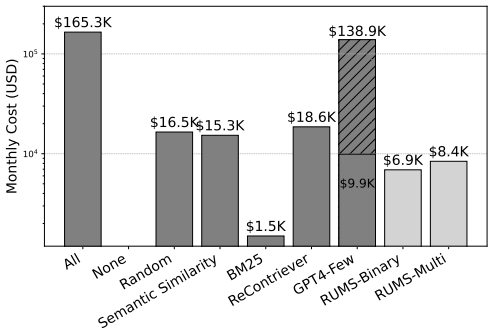
4.4. H3: Do Items Selected by RUMS-Models Improve Downstream Generation?

Having established that RUMS makes effective decisions about when and what to personalize (H1-H2), we now demonstrate that these selections translate to tangible improvements in response quality. This tests our core claim: principled selection leads to better personalization.

Experimental Setup. We evaluate RUMS on both synthetic (PersonaFeedback and FreebaseQA, 100 samples each)

(a) Average number of memory items by method and dataset type.

Method	Synthetic	Real World
All	50.00	50.00
None	0.00	0.00
Random	5.00	5.00
Semantic Similarity	4.63	4.79
BM25	0.60	0.35
ReContriever	5.00	5.00
GPT4-Few	2.80	3.00
RUMS-Binary	1.58	2.22



(b) Estimated additional average monthly cost by method. Diagonal line indicate added cost of LLM calls for memory selection.

Figure 3. Cost Analysis

and real-world datasets (WildChat, 50 personalized and 50 non-personalized) to assess generalization. Responses are generated using two models: GPT-4, a high-performing model with robust capabilities, and LLaMA 70B Instruct (AI, 2024a), a smaller open-source model that is relatively more susceptible to noise. We evaluate response quality using RUMS-Utility and RUMS-Models against all baselines described in Section 4.1. However, we are most interested in RUMS-Models performance, since RUMS-Utility is much more computationally expensive to run at inference. Performance is measured by GPT-4 win rate using chain-of-thought prompting and confidence thresholding (Appendix F.7).

RUMS improves response quality across diverse settings Table 2 reports generation win rates comparing our method against the baselines synthetic and real-world datasets. On the synthetic dataset, RUMS-Binary consistently outperforms all baselines, achieving at least a 15% higher win rate when responses are generated using the less robust LLaMA 70B model. Most notably, it has an 18% winrate over GPT4-Fewshot, a model $400\times$ it's size. Most of these gains come from non-personalized inputs, where a lower-capability model struggles to ignore noisy items, reducing its performance. In contrast, when using a more capable model such as GPT-4 for response generation, RUMS-Multi outperforms almost all baselines, primarily driven by personalized inputs. In this setting, the model can leverage more items, including potentially noisy ones, while still filtering out unhelpful items to a large extent. Real-world results follow the same trend. RUMS-Binary achieves a 12% higher win rate than the best SOTA baseline, ReContriever with LLaMA 70B, while RUMS-Multi exceeds GPT4-FewShot by 11% and BM25/ReContriever by 13% with GPT-4. The *All* baseline performs reasonably with only 50 items, but prior work shows scaling degrades performance (Rajeev et al., 2025). See Appendix A.12 for breakdowns. These results were validated with a small-scale human evaluation as well (A.9). These results empirically validate Theorem 3.1: entropy-reducing memory subsets consistently improve response quality, confirming that predictive entropy serves as a reliable proxy for user utility.

RUMS-Models performs similarly to RUMS-Utility. We find, that similar to the results of H2, the downstream results using RUMS-Utility are most similar to RUMS-Binary. Also, in general, RUMS-Utility performs very strongly against all baselines, with over 10% better winrate against almost all baselines when generating with LLaMA 70B, and all low-level baselines (None, All, Random, and BM25) when generating with GPT-4.

RUMS-Models reduces cost compared to other underperforming methods. Because personalization may be applied to every model input, efficiency is critical. We track both the number of memory items selected and the estimated monthly inference cost, computed as the extra tokens from selected items multiplied by a pricing model of \$0.01 per 1K tokens for one million daily queries (current GPT-4 rates). Figure 3a shows that RUMS-Models select substantially fewer memory items than most baselines, translating to significant cost savings (Figure 3b): RUMS-Binary costs \$6.9K/month and RUMS-Multi costs \$8.4K/month, compared to \$165.3K for All and \$18.6K for the competitive ReContriever. Both RUMS variants are cheaper than GPT4-FewShot, even when the additional cost of memory subset selection required by GPT4 baselines is excluded. Thus, RUMS balances informativeness and efficiency, enabling scalable personalization.

5. Conclusion

We introduced RUMS, an information-theoretic method for user memory selection that leverages response-aware utility as a more reliable signal for personalization than input-similarity. Our results show that RUMS accurately identifies when personalization is needed, selects memory subsets that align better with human judgment than SOTA baselines, and improves downstream generation quality while reducing computational cost by up to 95%. Future work includes applying RUMS to multi-turn conversations, enhancing RUMS-Models with stronger user profile generation, and extending evaluation across a broader range of LLMs.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.

The above statement can be used verbatim in such cases, but we encourage authors to think about whether there is content which does warrant further discussion, as this statement will be apparent if the paper is later flagged for ethics review.

References

AI, M. Llama 3.1 70b instruct, 2024a. URL https://huggingface.co/meta-llama/Llama-3.1-70B-Instruct. Accessed: 2025-09-16.

AI, M. Llama-3.1-8b, 2024b. URL https://huggingface.co/meta-llama/Llama-3.1-8B. Accessed: 2025-09-16.

Alemi, A. A., Fischer, I., Dillon, J. V., and Murphy, K. Deep variational information bottleneck. *International Conference on Learning Representations*, 2017.

Almudévar, A., Hernandez-Lobato, J. M., Khurana, S., Marxer, R., and Ortega, A. Aligning multimodal representations through an information bottleneck. In *Proceedings of the 2025 International Conference on Machine Learning (ICML 2025), Poster Session*, 2025. URL https://icml.cc/virtual/2025/poster/43456.

Bose, A., Xiong, Z., Chi, Y., Du, S. S., Xiao, L., and Fazel, M. Lore: Personalizing LLMs via low-rank reward modeling, 2025. URL https://arxiv.org/abs/2504.14439.

Brown, P. F., Della Pietra, S. A., Della Pietra, V. J., Lai, J. C., and Mercer, R. L. An estimate of an upper bound for the entropy of English. *Computational Linguistics*, 18(1):31–40, 1992. URL https://aclanthology.org/J92-1002/.

Chen, J. Memory assisted LLM for personalized recommendation system. 2025. URL https://arxiv.org/abs/2505.03824.

Chen, J., Song, L., Wainwright, M., and Jordan, M. Learning to explain: An information-theoretic perspective on model interpretation. In Dy, J. and Krause, A. (eds.), *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, volume 80 of *Proceedings of Machine Learning Research*, pp. 883–892. PMLR, 10–15 Jul 2018. URL https://proceedings.mlr.press/v80/chen18j.html.

Covert, I., Qiu, W., Lu, M., Kim, N., White, N., and Lee, S.-I. Learning to maximize mutual information for dynamic feature selection. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Dong, Y. R., Hu, T., and Collier, N. Can LLM be a personalized judge? In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2024*, pp. 10126–10141, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-emnlp.592. URL https://aclanthology.org/2024.findings-emnlp.592/.

Gadgil, S., Covert, I., and Lee, S.-I. Estimating conditional mutual information for dynamic feature selection. In *Proceedings of the 42nd International Conference on Learning Representations (ICLR)*. OpenReview.net, 2024. URL https://openreview.net/pdf?id=Oju2Qu9jvn.

Geifman, Y. and El-Yaniv, R. Selective classification for deep neural networks. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, pp. 4885–4894, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.

Gemma Team. Gemma 2: Improving open language models at a practical size, 2024. URL https://arxiv.org/abs/2408.00118.

Gneiting, T. and Raftery, A. E. Strictly proper scoring rules, prediction, and estimation. *Journal of the American Statistical Association*, 102(477):359–378, 2007.

Hagström, L., Kim, Y., Yu, H., goo Lee, S., Johansson, R., Cho, H., and Augenstein, I. Cub: Benchmarking context utilisation techniques for language models, 2025. URL https://arxiv.org/abs/2505.16518.

Handa, K., Gal, Y., Pavlick, E., Goodman, N., Andreas, J., Tamkin, A., and Li, B. Z. Bayesian preference elicitation with language models, 2024. URL https://arxiv.org/abs/2403.05534.

He, J. Z.-Y., Pandey, S., Schrum, M. L., and Dragan, A. Context steering: Controllable personalization at inference time. In *Proceedings of the 13th International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=xQCXInDq0m.

He, P., Liu, X., Gao, J., and Chen, W. DEBERTA: Decoding-enhanced BERT with disentangled attention. In *International Conference on Learning Representations*,

2021. URL https://openreview.net/forum?id=XPZIaotutsD.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Renard Lavaud, L., Lachaux, M.-A., Stock, P., Le Scao, T., Lavril, T., Wang, T., Lacroix, T., and El Sayed, W. Mistral 7B, 2023. URL https://arxiv.org/abs/2310.06825.

Jiang, K., Wu, D., and Jiang, H. FreebaseQA: A new factoid QA data set matching trivia-style question-answer pairs with Freebase. In Burstein, J., Doran, C., and Solorio, T. (eds.), *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 318–323, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1028. URL https://aclanthology.org/N19-1028/.

Joren, H., Nagpal, C., Heller, K. A., and Ustun, B. Participatory personalization in classification. In Oh, A., Naumann, T., Globerson, A., Saenko, K., Hardt, M., and Levine, S. (eds.), *Advances in Neural Information Processing Systems*, volume 36, pp. 14113–14133. Curran Associates, Inc., 2023.

Kang, W.-C., Ni, J., Mehta, N., Sathiamoorthy, M., Hong, L., Chi, E., and Cheng, D. In *Do LLMs Understand User Preferences? Evaluating LLMs On User Rating Prediction*, 05 2023. doi: 10.48550/arXiv.2305.06474.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z. R., Stevens, K., Barhoum, A., Nguyen, D., Stanley, O., Nagyfi, R., ES, S., Suri, S., Glushkov, D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., and Mattick, A. OpenAssistant conversations: Democratizing large language model alignment. In *NeurIPS 2023 Poster*, 2023. URL https://neurips.cc/virtual/2023/poster/73573.

Lei, Y., Ding, L., Cao, Y., Zan, C., Yates, A., and Tao, D. Unsupervised dense retrieval with relevance-aware contrastive pre-training. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 10932–10940, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.695. URL https://aclanthology.org/2023.findings-acl.695/.

Li, Y. Unlocking context constraints of LLMs: Enhancing context efficiency of LLMs with self-information-based content filtering. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP 2023)*, 2023. URL https://arxiv.org/abs/2304.12102.

Li, Y., Liang, D., Zhang, S., Yin, C., Xu, S., Yue, F., Zhou, G., Wei, Y., Xu, H., Wang, Y., and Zeng, J.-Y. Compressing context to enhance inference efficiency of large language models. In Bouamor, H., Pino, J., and Bali, K. (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 6348–6360, Singapore, December 2023. Association for Computational Linguistics. URL https://aclanthology.org/2023.emnlp-main.391.

Liu, J., Liu, C., Zhou, P., Lv, R., Zhou, K., and Zhang, Y. Is ChatGPT a good recommender? a preliminary study, 2023. URL https://arxiv.org/abs/2304.10149.

Liu, J., Qiu, Z., Li, Z., Dai, Q., Zhu, J., Hu, M., Yang, M., and King, I. In *A Survey of Personalized Large Language Models: Progress and Future Directions*, 02 2025. doi: 10.48550/arXiv.2502.11528.

Lù, X. H. Bm25s: Orders of magnitude faster lexical search via eager sparse scoring, 2024. URL https://arxiv.org/abs/2407.03618.

Ma, C., Tschiatschek, S., Palla, K., Hernández-Lobato, J. M., Nowozin, S., and Zhang, C. Eddi: Efficient dynamic discovery of high-value information with partial vae. *ICLR*, 2019. doi: 10.48550/arXiv.1809.11142.

Mysore, S., Lu, Z., Wan, M., Yang, L., Sarrafzadeh, B., Menezes, S., Baghaee, T., Gonzalez, E. B., Neville, J., and Safavi, T. Pearl: Personalizing large language model writing assistants with generation-calibrated retrievers. In Kumar, S., Balachandran, V., Park, C. Y., Shi, W., Hayati, S. A., Tsvetkov, Y., Smith, N., Hajishirzi, H., Kang, D., and Jurgens, D. (eds.), *Proceedings of the 1st Workshop on Customizable NLP: Progress and Challenges in Customizing NLP for a Domain, Application, Group, or Individual (CustomNLP4U)*, pp. 198–219, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.customnlp4u-1.16. URL https://aclanthology.org/2024.customnlp4u-1.16/.

OpenAI, Achiam, J., Adler, S., Agarwal, S., Ahmad, L., Akkaya, I., Aleman, F. L., Almeida, D., Altenschmidt, J., Altman, S., Anadkat, S., Avila, R., Babuschkin, I., Balaji, S., Balcom, V., Baltescu, P., Bao, H., Bavarian, M., Belgum, J., Bello, I., Berdine, J., Bernadett-Shapiro, G., Berner, C., Bogdonoff, L., Boiko, O., Boyd, M., Brakman, A.-L., Brockman, G., Brooks, T., Brundage, M., Button, K., Cai, T., Campbell, R., Cann, A., Carey, B., Carlson, C., Carmichael, R., Chan, B., Chang, C., Chantzis, F., Chen, D., Chen, S., Chen, R., Chen, J., Chen, M., Chess,

B., Cho, C., Chu, C., Chung, H. W., Cummings, D., Currier, J., Dai, Y., Decareaux, C., Degry, T., Deutsch, N., Deville, D., Dhar, A., Dohan, D., Dowling, S., Dunning, S., Ecoffet, A., Eleti, A., Eloundou, T., Farhi, D., Fedus, L., Felix, N., Fishman, S. P., Forte, J., Fulford, I., Gao, L., Georges, E., Gibson, C., Goel, V., Gogineni, T., Goh, G., Gontijo-Lopes, R., Gordon, J., Grafstein, M., Gray, S., Greene, R., Gross, J., Gu, S. S., Guo, Y., Hallacy, C., Han, J., Harris, J., He, Y., Heaton, M., Heidecke, J., Hesse, C., Hickey, A., Hickey, W., Hoeschele, P., Houghton, B., Hsu, K., Hu, S., Hu, X., Huizinga, J., Jain, S., Jain, S., Jang, J., Jiang, A., Jiang, R., Jin, H., Jin, D., Jomoto, S., Jonn, B., Jun, H., Kaftan, T., Łukasz Kaiser, Kamali, A., Kanitscheider, I., Keskar, N. S., Khan, T., Kilpatrick, L., Kim, J. W., Kim, C., Kim, Y., Kirchner, J. H., Kiros, J., Knight, M., Kokotajlo, D., Łukasz Kondraciuk, Kondrich, A., Konstantinidis, A., Kosic, K., Krueger, G., Kuo, V., Lampe, M., Lan, I., Lee, T., Leike, J., Leung, J., Levy, D., Li, C. M., Lim, R., Lin, M., Lin, S., Litwin, M., Lopez, T., Lowe, R., Lue, P., Makanju, A., Malfacini, K., Manning, S., Markov, T., Markovski, Y., Martin, B., Mayer, K., Mayne, A., McGrew, B., McKinney, S. M., McLeavey, C., McMillan, P., McNeil, J., Medina, D., Mehta, A., Menick, J., Metz, L., Mishchenko, A., Mishkin, P., Monaco, V., Morikawa, E., Mossing, D., Mu, T., Murati, M., Murk, O., Mély, D., Nair, A., Nakano, R., Nayak, R., Neelakantan, A., Ngo, R., Noh, H., Ouyang, L., O'Keefe, C., Pachocki, J., Paino, A., Palermo, J., Pantuliano, A., Parascandolo, G., Parish, J., Parparita, E., Passos, A., Pavlov, M., Peng, A., Perelman, A., de Avila Belbute Peres, F., Petrov, M., de Oliveira Pinto, H. P., Michael, Pokorny, Pokrass, M., Pong, V. H., Powell, T., Power, A., Power, B., Proehl, E., Puri, R., Radford, A., Rae, J., Ramesh, A., Raymond, C., Real, F., Rimbach, K., Ross, C., Rotsted, B., Roussez, H., Ryder, N., Saltarelli, M., Sanders, T., Santurkar, S., Sastry, G., Schmidt, H., Schnurr, D., Schulman, J., Selsam, D., Sheppard, K., Sherbakov, T., Shieh, J., Shoker, S., Shyam, P., Sidor, S., Sigler, E., Simens, M., Sitkin, J., Slama, K., Sohl, I., Sokolowsky, B., Song, Y., Staudacher, N., Such, F. P., Summers, N., Sutskever, I., Tang, J., Tezak, N., Thompson, M. B., Tillet, P., Tootoonchian, A., Tseng, E., Tuggle, P., Turley, N., Tworek, J., Uribe, J. F. C., Vallone, A., Vijayvergiya, A., Voss, C., Wainwright, C., Wang, J. J., Wang, A., Wang, B., Ward, J., Wei, J., Weinmann, C., Welihinda, A., Welinder, P., Weng, J., Weng, L., Wiethoff, M., Willner, D., Winter, C., Wolrich, S., Wong, H., Workman, L., Wu, S., Wu, J., Wu, M., Xiao, K., Xu, T., Yoo, S., Yu, K., Yuan, Q., Zaremba, W., Zellers, R., Zhang, C., Zhang, M., Zhao, S., Zheng, T., Zhuang, J., Zhuk, W., and Zoph, B. GPT-4 technical report, 2024. URL https://arxiv.org/abs/2303.08774.

Pan, Z., Wu, Q., Jiang, H., Luo, X., Cheng, H., Li, D., Yang, Y., Lin, C.-Y., Zhao, H. V., Qiu, L., and Gao, J. Secom: On memory construction and retrieval for personalized conversational agents. In *The Thirteenth International Conference on Learning Representations (ICLR)*, 2025. URL https://openreview.net/forum?id=xKDZAW0He3.

Rajeev, M. A., Ramamurthy, R., Trivedi, P., Yadav, V., Bamgbose, O., Madhusudhan, S. T., Zou, J., and Rajani, N. Cats confuse reasoning LLM: Query agnostic adversarial triggers for reasoning models. In *Second Conference on Language Modeling*, 2025. URL https://openreview.net/forum?id=VrEPiN5WhM.

Ravfogel, S., Elazar, Y., Gonen, H., Twiton, M., and Goldberg, Y. Null it out: Guarding protected attributes by iterative nullspace projection. In Jurafsky, D., Chai, J., Schluter, N., and Tetreault, J. (eds.), *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7237–7256, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.647. URL https://aclanthology.org/2020.acl-main.647/.

Richardson, C., Zhang, Y., Gillespie, K., Kar, S., Singh, A., Raeesy, Z., Khan, O. Z., and Sethy, A. Integrating summarization and retrieval for enhanced personalization via large language models. 2023. URL https://arxiv.org/abs/2310.20081.

Salemi, A., Kallumadi, S., and Zamani, H. Optimization methods for personalizing large language models through retrieval augmentation. SIGIR '24, pp. 752–762, New York, NY, USA, 2024a. Association for Computing Machinery. ISBN 9798400704314. doi: 10.1145/3626772.3657783. URL https://doi.org/10.1145/3626772.3657783.

Salemi, A., Mysore, S., Bendersky, M., and Zamani, H. LaMP: When large language models meet personalization, August 2024b. URL https://aclanthology.org/2024.acl-long.399/.

Sorensen, T., Moore, J., Fisher, J., Gordon, M., Mireshghallah, N., Rytting, C. M., Ye, A., Jiang, L., Lu, X., Dziri, N., Althoff, T., and Choi, Y. Position: a roadmap to pluralistic alignment. In *Proceedings of the 41st International Conference on Machine Learning*, ICML'24. JMLR.org, 2024.

Tan, Z., Liu, Z., and Jiang, M. Personalized pieces: Efficient personalized large language models through collaborative efforts. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 6459–6475. Association for Computational Linguistics, 2024. URL https://aclanthology.org/2024.emnlp-main.371.pdf.

Tao, M., Zhu, C., Ding, D., Wang, T., Jiang, Y. E., and Zhou, W. Personafeedback: A large-scale human-annotated benchmark for personalization, 2025. URL https://arxiv.org/abs/2506.12915.

Tiedemann, J. and Thottingal, S. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal, 2020.

Tishby, N., Pereira, F. C., and Bialek, W. The information bottleneck method. *arXiv preprint physics/0004057*, 1999.

Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. Minilm: Deep self-attention distillation for task-agnostic compression of pre-trained transformers, 2020. URL https://arxiv.org/abs/2002.10957.

Zhang, H., Zhang, X., Huang, H., and Yu, L. Prompt-based meta-learning for few-shot text classification. In Goldberg, Y., Kozareva, Z., and Zhang, Y. (eds.), *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 1342–1357, Abu Dhabi, United Arab Emirates, December 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.emnlp-main.87. URL https://aclanthology.org/2022.emnlp-main.87/.

Zhao, W., Ren, X., Hessel, J., Cardie, C., Choi, Y., and Deng, Y. Wildchat: 1m chatGPT interaction logs in the wild. In *The Twelfth International Conference on Learning Representations (ICLR 2024), Spotlight Poster*, 2024. URL https://openreview.net/forum?id=Bl8u7ZRlbM.

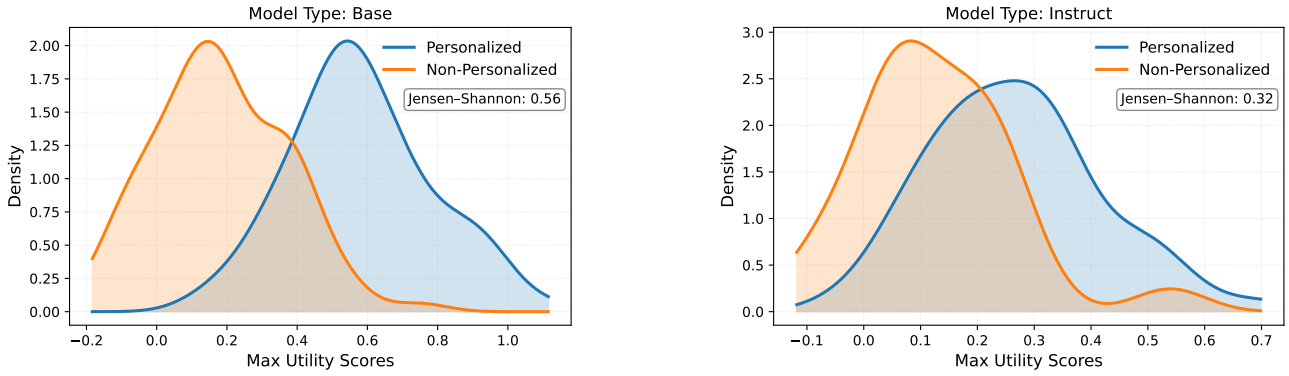
A. Additional Experimentation

In this section, we present additional experiments to supplement our main results.

A.1. Robustness Analysis of Utility Scores

In H1, we hypothesized that RUMS-Utility, the maximum utility scores, can reliably distinguish between user inputs that would benefit from personalization and those that would not. Here, we further investigate how hyperparameters in the RUMS-Utility algorithm affect this reliability and provide justification for these effects. Specifically, we examine the influence of model type, model size, model family, decoding method type, maximum number of tokens generated, number of Monte Carlo samples, and number of user profiles. For each analysis, all other variables are held constant using the LLaMa base model, 8B size, sampling decoding, $T = 20$ tokens, $n = 1$ sample, and averaged over $n = 3$ user profiles.

Hyperparameter: Model Type We first compare the LLaMa-8B base model to an LLaMa-8B instruction-tuned model. As shown in Figure 4, the base model produces more separated distributions of utility scores than the instruct model, with a Jensen-Shannon divergence score of 0.56 compared to 0.32 for the instruct model. This is consistent with prior research showing that instruction-tuning tends to pre-sharpen output distributions (Sorensen et al., 2024). Based on this observation, we chose to use the base model for our experiments.



(a) Model Type: Base

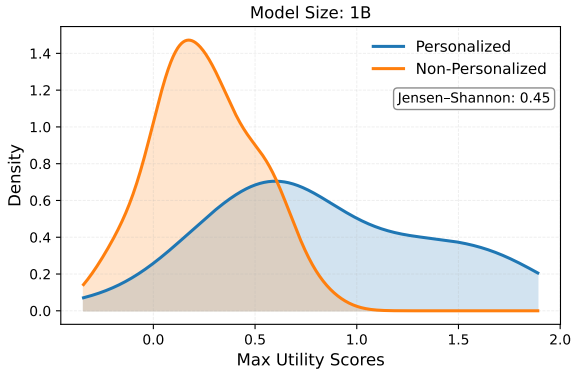
(b) Model Type: Instruct

Figure 4. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset differing by type of model used for entropy calculation.

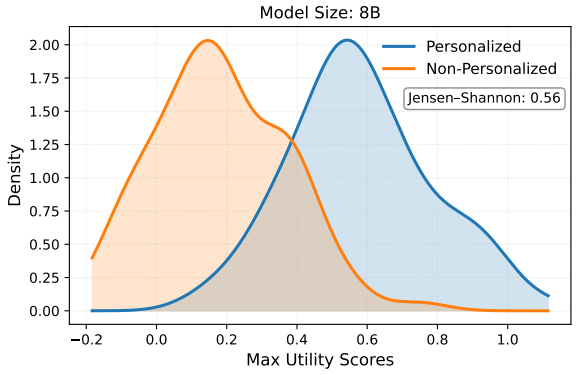
Hyperparameter: Model Size Next, we examined the effect of model size on the entropy calculation. As shown in Figure 5, the distribution of utility scores changes when using LLaMA-1B compared to LLaMA-8B. While the impact is less pronounced than the effect of model type, we observe an increase of 0.11 in the Jensen-Shannon divergence score when using the 8B model. We hypothesize that larger models are better able to capture subtle and complex patterns in their response distributions. Given that the improvement is relatively modest, we opted to use the 8B model to balance performance with computational efficiency and avoid the added cost of larger models.

Hyperparameter: Model Family We next examine the effect of model family, comparing Llama-8b (AI, 2024b), Mistral-7b (Jiang et al., 2023), and Gemma-9B (Gemma Team, 2024). As shown in Figure 6, we find that Mistral model performs similar to Llama (0.41 vs. 0.56), where the Gemma model performs badly (0.17). This could indicate that some model families better capture human preference than others. Based on this observation, we use Llama-7B in our experiments.

Hyperparameter: Decoding Type We next examine the effect of decoding type on the sequential average token entropy, comparing greedy and sampling decoding. As shown in Figure 7, sampling decoding produces more diverse distributions than greedy decoding. This is expected, as sampling introduces stochasticity into token selection, allowing the model to explore multiple plausible continuations, whereas greedy decoding always selects the highest-probability token at each step, leading to more concentrated and less variable distributions. Based on this observation, we use sampling decoding in our experiments.

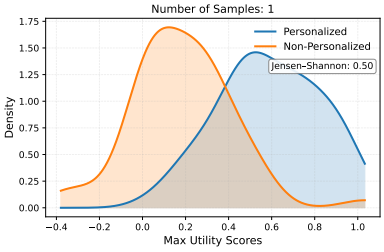


(a) Model Size: 1B

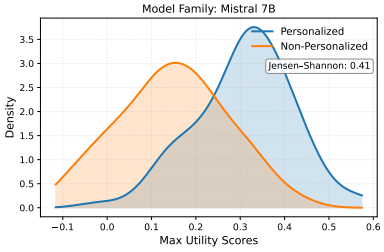


(b) Model Size: 8B

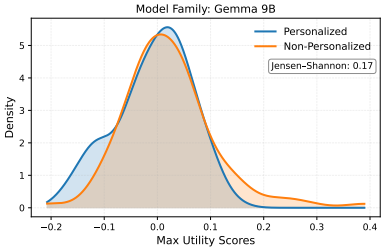
Figure 5. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset differing by size of model used for entropy calculation.



(a) Model Family: Llama

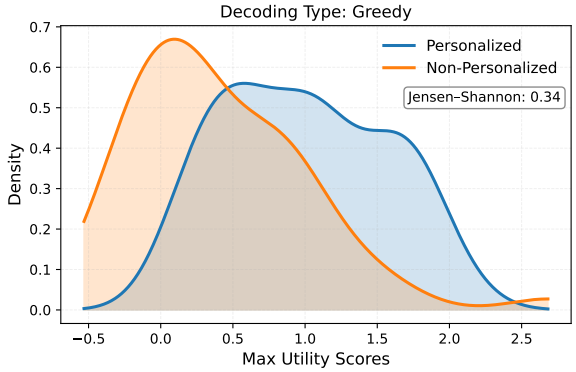


(b) Model Family: Mistral

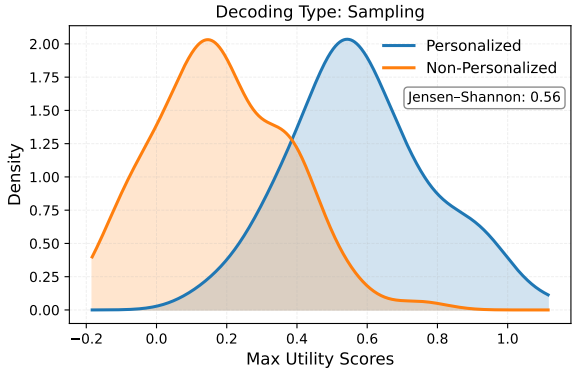


(c) Model Family: Gemma

Figure 6. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset calculated using models of different families.



(a) Decoding Type: Greedy



(b) Decoding Type: Sampling

Figure 7. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset differing by decoding type.

Hyperparameter: Number of Tokens As noted in the main text, we limit the number of tokens used in the calculation of Equation (3) to avoid bias introduced by longer generations, since longer outputs tend to have lower entropy. To study this, we randomly sampled $n = 10{,}000$ responses from the OpenAssistant dataset (Köpf et al., 2023), using only the first human input in each conversation that was labeled as English and non-toxic. We used this dataset since it is a real world dataset that could provide a wide diversity in types of user-LLM interactions. We then computed the average token-level entropy by token index, capped at 100 tokens. As shown in Figure 8, entropy begins to decrease significantly around $T = 20$ tokens, indicating that longer generations introduce a bias toward lower entropy.

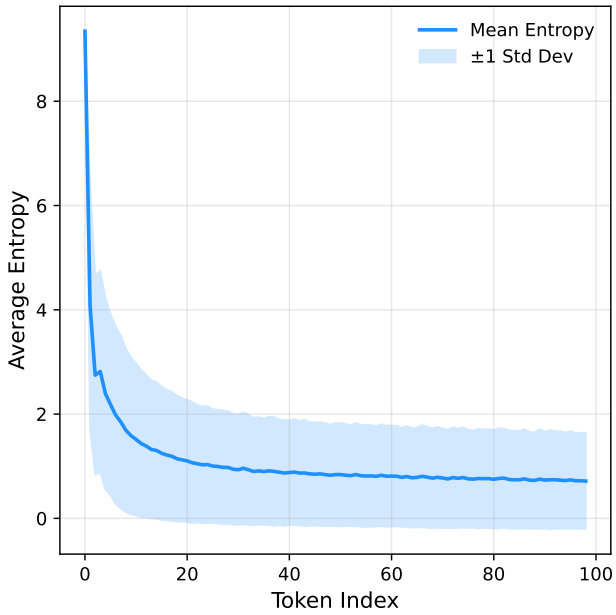
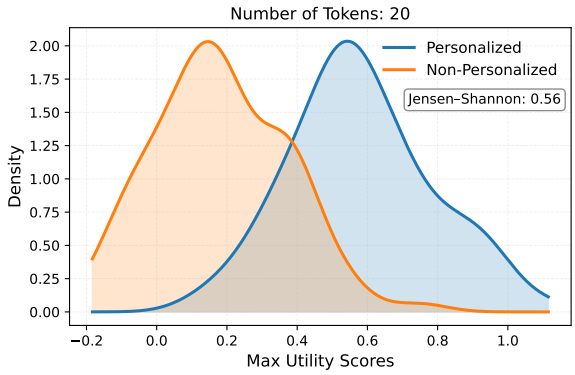
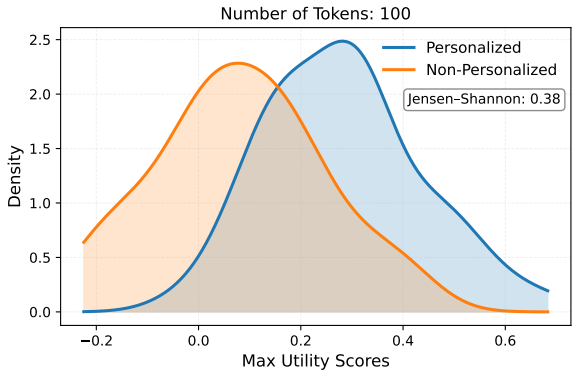


Figure 8. Average entropy by the index of the token being generated.

Further analysis in Figure 9 compares the distribution of maximum utility scores when using 20 versus 100 maximum tokens for entropy calculations. Using 100 tokens reduces the Jensen-Shannon divergence by 0.18 compared to using only 20 tokens. Based on these results, we limit the calculation to the first 20 tokens in our experiments.



(a) Number of Tokens: 20

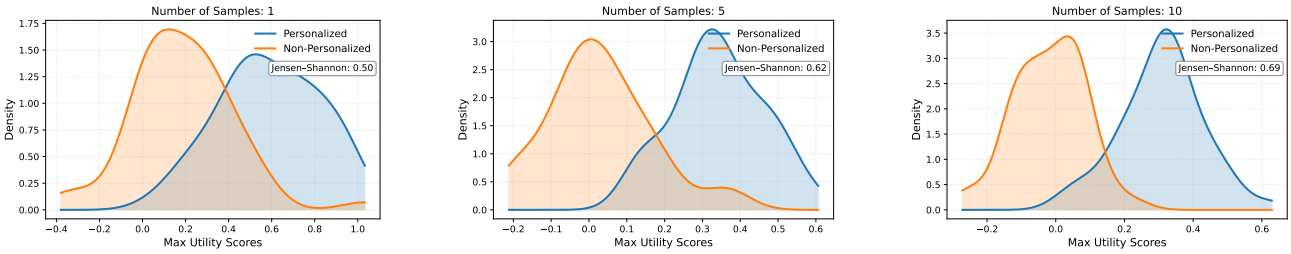


(b) Number of Tokens: 100

Figure 9. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset using different number of tokens.

Hyperparameter: Number of Samples Next, we examine the effect of the number of Monte Carlo samples used in Equation (3) on the robustness of the utility score. As shown in Figure 10, increasing the number of samples improves the separation between personalized and non-personalized inputs, which aligns with expectations since the estimate should converge toward the true parameter as the sample size increases if the samples are i.i.d. However, this comes with a tradeoff: more samples require additional computation, longer runtime, and higher resource costs. Balancing these considerations, we chose $n = 5$ samples for our experiments.

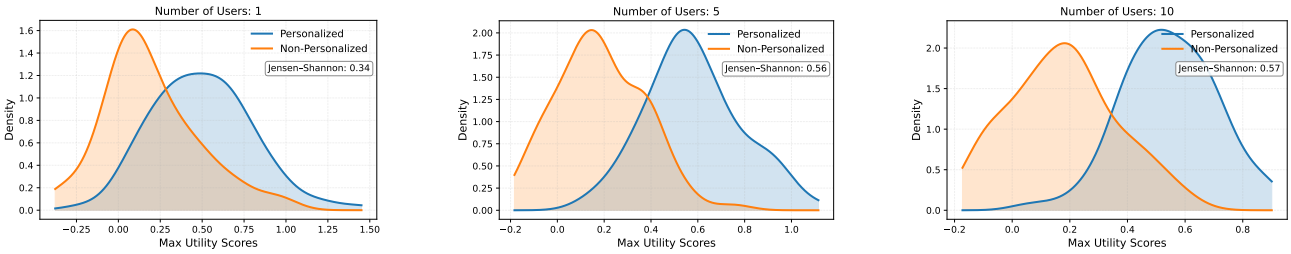
Hyperparameter: Number of User Profiles Lastly, we examine how the robustness of the average maximum utility score changes as we increase the number of (query,user profile) pairs included in the average. Consistent with estimation theory, we find that the difference between the personalized and non-personalized distributions grows as the sample size



(a) Number of Samples: 1 *(b)* Number of Samples: 5 *(c)* Number of Samples: 10

Figure 10. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset differing number of Monte Carlo samples average over.

increases. Notably, there is a substantial jump of 0.22 in Jensen–Shannon divergence when increasing from 1 to 5 users, but only a marginal increase of 0.01 when going from 5 to 10 users. Based on this, we choose $u = 5$ users for constructing our training data to balance performance with cost of inference.



(a) Number of Users: 1 *(b)* Number of Users: 5 *(c)* Number of Users: 10

Figure 11. Distribution of maximum utility scores for a personalized (blue) and non-personalized (orange) dataset differing by number of users averaged over.

A.2. Human-Chosen Memory Subset Performance on Downstream Response

In H2, we analyze the correlation between the memory subsets selected by RUMS-Utility, RUMS-Model and the baselines compared to human-selected subsets. Human labels are treated as the gold standard, as we hypothesize that they would lead to better downstream responses. To evaluate this, we calculate the generation win rate of responses produced using human-selected features compared to those produced by RUMS-Model and the baselines. We use the same evaluation technique outlined in Section F. As shown in Table 3, human-selected memory subsets yield better downstream responses than all baselines when using the less capable response model for generation (LLaMA 70B). With the closes' performance being RUMS-Binary. When generating responses with a more capable model, GPT-4, we still see slight better performance using the human selected memory items, although much less than when using a less capable mode. This shows that human chosen memory items should be used in H2 as a measure of a good subset of memory.

A.3. Analysis of Approximation Gap Between RUMS-Utility and RUMS-Models

A.3.1. Analysis of RUMS-Models Signal Learning

To better understand what signals the RUMS-Models are learning, we add an analysis which explores the correlation between memory items selected (included) and not selected (excluded) by these models, and the connection to the true RUMS-Utility score. Since we are interested in personalization, we use the PersonaFeedback test set (n=100).

In Figure 12 and Figure 13, we show the distribution of average utility score per sample. In Figure 14 and Figure 15, we show the distribution of difference of average utility score for included - excluded features per sample. We find the distribution of included and excluded utility scores are statistically different (p-value ≤ 0.03, Mann-Whitney test). Similarly, We find that the distribution of differences between the average utility score for items the RUMS-model included and the average utility score of memory items the RUMS-Model excluded is greater than 0 (p-value $< .001$, Wilcoxon one-sided

Table 3. Win rate of responses preferred using human selected memory subsets compared to other methods. A number above 0 is the percentage of responses generated by Humans chosen subsets that were preferred.

	LLaMA 70B	GPT-4
Method	Human Win Rate	Human Win Rate
None	**0.31**	**0.16**
All	**0.28**	**-0.26**
Random	**0.42**	0.06
Semantic Similarity	**0.38**	0.01
BM25	**0.18**	**0.26**
ReContriever	**0.52**	0.01
GPT4-Few	**0.27**	0.01
RUMS-Binary	0.01	-0.05
RUMS-Multi	**0.39**	0.02

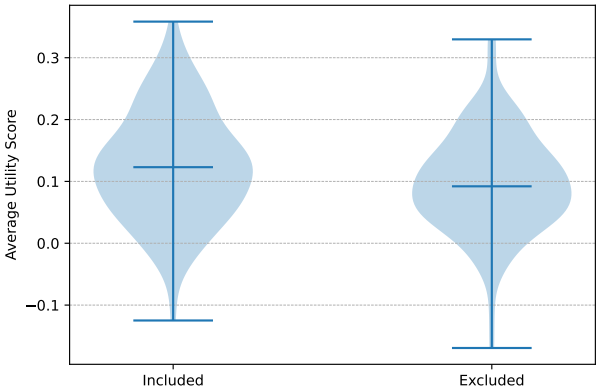


Figure 12. RUMS-Binary: Distribution

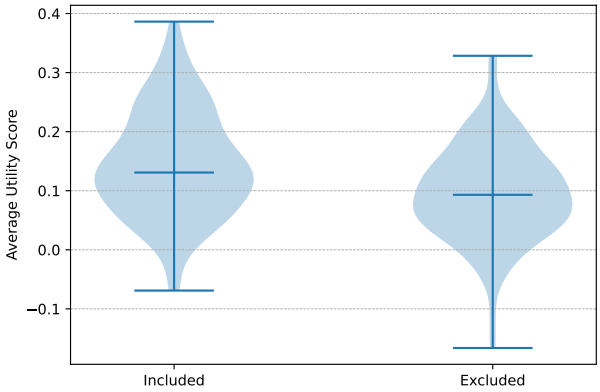


Figure 13. RUMS-Multi: Distribution

test), meaning that, on average, we find that memory items the RUMS-Model include have a truly higher utility score than those it does not. This lends strong evidence that RUMS-Models correctly learn signals of the raw utility scores.

A.3.2. Comparing RUMS-Utility to RUMS-Models on H2

RUMSAs mentioned in the main paper, we opt to use RUMS-Models, rather than the raw RUMS-Utility, due to its high efficacy and large decrease in computational compute. This is a common approach when using BOED methods, due to algorithmic components which make it time intensive. Although our H2 results show that RUMS-Models have similar, but lower, efficacy compared to RUMS-Utility, we want to further analyze this approximation gap.

To do this, we expand the results for H2. First, we calculated a majority feature list from the RUMS-Utility chosen data for the samples in this section, by including all features that were in a majority (¿=5) samples using the RUMS utility scores directly over 5 diverse profiles. Then, we calculated the same metrics (precision, recall, and F1), of this new gold label group to feature list selected by RUMS-Binary and RUMS-Multi.

In general, we find that the models have high agreement on in distribution datasets (synthetic) but perform moderate on out of distribution dataset (real world). In line with our results for H2, we find that RUMS-Binary has very high correlation with RUMS-Utility with a high F1 of 0.78 on the synthetic dataset. Although RUMS-Binary only has a moderate F1 of $.37$, we note that most of this comes from a decrease in performance on non-personalized datasets and the over-selection of RUMS-Multi. This is highlighted by the higher recall scores, in general, and slightly lower precision (e.g. 0.9 vs. $.73$ for RUMS-Binary).

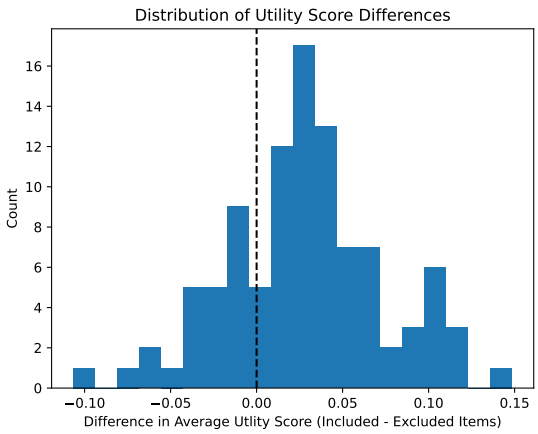


Figure 14. RUMS-Binary: Difference

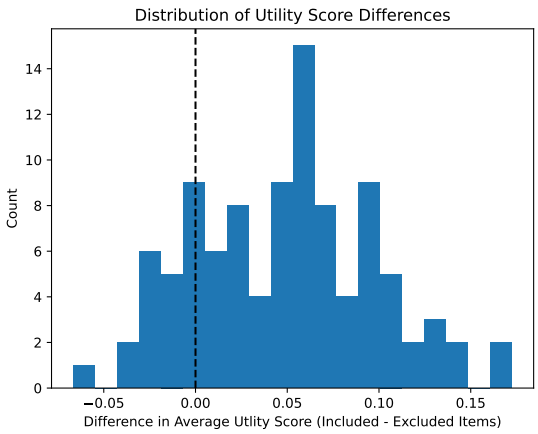


Figure 15. RUMS-Multi: Difference

Table 4. Precision, recall, and F1 scores of memory items selected compared to human gold label annotation across datasets and methods.

Dataset	Metric	RUMS-Binary	RUMS-Multi
Synthetic	Precision	0.73	0.33
	Recall	0.90	0.44
	F1	0.78	0.37
Real World	Precision	0.26	0.35
	Recall	0.42	0.35
	F1	0.32	0.34

A.4. Approximation of RUMS-Multi using RUMS-Utility

The RUMS-Utility objective evaluates subsets of memory items jointly, capturing interactions between features. In contrast, RUMS-Binary predicts each item independently. Since conditional mutual information is not generally additive, this raises the question of how much combinatorial information is lost when using the item-wise approximation.

To quantify this gap, we provide both a theoretical decomposition and empirical measurement.

Theoretical Decomposition Let $S = \{m_1, ..., m_k\}$ be a subset of memory items. The utility function used in RUMS is

$$U(S) = I(Y; M_S|x).$$

Using the chain rule of conditional mutual information,

$$I(Y; M_S|x) = \sum_{i=1}^{k} I(Y; m_i|x, m_{<i}),$$

where $m_{<i} = \{m_1, ..., m_{i=1}\}$.

Next, the independent approximation used by RUMS-Binary replaces conditional gains with marginal gains,

$$I(Y; m_i|x, m_{<i}) \approx I(Y; m_i|x).$$

This yields the additive approximation,

$$\tilde{U}(S) = \sum_{i=1}^{k} I(Y; m_i|x).$$

Dataset	Subset Gap	Sequential Gap
Synthetic	0.23	0.11
Real World	1.01	0.09

Table 5. Results from the analysis of the combinatorial approximation gap between RUMS-Utility and RUMS-Binary. We show the average normalized subset interaction gap and the average sequential conditional interaction gap for personalized datasets.

Therefore, the combinatorial interaction gap is expressed,

$$\Delta(S) = U(S) - \tilde{U}(S) = \sum_{i=1}^{k} \left(I(Y; m_i | x, m_{<i}) - I(Y; m_i | x) \right). \tag{5}$$

Therefore, Equation (5) shows that the total approximation error equals the sum of per-feature interaction terms.

Empirical Measurement To better understand this approximation error in our experimentation, we estimate interaction effects using the utilities computed by RUMS-Utility for the test set of human annotated examples (n=60) using only the personalized datasets. For each query and selected subset S in our four test sets, we compute two metrics.

1. **Subset Interaction Gap**: $\Delta(S) = U(S) - \sum_{i \in S} U(\{\})$, which measures the total combinatorial gain.
2. **Sequential Conditional Interaction Gap** Given an ordering of memory items, we define $\delta_i = [U(S_i) - U(S_{i-1}] - U(\{i\})$, where S_i is the prefix subset. This measures how much the conditional contributions of each memory item deviates from independence.

Table 5 summarizes the measured combinatorial gaps across both synthetic and real-world datasets. We find the following:

- **Subset interaction gap**: The average normalized subset gap is 0.23 for the synthetic dataset and 1.01 for the real-world dataset, indicating that joint subsets can provide substantial additional utility beyond the sum of individual feature contributions.
- **Sequential conditional interaction gap**: The average per-feature sequential gap is 0.11 for synthetic data and 0.09 for real-world data, showing that individual features' conditional contributions deviate only slightly from the independent approximation.

These results highlight two key patterns. First, joint feature subsets offer additional utility, particularly in real-world datasets where the subset gap is larger. However, despite the larger subset-level gains, sequential conditional gaps remain small, suggesting that no single feature strongly violates the independence assumption. Together, these findings indicate that while higher-order interactions contribute to total utility, they are distributed across features and individually weak.

The analysis confirms that RUMS-Binary's item-wise approximation captures most of the subset utility. Even in datasets with substantial subset-level gains, the limited per-feature sequential gaps explain why RUMS-Binary performs well in practice while remaining computationally efficient. This supports its use as a scalable approximation to full subset-based RUMS-Utility selection.

A.5. RUMS-Utility Threshold Analysis

A.5.1. RUMS-Utility Threshold Effect on H1' Personalization Accuracy

We examined how personalization accuracy (agreement rate and F1), measured in H1', changes as we vary the threshold from 0 to 1 in 0.01 increments, on the test set used in our paper. For both datasets, Synthetic and Real World, we provide a graph comparing the metric (% agreement or F1) across thresholds. We also mark the optimal threshold (decided on a separate evaluation set), which we used in all experimentation. Lastly, we also show a zoomed-in view around the optimal threshold ± 0.005 for each graph (see Figure 16 - Figure 23).

In general, we found that personalization decisions are not very sensitive in respect to the threshold chosen. As we can see, for the Synthetic test set, when we change the threshold $+/-0.1$ we see only about a 15% decrease in agreement rate and only a 10% for the Real World Data. So, although the optimal threshold does seem to be heavily dataset dependent, it does seem moderately robust.

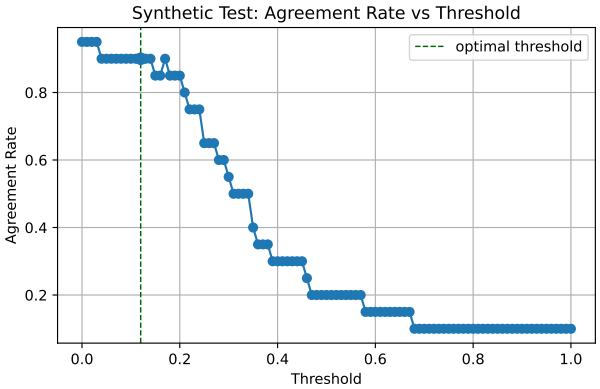


Figure 16. Synthetic: % Agreement vs. Threshold

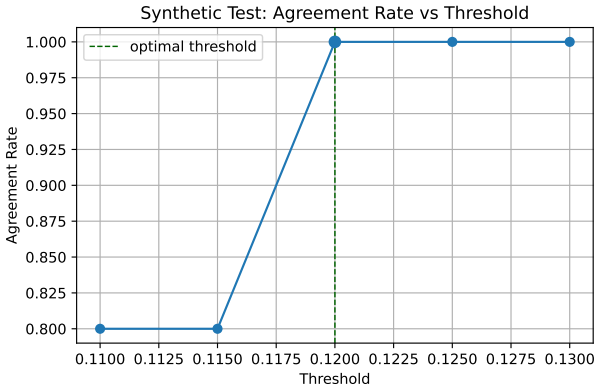


Figure 17. Synthetic: % Agreement vs. Threshold (zoomed in)

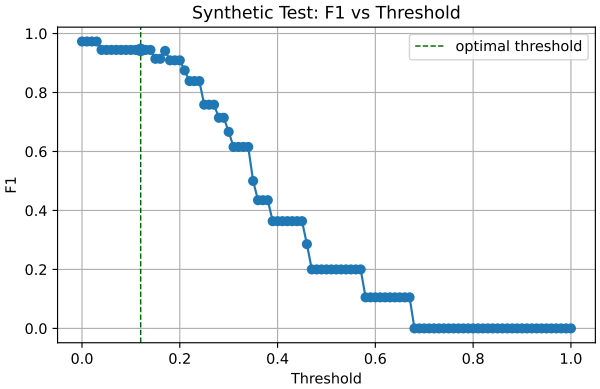


Figure 18. Synthetic: F1 vs. Threshold

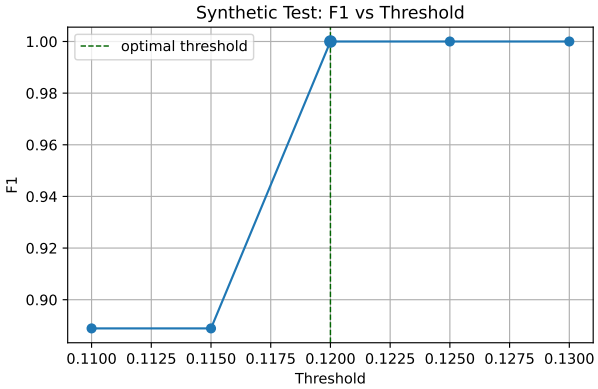


Figure 19. Synthetic: F1 vs. Threshold (zoomed in)

A.5.2. User-specific RUMS-Utility Threshold Effect on Personalization

We explore the idea of adaptive threshold per-user profile might change. To do this, we use a sample of $n = 100$ user inputs from FreebaseQA (non-personalized dataset) and PersonaFeedback (personalized dataset) respectively. For $u = 10$ static user profiles, we compute the maximum utility score for each. Then, we use the same procedure for finding the optimal threshold in the paper, to find the optimal threshold for each user. We then analyze the distribution of optimal thresholds for robustness using mean and standard deviations.

In Figure 24, we present the distribution of optimal thresholds for the 10 users (solid blue line), as well as the mean threshold (dotted red line) and ± 1 standard deviation (gray shaded area). We found that 70% users fell within a threshold of $[0.24, 0.30]$. So although there is some outlier profiles, we do not think there is enough variability that compensate for the added computational cost of per-user thresholds. We thank the reviewer for this query and find it interesting so we will add this analysis to our appendix.

A.5.3. RUMS-Utility Threshold Effect on Number of Features Selected

As mentioned in the main text, RUMS-Utility has a threshold hyperparameter that controls the sensitivity of memory item selection. Adjusting this threshold results in either more or fewer memory items being chosen. Table 6 illustrates how the average number of selected memory items changes as the threshold is varied. As the threshold increases, a stronger reduction in entropy is required for a memory item to be selected, leading to smaller average subset sizes. This analysis highlights that the threshold provides a simple yet effective way to balance the tradeoff between including more context and avoiding unnecessary or low-utility memory items.

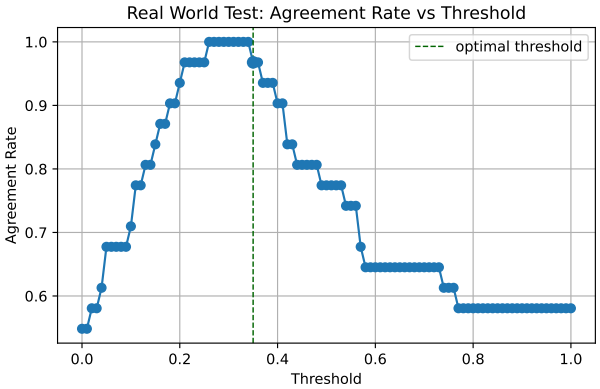


Figure 20. Real World: % Agreement vs. Threshold

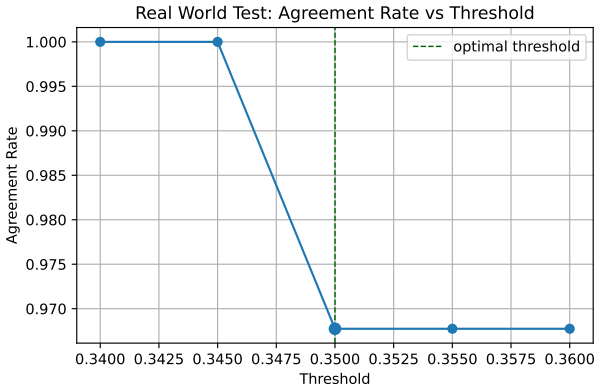


Figure 21. Real World: % Agreement vs. Threshold (zoomed in)

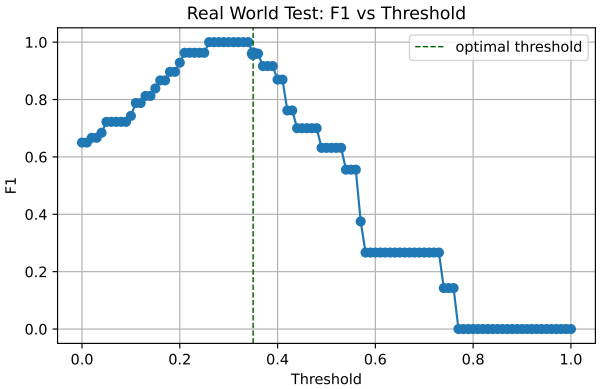


Figure 22. Real World:F1 vs. Threshold

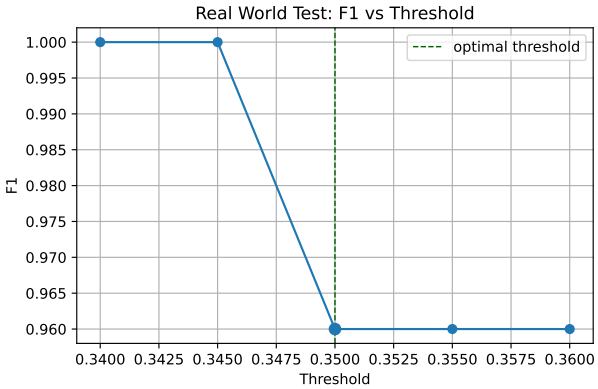


Figure 23. Real World: F1 vs. Threshold (zoomed in)

Table 6. Average number of memory items selected at different thresholds used in RUMS-Utility.

Threshold	PersonaFeedback	FreebaseQA	WildChat-Personal	WildChat-NonPersonal
-100	2.76	1.83	2.77	2.70
0.00	2.76	1.56	2.73	2.57
0.29	2.43	0.21	1.31	1.38
0.50	1.11	0.00	0.29	0.27
0.80	0.14	0.00	0.00	0..00

A.6. Examples of Selected Memory Items

Table 7 illustrates one personalized and one non-personalized case. Consistent with Table 1, GPT4-ZeroShot and Semantic Similarity over-select, adding noise in non-personalized cases; GPT4-FewShot selects less in general but still selects items for non-personalized inputs. RUMS-Binary balances personalization by matching human annotations in the personalized case and abstaining in the non-personalized case.

A.7. Cost Analysis

To better understand the practical implications of our approach, we analyze the tradeoff between computational cost and model performance across all methods. Since our methods and baselines differ in the number of memory items they select, and hence the number of additional tokens they introduce into the prompt, it is important to quantify how these differences affect overall efficiency at scale.

We measure computation overhead with two complementary metrics. First, we calculate the *average number of memory*

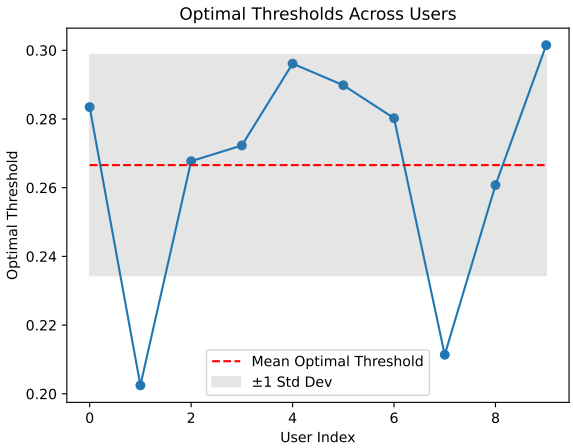


Figure 24. Optimal threshold calculated for each $n = 10$ users based on utility scores. It shows the optimal thresholds for the 10 users (solid blue line), as well as the mean threshold (dotted red line) and ± 1 standard deviation (gray shaded area)

Table 7. Examples of memory items selected by each method compared to human gold labels. Overlapping memory with items human annotation is **bolded**.

User Input	Humans*	RUMS-Binary	GPT4-ZeroShot	GPT4-FewShot	Semantic Similarity
[PERSONALIZED] I became a millionaire recently and want to give money to some local businesses	Location Hobbies and Interests Volunteer Activities	**Location** **Hobbies and interests** **Volunteer activities** Personal values Industry of employment	**Location** Occupation Industry of employment Current projects Long-term aspirations	**Location** Recent life events	**Location** **Volunteer activities** Financial situation Annual Income Pet ownership
[NON-PERSONALIZED] Who was the Prime Minister of Gt. Britain at the outbreak of The First World War?	No features selected	**No features selected**	Location Occupation Name Political affiliation	Location Occupation	Travel history Occupation Name Political affiliation Current projects

items selected per dataset and method. This captures how the amount of items each method chooses in order to fulfill personalization or how little for no personalization. Second, we compute the *average number of additional tokens* that must be appended to the prompt during inference in order to include the selected memory items. This provides a direct measure of the inference overhead. Table 8 and Table 9 summarize these results, showing clear differences across methods: baselines such as All or Random predictably add large numbers of items, while information-retrieval heuristics like BM25 are highly selective. Our methods lie in between, striking a balance between selecting informative items and avoiding unnecessary additions.

To translate these measures into practical cost, we estimate the monthly cost of running one million queries per day for each method using pricing aligned with GPT-4. Specifically, we convert the additional tokens from Table 2 into dollar cost under the assumption that each token incurs a fixed marginal cost of \$0.01 per 1K tokens and 30 days per month. This allows us to approximate the total monthly inference cost for each method, and thus assess the economic tradeoff of personalization strategies at scale.

The results are visualized in the Figure 25. As expected, the All baseline is by far the most expensive, exceeding \$160K per month due to its maximal inclusion of all memory items. On the other hand, our model RUMS-Binary yields much lower additional cost of only \$6.9K which is about 30% lower than just the inference cost of including the memory features selected by GPT4-FewShot, not including the cost to generate these memory items. Our other model, RUMS-Multi, also has low cost of only \$8.4K, keeping costs well below stronger baselines like GPT-4, ReContriever, and Semantic Similarity. Overall, this analysis highlights that RUMS-Models can be deployed at scale in a cost-efficient manner, with our methods demonstrating favorable tradeoffs between informativeness and economic feasibility.

Table 8. Average number of memory items selected by method per dataset.

Method	FreebaseQA	PersonaFeedback	WildChat-Personal	WildChat-NonPersonal
All	50	50	50	50
None	0	0	0	0
Random	5	5	5	5
Semantic Similarity	4.71	4.54	4.84	4.74
BM25	0.39	0.81	0.28	0.42
ReContriever	5	5	5	5
GPT4-Few	1.21	4.39	3.90	2.10
RUMS-Binary	0.02	3.14	2.80	1.64
RUMS-Multi	1.45	3.58	2.36	1.76

Table 9. Average number of extra tokens required for response generation.

Method	FreebaseQA	PersonaFeedback	WildChat-Personal	WildChat-NonPersonal
All	551.59	551.28	552.16	550.40
None	0	0	0	0
Random	54.87	54.98	54.34	54.20
Semantic Similarity	48.04	51.44	53.04	49.58
BM25	1.48	8.95	3.20	4.86
ReContriever	60.87	62.93	61.22	61.72
GPT4-Few	12.96	49.31	46.02	23.58
RUMS-Binary	0.23	37.39	34.88	19.10
RUMS-Multi	13.88	45.34	30.90	21.08

A.8. Time Analysis

We ran a wall-clock analysis to showcase the time resources needed to run each method (Table 10. We calculated the average time for doing inference on one sample using $n = 3$ runs. We find that the RUMS-Multi have runtimes comparable to the retrieval methods (Semantic Similarity, BM25, and ReContriever) and RUMS-Binary has times comparable to GPT4-Fewshot. As mentioned, the RUMS-Utility has a very high runtime and therefore we do not recommend it to be used during inference.

A.9. Results of H3 using Human Judgment

For evaluation in H3, LLM-as-Judge was chosen due to it's validation as similar to third-party human annotation for personalization when both in-context examples and model confidence scores (Dong et al., 2024). However, to validate that selection alignment translates to better responses, we conducted a human study comparing RUMS-Binary with GPT-4 Few-Shot (strongest baseline). We had two human annotators (CSE PhD graduate students) annotate which of the two response they felt were better using the same rubric as the LLM-as-Judge in H3.

We found RUMS-Binary outperforms GPT4-Few by 33% in human preference, despite GPT-4 Few-Shot using a model $400\times$ larger for both selection and generation. This result (1) validates our LLM-as-Judge evaluation methodology (33% human win rate closely matches the 18% win rate from Table 3 against GPT-4 Few-Shot on synthetic data), (2) Confirms that RUMS's human-aligned selection decisions (from H2) translate to better final outputs (H3).

A.10. Analysis of Selected Entropy Calculation Model

As shown in Section A.1, we found that Llama-7 was the best performing model on calculating entropy scores that correctly seperate personalized versus non-personlized inputs. However, given that Mistral-7B was also moderately good at this task, we test how RUMS would work using Mistral-7B on both binary decisions of personalization (H1), and choosing memory item subsets (H2).

Table 12 show the results from H1 and Table 14 show the results from H2 from using Mistral-7B compared to Llama-7B. We find that although in general Llama-7B beats Mistral-7B, they perform quite closely. This might indicate that even if

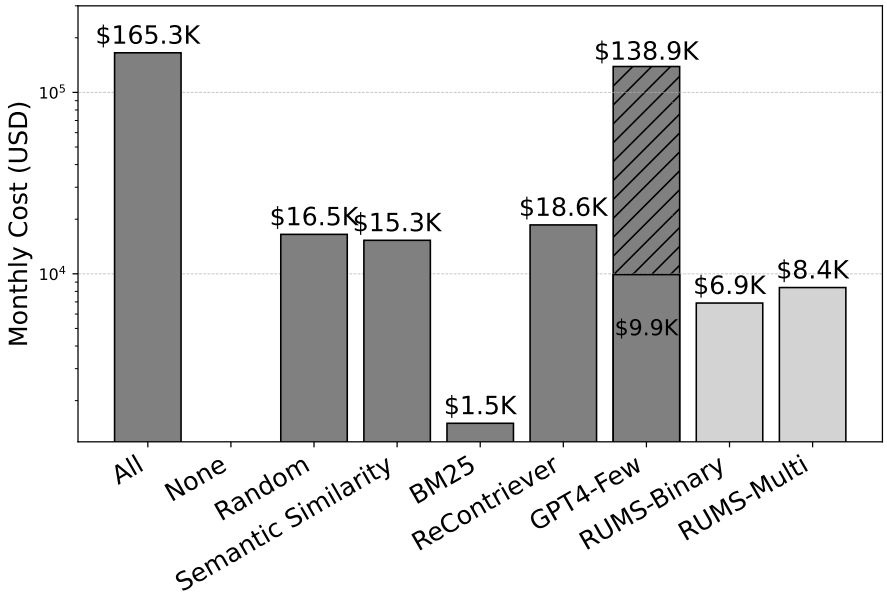


Figure 25. Estimated monthly cost of additional prompt tokens (1M queries per day, 30 days, GPT-4 input pricing at $0.01 per 1K tokens).

Table 10. Average time for inference of one sample, calculated over $n = 3$ trials. All values are in units of seconds.

Method	Average Time (s)
All	0.0
None	0.0
Random	0.0
Semantic Similarity	0.21
BM25	0.01
ReContriever	0.08
GPT4-Few	1.22
RUMS-Binary	0.14
RUMS-Multi	0.04
RUMS-Utility	7.28

Table 11. Winrate of responses generated using memory subset of RUMS-Utility vs. GPT4-Few, determined by human evaluation.

Dataset	RUMS-Utility Winrate
Synthetic	0.33
Real World	-0.03

slightly less separation is seen between personalized and non-personalized entropy scores they can still be useful signal.

A.11. Analysis of *Optional* Filter Step

In our methods section, we mention that if a large number of candidate memory items exist it can be useful to use a more coarse method (such as a prompted LLM) to reduce the number of memory features for RUMS calculation. In our experimentation, we use GPT-4 as this method of coarse deselection which chooses up to 5 memory items from the original 50 to measure. In this section we validate the use of GPT-4 for this purpose.

To validate the use of GPT-4 and other possible other models, we use the randomly sampled personalized inputs which

Table 12. H1 Results

Dataset	Model	Recall	Specificity
Synthetic	Llama-7B	**0.94**	**0.94**
Real World	Llama-7B	0.92	**0.96**
Synthetic	Mistral-7B	0.83	0.88
Real World	Mistral-7B	0.92	0.77

Table 13. Results of using Llama-7B or Mistral-7B to calculate entropy scores on H1 experiments. Bolded is the higher value between the two models.

Table 14. H1 Results

Dataset	Model	Precision	Recall	F1
Synthetic	Llama-7B	0.61	**0.70**	**0.70**
Real World	Llama-7B	0.48	**0.36**	0.53
Synthetic	Mistral-7B	0.61	0.62	0.69
Real World	Mistral-7B	**0.61**	0.30	**0.69**

Table 15. Results of using Llama-7B or Mistral-7B to calculate entropy scores on H2 experiments. Bolded is the higher value between the two models.

we have a consensus human annotated memory items selected (n=60) used in H2. Then, we analyzed how well the large general purpose models (GPT-4, Llama-70B, Qwen-72B, and Gemma-27B) filtered memory items (set to k=5) created by prompting the model, compared to the human selected memory items. We note that since these models are instructed to overselect (i.e. filter to a list of 5) we focus mostly on the recall metric, rather than precision of F1. Also, we only show the results for the personalized datasets, as this is when this is mostly used.

Table 16. Model Performance on Personalized Datasets

Dataset	Model	Precision	Recall	F1
Synthetic Personalized	GPT-4	0.24	**0.77**	0.49
	Llama-70B	0.23	0.74	0.47
	Qwen-72B	0.21	0.74	0.45
	Gemma-27B	0.22	0.66	0.45
Real World Personalized	GPT-4	0.46	**0.70**	0.59
	Llama-70B	0.41	0.69	0.53
	Qwen-72B	0.44	0.67	0.55
	Gemma-27B	0.48	0.64	0.56

Table 17. Results of using large general purpose models for filtering. We compare the memory subsets filtered by each model, through prompting, with a human annotated list.

Table 17 shows the results from this analysis. We find that in general, all the models perform similarly with high recall to human annotations. However, we do see that GPT-4, the model we used, was the best at 0.77 and 0.70 recall for synthetic and real world data respectively. In general, this supports this method of coarse deselection.

To further validate these results, we were interested in if there is any bias in the type of features selected by the different models. So, we compared the top-10 most selected memory items per model. We found that on average, the models shared 6.8 memory items with other models and 4.75 items with humans. Table 19 shows the full results. This gave evidence that different models did not seem to have unique preferences or selections, thus adding further robustness to this method.

Table 18. Pairwise Agreement Between Models and Humans

	Humans	GPT-4	Llama-70B	Qwen-72B	Gemma-27B
Humans	—	—	—	—	—
GPT-4	0.4	—	—	—	—
Llama-70B	0.5	0.7	—	—	—
Qwen-72B	0.5	0.8	0.6	—	—
Gemma-27B	0.5	0.6	0.7	0.7	—

Table 19. Pairwise overlap (percentage) of the top-10 most frequently selected memory items across models and humans.

A.12. Full Results of H3

In this section we expand H3 by breaking down results for the Synthetic and Real World datasets by inputs intended to be personalized (left) and not personalized (right). This comparison highlights differences between RUMS-Binary and RUMS-Multi. With the less capable response model, LLaMA-70B, RUMS-Binary (which favors fewer memory items) outperforms most baselines, showing especially strong gains on non-personalized inputs, with a 38% average win rate over baselines on the Synthetic dataset. Similar patterns hold on the Real World dataset, except for the None and BM25 baselines, which never select items.

By contrast, RUMS-Multi (which selects more memory items) performs better with the stronger GPT-4 model, particularly on personalized inputs, achieving 25% and 26% average win rates on the personalized Synthetic and Real World datasets respectively. These results demonstrate how RUMS can be tuned to different settings, adapting to the response model and personalization needs.

Table 20. Average winrate comparison on Synthetic data under Not Personal (left) and Personal (right) conditions. Positive values (green) indicate better performance, negative values (red) indicate worse performance. Values greater than 0.1 in magnitude are bolded.

(a) Non-personalized inputs

	LLaMA 70B		GPT-4	
	RUMS-Binary.	RUMS-Multi	RUMS-Binary.	RUMS-Multi
None	-0.03	**-0.79**	0.02	0.00
All	**0.45**	**-0.24**	0.00	0.02
Random	**0.50**	**-0.12**	-0.03	-0.01
Semantic Similarity	**0.60**	0.01	0.02	0.00
BM25	**0.12**	**-0.66**	-0.02	0.01
ReContriever	**0.62**	**-0.14**	0.01	0.03
GPT4-Few	**0.42**	**-0.34**	0.00	0.01

(b) Personalized inputs

	LLaMA 70B		GPT-4	
	RUMS-Binary.	RUMS-Multi	RUMS-Binary.	RUMS-Multi
none	**0.52**	**0.53**	**0.50**	**0.72**
All	**-0.14**	**-0.14**	**-0.31**	0.01
Random	**0.18**	**0.37**	**0.15**	**0.39**
Semantic Similarity	-0.06	**0.17**	-0.07	0.09
BM25	**0.25**	**0.40**	**0.42**	**0.40**
ReContriever	-0.09	0.09	-0.09	0.09
GPT4-Few	-0.07	0.03	**-0.17**	0.03

Table 21. Average winrate comparison on Real World data under Not Personal (left) and Personal (right) conditions. Positive values (green) indicate better performance, negative values (red) indicate worse performance. Values greater than 0.1 in magnitude are bolded.

(a) Non-personalized inputs

	LLaMA 70B		GPT-4	
	RUMS-Binary	RUMS-Multi	RUMS-Binary	RUMS-Multi
None	**-0.36**	**-0.64**	**-0.12**	**-0.26**
All	0.08	**-0.14**	**-0.14**	**-0.16**
Random	-0.02	-0.06	-0.06	**-0.26**
Semantic Similarity	0.04	-0.02	-0.02	**-0.12**
BM25	**-0.40**	**-0.16**	**-0.16**	**-0.26**
ReContriever	**0.12**	0.04	0.04	0.02
GPT4-Few	0.08	**0.12**	**0.12**	-0.02

(b) Personalized inputs

	LLaMA 70B		GPT-4	
	RUMS-Binary	RUMS-Multi	RUMS-Binary	RUMS-Multi
None	**0.46**	**0.70**	**0.52**	**0.60**
All	**-0.16**	-0.04	**-0.14**	0.04
Random	**0.24**	**0.42**	**0.26**	**0.34**
Semantic Similarity	-0.10	0.00	-0.02	**0.12**
BM25	**0.42**	**0.58**	**0.52**	**0.52**
ReContriever	0.00	**0.24**	**-0.14**	**0.24**
GPT4-Few	-0.08	-0.04	-0.04	-0.04

A.13. Qualitative Analysis and Results

To better explore the cases when memory items selected by RUMS-Utility does not match the human selected, we further analyze the n=60 samples from H2. First, we define the different categories of cases we found:

- Correct: RUMS-Utility matches Human chosen list exactly
- Underselect: Human selected items and RUMS-Utility did not
- Some: RUMS-Utility selected a mixture of items selected at least one human, and not selected by humans

Table 22. Breakdown of selection behavior categories with representative user inputs and selected memory items by humans (all agree), humans (at least one agree), and RUMS Utility.

Category	**%**	**Example**		**Chosen Items**	
		User Input	Humans (All)	Humans (> 1)	RUMS-Utility
Correct	38	"In May 1994 the Channel Tunnel was opened by Queen Elizabeth II and which French President?"	[]	[]	[]
Underselect	15	"How to use a vpn while vacationing in china"	[Tech savvy]	[Tech savvy, Feedback style, Comm mode, Tech use]	[none]
All	13	"Choose between new zara sneaker or used new balance"	[]	[Hobbies, Income, Aspirations, Finances, Exercise, Personality, Job, Industry, Pastimes, Events, Gender, Location]	[Hobbies, Income]
Some	22	"Buying a new notebook to write a script"	[]	[Location, Income, Tech savvy, Finances, Personality]	[Hobbies, Tech savvy, Fav books]
None	5	"My PC is r6 3600 and 1660s need new GPU"	[Projects, Finances]	[Income, Job, Hobbies, Industry, Projects, Aspirations, Finances, Pastimes]	[Tech savvy]
Underselect - All	7	"Im hungry"	[Foods, Diet needs, Fitness, Location]	[Foods, Diet needs, Fitness, Location, Income, Finances, Travel, Gender]	[Foods, Diet needs, Fitness]

- All: RUMS-Utility selected all items which at least one human selected (but possibly not all humans)
- None: RUMS-Utility only selected items which no human selected
- Underselect - All: RUMS-Utility did select some items, which were chosen by all humans, but humans selected more items as well which RUMS-Utility did not.

Using these categories, we count the number of samples for each case.

Table Table 22 shows the results from this analysis, including the type of category, percentage of samples, and an example of each. We find majority of non-correct samples are from RUMS-Utility selecting values which at least one human selected but not all humans. This makes sense, as memory selection for personalization can be highly subjective and so this just shows that RUMS-Utility may select some of the more on the edge items. There is also a moderate level of underselect. This lends more weight to the idea that RUMS-Utility is a more conservative method, however, this could be changed based on the threshold selected by the user.

B. Formulas

B.1. Shannon Entropy

We define the Shannon Entropy as a measure of uncertainty within a system. For a random variable $x \in X$ and a distribution according to $p : X \to [0, 1]$ such that $p(x) := \mathbb{P}[X = x]$. Then we formalize the Shannon entropy H as,

$$H(x) = -\sum_{x \in X} p(x) \log p(x)$$

C. Optimality of RUMS Proof

Proof. We begin with the joint conditional entropy of (y, z_u) given $(x, m_{u,S})$. By the chain rule,

$$\begin{aligned} H(y, z_u \mid x, m_{u,S}) &= H(y \mid x, m_{u,S}) + H(z_u \mid x, y, m_{u,S}) \\ &= H(z_u \mid x, m_{u,S}) + H(y \mid x, z_u, m_{u,S}). \end{aligned}$$

Equating the two expressions and rearranging yields

$$H(y \mid x, m_{u,S}) = H(y \mid x, z_u, m_{u,S}) + H(z_u \mid x, m_{u,S}) - H(z_u \mid x, y, m_{u,S}).$$

By Assumption 2, memory $m_{u,S}$ provides little additional information about y beyond (x, z_u). We therefore approximate

$$H(y \mid x, z_u, m_{u,S}) \approx H(y \mid x, z_u).$$

Substituting this approximation gives

$$H(y \mid x, m_{u,S}) \approx H(y \mid x, z_u) + H(z_u \mid x, m_{u,S}) - H(z_u \mid x, y, m_{u,S}).$$

The first term, $H(y \mid x, z_u)$, does not depend on the choice of subset S. In addition, Assumption 2 implies that conditioning on $m_{u,S}$ does not substantially refine the posterior over z_u once (x, y) are observed, so

$$H(z_u \mid x, y, m_{u,S}) \approx H(z_u \mid x, y),$$

which is also independent of S.

Combining these observations, we obtain the approximation

$$H(y \mid x, m_{u,S}) \approx C + H(z_u \mid x, m_{u,S}),$$

where C is a constant that does not depend on S.

Therefore, minimizing predictive entropy $H(y \mid x, m_{u,S})$ is approximately equivalent to minimizing uncertainty about the latent variable z_u given the selected memory.

Now consider expected utility as defined in Assumption 1,

$$\mathbb{E}[U_u(y)] = -\mathbb{E}_{z_u}\mathbb{E}_{y \sim p_\theta(\cdot \mid x, m_{u,S})}[\ell(y, z_u)].$$

Because ℓ is a strictly proper scoring rule, the expected loss is minimized when the predictive distribution $p_\theta(y \mid x, m_{u,S})$ matches the conditional distribution induced by z_u. Reducing uncertainty about z_u therefore strictly improves expected utility.

Consequently, any subset S^* that approximately minimizes $H(y \mid x, m_{u,S})$ also approximately maximizes expected utility. □

D. Sequence Entropy Equivalence Proof

Proof. Let $p_\theta(y \mid x)$ denote the model distribution over token sequences $y = (y_1, \ldots, y_T)$ given input x. At each step t, the conditional next-token distribution is

$$p_\theta(\cdot \mid y_{<t}, x),$$

where $y_{<t}$ is the prefix of tokens before step t.

By the chain rule of probability,

$$-\log p_\theta(y \mid x) = \sum_{t=1}^{T} -\log p_\theta(y_t \mid y_{<t}, x).$$

Taking the expectation with respect to $y \sim p_\theta(\cdot \mid x)$ on both sides gives

$$
\begin{aligned}
\mathbb{E}_{y\sim p_\theta}\left[-\log p_\theta(y \mid x)\right] &= \mathbb{E}_{y\sim p_\theta}\left[\sum_{t=1}^{T} -\log p_\theta(y_t \mid y_{<t}, x)\right] \\
&= \sum_{t=1}^{T} \mathbb{E}_{y\sim p_\theta}\left[-\log p_\theta(y_t \mid y_{<t}, x)\right] && \text{linearity of expectation} \\
&= \sum_{t=1}^{T} \mathbb{E}_{y_{<t}\sim p_\theta}\left[\sum_{y_t} p_\theta(y_t \mid y_{<t}, x) - \log p_\theta(y_t \mid y_{<t}, x)\right] && \text{law of iterated expectation} \\
&= \mathbb{E}_{y_{<t}\sim p_\theta}\left[\sum_{t=1}^{T} H(p_\theta(\cdot \mid y_{<t}, x))\right].
\end{aligned}
$$

□

E. RUMS-Utility Algorithm

In Algorithm 1 we provide the formalized algorithm for RUMS-Utility.

Algorithm 1 Memory Selection via RUMS

Require: User input x, user memory $u \in \mathbb{R}^d$
Ensure: Selected memory subset $m^*_{u,S}$ that maximizes utility

Candidate Memory Item Identification:
Use LLM to generate candidate subsets S for x
Subset Evaluation:
for each $s \in S$ **do**
    Map candidate attritubes to user values $m_{u,S}$
    Prompt Reference LM with:
        (i) x only
        (ii) x and $m_{u,S}$
    Compute estimated utility $\hat{\text{EIG}}(S)$ using Equation (3)
    Store utility score for subset s
Subset Selection:
$s^* \leftarrow \arg\max_{s\in S} \hat{\text{EIG}}(S)$
if all MI scores $<$ thresholdτ **then**
    $s^* \leftarrow \emptyset$ ▷ Select no features

F. Experimental Details

This section provides the full details of the experiments described in Section 4.

F.1. Experimental Setup

Across all experiments (H1–H3), we consider a fixed set of $n = 50$ memory items (e.g., *location, age, dietary restrictions*). The task is to select a subset of up to $n = 5$ memory items that can improve personalization.

F.2. Datasets

To evaluate scenarios where personalization is beneficial as well as those where it is not, we use a diverse mixture of user inputs drawn from prior work, including both synthetic and real-world sources. The three datasets we used are described below:

PersonaFeedback (Tao et al., 2025) This is a synthetically created dataset consisting of user-specific queries designed to reflect inputs from realistic and diverse user personas. First, they constructed personas by representing them as a structured dictionary of memory items, including demographics, personality traits, and preferences. Then, open-source content from social media, reviews, and forums were used to populate these items with specific values. Lastly, an LLM generator was used to produce queries based on a sampled persona and specific sub-scenarios. Additional steps such as similarity-based filtering, rephrasing to avoid directly revealing persona information, and manual annotations ensured diversity and realism. The final dataset contains more than 4,000 persona-specific questions. The original data is in Chinese and therefore, we translated to English using a machine translation model (Tiedemann & Thottingal, 2020). Example: *"Do you have any music for the trip?"*

FreebaseQA (Jiang et al., 2019) A dataset of trivia-style questions and answers collected from human trivia competitions. The original dataset contains approximately 28,000 question–answer pairs. Example: *"What type of snake is a sidewinder?"*

WildChat (Zhao et al., 2024) A large-scale collection of more than 1 million real-world user–ChatGPT interactions, encompassing over 2.5 million dialogue turns. The dataset was compiled by offering free access to ChatGPT and GPT-4 in exchange for user consent to share chat histories. It covers diverse conversational scenarios such as ambiguous requests, code-switching, topic shifts, and political discussions. We filter the data to retain only English conversations labeled as non-toxic, and we use only the first human-written input per conversation to ensure that queries contain sufficient context.

To determine whether an input is personalized, two human annotators labeled a random selection of WildChat queries as either beneficial for personalization (WildChat-Personal) or not (WildChat-NonPersonal). Only queries with full annotator agreement were retained. Examples: *"I'm hungry"; "Who is Mohamed Gamal Mohamed Maniieh?"*

Synthetic vs. Real-World Datasets By design, PersonaFeedback is treated as a personalized input set, and FreebaseQA as non-personalized. We collectively refer to these as the "Synthetic" dataset, since they were not originally created for LLM input. WildChat is considered a "Real-World" dataset, as it was collected directly from real user interactions with an LLM. We use both synthetic and real-world datasets to evaluate the robustness of RUMS across many diverse inputs.

Sample Size by Hypothesis The number of samples varies across hypotheses. For H1, we use $n = 500$ samples from PersonaFeedback and FreebaseQA to analyze RUMS-Utility for binary personalization decisions. For H1', we use $n = 25$ inputs each from PersonaFeedback, FreebaseQA, WildChat-Personal, and WildChat-NonPersonal to extend the H1 analysis to correlation with human judgment. For H2, we use $n = 15$ inputs each from PersonaFeedback, FreebaseQA, WildChat-Personal, and WildChat-NonPersonal to examine the correlation between RUMS memory selection and human preference. Finally, for H3, we use $n = 100$ samples from PersonaFeedback and FreebaseQA along with $n = 50$ samples each from WildChat-Personal and WildChat-NonPersonal to analyze the downstream impact of subsets chosen by RUMS-Model.

F.3. Profile Generation

User profiles are incorporated during the training data curation phase, where specific user values are used to compute the entropy difference of the model required for the utility function in Equation (3). Also, we use profiles in our test sets for evaluation.

Memory Item Creation To construct user profiles, we prompted GPT-4 (OpenAI et al., 2024) to generate 50 commonly occurring user memory items. These items span demographic, behavioral, and preference-based attributes, providing a diverse set of values for personalization. The complete list is as follows:

- Name
- Gender
- Location (City/State/Country)
- Ethnicity
- Race
- Annual Household Income
- Language spoken at home
- Language preference
- Sexual Preference
- Disabilities
- Occupation
- Political affiliation
- Education level
- Hobbies and interests
- Favorite sports
- Preferred music genre
- Favorite books
- Preferred movie genre

- Favorite TV shows
- Favorite foods
- Dietary restrictions (e.g., vegetarian, vegan)
- Relationship status
- Number of children
- Pet ownership (types of pets)
- Travel history (countries visited)
- Travel preferences (adventurous, relaxing)
- Social media platforms used
- Level of tech-savviness
- Preferred mode of communication (text, voice, video)
- Work schedule (full-time, part-time)
- Industry of employment
- Current projects or goals
- Long-term aspirations
- Financial situation (broad categories)
- Health and fitness goals
- Preferred exercise activities
- Level of environmental consciousness
- Volunteer activities or interests
- Current challenges or pain points
- Personality traits (e.g., introverted, extroverted)
- Favorite pastimes
- Personal values or beliefs
- Cultural background
- Religious beliefs
- Recent life events (e.g., change in job, moved)
- Technology usage (devices, apps)
- Frequency of using the service (daily, weekly)
- Reasons for using the service
- Feedback preferences (detailed, brief)
- Preferred tone of communication (formal, casual)

Memory Item Values Next, we used GPT-4 (OpenAI et al., 2024) to generate consistent user profiles without contradictions (e.g., "vegan" with "favorite food = steak"), following the prompt in Prompt 1. To further ensure logical consistency, we employed GPT-4 to filter out any profiles that appeared illogical, as described in Prompt 2. This process was repeated until we obtained $n = 100$ unique, logically consistent profiles for both training and testing.

This method yields a diverse set of user profiles, as illustrated in the bar charts of memory item distributions shown in Figure 26 through Figure 75.

Prompts used for Profile Generation

Prompt 1. For these 50 features, I want you to create 10 very diverse persona profiles. So for each of the 50 features, I want you to select a value (e.g. "age" = 50) that would make up the persona profile. Make sure the values make sense for each other, meaning don't have a person who is 12 years old have a job as a accountant.

Output a json file with a dictionary with 10 keys, "Persona_1", "Persona_2",...,"Persona_10", whose corresponding values is a list of the selected values in the same order as the features (e.g. [John, Male, Seattle/WA/USA,...]).

Do not include any other markdown or text. Here are the Features:

Prompt 2. You are an AI assistant that needs to decide if a user profile is logically sound. This means that the information in the user profile should be consistent and non-conflicting.

Only output a single word, either 'yes' or 'no', to indicate if the user profile is logically sound.

Example 1: If a user profile states that the user is allergic to peanuts, it should not also state their favorite food is peanuts.

Example 2: If a user profiles states highest education is a GED, it should not also state their profession is a doctor.

Example 3: If a user profile states that the user is 25 years old, it should not also state that they have been working for 30 years.

Example 4: If a user profile states that they live in Columbia, it should not state that their main language is French.

User Profile:

F.4. Baselines

In this section we describe each baseline that is used in our experimentation.

GPT-4 Prompting (OpenAI et al., 2024): We use GPT-4o as a baseline across H1, H2, and H3. In H1, we apply Prompt 3 for the zero-shot setting and Prompt 4 for the few-shot setting to decide whether personalization is needed for a given query. For the few-shot case, we provide four in-context examples: two requiring personalization and two not requiring personalization.

In H2 and H3, we adapt the prompts to instead elicit the subset of specific user memory items that would be beneficial for personalization. We use Prompt 5 for zero-shot and Prompt 6 for few-shot. In the few-shot setting, we provide two in-context examples: one requiring personalization and one not.

Prompt 3. You are a classifier that determines whether a given user input would benefit from user-specific information for the model to respond.

Your response must be a JSON object with a two key "Decision" with a value of either "True" or "False" and "Features" with either a list of user-specific features (at most 10 features) if "Decision" is "True" or "None" if "Decision" is "False" :

Output "True" if the input would benefit from user-specific information to better respond to the query.

Output "False" if the input would not benefit from user-specific information or would be the same answer regardless of the user.

Respond with only the JSON object. Do not include any explanation or additional text.

Input: user_input

Prompt 4. 'You are a classifier that determines whether a given user input would benefit from user-specific information for the model to respond.

Your response must be a JSON object with a two key "Decision" with a value of either "True" or "False" and "Features" with either a list of user-specific features (at most 10 features) if "Decision" is "True" or "None" if "Decision" is "False" :

Output "True" if the input would benefit from user-specific information to better respond to the query.

Output "False" if the input would not benefit from user-specific information or would be the same answer regardless of the user.

Respond with only the JSON object. Do not include any explanation or additional text.

Example 1:

Input: "Where should I eat tonight?"

Output: "Decision": "True", "Features": ["location", "dietary restrictions", "income"]

Example 2:

Input: "Can you help me write an essay on feminism?"

Output: "Decision": "True", "Features": ["education", "political ideology", "gender"]

Example 3:

Input: "What is the capital of Austin, Texas?"

Output: "Decision": "False", "Features": "None"

Example 4:

Input: "Can you explain what AGI is?"

Output: "Decision": "False", "Features": "None"

Now classify the following input:

Input: user_input'

Prompt 5. Given a user-input and list of features, select which features might be important for answering the user-input.

Choose up to 5 features. Respond in a JSON with one key "Chosen Features", which has a list of chosen features. Do not include any other text or formatting.

User Input: user_input

Possible Features: candidate_features_ls

Prompt 6. Given a user-input and list of features, select which features might be important for answering the user-input. Choose up to 5 features.

Respond in a JSON with one key "Chosen Features", which has a list of chosen features. Do not include any other text or formatting.

Example Input 1: "whats the weather today"

Example Output 1: "Chosen Features": ["Location (City/State/Country)"]

Example Input 2: "What is the capital of Texas?"

Example Output 2: "Chosen Features": []

User Input: user_input

Possible Features: candidate_features_ls'

Semantic Similarity: In H1–H3, we also include a simple baseline based on semantic similarity. Here, we compute the cosine similarity between user query embeddings and each memory item embeddings using a lightweight sentence-transformer model (Wang et al., 2020).

In H1, we use the validation set, defined as a 20% hold-out of the data, to tune a threshold for deciding whether personalization is required. Thresholds are determined by grid search over $[-1, 1]$ in increments of 0.5, selecting the value that maximizes agreement with human annotations for each dataset. A query is labeled as requiring personalization if at least one memory item has a similarity score above the selected threshold. The exact thresholds used for each dataset are reported in Table 23.

Dataset	Threshold
All	-1.0
Synthetic	-1.0
Real World	0.5

Table 23. Threshold for H1 analysis of semantic similarity baseline. If at least one memory item had a semantic similarity higher than this threshold, then the query was labeled as needing personalization.

Since H2 and H3 require selecting specific memory items for personalization rather than making a simple binary decision, we set an intuitive threshold of $t = 0$, corresponding to no similarity between the user query and a memory item. If more than five memory items exceed this threshold, we select the top five with the strongest correlations.

Simple Baselines: We also compare against several naive baselines. The "None" baseline provides no features ($n = 0$), serving as a lower bound for personalized inputs and an upper bound for non-personalized ones. The "All" baseline includes all available features ($n = 50$), testing the effect of providing maximal but noisy context. Finally, the "Random" baseline samples $n = 5$ items at random, establishing a chance-level reference.

SOTA Retrieval Methods: We additionally evaluate two state-of-the-art retrieval methods.

First, BM25 (Lù, 2024), a widely used ranking function in information retrieval, can be adapted for memory selection. It scores memory items (treated as terms) based on their frequency in each instance relative to the entire dataset, balancing how common a memory item is with how discriminative it is. Items with higher BM25 scores are considered more informative and are selected as the most relevant, making BM25 well-suited for sparse or high-dimensional text and categorical data.

Second, ReContriever (Lei et al., 2023) is a retrieval-based method that scores memory items by their ability to help a model retrieve relevant information. It leverages dense embeddings to measure semantic similarity between queries and candidate items, effectively identifying the most informative items for downstream tasks. Memory items that yield higher retrieval relevance are prioritized, making ReContriever particularly effective for selecting contextually important or predictive inputs in text or structured datasets. Similarly to semantic similarity a threshold of $t = 0$ is used as threshold for selecting memory items, with a maximum of up to $n = 5$ selected.

F.5. RUMS-Utility and RUMS-Model Details

In this paper, we describe two main methods: RUMS-Utility, which selects a subset of user memory items with the highest utility for personalization directly, and RUMS-Model, a lightweight trained model built on data generated by RUMS-Utility.

RUMS-Utility: We use RUMS-Utility not only for the analyses in H1 and H2 but also to generate training data for fine-tuning RUMS-Models. In general, we compute entropy in Equation (4) using Llama-3.1-8B (AI, 2024b) with a temperature of 0.7 and sampling-based decoding to obtain the response distribution. We restrict the maximum generation length to $T = 20$ tokens, chosen to mitigate bias from longer generations having naturally lower entropy.

Following the algorithm in Section 3.4, we first prompt GPT-4 to reduce the candidate memory items from 50 to at least 1 and at most 10 before computing RUMS-Utility (see Prompt 7). Utility scores are then estimated using $n = 5$ Monte Carlo samples. For each input, we generate $n = 5$ distinct user profiles as separate training points, which can yield entirely unique best memory subsets of data depending on the values in the user profile.

The chosen hyperparameters (e.g., model size, number of Monte Carlo samples, etc.) reflect a tradeoff: obtaining reliable signals of memory item importance while keeping computations lightweight and feasible for large-scale inference. Further discussion of these choices is provided in Section A.1.

Prompt 7. Given a user-input and list of features, select which features might be important for answering the user-input.

Choose at least 1 and up to 10 features.

Respond in a JSON with one key "Chosen Features", which has a list of chosen features. ' Do not include any other text or formatting.

Choosing the Personalization Threshold: To determine the threshold for deciding whether to apply personalization or select no personalization, we used a held-out validation set of size $n = 100$ from each of PersonaFeedback and FreebaseQA. For each query, we computed the average maximum utility score over the $n = 5$ generated user profiles. We labeled inputs from PersonaFeedback as personalized (1) and inputs from FreebaseQA as non-personalized (0). These scores were combined and labeled by dataset to construct an ROC curve.

We then selected the threshold that maximized Youden's J statistic ($J = \text{TPR} - \text{FPR}$), which identifies the point that best balances sensitivity and specificity. Using this procedure, we determined an optimal threshold of $t = 0.29$, which was applied in all downstream analyses.

RUMS-Models For training the downstream models, we sampled approximately 2.5K unique PersonaFeedback queries and 2K unique FreebaseQA queries. Each query was paired with five user profiles, and RUMS-Utility was applied to select the best memory subsets, yielding 13K and 9K training examples, respectively. We excluded all FreebaseQA queries that exceeded the personalization threshold, thereby removing any noisy non-personalized samples. This design choice emphasized that non-personalized inputs should not include any memory items.

We then used these (query, memory subset) pairs to train two models. The first model, used in H1–H3, was a binary logistic regression implemented with DeBERTa-v3-large (He et al., 2021) and a final linear classification layer. To train this model, we converted each (query, memory subset) pair into multiple (query, memory item, label) triplets. Training was performed with a learning rate of 5×10^{-6} for 12 epochs, and the best model was selected based on the highest F1 score on a held-out validation set ($n = 1K$).

The second model, used for H3, was a multiclass, multilabel logistic regression implemented again with DeBERTa-v2-large (He et al., 2021) and a final multiclass classification layer. Here, labels were constructed by merging the selected subsets from the $n = 5$ diverse user profiles per query. This produced a single label vector in which all memory items chosen in any of the five samples were marked as significant. This resulted in more dense labels, training the model to select more memory items. Training was performed with a learning rate of 2×10^{-5} for 12 epochs, and the best model was again chosen based on the highest F1 score on a held-out validation set ($n = 1K$).

Selecting Regression Threshold Since RUMS-Models is trained as a logistic regression model (either binary or multiclass), its output represents the probability of selecting a particular memory item. A common approach is to use a cutoff of 0.5 to make a binary decision, however, since our task is more specialized, instead we tuned this cutoff using a held-out evaluation

Table 24. Baseline methods and their corresponding model sizes.

Method	Model	# Parameters
RUMS	microsoft/deberta-v3-large	418M
Semantic Similarity	all-MiniLM-L6-v2	$\sim$22.7M
GPT4o	GPT4o	Not disclosed (est. >200B)
ReContriever	Yibin-Lei/ReContriever	$\sim$110M

set ($n = 1K$). Specifically, we perform a grid search over boundaries in $[0, 1]$ and select the cutoff that achieved the highest F1 score on the evaluation set. This yielded $b = 0.81$ for RUMS-Binary and $b = 0.93$ for RUMS-Multi.

F.6. Comparing Size of Methods

In Table 24, we summarize the size of each model used in both our method and the baselines. Our approach uses a model comparable in scale to SOTA baselines, yet it is over $400\times$ smaller than GPT-4.

F.7. Evaluation Metric and Analysis

H1 Evaluation: In H1, we compare the distribution of maximum utility scores between assumed personalized and non-personalized inputs. Specifically, we evaluate utility scores on a non-personalized dataset (FreebaseQA) and a personalized dataset (PersonaFeedback). We sampled $n = 500$ inputs from each dataset and computed the maximum utility score using Equation (3) for every input. The resulting distributions were then analyzed using divergence metrics and hypothesis testing.

For divergence metrics, we use Jensen–Shannon (JS) divergence, a symmetric measure of similarity between two probability distributions P and Q. It is defined as:

$$\text{JS}(P\|Q) = \frac{1}{2}\,\text{KL}(P\|M) + \frac{1}{2}\,\text{KL}(Q\|M), \quad \text{where } M = \frac{P+Q}{2}$$

Here, $\text{KL}(P\|M)$ is the Kullback–Leibler divergence from P to the average distribution M. JS divergence is always finite, symmetric, and bounded between 0 and $\log 2$.

For hypothesis testing, we use the two-sample Kolmogorov–Smirnov test to assess whether the two samples come from the same distribution. The null hypothesis states that they do, and we test significance at the $\alpha = 0.05$ level.

H1' Evaluation: In H1', performance was evaluated using human labels as the gold standard. For each sample, four human annotators rated whether $n = 100$ samples would benefit from personalization using a 0–2 scale. If the annotators' confidence in personalization was 1 or higher, we considered the input to be personalized. Instructions provided to annotators are given in Prompt 8. We only included inputs with unanimous agreement ($n = 64$) in the evaluation. The average interrater agreement between pairs of annotators was $\kappa = 0.59$ using the Cohen's kappa. The primary metrics used were recall, measuring accuracy in identifying personalized inputs, and specificity, measuring accuracy in identifying non-personalized inputs.

Prompt 8. Instructions: Read the user input and then decide if the input would benefit from specific user information.

0 = No Personalization Needed, 1 = Personalize is might be helpful, 2= Personalization is definitely helpful

H2 Evaluation: For H2, we evaluated how closely each method's selected subset of memory items matched the subsets chosen by humans. To create the human-selected subsets, four annotators were asked to select all memory items (from the full set of 50) that they felt would be helpful in generating a response. Instructions provided to annotators are given in Prompt 9. For each input, we included only the memory items selected by at least three annotators to form the majority subset. This process resulted in an average of 4.19 memory items per query for inputs where at least one item was chosen. This average guided our experimental design for H2 and H3, as we restricted each method to select no more than $n = 5$ items.

We evaluated performance using precision, recall, and F1 score. Here, precision measures the fraction of items selected by a method that were also in the human majority subset, recall measures the fraction of items in the human majority subset that were correctly selected by the method, and F1 is the harmonic mean of precision and recall, providing a single metric that balances both.

Table 25. Average number of features selected for personalized vs. not-personalized inputs. Methods closes to human performance are **bolded**.

	Personalized	Not-Personalized	Difference
Human Annotation	6.45	1.43	5.02
RUMS-Utility	1.87	0.80	1.07
RUMS-Binary	4.87	**1.56**	**3.31**
RUMS-Multi	3.27	1.67	1.60
Semantic Similarity	**5.00**	5.00	0.00
GPT4-ZeroShot	3.30	2.61	0.69
GPT4-FewShot	1.80	1.57	0.23

Number of Memory Items Selected in H2 Lastly, we include table 25 which compares the average number of memory items selected by each method in H2. Of note, we see that human annotations show a clear separation: personalized inputs have on average 6.5 features selected, compared to only 1.4 for non-personalized inputs, a difference of 5. Among automatic methods, RUMS-Model and RUMS-Utility comes closest to this behavior, achieving a range of 3.3 and 1.08, whereas the next closest method is GPT4-ZeroShot which achieves only a 0.67 difference. This further highlights how RUMS excels at both distinguishing when memory items are beneficial as well as which features are beneficial.

Prompt 9. For each input, decide which (if any) user-features would be helpful to better respond to the input.

H3 Evaluation: In H3, we evaluate the downstream impact of the memory subsets chosen by each method on generated responses. For each user input, responses are generated using the query combined with the subset of memory items selected by each method. This evaluation is performed with two models: GPT-4 (OpenAI et al., 2024), a high-performing model with robust capabilities, and LLaMA 70B Instruct (AI, 2024a), a smaller model more susceptible to noise. This setup demonstrates RUMS-Models's flexibility in adapting to models with varying capacities and usage scenarios.

Performance is measured using *generation win rate*, defined as the proportion of pairwise comparisons in which RUMS-Models's output is judged preferable to that of a baseline. GPT-4 is used as the judge, comparing responses generated with RUMS-Models's selected subset against those generated with each baseline's subset. LLM-as-Judge is validated to align closely with third-party human annotations for personalization when combined with in-context examples, model confidence scores (Dong et al., 2024).

To ensure robust evaluation, our methodology includes:

- *In-context examples*: Three examples are provided in the prompt for each evaluation to demonstrate the expected style of reasoning and scoring.
- *Confidence thresholds*: Only responses for which the model reports a self-generated confidence score above the 70% percentile are included in the final analysis.
- *Chain-of-thought prompting*: Evaluation prompts encourage the model to reason step-by-step and justify its preference for one response over another, improving consistency and interpretability of the win-rate comparisons.

For personalized datasets (PersonaFeedback and WildChat-Personal), the evaluation metrics in the chain-of-thought include:

- *Completeness*: Does the response fulfill the user's request?
- *Personalization*: Does the response appropriately incorporate relevant details from the user profile?
- *Overpersonalization*: Does the response include irrelevant, distracting, or unnecessary user details?

For non-personalized datasets (FreebaseQA and WildChat-NonPersonal), the metrics are:

- *Completeness/Accuracy*: Does the response correctly and fully answer the user's request?
- *Neutrality*: Is the response unbiased and general, rather than tailored to the user?

Full evaluation prompts can be found in Prompt 11 for personalized datasets, Prompt 12 for non-personalized datasets, and Prompt 10 for the three in-context examples used in all evaluation.

Prompt 10. Example 1:

"user profile": ['Preferred exercise activities: Weightlifting', "Hobbies and interests: ['Surfing', 'Photography']", 'Location (City/State/Country): Melbourne/Victoria/Australia', "Social media platforms used: ['Facebook', 'LinkedIn']", "Technology usage (devices, apps): ['Smartphone', 'Laptop']],

"user input": " Why are there five rings in Olympic logo? And what does they symbolize?",

"response_1": " The five rings in the Olympic logo represent the five continents: Africa, the Americas, Asia, Europe, and Oceania. They are interlinked to symbolize the unity and friendship among nations, highlighting the idea of bringing together athletes from around the world to compete in the spirit of sportsmanship. The colors of the rings (blue, yellow, black, green, and red) were chosen because at least one of these colors appears in the national flags of all participating countries.",

"response_2": "The five rings in the Olympic logo represent the five continents of the world, Africa, the Americas, Asia, Europe, and Oceania. This design, created by Pierre de Coubertin in 1913, symbolizes the unity and connection among the nations and cultures participating in the Olympic Games. The rings are interlocked, indicating that these continents are united through the spirit of sport, competition, and the Olympic ideals of excellence, friendship, and respect. The colors of the rings (blue, yellow, black, green, and red) along with the white background were chosen because at least one of these colors appears on the flag of every nation competing in the Olympics.",

"overall": "0",

"overall_explanation": "Although they are worded a bit differently, they generally have the same information and do not include unhelpful user-information in the answer."

Example 2:

"user profile": ["Social media platforms used: ['Instagram', 'Facebook']", 'Dietary restrictions (e.g., vegetarian, vegan): None', 'Gender: Female', "Pet ownership (types of pets): ['Cat']", 'Current projects or goals: Continue education', 'Religious beliefs: Christian', 'Favorite TV shows: ["Greyś Anatomy"]', 'Reasons for using the service: Medical updates', 'Current challenges or pain points: Shift work', 'Travel preferences (adventurous, relaxing): Relaxing'],

"user input": "The 2011 album 'The King of Limbs' was released by which English rock band?",

"response_1": "The 2011 album 'The King of Limbs' was released by the English rock band Radiohead.",

"response_2": "The album 'The King of Limbs' was released by the English rock band Radiohead",

"overall": "0",

"overall_explanation": "These answer are the same, minus only minor differences"

Example 3:

"user profile": ['Current challenges or pain points: Commuting stress', 'Location (City/State/Country): Bangalore/Karnataka/India', 'Favorite sports: Cricket', "Personal values or beliefs: ['Kindness', 'Resilience']", 'Political affiliation: Centrist', 'Name: Arjun Mehta', 'Reasons for using the service: Access tools', 'Financial situation (broad categories): Financially modest', 'Level of tech-savviness: Moderate', 'Cultural background: South Asian', 'Frequency of using the service (daily, weekly): Weekly', "Favorite TV shows: ['Sacred Games']", 'Travel preferences (adventurous, relaxing): Relaxing', 'Preferred mode of communication (text, voice, video): Voice', "Favorite foods: ['Dosa', 'Chaat']", 'Health and fitness goals: Improve health'],

"user input": " Are there any recommended effective ways to enhance the fun of English grammar learning?",

”response_1”: ”Hi Arjun! Here are some recommended ways to enhance the fun of learning English grammar: 1. **Games and Quizzes**: Engage in grammar games or online quizzes that challenge your understanding in a fun way. Websites like Kahoot! or Quizlet offer interactive tools. 2. **Group Learning**: Form a study group with friends or peers. This can make the learning process more social and less intimidating. You can discuss grammar topics over chai or dosa! 3. **Music and Lyrics**: Listen to English songs and try to analyze the lyrics. Identifying verb tenses or other grammatical structures within your favorite songs can be both educational and entertaining. 4. **Humor in Learning**: Explore humorous content, such as grammar memes or funny videos about grammar mistakes, to lighten the mood. 5. **Role Play**: Practice dialogues using correct grammar in fun role-playing scenarios that interest you, perhaps even related to cricket! By combining these methods with your existing interests and values, like resilience and kindness, you can make grammar learning a much more enjoyable experience. Happy learning!”,

”response_2”: “Improving the enjoyment of English grammar learning can be approached in various fun and engaging ways. Here are some recommendations that might resonate with you: 1. **Gamify Learning**: Use educational games and apps that focus on grammar. Many platforms have quizzes, challenges, and rewards that make learning feel like a fun competition. 2. **Interactive Activities**: Engage in activities like grammar charades or sentence-building races with friends or family. This adds a social element to learning. 3. **Threshold Learning**: Set small, achievable goals and reward yourself when you reach them. This creates a sense of accomplishment and makes the process more enjoyable. 4. **Join a Language Group**: Participate in local or online language exchange groups where you can practice speaking with others. This can be a relaxing way to enhance your skills without the pressure of formal study. Would you like any specific resources or activities tailored to your preferences?”,

“overall”: “2”,

“overall_explanation”: “Response 1 is over personalized to the profile and inserts user-information in an unnatural and unhelpful way”

Prompt 11. You are an impartial and knowledgeable evaluator. You will be shown a user profile, a user input, and two model-generated responses. Your task is to decide if **only one of the responses** meets the evaluation criteria. If both responses do (or neither does), mark it as a tie (0).

Evaluation Criteria:

- **Completeness**: Does the response fulfill the user’s request?

- **Personalization**: Does the response appropriately use relevant details from the user profile?

- **Overpersonalization**: Does the response include irrelevant, distracting, or unnecessary details about the user?

- **Overall**: Taking into account completeness, and being appropriately personalized without being overly personal, is one response clearly better than the other?

Important Notes:

- Assign a score of 1 if **only response 1** satisfies the criterion.

- Assign a score of 2 if **only response 2** satisfies the criterion.

- Assign a score of 0 if **both responses** satisfy it or **neither does**.

- Do not favor length or style unless it clearly affects the criterion.

- Be strict: if the difference is marginal, mark it as a tie (0).

Output Instructions:

Each of the 4 score fields must be:

- ‘”1”‘ = only response 1 satisfies the criterion

- ‘”2”‘ = only response 2 satisfies the criterion

- ‘”0”‘ = both or neither satisfy the criterion

Additionally, assess your confidence in this decision by assigning a certainty level from 1 to 100. Use the following guidelines to assign the certainty level:

1–20 (Uncertain): There is insufficient or minimal evidence. The decision is largely based on weak or indirect hints.

21–40 (Moderately Confident): There is noticeable evidence supporting a preference, though it is not comprehensive, and other interpretations are possible.

41–60 (Quite Confident): There is clear and convincing evidence that supports this decision, though it is not entirely decisive.

61–80 (Confident): There is strong evidence that clearly supports this decision, with very little ambiguity.

81–100 (Highly Confident): There is direct and explicit evidence that decisively supports this prediction.

Lastly, include a **brief (1–2 sentences)** and justify the score using the criteria above.

You must return a JSON object with these 12 keys:

- ”completeness”, ”completeness_explanation”, ”completeness_confidence”

- ”personalization”, ”personalization_explanation”, ”personalization_confidence”

- ”overpersonalization”, ”overpersonalization_explanation”, ”overpersonalization_confidence”

- ”overall”, ”overall_explanation”, ”overall_confidence”

Do not include any text outside the JSON block. Here is the evaluation task:

Given: ”user input”: ”user_input”, ”response_1”: ”response_a”, ”response_2”: ”response_b” Generation:

Prompt 12. f”’You are an impartial and knowledgeable evaluator. You will be shown a user input and two model-generated responses. Your task is to decide if **only one of the responses** meets the evaluation criteria. If both responses do (or neither does), mark it as a tie (0).

Evaluation Criteria:

- **Completeness & Accuracy**: Does the response correctly and fully answer the user’s request?

- **Neutrality**: Is the response unbiased and general, rather than tailored to the user?

- **Overall**: Taking into account completeness, accuracy, and neutrality is one response clearly better than the other?

Important Notes:

- Assign a score of 1 if **only response 1** satisfies the criterion.

- Assign a score of 2 if **only response 2** satisfies the criterion.

- Assign a score of 0 if **both responses** satisfy it or **neither does**.

- Do not favor longer responses unless they add essential information.

- Be strict: if the difference is marginal, mark it as a tie (0).

Output Instructions: Each of the 4 score fields must be:

- ‘”1”‘ = only response 1 satisfies the criterion

- ‘”2”‘ = only response 2 satisfies the criterion

- ‘”0”‘ = both or neither satisfy the criterion

Additionally, assess your confidence in this decision by assigning a certainty level from 1 to 100. Use the following guidelines to assign the certainty level:

1–20 (Uncertain): There is insufficient or minimal evidence. The decision is largely based on weak or indirect hints.

21–40 (Moderately Confident): There is noticeable evidence supporting a preference, though it is not comprehensive, and other interpretations are possible.

41–60 (Quite Confident): There is clear and convincing evidence that supports this decision, though it is not entirely decisive.

61–80 (Confident): There is strong evidence that clearly supports this decision, with very little ambiguity.

81–100 (Highly Confident): There is direct and explicit evidence that decisively supports this prediction.

Lastly, include a **brief (1–2 sentences)** and justify the score using the criteria above.

You must return a JSON object with these 12 keys:

- ”completeness_accuracy”, ”completeness_accuracy”_explanation”, ”completeness_accuracy”_confidence”

- ”neutrality”, ”neutrality_explanation”, ”neutrality_confidence”

- ”overall”, ”overall_explanation”, ”overall_confidence”

Do not include any text outside the JSON block.

Here is the evaluation task:

Given: ”user input”: ”user_input”, ”response_1”: ”response_a”, ”response_2”: ”response_b” Generation:

G. Other Details

G.1. Software

We used Python 3.13.2, Pytorch 2.7.0, and HuggingFace Transformers 4.52.4.All code is licensed under the Apache License 2.0.

G.2. Hardware

All experiments were run on a cluster with 4 NIVIDIA H100 GPUs and 1 NIVIDIA A100 GPU with 80B memory. For most inference jobs we used one GPU. For supervised fine-tuning, we used one GPU. The training for twelve epochs took about 2 hours.

G.3. Artifact Terms of Use

PersonaFeedback(Tao et al., 2025): Apache-2.0 FreebaseQA (Jiang et al., 2019): Creative Commons Attribution-NonCommercial-ShareAlike 3.0 International License WildChat (Zhao et al., 2024): AI2 ImpACT License

H. Information About Use Of AI Assistants

In this project, AI assistants were used for results visualization code (e.g., figures and tables) and for minor writing edits.

Distribution of Processed Profile Features

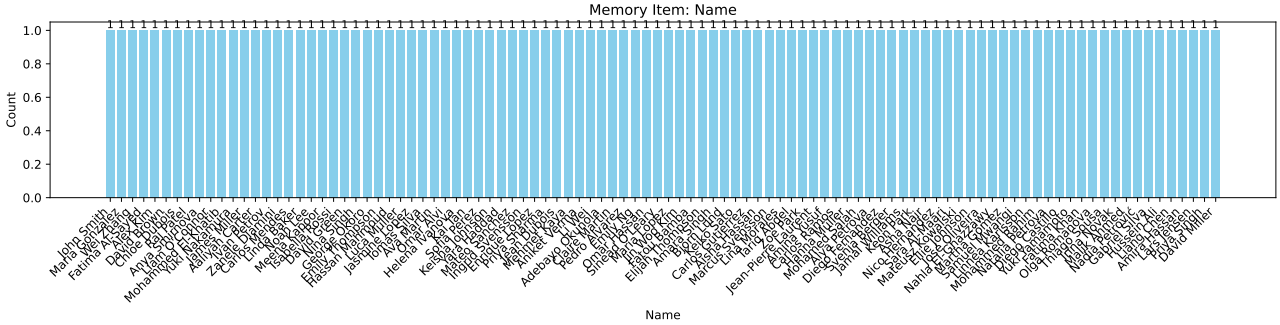


Figure 26. Distribution of Name

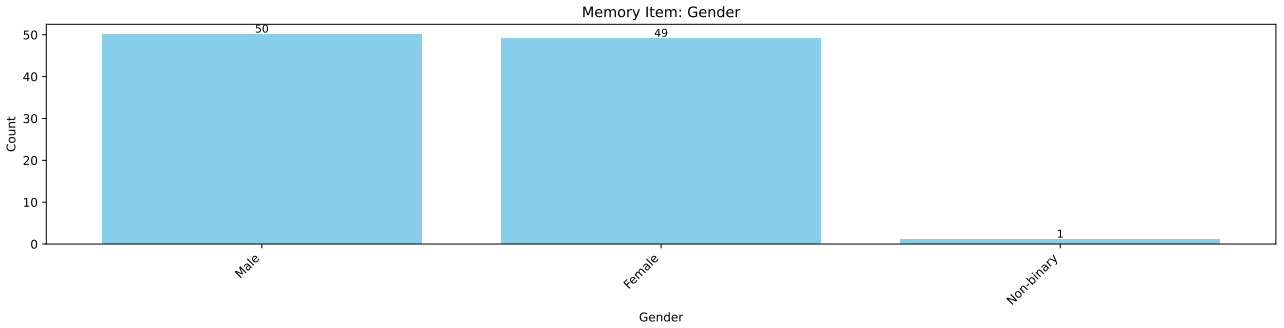


Figure 27. Distribution of Gender

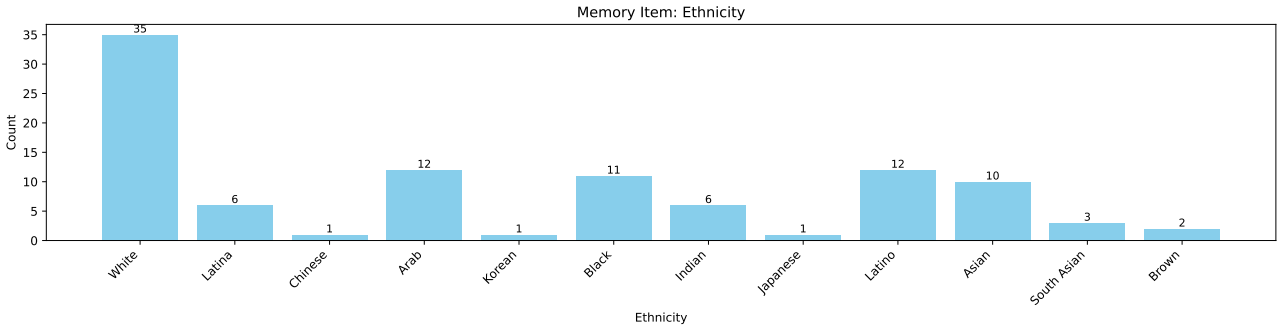


Figure 28. Distribution of Ethnicity

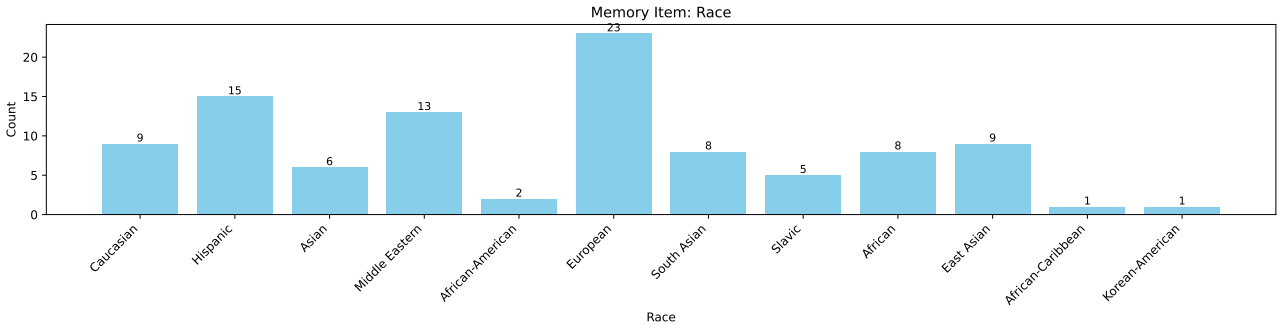


Figure 29. Distribution of Race

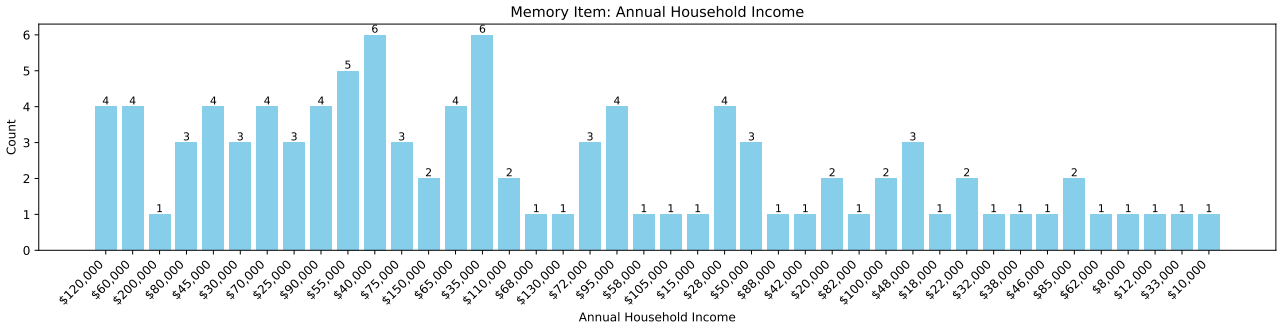


Figure 30. Distribution of Annual Household Income

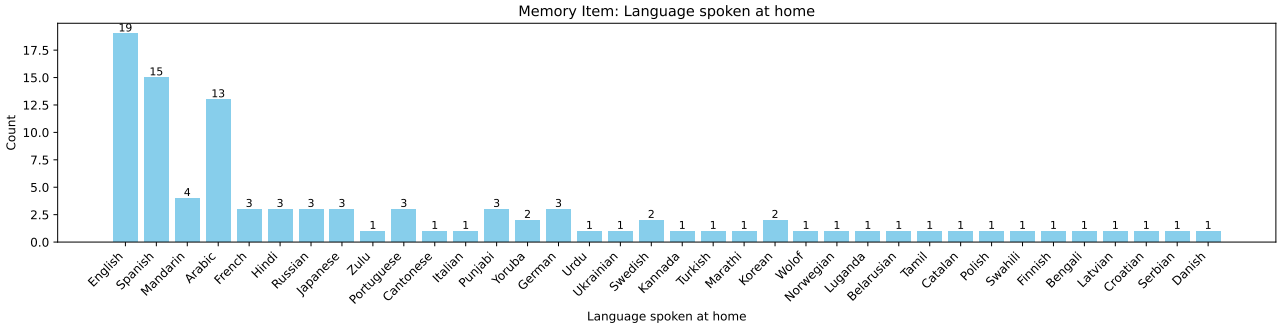


Figure 31. Distribution of Language spoken at home

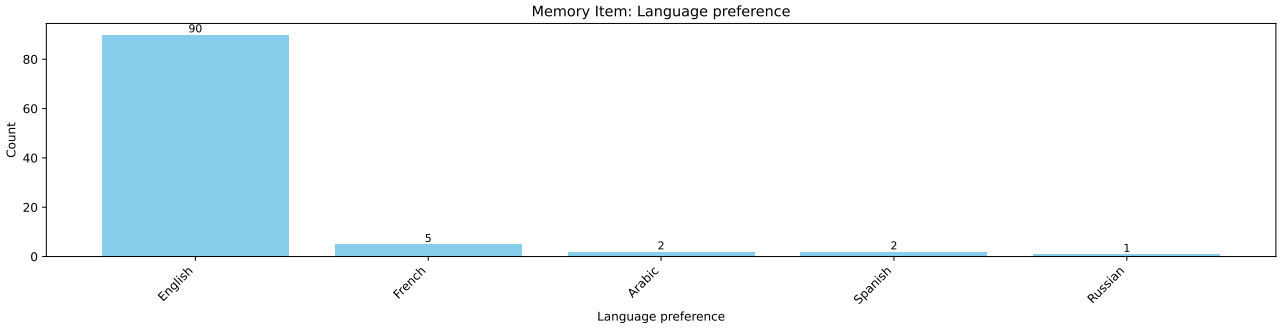


Figure 32. Distribution of Language preference

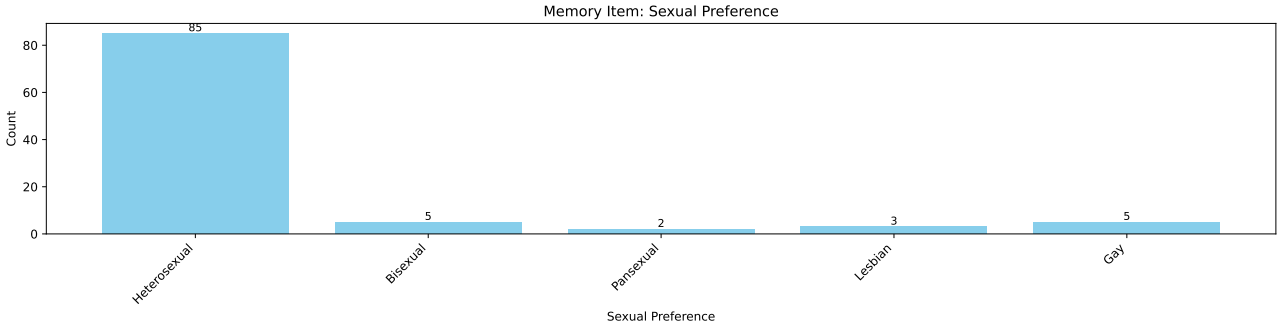


Figure 33. Distribution of Sexual Preference

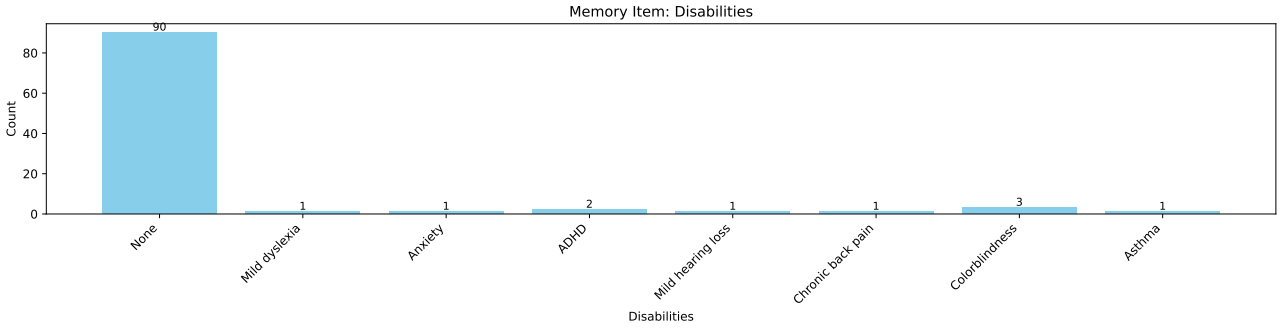


Figure 34. Distribution of Disabilities

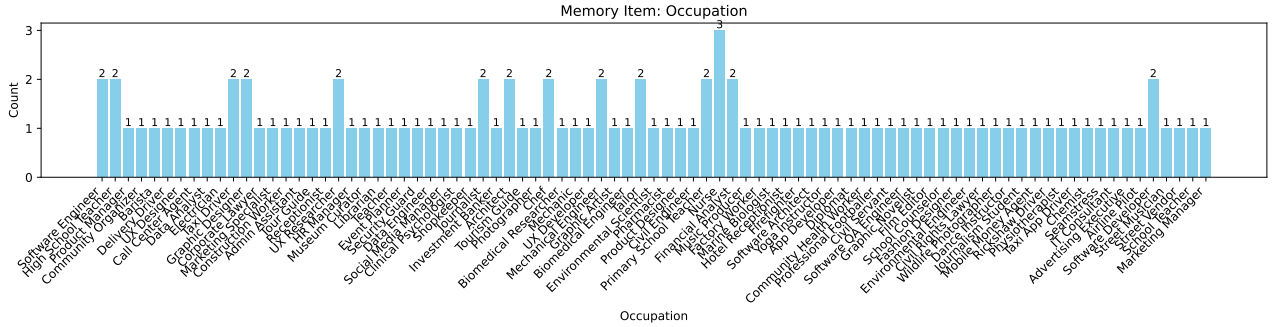


Figure 35. Distribution of Occupation

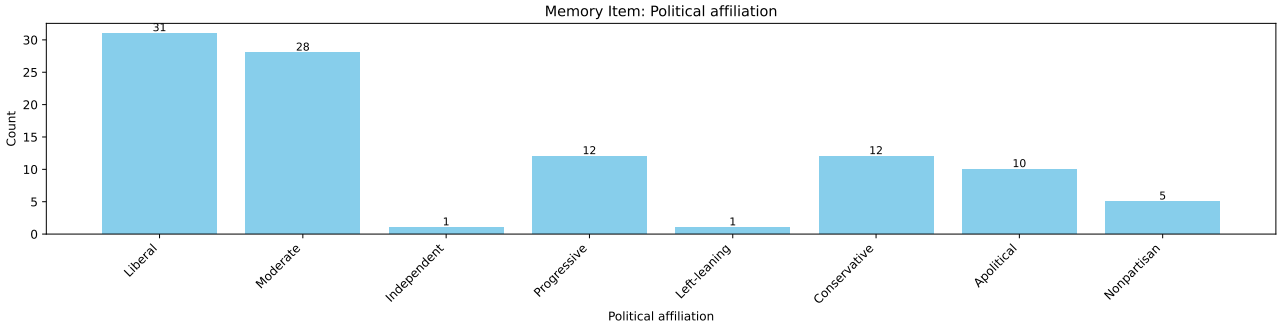


Figure 36. Distribution of Political affiliation

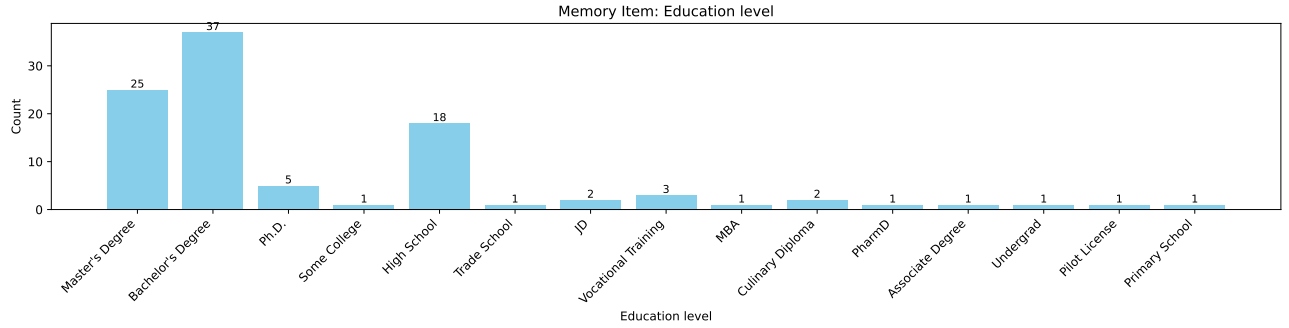


Figure 37. Distribution of Education level

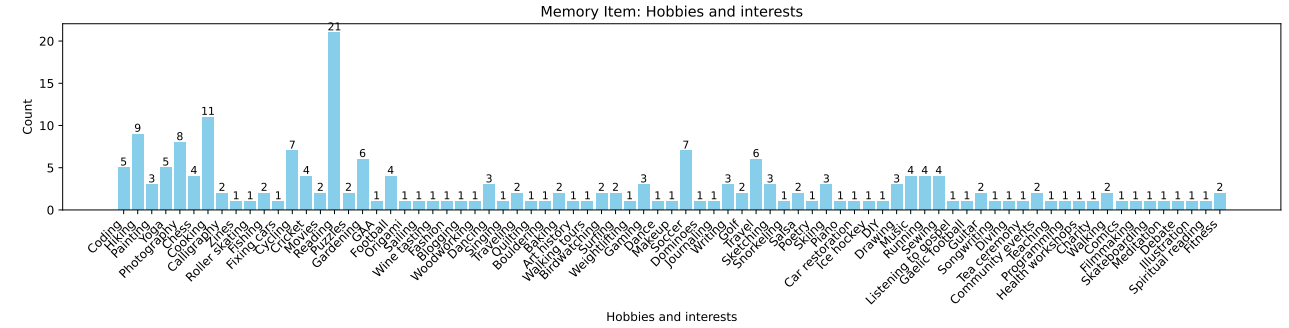


Figure 38. Distribution of Hobbies and interests

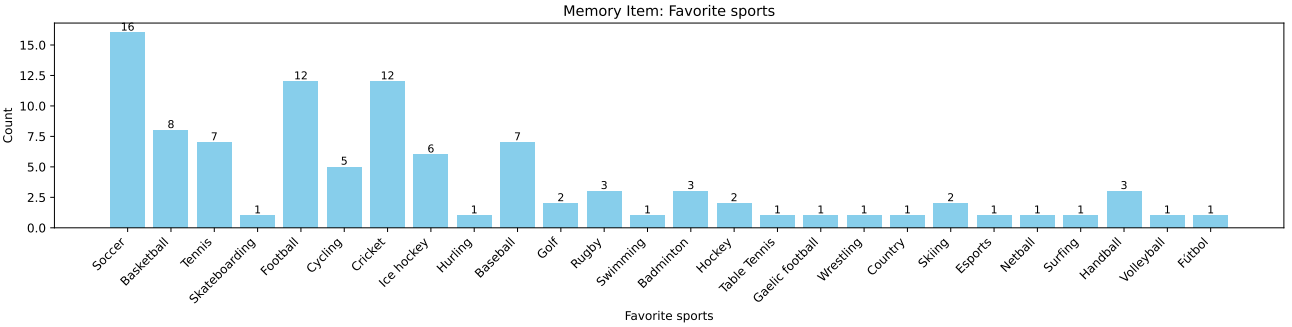


Figure 39. Distribution of Favorite sports

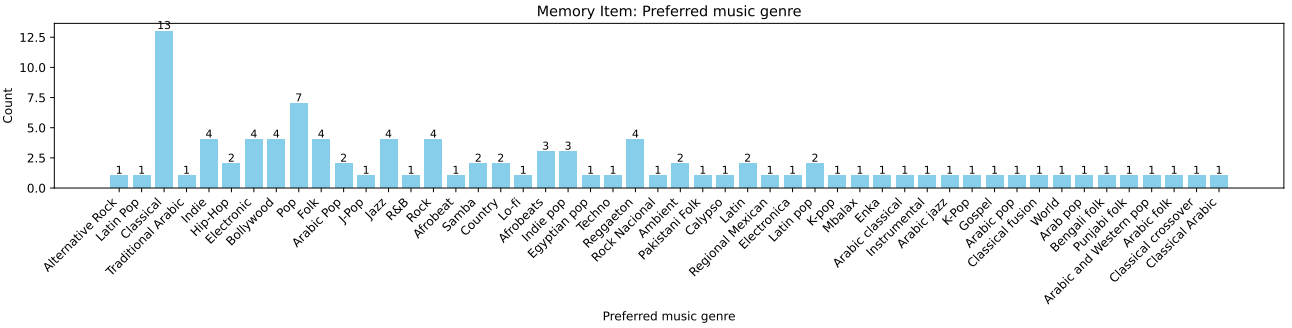


Figure 40. Distribution of Preferred music genre

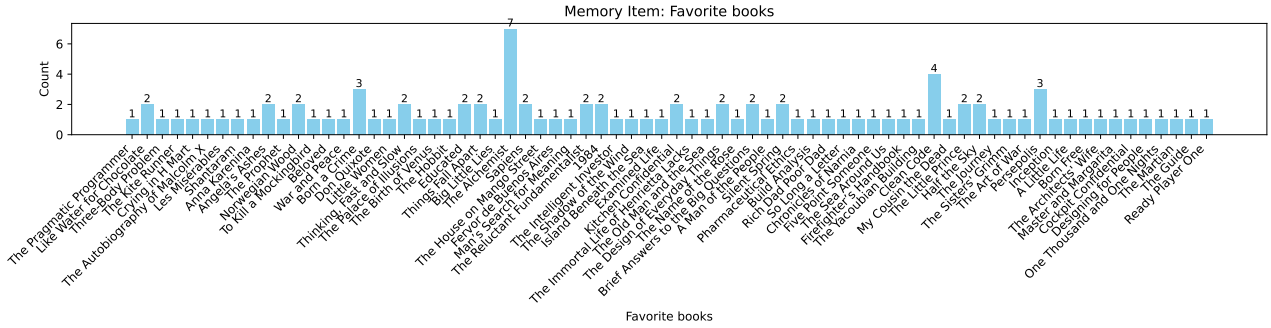


Figure 41. Distribution of Favorite books

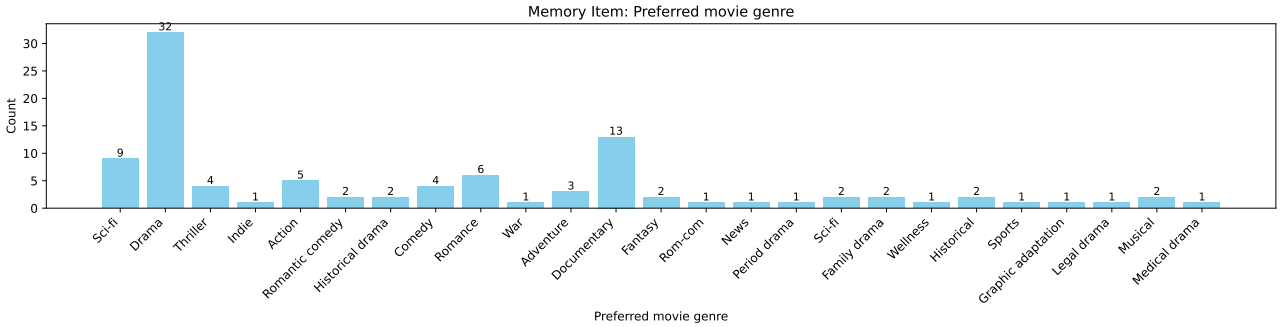


Figure 42. Distribution of Preferred movie genre

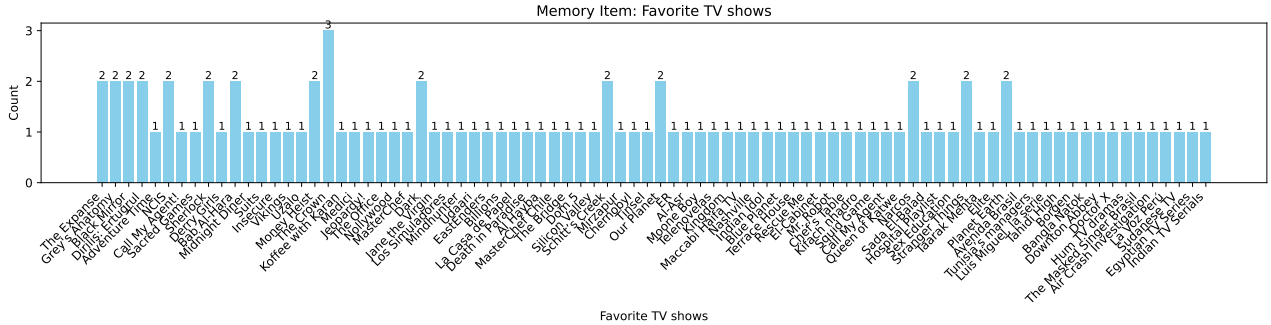


Figure 43. Distribution of Favorite TV shows

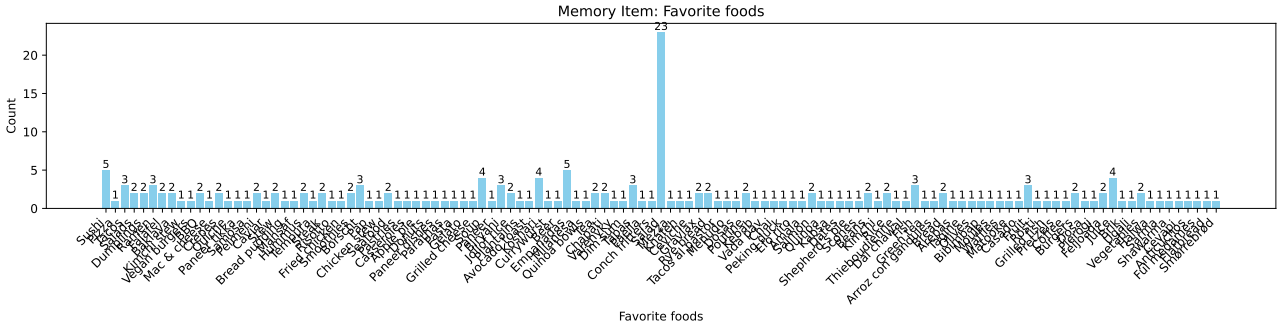


Figure 44. Distribution of Favorite foods

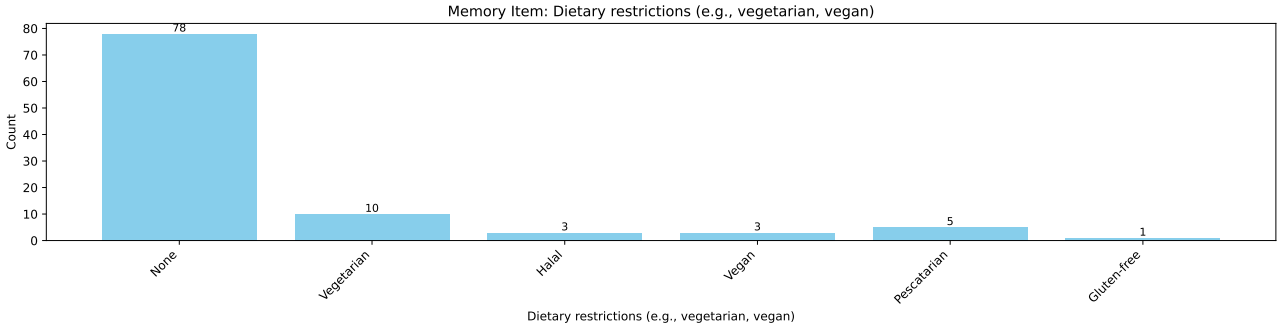


Figure 45. Distribution of Dietary restrictions (e.g., vegetarian, vegan)

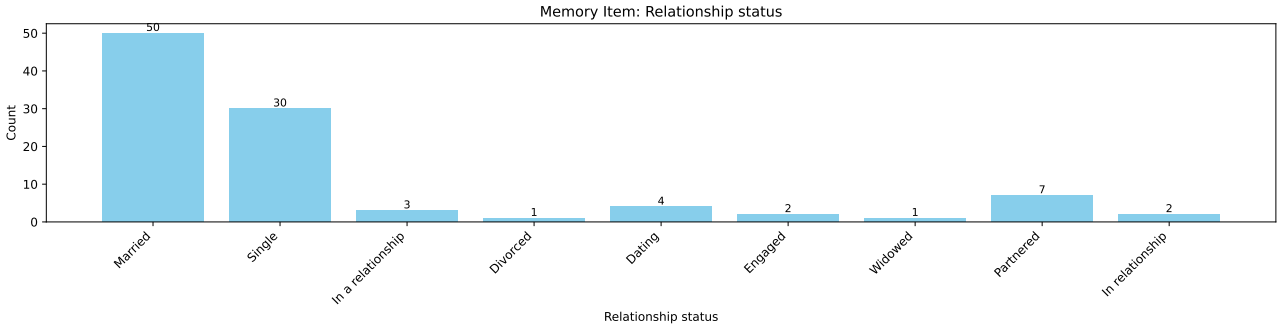


Figure 46. Distribution of Relationship status

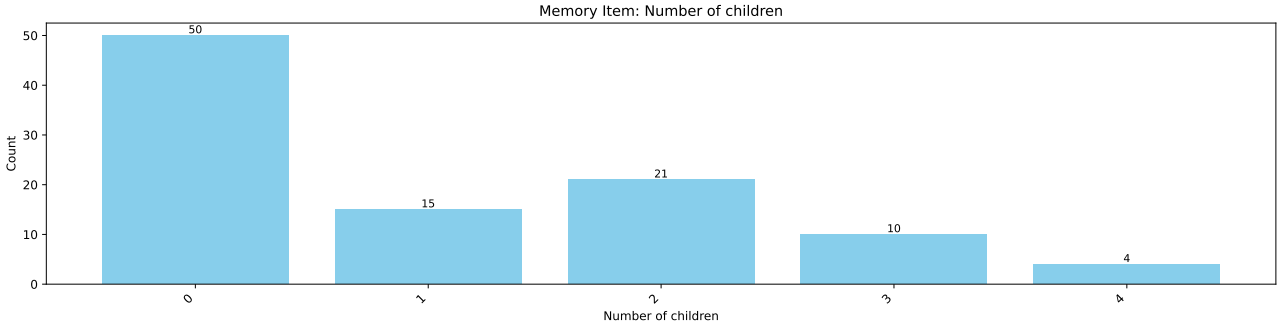


Figure 47. Distribution of Number of children

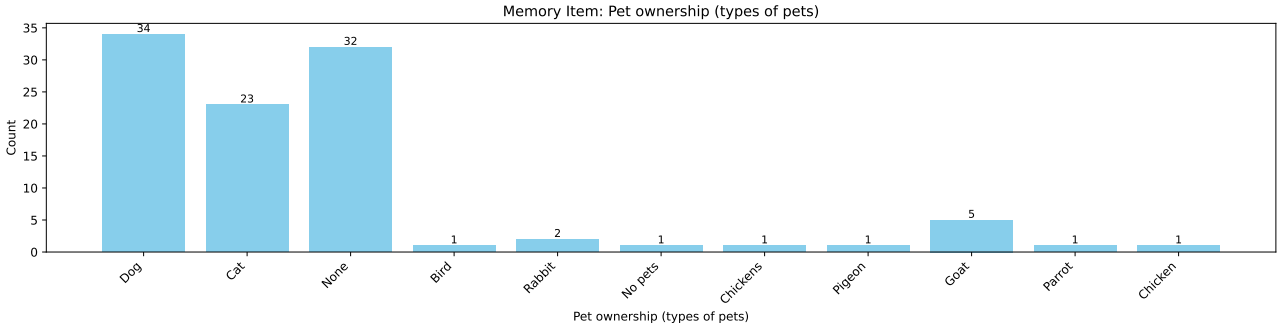


Figure 48. Distribution of Pet ownership (types of pets)

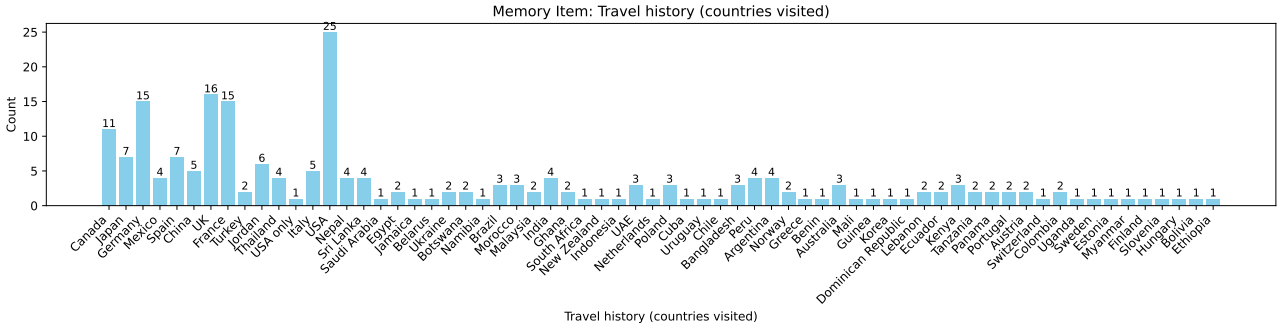


Figure 49. Distribution of Travel history (countries visited)

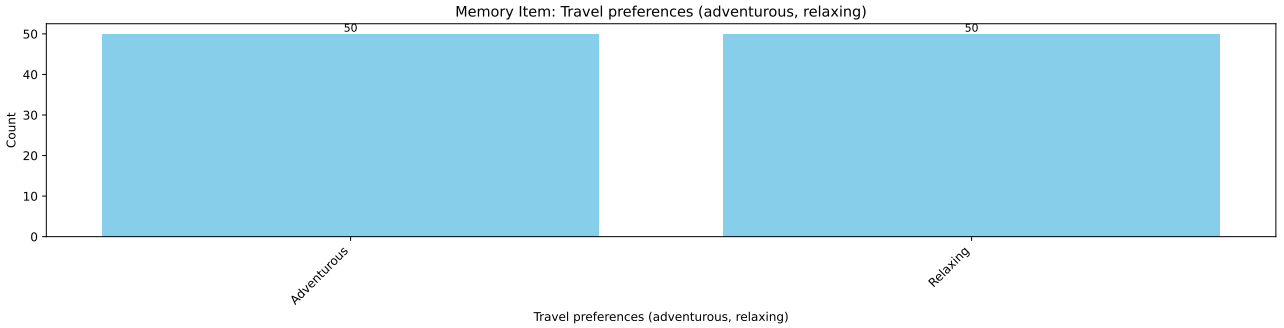


Figure 50. Distribution of Travel preferences (adventurous, relaxing)

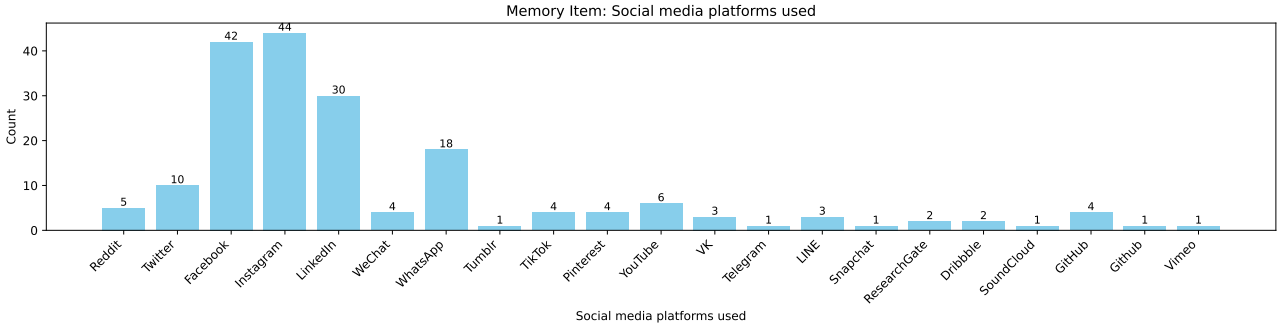


Figure 51. Distribution of Social media platforms used

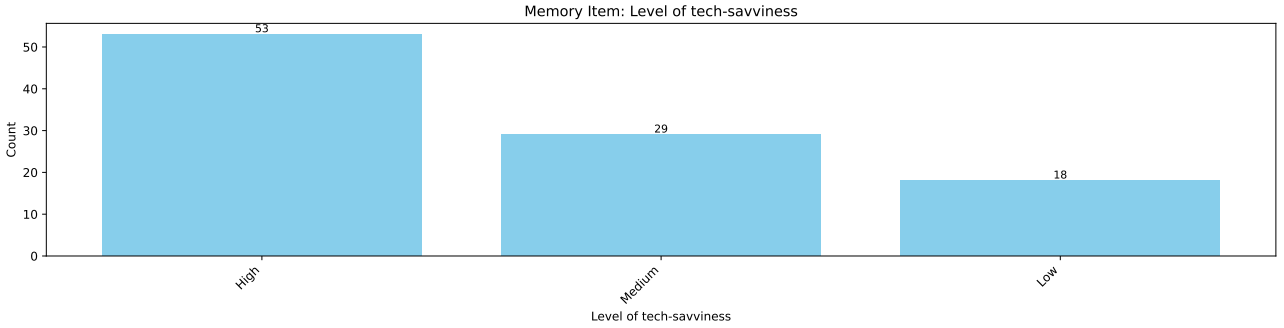


Figure 52. Distribution of Level of tech-savviness

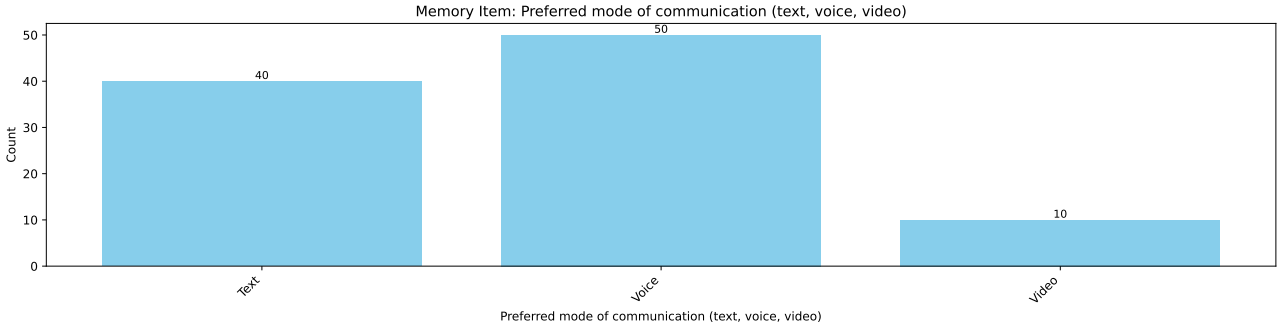


Figure 53. Distribution of Preferred mode of communication (text, voice, video)

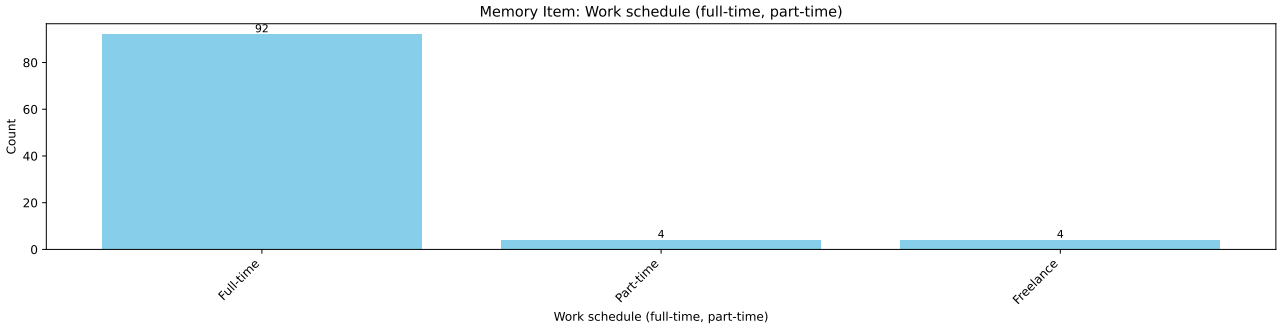


Figure 54. Distribution of Work schedule (full-time, part-time)

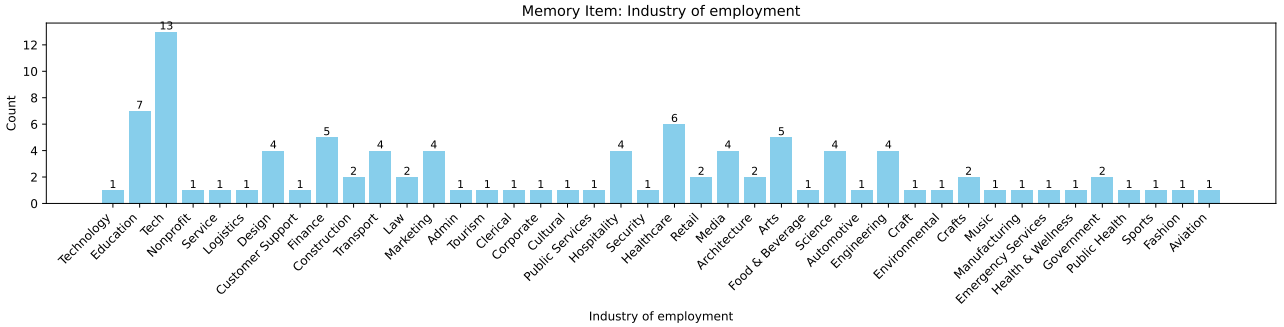


Figure 55. Distribution of Industry of employment

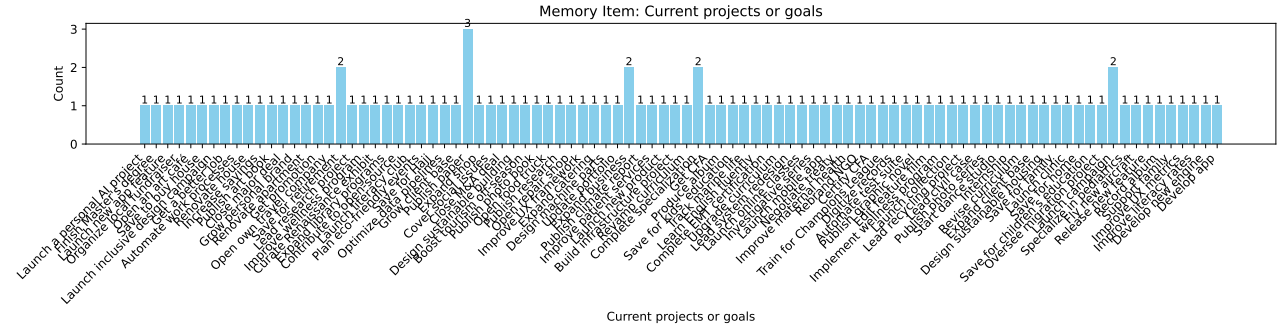


Figure 56. Distribution of Current projects or goals

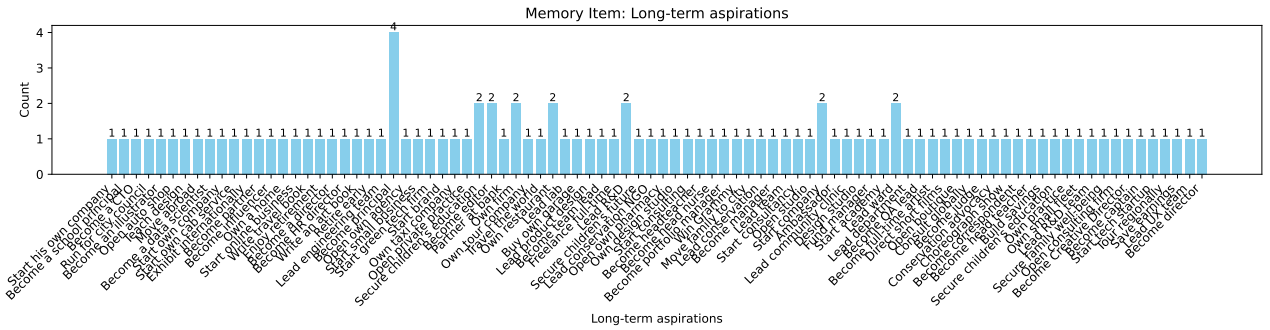


Figure 57. Distribution of Long-term aspirations

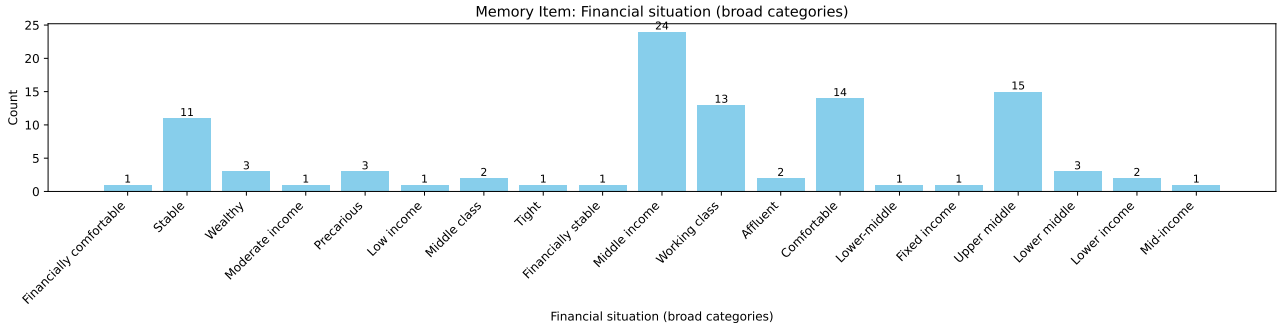


Figure 58. Distribution of Financial situation (broad categories)

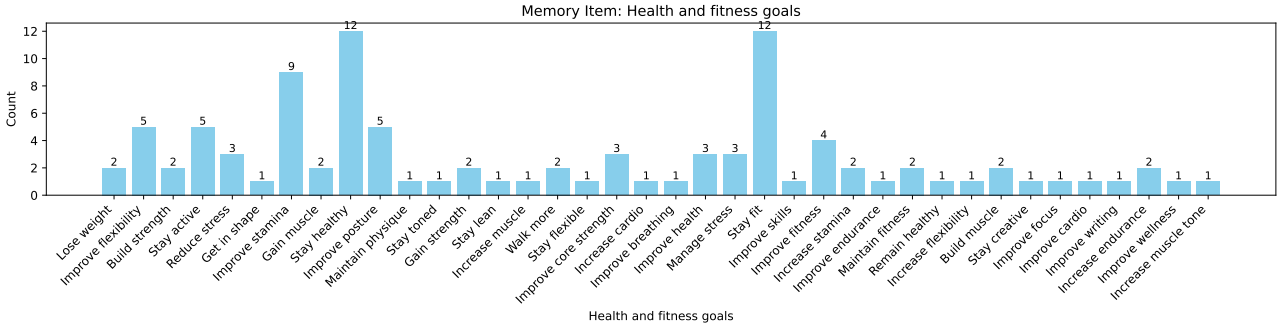


Figure 59. Distribution of Health and fitness goals

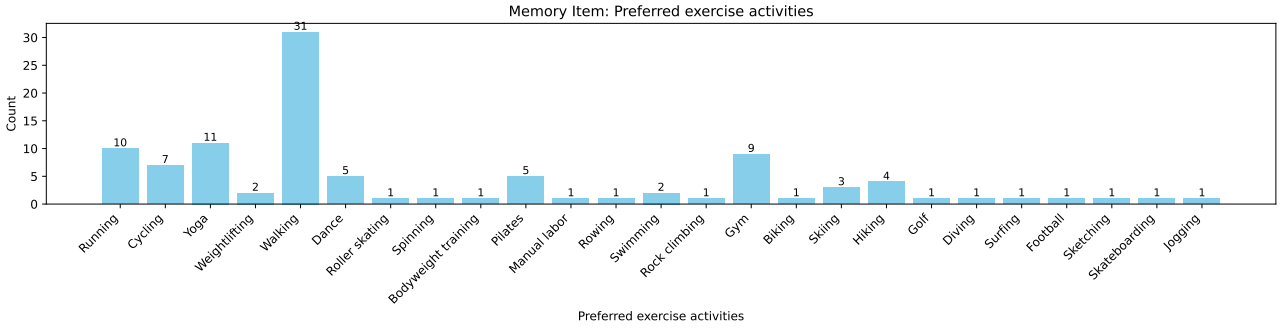


Figure 60. Distribution of Preferred exercise activities

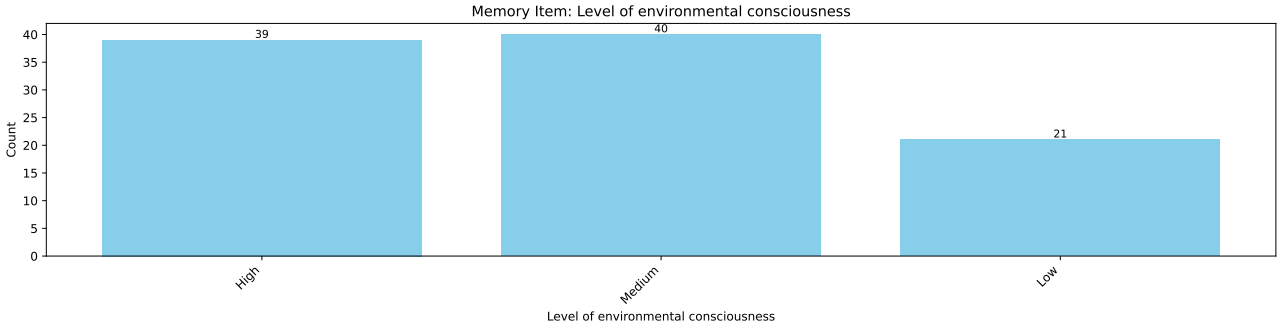


Figure 61. Distribution of Level of environmental consciousness

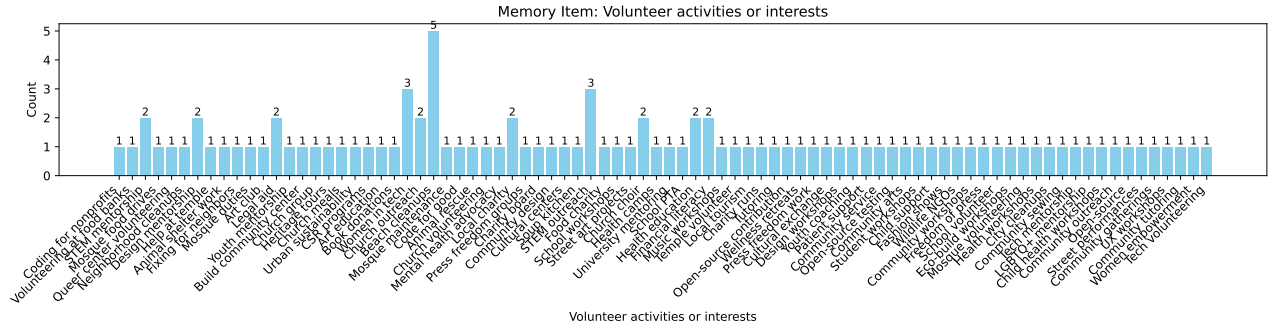


Figure 62. Distribution of Volunteer activities or interests

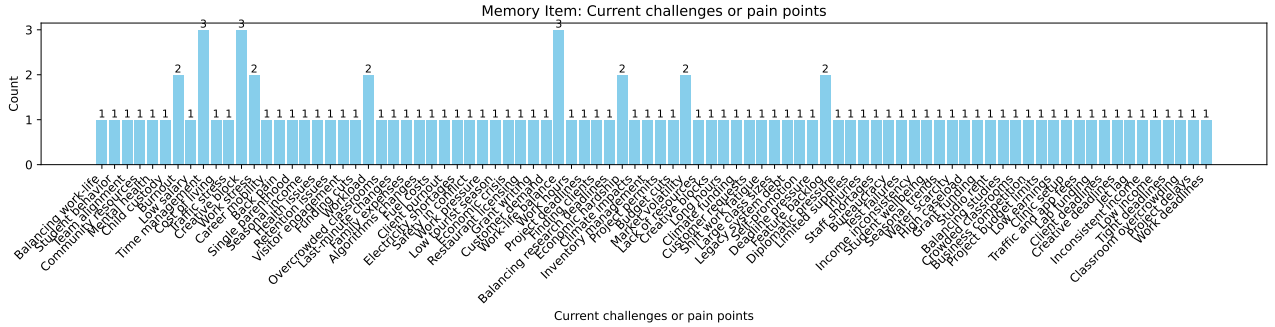


Figure 63. Distribution of Current challenges or pain points

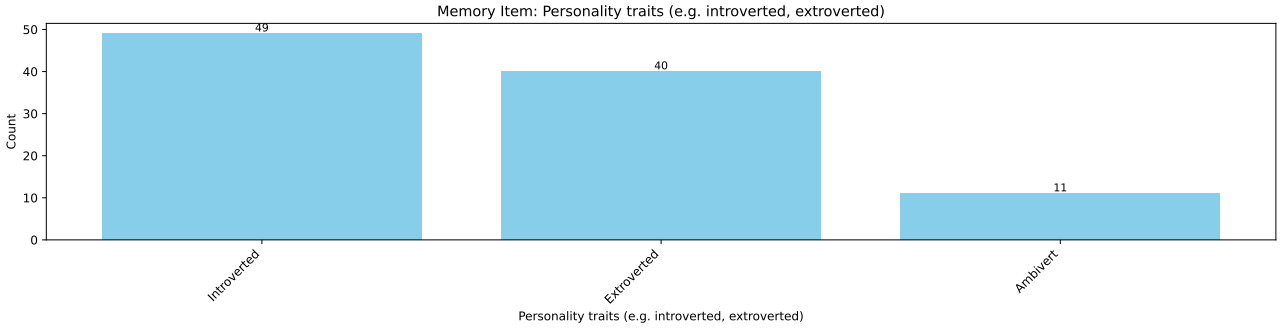


Figure 64. Distribution of Personality traits (e.g. introverted, extroverted)

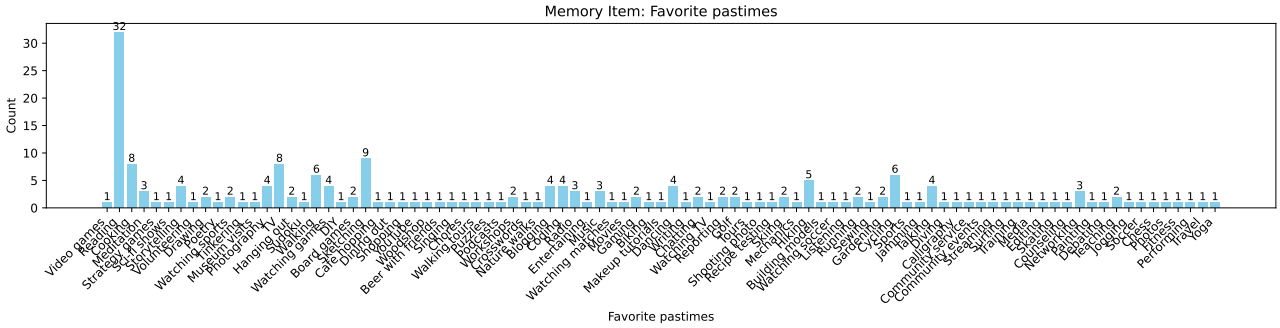


Figure 65. Distribution of Favorite pastimes

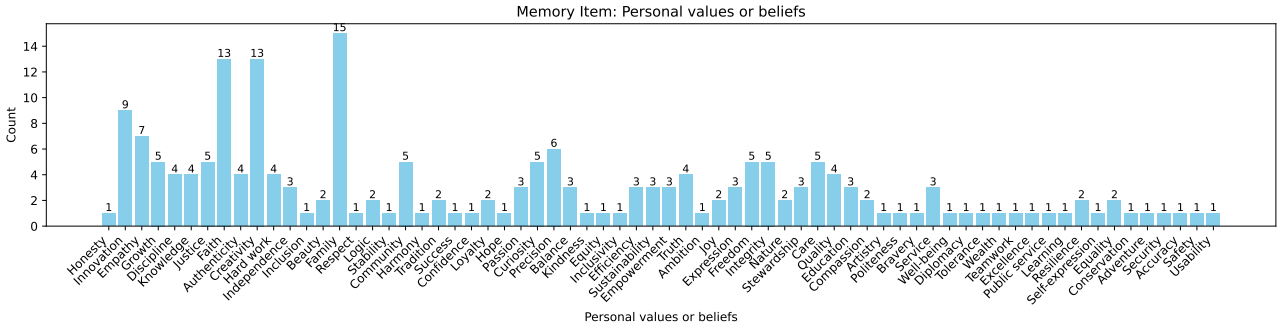


Figure 66. Distribution of Personal values or beliefs

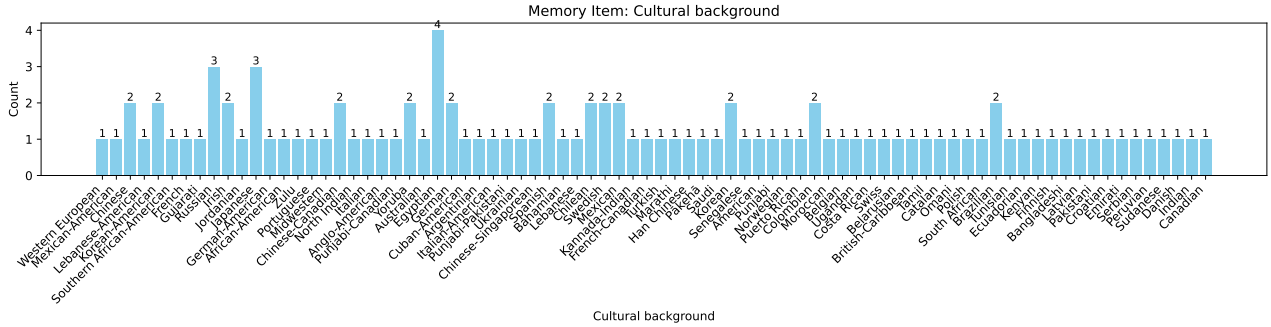


Figure 67. Distribution of Cultural background

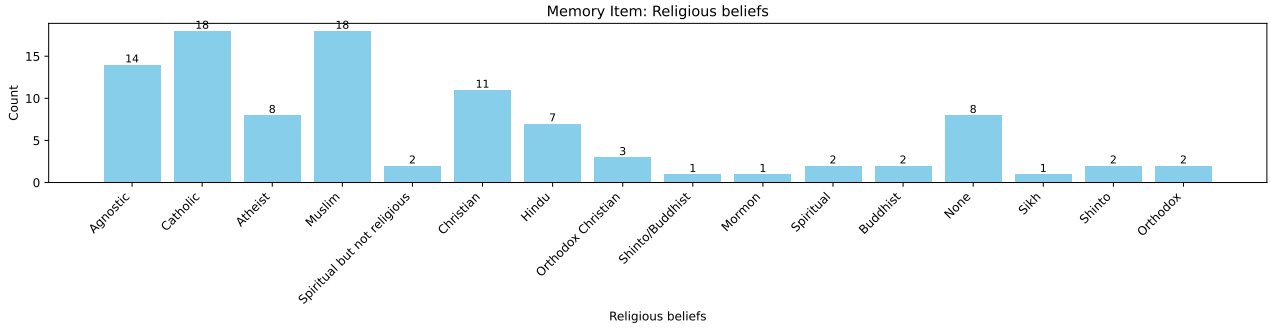


Figure 68. Distribution of Religious beliefs

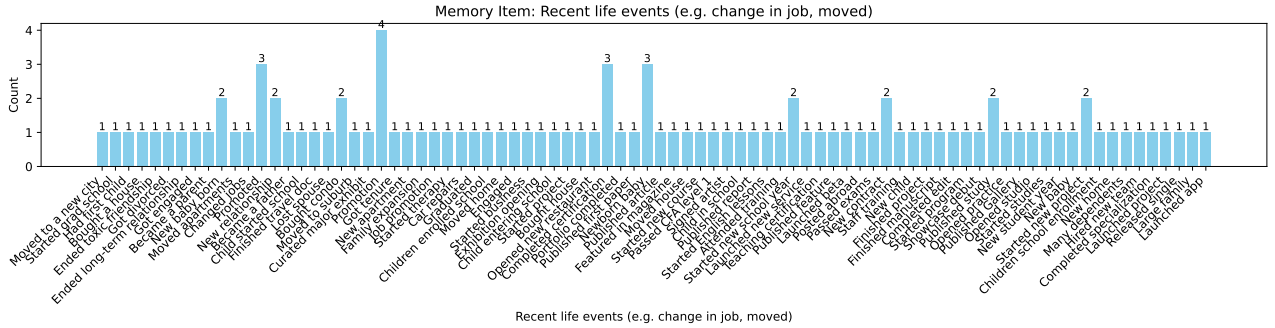


Figure 69. Distribution of Recent life events (e.g. change in job, moved)

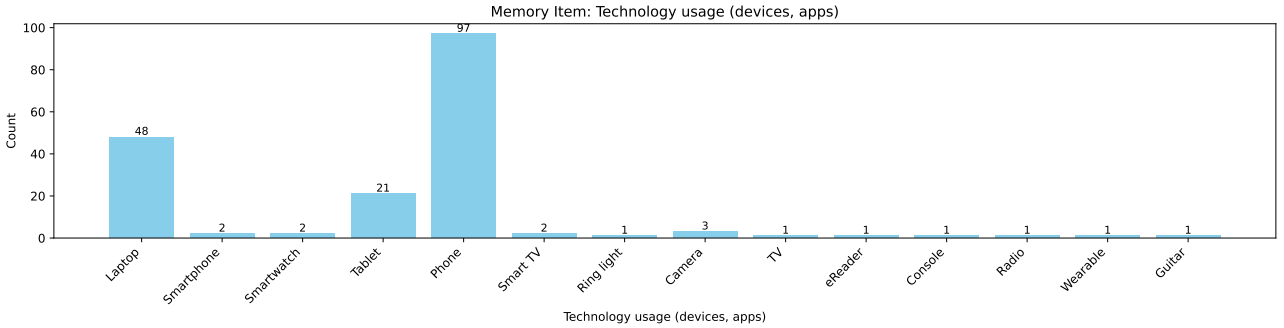


Figure 70. Distribution of Technology usage (devices, apps)

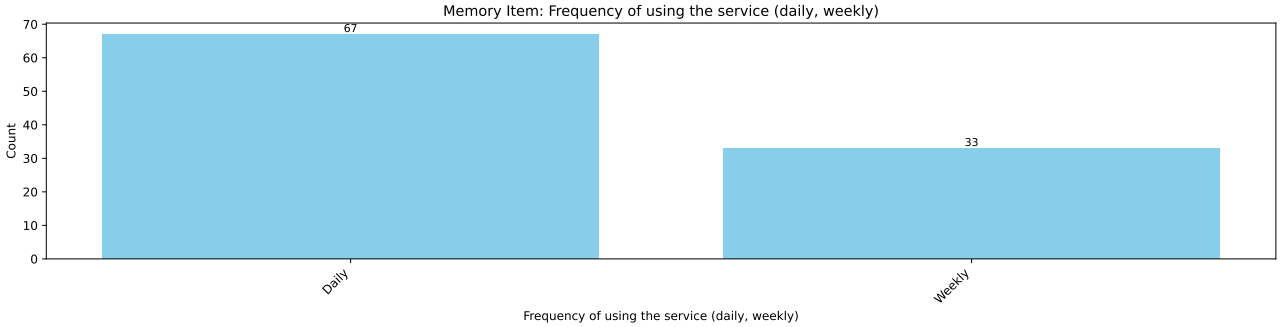


Figure 71. Distribution of Frequency of using the service (daily, weekly)

Figure 72. Distribution of Reasons for using the service

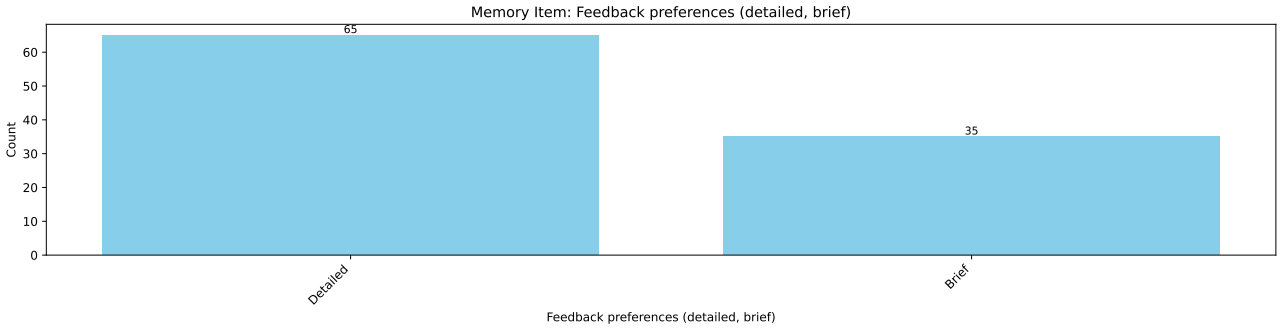


Figure 73. Distribution of Feedback preferences (detailed, brief)

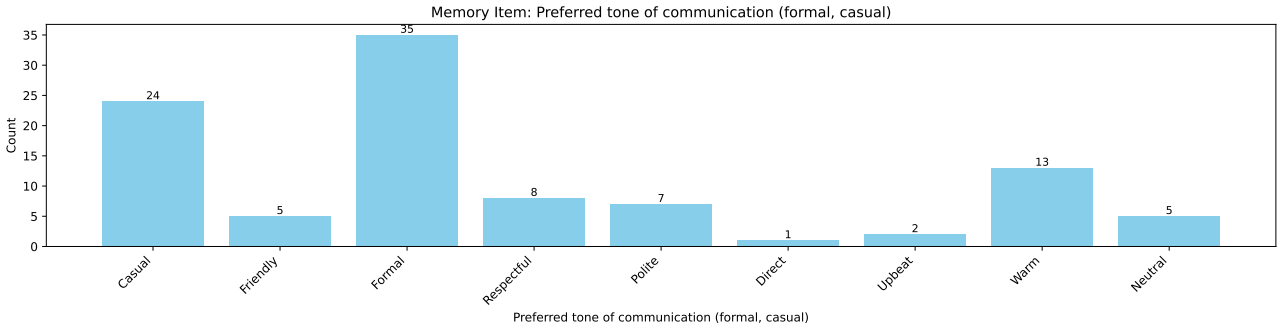


Figure 74. Distribution of Preferred tone of communication (formal, casual)

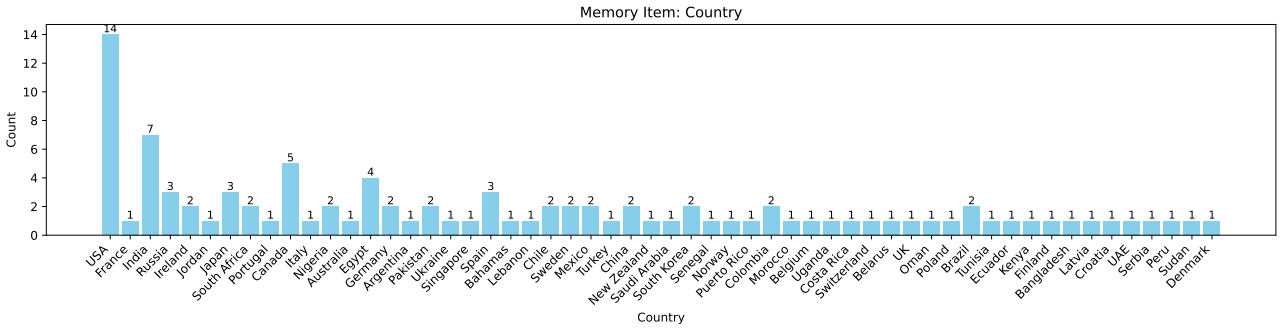


Figure 75. Distribution of Country