The PROPER Approach to Proactivity: Benchmarking and Advancing Knowledge Gap Navigation

Anonymous Author(s)

Abstract

Most language-based assistants follow a reactive *ask-and-respond* paradigm, requiring users to explicitly state their needs. As a result, relevant but unexpressed needs often go unmet. Existing proactive agents attempt to address this gap either by eliciting further clarification, preserving this burden, or by extrapolating future needs from context, often leading to unnecessary or mistimed interventions. We introduce **PROPER**, **Pro**activity-driven **Per**sonalized agents, a novel two-agent architecture consisting of a Dimension Generating Agent (DGA) and a Response Generating Agent (RGA). DGA, a fine-tuned LLM agent, leverages explicit user data to generate multiple implicit dimensions (latent aspects relevant to the user's task but not considered by the user) or knowledge gaps. These dimensions are selectively filtered using a reranker based on quality, diversity and task-relevance. RGA then balances explicit and implicit dimensions to tailor personalized responses with timely and proactive interventions. We evaluate PROPER across multiple domains using a structured, gap-aware rubric that measures coverage, initiative appropriateness, and intent alignment. Our results show that PROPER improves on quality-scores and win rates across all domains, achieving up to 84% gains in single-turn evaluation and consistent dominance in multi-turn interactions.

1 Introduction

Traditional conversational agents interact in a reactive manner and rely on end users to formulate specific queries and ask the right questions. While recent language-based agents introduce proactive behaviors, most forms of proactivity remain grounded in what users already know. Clarification-based interventions address either *known unknowns*, by eliciting missing specifications (Zhang et al., 2024b; Deng et al., 2024b; Hahn et al., 2025), or *unknown knowns*, by prompting users to surface implicit assumptions (Huang et al., 2024). Other approaches extrapolate from observable context or environment (Lu et al., 2025; Dong et al., 2025). However, such interventions, when insufficiently grounded in user intent, risk being mistimed or misaligned, ultimately degrading trust and interaction quality (Zargham, 2022; Chen et al., 2025; Brîncoveanu, 2025; Lei et al., 2025).

Additionally, existing methods do not explicitly address *unknown unknowns* (Kerwin, 1993): task-relevant considerations that are neither articulated by the user nor readily inferable from context. Such gaps often drive learning and exploration, but introducing them indiscriminately risks disrupting user intent. We therefore frame effective assistance as a problem of *calibrating personalization and proactivity*: deciding when to remain anchored in explicit user intent and when to activate implicit dimensions. As illustrated in Figure 1, in a coding query these implicit dimensions may include broader goals, input structure, project longevity, technology preferences, or scaling considerations.

We operationalize this calibration through a modular two-agent architecture. A Dimension Generating Agent (DGA) models how users express explicit dimensions and how effective assistance reasons over these dimensions, enabling it to identify both stated aspects of a query and plausible task-relevant gaps that remain unarticulated. These candidate dimensions are then evaluated by an activation module that assesses their relevance, timeliness, and compatibility with the user's expressed intent. Rather than surfacing all inferred gaps, this module selectively activates a subset of dimensions appropriate to the current interaction. The activated dimensions are then provided to a Response Generating Agent (RGA), which conditions generation on this curated set to ensure that additional information is introduced in a focused and non-disruptive manner. By regulating the balance between explicit personalization and proactive gap activation, PROPER introduces additional informa-

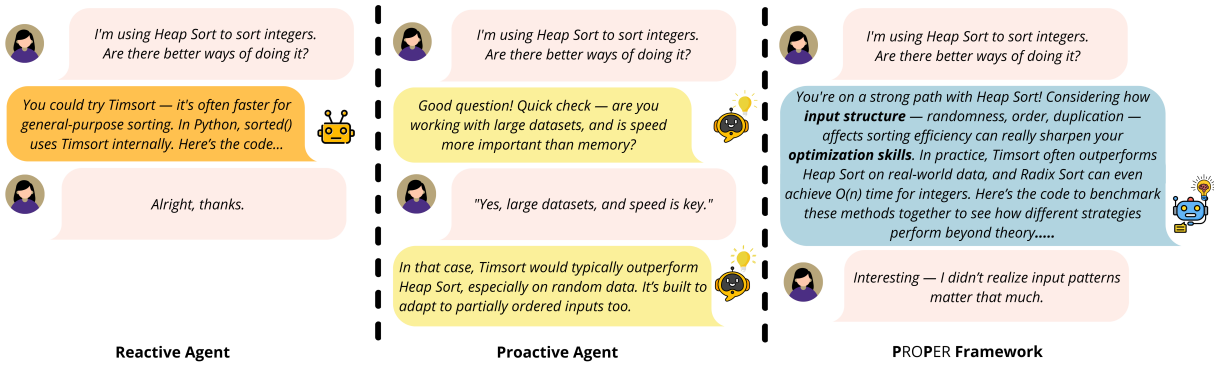

Figure 1: Different agent responses to a user query. The **Reactive Agent** provides immediate task fulfillment without exploring user context, goals, or learning needs. The **Proactive Agent** clarifies task-related ambiguities to optimize the immediate solution but remains confined to the user's explicitly stated problem space. PROPER, on the other hand, embodies a higher-order interaction strategy established through a learning-centric response structure.

tion only when it is timely and aligned with user intent, rather than treating proactivity as a uniform increase in initiative.

Finally, we evaluate PROPER using a structured, gap-aware rubric designed to assess proactive assistance beyond surface correctness. The rubric measures response quality along three dimensions: coverage of task-relevant knowledge gaps, appropriateness of initiative, and alignment with user intent. Our results show that PROPER consistently improves task utility over strong base LLMs and CoT prompting, especially in medical tasks where hidden risks, constraints, and user needs matter. The gains are smaller in tightly defined tasks like coding. To summarize, the key contributions are:

- We formalize *proactivity as a calibration problem*, emphasizing selective intervention over task-relevant *unknown unknowns*.
- We introduce a *dimension-based representation* of user needs and instantiate it through *ProPerBench*, a benchmark with dimension-level supervision.
- We propose PROPER, a modular agent architecture that decouples knowledge gap discovery from response generation to balance personalization and proactivity.
- We demonstrate across medical, recommendation, and coding domains that PROPER produces more helpful and better-timed responses than strong baselines.

2 Related Work

Prior work on proactive conversational agents has explored initiating assistance beyond explicit user queries. Clarification-based dialogue systems refine underspecified requests by eliciting additional input, effectively addressing *known unknowns* (Zhang et al., 2024b; Deng et al., 2024b; Hahn et al., 2025). Structured prompting and planning-based methods further improve reasoning and task handling through internal deliberation or decomposition, but remain grounded in information that is explicitly stated or readily inferable from context (Deng et al., 2023; Liu et al., 2024; Madaan et al., 2023).Recent approaches extend proactivity through autonomous planning, task monitoring, or adaptive policies (Dong et al., 2025; Lu et al., 2025), including multimodal and embodied agents that leverage environmental signals (Bandyopadhyay et al., 2025; Hahn et al., 2025). While these systems move beyond purely reactivity, their proactive behaviors are typically driven by observable state, predefined heuristics, or task progress, instead of modeling unarticulated user needs (Zhang et al., 2024a; Park et al., 2024; Ribeiro et al., 2020).

Human-centered studies emphasize that poorly calibrated proactivity can undermine user trust, agency, and interaction quality when assistance is mistimed or misaligned (Zargham, 2022; Chen et al., 2025; Brîncoveanu, 2025; Lei et al., 2025; Deng et al., 2024a). In parallel, personalization and user modeling (Park et al., 2024; Ribeiro et al., 2020) capture latent preferences or goals from history, but focus on what users want rather than what they do not yet know. In contrast, our work treats unarticulated, task-relevant knowledge gaps as first-class elements of reasoning and frames proactive assistance as a calibration problem. By separating explicit needs from implicit gaps and regulating their use, PROPER enables calibrated proactivity aligned with user intent.

3 Problem Formulation

Let $q \in \mathcal{Q}$ denote the current user query, $h \in \mathcal{H}$ the interaction history, and $p \in \mathcal{P}$ persona-related explicit attributes. We define the user state as $u = (q, h, p) \in \mathcal{U}$. Given u, a baseline system produces an initial response $r_0 \in \mathcal{R}$ based on the explicitly available information. However, some task-relevant aspects may remain unarticulated in (q, h) or unaddressed in r_0. The objective is to generate a final response $r \in \mathcal{R}$ by selectively intervening when warranted by the user state.

Definition 1 (Interaction Dimensions) *Let $\mathcal{D}$ denote a shared universe of interaction dimensions. Given an interaction state $I = (u, r_0)$, the set of interaction dimensions is defined as:*

$$\mathcal{D}(I) := \mathcal{D}^{user}_{exp}(u) \cup \mathcal{D}^{sys}_{exp}(r_0) \cup \mathcal{D}_{imp}(u), \quad (1)$$

where,

$$\mathcal{D}^{user}_{exp}(u) = \{d \in \mathcal{D} \mid d \prec (q, h)\}, \quad (2)$$

$$\mathcal{D}^{sys}_{exp}(r_0) = \{d \in \mathcal{D} \mid d \prec r_0\}, \quad (3)$$

$$\mathcal{D}_{imp}(u) = \{\tilde{d} \in \mathcal{D} \mid \tilde{d} \not\prec (q, h) \ \wedge \ \tilde{d} \sim u\}. \quad (4)$$

Here, $\mathcal{D}(I)$ comprises user-explicit dimensions $\mathcal{D}^{\text{user}}_{\text{exp}}(u)$, system-explicit dimensions $\mathcal{D}^{\text{sys}}_{\text{exp}}(r_0)$, and implicit task-relevant dimensions $\mathcal{D}_{\text{imp}}(u)$, enabling reasoning about both unaddressed explicit needs and unarticulated knowledge gaps.

Definition 2 (Selective Activation) *Given an interaction state $I = (u, r_0)$, selective activation identifies a set of interaction dimensions $\mathcal{D}_{act}(I) \subseteq \mathcal{D}(I)$ such that*

$$\mathcal{D}_{act}(I) \subseteq \left(\mathcal{D}^{user}_{exp}(u) \backslash \mathcal{D}^{sys}_{exp}(r_0)\right) \cup \mathcal{D}_{imp}(u). \quad (5)$$

The set $\mathcal{D}_{\text{act}}(I)$ captures dimensions that warrant proactive intervention, including user-explicit dimensions not addressed by the baseline response and task-relevant implicit dimensions that remain unarticulated. Selective activation thus unifies two sources of incompleteness: unmet explicit needs and latent knowledge gaps.

Definition 3 (Post-hoc Calibrated Ranking) *Given an interaction state $I = (u, r_0)$ and a selectively activated candidate set $\mathcal{D}_{act}(I)$, post-hoc calibrated ranking defines the problem of selecting a budgeted subset of interaction dimensions $S_k \subseteq \mathcal{D}_{act}(I)$, where k denotes a fixed intervention budget.*

The selected set S_k represents the dimensions that warrant proactive intervention in the final response. Calibrated ranking balances three considerations: quality, whether it addresses explicit user needs, and the degree of redundancy.

Definition 4 (PROPER Calibration) *Given $I = (u, r_0)$, user's interaction state and $\mathcal{D}(I) = \mathcal{D}^{user}_{exp}(u) \cup \mathcal{D}^{sys}_{exp}(r_0) \cup \mathcal{D}_{imp}(u)$, the corresponding interaction dimensions. Let $U_u := \mathcal{D}^{user}_{exp}(u)$ and $S^*_k(I) \subseteq \mathcal{D}_{act}(I)$ denote the user-explicit dimensions and a calibrated set of activated dimensions, respectively. Let $\mathcal{F} : \mathcal{R} \times \mathcal{U} \times \mathcal{R} \times 2^{\mathcal{D}} \times 2^{\mathcal{D}} \to \mathbb{R}$ be a utility function. The learning objective is*

$$\hat{r} = \arg\max_{r \in \mathcal{R}} \mathcal{F}(r \mid u, r_0, U_u, S^*_k(I)). \quad (6)$$

We formulate proactive assistance as a calibration problem over response repair, asking when and which unaddressed or implicit dimensions should be introduced given a user state and baseline response. The objective $J(\cdot)$ defines this selection problem, and the utility $\mathcal{F}$ evaluates candidate responses conditioned on the selected dimensions.

4 Proposed Methodology

This section describes our methodology, which operationalizes the formulation in Section 3 via three stages. First, we construct *interaction dimensions* by separating user-explicit dimensions, system-covered dimensions, and implicit task-relevant gaps (Def. 1). Second, we *selectively activate* a candidate set and perform *calibrated ranking* to obtain a budgeted intervention set (Def. 2 and Def. 3). Finally, we generate a response conditioned on the user-explicit dimensions and the calibrated intervention set (Def. 4).

4.1 Dimension Generating Agent (DGA)

DGA is responsible for inferring implicit dimensions $\mathcal{D}_{\text{imp}}(u) \subseteq \mathcal{D}$ that correspond to task-relevant knowledge gaps. For this, we fine-tune an LLM using supervision derived from successful assistance trajectories, where interactions are annotated with dimension-level labels (details in Appendix A.1). Through this, DGA learns to generate plausible missing dimensions conditioned on the user state.

At inference time, given $u = (q, h, p)$, the DGA produces a set of candidate implicit dimensions $\mathcal{D}_{\text{imp}}(u)$ along with log probabilities as confidence scores (see Appendix A.1.1 for the exact prompt used for DGA). User-explicit dimensions $\mathcal{D}^{\text{user}}_{\text{exp}}(u)$

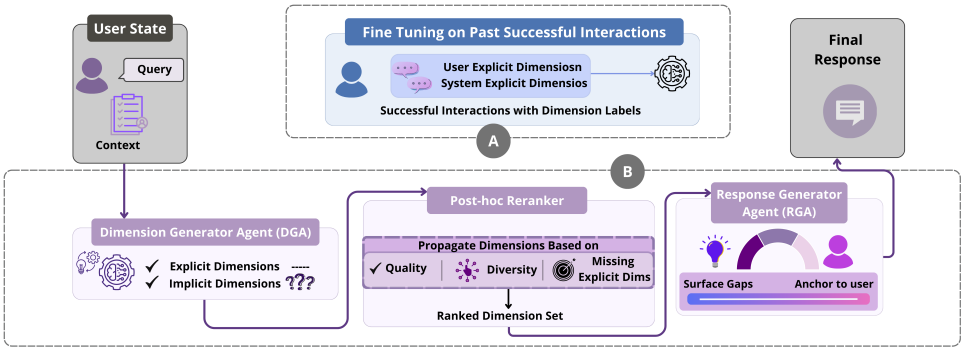

Figure 2: Overview of the PROPER framework. During training (A), the Dimension Generating Agent (DGA) is fine-tuned on successful interactions annotated with user- and system-explicit dimensions, learning task-specific priors. At inference (B), the DGA identifies explicit and candidate implicit dimensions from the user state. A post-hoc reranker selects a calibrated subset, and the Response Generating Agent (RGA) updates the base response by selectively integrating them, balancing proactivity with user intent.

and baseline-covered dimensions $\mathcal{D}^{\text{sys}}_{\text{exp}}(r_0)$ are extracted separately, enabling construction of $\mathcal{D}(I)$ for $I = (u, r_0)$ (Def. 1).

Through fine-tuning, DGA goes beyond text to internalize all the key dimensions considered to assist a user. At inference time, given a user query, DGA leverages its internalized dimensions of other tasks to produce a set of candidate interaction dimensions $\mathcal{D}(u)$ which were used for successful assistance and the current user might be missing.

4.2 Post-hoc Calibrated Reranker

The reranker determines which interaction dimensions inferred by the DGA should be activated for response generation. It operationalizes selective activation and calibrated ranking (Def. 2, Def. 3), treating proactivity as a controlled decision.

Given an interaction state $I = (u, r_0)$, the reranker constructs the activation pool $\mathcal{D}_{\text{act}}(I)$ comprising (i) user-explicit dimensions not addressed by the baseline response and (ii) diverse and non-redundant implicit task-relevant dimensions. All dimensions are encoded using BGE-small embeddings. From this pool, the reranker selects a budgeted set $S^*_k(I)$ by maximizing the objective in Eq. 7. The objective favors dimensions with high DGA confidence, encourages alignment with unmet explicit user needs via semantic similarity, and penalizes redundancy among selected dimensions to promote complementary coverage. The budget k controls the degree of proactivity by limiting how many dimensions are activated.

$$S^*_k(I) = \arg \max_{S \subseteq \mathcal{D}_{\text{act}}(I),\, |S|=k} \Bigg[\underbrace{\sum_{d \in S} \log P(d \mid u)}_{\text{quality}} - \lambda_1 \underbrace{\sum_{d \in S} \max_{e \in \mathcal{E}(I)} \text{sim}(d, e)}_{\text{unmet explicit alignment}} - \lambda_2 \underbrace{\sum_{i<j} \text{sim}(d_i, d_j)}_{\text{implicit diversity}} \Bigg]. \tag{7}$$

where $I = (u, r_0)$ and $\mathcal{E}(I) = \mathcal{D}^{\text{user}}_{\text{exp}}(u) \setminus \mathcal{D}^{\text{sys}}_{\text{exp}}(r_0)$. All similarity terms are computed as cosine similarity between BGE-encoded dimension representations.

4.3 Response Generating Agent (RGA)

The Response Generating Agent (RGA) generates the final system response by selectively updating a baseline response using the calibrated set of activated interaction dimensions. It is implemented as a prompt-driven generation module that conditions on the user query, interaction context, the baseline response, and the activated dimensions selected by the post-hoc reranker.

Given an interaction state $I = (u, r_0)$ and a calibrated intervention set $S^*_k(I)$, the RGA produces an updated response $\hat{r}$ that preserves the intent and structure of r_0 while selectively incorporating additional information. The prompt explicitly encodes *initiative calibration*: the RGA infers how proactive to be from the user query itself, expanding when signals of confusion, risk, or uncertainty are present, and remaining focused when the query is narrow or constrained.

The prompt further distinguishes between explicit and implicit dimensions, requiring explicit

Dataset →	Medical (MD)		Code-Contests		PWAB	
Models ↓	μScore	Win%	μScore	Win%	μScore	Win%
LLAMA-8B	2.19	10.52	1.26	15.51	2.34	6.83
vs LLAMA-8B + PROPER	**3.86**†	**89.48**†	**2.13**†	**84.49**†	**4.06**	**93.17**†
QWEN-8B	2.93	18.73	2.24	24.76	3.12	12.50
vs QWEN-8B + PROPER	**4.03**†	**81.27**†	**2.84**†	**75.24**†	**4.29**†	**87.50**†
GPT-4	3.28	29.74	**3.19**†	**68.93**†	3.46	23.61
vs LLAMA-8B + PROPER	**3.73**	**70.26**	2.08	31.07	**4.11**†	**76.39**†
GPT-4	3.26	19.26	3.11	43.63	3.53	17.40
vs QWEN-8B + PROPER	**4.03**†	**80.74**†	**2.71**	**56.37**	**4.24**†	**82.60**†

Table 1: One-on-one comparison of LLAMA-8B, QWEN-8B, GPT-4, and our model, with GPT-5 as judge. **μScore** denotes the average score across all entries in the dataset; **Win%** denotes the percentage of samples where the model outperformed the paired model. PROPER improves over base LLMs and surpasses GPT-4 in Medical (MD) and PWAB; gains are smaller in Code-Contests, as both models may find optimal solutions. † indicates $p \leq 0.01$.

gaps to be addressed by default while incorporating implicit gaps only when warranted by the interaction context. To avoid overreach, the RGA is constrained to favor concise additions over full rewrites and to ask at most one clarifying question when resolving an implicit gap would require user-specific information. These constraints ensure that proactivity remains targeted, non-assumptive, and aligned with user intent.

We use domain-specific instantiations of this prompt for coding assistance, clinical support, and recommendation settings. The full prompts for all three domains are provided in Appendix A.4.

4.4 PROPER Framework

We now describe the end-to-end execution of the PROPER framework. The process begins with the construction of a user state composing user query, interaction history, and persona related information (if available). Together, these components define the interaction context used throughout the pipeline. A baseline system response r_0 is generated using a standard assistant, providing a reference point for later phases. The first stage is handled by the Dimension Generating Agent (DGA). Given the interaction context, the DGA infers a set of interaction dimensions $\mathcal{D}(u)$ by distinguishing between explicit dimensions surfaced in the interaction and implicit gaps, task-relevant dimensions that are not currently addressed. The DGA outputs a pool of candidate dimensions along with confidence scores reflecting their relevance under the user state.

In the second stage, a post-hoc calibrated reranker selects a budgeted subset $S_k^*(u, r_0)$ by optimizing quality, alignment with user-explicit needs, and diversity. This frames proactivity as a controlled choice of which missing aspects to address, rather than simple response expansion.

Finally, RGA takes the input user query, the baseline response, explicit dimensions, and the calibrated set of activated dimensions. Conditioned on this structured guidance, the RGA produces an updated response $\hat{r}$ that preserves the intent and structure of the baseline output while selectively addressing the activated dimensions. Explicit dimensions anchor the response to articulated user intent, while activated dimensions guide targeted expansion over task-relevant aspects that were previously unaddressed.

5 Experiments

We evaluate PROPER to examine whether *calibrated activation of implicit, task-relevant dimensions* improves assistance quality beyond reactive and clarification-based baselines. Our evaluation focuses not only on response correctness, but on whether proactive intervention is timely, appropriate, and aligned with user needs across multiple domains.

Our experiments are structured around four research questions:

RQ1 Does PROPER improve end-to-end task utility across domains?

RQ2 How do DGA, reranking, and RGA individually contribute to performance?

RQ3 Are observed gains driven by calibrated proactivity rather than increased verbosity?

RQ4 Does PROPER remain robust in multi-turn conversational settings?

Unless otherwise specified, DGA and RGA share the same underlying LLM architecture.

5.1 Experimental Setup

Datasets: We evaluate PROPER across three domains that differ in how and when proactive assistance is appropriate. **MedDG** (Liu et al., 2022) involves patient-facing medical queries related to diagnosis, treatment, or symptoms, where proactive behavior must be cautious, uncertainty-aware, and emotionally appropriate. **Code-Contests** (Li et al., 2022) contains programming tasks in which coding assistance. **PWAB** (Cai et al., 2025) is an online Amazon-shopping dataset comprising mulitiple recommendation and web search queries. Together, these datasets stress complementary aspects of calibrated proactivity, ranging from restraint (medical) to guidance (coding) to preference balancing (shopping). Additional preprocessing details are provided in Appendix A.1.

Baselines: Following prior work on proactive agents (Lu et al., 2025), we use LLaMA-3.1-8B-Instruct (Grattafiori et al., 2024) and Qwen-3-8B (Yang et al., 2025) as our primary baselines. These models represent strong, instruction-tuned LLMs of comparable scale and are evaluated both as standalone assistants and as the backbone architectures for PROPER. This design isolates the effect of calibrated dimension generation and response orchestration from raw model capacity. Full fine-tuning configurations and hyperparameters for DGA are provided in Appendix A.2.

Evaluation Protocol: We evaluate responses using an external LLM-based judge (GPT-5), following recent work on holistic evaluation of open-ended assistance. For each input, we generate responses from all compared models and ask GPT-5 to assign a quality score on a 0–5 scale, accompanied by a brief justification. The evaluation rubric emphasizes overall helpfulness, relevance, and the appropriateness of proactive guidance, rather than surface-level correctness alone.

We report the mean quality score across samples (μScore) and the percentage of samples in which a model achieves the highest score (Win%). Unless otherwise specified, evaluations consider complete responses rather than isolated factual accuracy, reflecting our focus on calibrated assistance.

5.2 Results and Discussions

RQ1: End-to-End Task Utility: To evaluate how PROPER improves proactivity among base LLMs, we compare the responses generated by the base LLM (LLAMA-8B or QWEN-8B) with those from PROPER with the same LLM as the base. We also include GPT-4 for reference. Table 1 results clearly show that, on average, PROPER outperforms the base LLM on 84% of entries across all datasets. PROPER achieves substantially higher mean scores and win rates when paired with either LLAMA-8B or QWEN-8B, reflecting the importance of proactively surfacing unarticulated risks, constraints, and preferences that are often omitted from the initial query. In contrast, improvements on Code-Contests are more modest: while PROPER still wins more frequently than base models, gains in mean score are smaller, suggesting diminishing returns from proactive intervention in highly structured tasks with well-defined solution spaces.

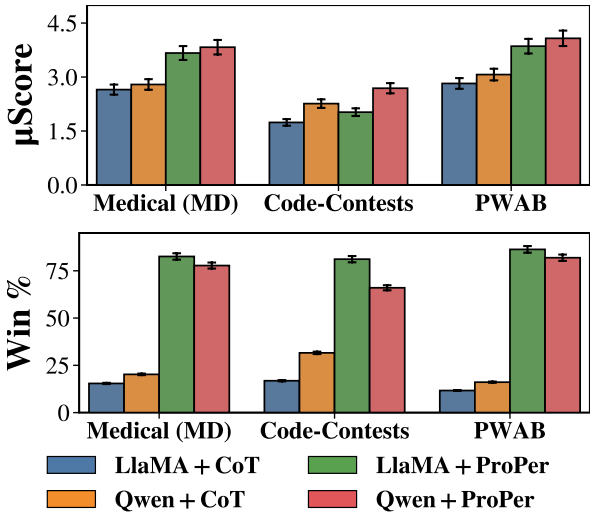

Figure 3: Comparison of PROPER with CoT prompting applied to LLAMA-8B and QWEN-8B across all datasets. PROPER consistently outperforms other models even when CoT prompting enhances baseline LLMs.

In addition, we also compare PROPER with chain-of-thought (CoT) prompting applied to LLAMA-8B and QWEN-8B (refer to Appendix A.5 for COT prompts). Following (Madaan et al., 2023), we construct a chain of prompting that first asks the LLM to generate a response to a user query, then analyze which aspects the response did not sufficiently address, and finally reiterate the process to improve the response. The comparison of CoT models with PROPER, shown in Figure 3, show that even after CoT prompting, which doe improve the base LLM, PROPER still outperforms. In addition, while CoT improves standard fine-tuned models, the gains are marginal.

To complement the quantitative results, we examine a representative query from the Medical (MD) dataset: *"Because I have type 1 diabetes, what is*

(λ_1, λ_2)	Medical (MD)		Code-Contests		PWAB	
	LLaMA+ProPer	Qwen+ProPer	LLaMA+ProPer	Qwen+ProPer	LLaMA+ProPer	Qwen+ProPer
(8.0, 1.0)	4.00	4.15	2.11	2.81	3.96	3.71
(2.0, 0.5)	3.75	4.01	2.12	2.89	4.06	3.91
(0.0, 0.2)	3.70	3.91	2.08	2.79	4.17	3.80

Table 2: Tabular illustration of μScore variations for different (λ_1, λ_2) values controlling missing dimension activation and diversity, respectively. Performance degrades as λ decreases in Medical and PWAB, while Code-Contests remains comparatively stable, reflecting domain-specific sensitivity to proactive calibration.

likely to happen to me if I get the coronavirus?" Below, we report the full responses generated by different models.

> **LLaMA-8B *(Fine-Tuned)*:** *People with diabetes are at increased risk for severe illness and death from COVID-19. It is not a guarantee that you would die from it, but the risk is higher. The virus spreads mainly through droplets of saliva, mucus, and other respiratory secretions, typically released when an infected person coughs, sneezes, or breathes.*
>
> **GPT-4*:*** *If you have type 1 diabetes and get COVID-19, you'll most likely have mild symptoms and recover, especially if you're young. The main issue is that illness can cause higher blood sugars and ketones, so you may need more insulin and closer monitoring. Serious problems are uncommon when diabetes is well managed.*
>
> **LLaMA-8B + PROPER*:*** *Because you have Type 1 diabetes, COVID-19 might be more severe. Your immune system may struggle raising the risk of breathing complications, and can affect blood sugar control. Vaccination helps, avoid crowds, practice hygiene, and follow your doctor's advice. Other conditions like hypertension, kidney, or heart disease increase risk.*

While all three responses are factually plausible, they differ markedly in how implicit, task-relevant dimensions are addressed. The evaluator model (GPT-5) rated PROPER as the highest, noting: "*Provides diabetes-specific risk framing, vaccination context, practical precautions*". The base LLaMA-8B response focuses on general risk and transmission mechanisms and GPT-4 narrows the scope to likely outcomes and metabolic management but avoids broader risk framing or preventive action. In contrast, PROPER integrates multiple latent considerations—disease severity, glycemic control, vaccination, behavioral precautions, and comorbidities—into a single coherent response.

RQ2: Component Contributions.: We analyze the contribution of PROPER's core components through targeted ablations. As shown in Figure 4, removing the Dimension Generating Agent (DGA) results in a substantial performance drop across all datasets, while removing post-hoc dimension reranking (PROPER-RGA) leads to a smaller but consistent degradation. This indicates that explicitly generating task-relevant implicit dimensions is foundational to PROPER's effectiveness: without DGA, the system lacks a structured representation of what is missing from the user query, and downstream response generation becomes less targeted and less calibrated.

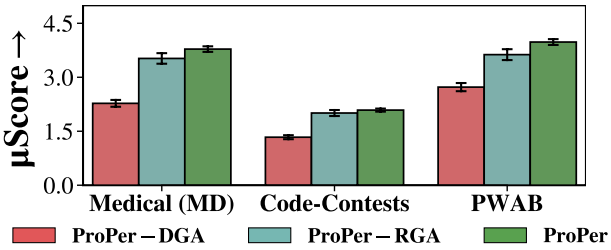

Figure 4: Performance comparison of PROPER and its variants (PROPER-DGA and PROPER-RGA), showing the impact of removing DGA or RGA on overall results. The drop without DGA is more pronounced.

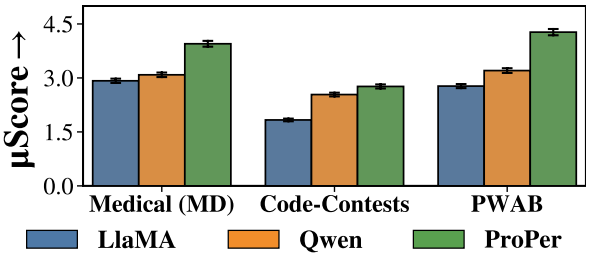

Figure 5: Quality comparison of dimensions generated by DGA in PROPER, LLaMA-8B, and QWEN-8B. PROPER produces the most effective dimensions, with QWEN-8B outperforming LLaMA-8B.

To further understand this effect, Figure 5 compares the quality of dimensions produced by DGA against those generated directly by strong base LLMs. DGA consistently generates higher-quality dimensions than both LLaMA-8B and QWEN-8B, suggesting that the gains observed in the full system stem from learning to surface relevant latent needs rather than from response-generation capac-

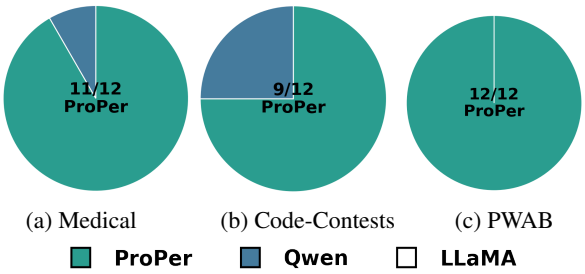

Figure 6: Dominance disks summarizing evaluator preferences across 12 randomly sampled multi-turn conversations per dataset.

ity alone. Taken together, these results reveal a clear division of labor: DGA enables anticipation by identifying meaningful implicit dimensions, while reranking and RGA modulate their influence to prevent over-proactivity.

Proactivity vs. Personalization (RQ3): Table 2 evaluates whether PROPER's gains arise from calibrated proactivity rather than indiscriminate response expansion by varying the calibration regime (λ_1, λ_2) controlling implicit-dimension activation and diversity. In Medical and PWAB, reducing (λ_1, λ_2) consistently lowers response quality for both backbones, indicating that under-activating implicit dimensions, such as safety considerations, latent constraints, and comparison criteria, leads to less helpful responses even when outputs remain concise and on-topic. In contrast, Code-Contests exhibits minimal sensitivity to calibration, reflecting a domain where objectives are tightly specified and additional proactive breadth offers limited benefit beyond producing a correct solution. Across all settings, the shared directionality of these trends for both LLAMA-8B +PROPER and QWEN-8B +PROPER suggests that the observed effects are not driven by model scale, but by the extent to which proactive dimension activation aligns with domain-specific task structure.

On Multi-turn Conversations (RQ4): To assess whether calibrated proactivity remains stable over time, we conduct a small-scale multi-turn evaluation on 12 randomly sampled conversations per domain, comparing full interaction trajectories generated by PROPER against strong base-model baselines. As summarized in Figure 6, PROPER is preferred in the majority of conversations across domains (11/12 in Medical, 9/12 in Code-Contests, and 12/12 in PWAB). Baseline wins occur primarily in narrowly specified tasks where conservative responses suffice. These results suggest that PROPER's advantages extend beyond single-turn settings, maintaining appropriate initiative across turns, particularly in domains where implicit needs and constraints emerge gradually.

6 Conclusion and Discussion

This work reframes proactive assistance as an epistemic calibration problem centered on identifying and addressing *what is missing*. Rather than treating proactivity as response expansion, we view it as the controlled surfacing of latent, task-relevant knowledge gaps, including needs the user may not yet articulate or even recognize. To operationalize this view, we introduce PROPER, a modular framework that elevates explicit and implicit dimensions to first-class representations, allowing proactive behavior to be generated, prioritized, and applied in a principled manner. By reasoning over dimensions instead of raw text alone, we move beyond reactive personalization toward anticipatory assistance.Across three domains, PROPER consistently improves task utility over strong base models as well as CoT variations, with the largest gains appearing in settings where unarticulated risks, constraints, or trade-offs materially shape outcomes. Ablation results further show that explicitly modeling the unknowns is more critical than response-generation capacity alone. Importantly, the benefits persist in multi-turn interactions, where miscalibrated initiative would otherwise compound over time.

Looking forward, this work opens a new space for proactive agents that reason over epistemic states rather than surface text. Treating dimensions as intermediate representations makes it possible to study when gaps should be surfaced, how initiative should be modulated as understanding evolves, and how uncertainty should be acknowledged rather than concealed. This perspective, further, invites future work on adaptive calibration policies, concept-grounded dimensions (Sun et al., 2025), and mechanisms for detecting when the dimension set itself is incomplete. More broadly, it suggests a path toward proactive agents that do not merely assist with stated goals, but actively support sensemaking and exploration by identifying what questions have not yet been asked - *an ability that may prove critical in domains ranging from safety-critical decision making to open-ended scientific inquiry.*

7 Limitations

Evaluation choice and what it enables. Our goal in this paper is to isolate whether *calibrated* activation of implicit dimensions improves assistance quality across domains. To make that comparison feasible at scale and under controlled conditions, we primarily use a strong LLM judge that applies a consistent rubric across many paired responses. This choice trades off some ecological validity for comparability: an LLM judge may reward certain writing styles or forms of reasoning, and it is not a substitute for human satisfaction or downstream task outcomes. We view this as an appropriate first step for studying calibration effects cleanly, and we expect the most valuable next step to be human-centered validation that measures trust, perceived intrusiveness, and long-horizon utility in addition to response quality.

Why dimensions are free-form (for now). We represent implicit dimensions in free-form language rather than grounding them in ontologies, taxonomies, or structured variables. This is deliberate: domain-specific grounding would hard-code priors that blur the causal question we study (does explicit implicit-dimension modeling help, independent of domain knowledge engineering?). The downside is reduced interpretability and weaker guarantees against redundancy or inconsistent phrasing. At the same time, the dimension interface creates a natural bridge to concept modeling: future variants could replace or augment textual dimensions with concept bottlenecks, ontology-aligned slots, or hybrid symbolic–neural representations, making proactivity more controllable and auditable without sacrificing generality.

Calibration as a controlled knob rather than a learned policy. We sweep fixed (λ_1, λ_2) regimes to expose how proactivity and diversity affect utility, instead of learning a policy that adapts these parameters online. We make this choice because fixed regimes yield clearer ablations and more interpretable conclusions about when proactivity helps versus when it saturates. The limitation is that real users likely require context- and user-dependent calibration. A natural next direction is to learn adaptive calibration that conditions on uncertainty, conversation stage, and user feedback, effectively turning (λ_1, λ_2) into a dynamic control policy rather than static hyperparameters.

Multi-turn evidence as a robustness check. Our multi-turn experiment is intentionally small-scale and qualitative. We use it to test a specific failure mode of proactive systems, whether initiative drifts or compounds across turns, rather than to claim comprehensive coverage of long-horizon behavior. Scaling this evaluation to longer trajectories with diverse interaction patterns is important, but it requires different study design and substantially more annotation. We see this as a key direction for advancing proactive agents: measuring not only per-turn helpfulness, but also stability of calibration, recovery from missteps, and user-perceived intrusiveness over time.

Personalization and modality beyond text. Although PROPER reasons over explicit and implicit dimensions, it does not yet maintain persistent user models (e.g., expertise, risk tolerance, or preference for initiative), and it operates purely over text. We adopt this restriction to stay compatible with widely used benchmarks and to keep the study focused on the proactivity mechanism itself. Extending the framework to user-specific proactivity thresholds and to multimodal signals (behavioral traces, structured task state, or environment feedback) is a promising path to make calibration both more personalized and more robust in real deployment settings.

Toward unknown unknowns and epistemic proactivity. Finally, while PROPER surfaces missing task-relevant dimensions, it does not explicitly distinguish between known unknowns (recoverable gaps) and deeper epistemic gaps where the system may not even know what it is missing. We view this paper as a step toward that broader goal: by elevating dimensions into an explicit intermediate representation, we create a scaffold for future work to model epistemic uncertainty, detect when the dimension set itself is incomplete, and reason about when to ask, when to caution, and when to defer.

References

Saptarashmi Bandyopadhyay, Vikas Bahirwani, Lavisha Aggarwal, Bhanu Prakash Reddy Guda, Lin Li, and Andrea Colaco. 2025. Yeti (yet to intervene) proactive interventions by multimodal ai agents in augmented reality tasks. *ArXiv*, abs/2501.09355.

Constantin Brîncoveanu. 2025. Trust me, i'm funny: Humor, personalization, and trust in conversational

agents – a systematic literature review on user engagement, educational adoption, and responsible use. In *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Hongru Cai, Yongqi Li, Wenjie Wang, Fengbin Zhu, Xiaoyu Shen, Wenjie Li, and Tat-Seng Chua. 2025. Large language models empowered personalized web agents. In *Proceedings of the ACM Web Conference 2025*, WWW '25.

Yue Chen, Chen Huang, Yang Deng, and Tat-Seng Chua. 2025. Style: Improving domain transferability of asking clarification questions in large language model powered conversational agents. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yang Deng, Lizi Liao, Liang Chen, Hongru Wang, Wenqiang Lei, and Tat-Seng Chua. 2023. Prompting and evaluating large language models for proactive dialogues: Clarification, target-guided, and non-collaboration. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 10602–10621, Singapore. Association for Computational Linguistics.

Yang Deng, Lizi Liao, Zhonghua Zheng, Grace Hui Yang, and Tat-Seng Chua. 2024a. Towards human-centered proactive conversational agents. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '24, page 807–818, New York, NY, USA. Association for Computing Machinery.

Yang Deng, Yong Zhao, Moxin Li, See-Kiong Ng, and Tat-Seng Chua. 2024b. Don't just say "I don't know"! self-aligning large language models for responding to unknown questions with explanations. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 13652–13673, Miami, Florida, USA. Association for Computational Linguistics.

Wenjie Dong, Sirong Chen, and Yan Yang. 2025. ProTOD: Proactive task-oriented dialogue system based on large language model. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9147–9164, Abu Dhabi, UAE. Association for Computational Linguistics.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, and 1 others. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Meera Hahn, Wenjun Zeng, Nithish Kannen, Rich Galt, Kartikeya Badola, Been Kim, and Zi Wang. 2025. Proactive agents for multi-turn text-to-image generation under uncertainty.

Guangming Huang, Yunfei Long, Cunjin Luo, Jiaxing Shen, and Xia Sun. 2024. Prompting explicit and implicit knowledge for multi-hop question answering based on human reading process. In *International Conference on Language Resources and Evaluation*.

Ann Kerwin. 1993. None too solid: Medical ignorance. *Knowledge*, 15(2):166–185.

Wenqiang Lei, Tat-Seng Chua, Yang Deng, and Wai Lam. 2025. A survey on proactive dialogue systems: Problems, methods, and prospects. *Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, and 7 others. 2022. Competition-level code generation with alphacode. *arXiv preprint arXiv:2203.07814*.

Wenge Liu, Jianheng Tang, Yi Cheng, Wenjie Li, Yefeng Zheng, and Xiaodan Liang. 2022. Meddg: An entity-centric medical consultation dataset for entity-aware medical dialogue generation. In *Natural Language Processing and Chinese Computing: 11th CCF International Conference, NLPCC 2022, Guilin, China, September 24–25, 2022, Proceedings, Part I*, page 447–459, Berlin, Heidelberg. Springer-Verlag.

Xingyu Bruce Liu, Shitao Fang, Weiyan Shi, Chien-Sheng Wu, Takeo Igarashi, and Xiang 'Anthony' Chen. 2024. Proactive conversational agents with inner thoughts. *Proceedings of the 2025 CHI Conference on Human Factors in Computing Systems*.

Yaxi Lu, Shenzhi Yang, Cheng Qian, Guirong Chen, Qinyu Luo, Yesai Wu, Huadong Wang, Xin Cong, Zhong Zhang, Yankai Lin, Weiwen Liu, Yasheng Wang, Zhiyuan Liu, Fangming Liu, and Maosong Sun. 2025. Proactive agent: Shifting LLM agents from reactive responses to active assistance. In *The Thirteenth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, and 1 others. 2023. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36:46534–46594.

Joon Sung Park, Joseph C. O'Brien, Carrie J. Cai, Meredith Ringel Morris, Percy Liang, and Michael S. Bernstein. 2024. Generative agents: Interactive simulacra of human behavior. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

Marco Tulio Ribeiro, Tongshuang Wu, Carlos Guestrin, and Sameer Singh. 2020. Beyond accuracy: Behavioral testing of nlp models with checklist. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics (ACL)*, pages 4902–4912.

Chung-En Sun, Sungjin Lee, Yiyang Zhang, and Danqi Chen. 2025. Concept bottleneck large language models. In *Proceedings of the International Conference on Learning Representations (ICLR)*.

An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, Chujie Zheng, Dayiheng Liu, Fan Zhou, Fei Huang, Feng Hu, Hao Ge, Haoran Wei, Huan Lin, Jialong Tang, and 41 others. 2025. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*.

et al. Zargham. 2022. Understanding circumstances for desirable proactive behaviour of voice assistants. In *Proceedings of the ACM Conference on Research and Development in Information Retrieval (SIGIR)*.

Ceyao Zhang, Kaijie Yang, Siyi Hu, Zihao Wang, Guanghe Li, Yihang Sun, Cheng Zhang, Zhaowei Zhang, Anji Liu, Song-Chun Zhu, Xiaojun Chang, Junge Zhang, Feng Yin, Yitao Liang, and Yaodong Yang. 2024a. Proagent: building proactive cooperative agents with large language models. In *Proceedings of the Thirty-Eighth AAAI Conference on Artificial Intelligence and Thirty-Sixth Conference on Innovative Applications of Artificial Intelligence and Fourteenth Symposium on Educational Advances in Artificial Intelligence*, AAAI'24/IAAI'24/EAAI'24. AAAI Press.

Xuan Zhang, Yang Deng, Zifeng Ren, See-Kiong Ng, and Tat-Seng Chua. 2024b. Ask-before-plan: Proactive language agents for real-world planning. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10836–10863, Miami, Florida, USA. Association for Computational Linguistics.

Yaowei Zheng and 1 others. 2024. Llama-factory: Unified efficient fine-tuning of large language models. `https://github.com/hiyouga/LLaMA-Factory`.

A Appendix

A.1 Fine-tuning Data Construction Pipeline

To enable models to surface missing knowledge gaps, we construct fine-tuning data at the level of *interaction dimensions* that goes beyond raw text and elicits major aspects and considerations unique to an interaction that guide the successful completion of a certain task. Dimension-level supervision abstracts away surface form variation and allows the DGA to learn patterns of well-articulated user needs and task-relevant aspects considered to solve that task. Below we describe how such dimension-level supervision is defined, generated, and used to fine-tune the DGA.

A.1.1 Dimension Annotation Schema

This subsection defines the annotation schema used to construct supervision for training the Dimension Generating Agent (DGA). Importantly, our DGA is trained only on *observed* dimensions i.e., dimensions that are explicitly present in the user input and/or explicitly addressed in the assistant response, to learn what aspects are salient for a given type of task. At inference time, we then prompt the DGA to *elicit implicit dimensions* by proposing which salient dimensions are likely missing from the current interaction, conditioned on the user state.

We represent each interaction using a shared universe of dimensions, where a dimension is a short descriptor of a task-relevant aspect that may influence response quality (e.g., "input validation," "safety constraints," "preference trade-offs"). For a given interaction state, we annotate two explicit sets. *User-explicit dimensions* capture aspects explicitly stated or clearly implied by the user query or interaction history. *System-explicit dimensions* capture aspects addressed by a baseline (or reference) assistant response. These explicit annotations provide the DGA with dimension-level signals of what matters for similar tasks and how such aspects tend to be expressed or addressed.

Implicit dimensions are not directly supervised during fine-tuning. Instead, they are elicited at inference by conditioning the DGA on the current interaction and prompting it to propose task-relevant dimensions that are likely missing from the user input and baseline response, leveraging the dimension vocabulary and regularities learned from explicit supervision.

A.1.2 Fine-tuning Data Generation Procedure

While the concrete data generation procedures differ across datasets and domains, all fine-tuning data for the Dimension Generating Agent (DGA) follows a shared high-level pipeline. This pipeline defines how raw interactions are transformed into structured, domain-independent supervision at the dimension level.

At a high level, the process begins with a user query and the corresponding assistant response observed in an existing interaction. Given this interaction context, we use **GPT-5** and apply a curated elicitation prompt designed to surface task-relevant dimensions expressed on the user side and dimensions addressed by the assistant response. These prompts are tailored to the characteristics of each dataset and domain, and are described in detail in subsequent subsections.The elicitation prompt produces a structured output in the form of a JSON object, containing the inferred user-explicit dimensions and system-explicit dimensions for the interaction. These dimension annotations are then stored and aggregated to form the fine-tuning dataset. Each training example thus consists of an interaction context paired with a structured representation of explicit dimensions, abstracting away surface-level textual variation. In the following subsections, we describe the concrete instantiations of this pipeline for each dataset, including the specific prompts used and any domain-specific heuristics or filtering applied during data construction.

(i) **CodeContest Annotation Pipeline:** We first instantiate the dimension annotation pipeline on the CodeContest dataset, which consists of competitive programming problems and corresponding reference solutions. Since CodeContest does not natively contain user queries, we construct realistic user–assistant interaction contexts by eliciting user-style queries from problem descriptions. This enables downstream dimension annotation while preserving the characteristics of real coding assistance scenarios.

Initial Data Preprocessing. We begin with raw CodeContest data dumps containing programming problems and associated metadata. All available splits are concatenated into a unified dataset comprising 13,610 problem instances. Each instance includes attributes such as problem description, difficulty, test cases, and multiple reference solutions.

For the scope of our work, we retain only the

following fields:

- **Problem Description**: Natural language description of the programming task.
- **Difficulty**: Integer-coded difficulty level.
- **Solutions**: A list of reference code solutions with language annotations.

We further restrict the dataset to **Python 3** solutions to ensure consistency and ease of downstream processing, as Python 3 is the most prevalent and actively maintained language in the dataset.

Difficulty-Aware Dataset Construction. After filtering, the dataset spans 18 distinct difficulty levels. To evaluate both in-distribution learning and generalization, we partition the dataset into *warm* and *cold-start* subsets. The warm set includes up to 15 problems per difficulty level (or all problems when fewer are available), and is used for training and validation. The remaining problems constitute the cold-start set, enabling evaluation on entirely unseen problem instances.

For each problem, we retain at most 50 Python 3 solutions. Each entry is flattened into the following structure:

$$\langle \text{Problem Description}, \text{ Difficulty}, \text{ Python 3 Solutions} \rangle$$

The warm subset is randomly split into training and test sets using a 70/30 ratio.

Query Elicitation via Prompting. Because CodeContest does not include user-issued queries, we use a large language model (GPT-5) to elicit realistic user prompts conditioned on the problem description. The goal is not to generate solutions, but to simulate how users with different levels of programming experience might articulate their intent when interacting with a coding assistant.

Specifically, for each problem, we generate three query variants corresponding to increasing levels of user expertise. This design allows the annotation pipeline to capture variation in how explicit or underspecified user intent may be expressed.

Prompt Templates. We use carefully designed prompt templates to elicit these query variants. Each prompt conditions on the problem description and instructs the model to generate a single realistic user query matching the desired expertise level. The prompts are shown below for completeness.

Query Level	Description
Level 1	Vague prompts with minimal technical detail, reflecting beginner-level understanding.
Level 2	Moderately structured prompts with partial technical awareness and some ambiguity.
Level 3	Precise, code-focused prompts reflecting strong understanding of algorithms or data structures.

Table 3: Query elicitation levels used to simulate user expertise in CodeContest.

Level 1 (Beginner) Query Prompt

```
You have been given a detailed
description of a coding problem and its
solution. Your task is to generate one
realistic user prompt that a beginner
programmer might naturally ask GitHub
Copilot to generate similar code.

Requirements:
- Natural phrasing that sounds like a
real beginner developer.
- Very vague understanding of the
problem.
- General intent without clear technical
details.
- Concise and Copilot-friendly.

Input: <problem description>

Output: A single vague, beginner-level
user prompt a programmer would use.
```

Level 2 (Intermediate) Query Prompt

```
You have been given a detailed
description of a coding problem and its
solution. Your task is to generate
one realistic user prompt that an
intermediate programmer might naturally
ask GitHub Copilot to generate similar
code.

Requirements:
- Natural phrasing reflecting partial
understanding.
- References to basic techniques (e.g.,
loops, conditionals, functions).
- Some structure, but remaining ambiguity
in solution design.
- Clear and concise.

Input: <problem description>

Output: A single moderately clear user
prompt showing partial understanding of
the task.
```

Level 3 (Advanced) Query Prompt

```
You have been given a detailed
description of a coding problem and its
solution. Your task is to generate
one realistic user prompt that an
experienced programmer might naturally
ask GitHub Copilot to generate similar
code.

Requirements:
- Precise, code-focused phrasing.
- Explicit references to relevant
```

Query Level	Example Excerpt
Beginner	"How can I write a program to find the minimal number of groups where the words in each group denote the same name given a list of words where some letters can be replaced by others and still be considered the same name?"
Intermediate	"Given a string s of length at most 1000 composed of lowercase letters, find whether it contains heidi as a subsequence, return YES or NO. Constraints: s is nonempty. Goal is to determine if s contains heidi as a subsequence without looking for heidi directly. Tricky part: Handle the case where heidi is a substring of s. Should I implement a function to check if s contains heidi as a subsequence or directly compare s and heidi?"
Advanced	"How to efficiently group words into equivalence classes under certain transformations with respect to the Latin alphabet, allowing for u to oo and h to kh replacements, while minimizing the number of groups, considering a large input size up to 400 words and time complexity of O(n*m) or better, where n is the number of words and m is the maximum word length, ensuring correct handling of edge cases such as single-letter words and empty strings?"

Table 4: Representative query excerpts illustrating variation in specificity and technical detail across elicited expertise levels.

```
data structures, algorithms, or
optimizations.
- Highly targeted intent guiding the
solution approach.
- Concise and technically specific.

Input: <problem description>

Output: A single clear and specific user
prompt capturing all relevant solution
details.
```

Structured Annotation Output. The generated queries, along with the original problem descriptions and reference solutions, are passed to dataset-specific elicitation prompts (described in subsequent subsections) that infer user-explicit and system-explicit dimensions. The resulting annotations are stored as structured JSON objects and form the basis of the fine-tuning dataset used to train the Dimension Generating Agent.

Illustrative Query Variants. The elicited queries differ substantially in how explicitly users articulate goals, constraints, and uncertainty. Table 4 shows representative examples from the three query levels used in CodeContest. These variations allow the annotation pipeline to capture differences in how task intent and missing information are expressed across user expertise levels.

Dimension Extraction via Prompted Annotation. Given a generated query and a reference solution, we next extract structured interaction dimensions using a curated annotation prompt. This prompt is designed to infer *explicit user-side dimensions* and *system-side dimensions* grounded in the solution code, without speculative reasoning.

The prompt instructs the model to reason along interpretable axes such as user intent, uncertainty, task structure, and constraints on the user side, and algorithmic patterns, complexity, and robustness on the system side. Importantly, the prompt requires all extracted dimensions to be justified using evidence from the input text, producing structured JSON outputs suitable for downstream processing.

This prompt design serves two purposes. First, it exposes the Dimension Generating Agent to consistent, semantically meaningful signals about what aspects are explicitly present in real coding interactions, despite wide variation in surface phrasing. Second, by anchoring extraction to both the query and the solution, it allows the model to internalize which technical dimensions tend to matter for different classes of programming tasks. These learned regularities later enable the DGA to elicit plausible *implicit* dimensions at inference time when such aspects are missing from a new interaction.

Annotation Prompt. For completeness, we show the annotation prompt used to extract dimensions below.

```
You are an expert annotator for
programming tasks. Extract latent
dimensions from:
(1) User query, (2) Solution code.

Objectives:
- Infer concise, latent aspects grounded
in the text.
- Avoid speculation beyond weak,
text-supported inference.

User-side axes:
goal intent, user state, expertise
level, uncertainty indicators,
constraint awareness, task structure,
missing information.

System-side axes:
algorithm pattern, data structures,
time and space complexity,
optimization decisions, robustness,
constraint handling.

Output format (JSON only):
{
"user_aspects": [
{"name": str, "value": str,
"justification": str}
```

Speaker	Utterance
Patient	*"I have had cold symptoms for over a week and a low-grade fever last week. For the past two days I have been feeling dizzy. Should I contact my doctor?"*
Doctor	*"These symptoms alone are not enough to classify you as a COVID-19 suspect. However, if you have had contact with a confirmed case, experience persistent cough or shortness of breath, or have chronic conditions such as diabetes or immune suppression, you should seek medical attention and follow protective measures."*

Table 5: Example patient–doctor interaction from the MD dataset.

```
],
"solution_aspects": [
{"name": str, "value": str,
"justification": str}
]
}
Justifications must cite evidence from
the input or be null if implicit.
```

(ii) **Medical Diagnosis (MD) Annotation Pipeline:** We next instantiate our annotation pipeline on the Medical Diagnosis (MD) dataset, which consists of 280 clinical question–answer interactions between patients and medical professionals. We also apply our annotation pipeline to the Medical Diagnosis (MD) dataset, which consists of real-world clinical question–answer interactions between patients and medical professionals. Unlike CodeContest, this dataset natively contains user queries and expert responses, making it directly suitable for dimension-level annotation without additional preprocessing or query elicitation.

The MD dataset captures medically grounded interactions that include symptom descriptions, clinical uncertainty, risk considerations, and professional guidance. As a result, we directly operate on the original patient utterances and doctor responses. Additional details about the dataset are available at the public release link.[1]

Illustrative Interaction Example. Table 5 shows a representative interaction from the MD dataset. Such examples highlight how patient queries often combine explicit symptoms with uncertainty and concern, while doctor responses balance reassurance, risk assessment, and guidance.

[1] https://drive.google.com/drive/folders/11sglwm6-cY7gjeqlZaMxL_MDKDMLdhym

Dimension Extraction Prompt. For each interaction, we apply a curated annotation prompt that extracts explicit and latent dimensions from both the patient query and the doctor response. The prompt emphasizes medically meaningful distinctions, including symptom patterns, risk indicators, and uncertainty handling, while explicitly discouraging speculative inference. The resulting structured annotations are used as supervision for training the Dimension Generating Agent.

```
You are an expert annotator analyzing
clinical question-answer interactions.
Extract both explicit and latent
dimensions from:
(1) User medical query, (2) Doctor
response.

Objectives:
- Infer concise, latent, and
non-trivially stated aspects.
- Avoid speculation beyond weak,
text-grounded inference.
- Emphasize medically relevant,
domain-specific distinctions.

User-side axes:
clinical goal intent, explicit symptoms
and history, latent symptom patterns,
disease-specific indicators,
multisystem interactions, risk or
red-flag indicators,
constraints, missing information,
emotional state, task structure.

System-side axes:
medical reasoning patterns, diagnostic
hypotheses, treatment guidance,
risk assessment, uncertainty handling,
reassurance strategies, guideline
alignment.

Output format (JSON only):
{
"user_aspects": [
{"name": str, "value": str,
"justification": str}
],
"solution_aspects": [
{"name": str, "value": str,
"justification": str}
]
}

Justifications must cite evidence from
the input or be null if implicit.
```

(iii) **PersonalWAB Dataset**

We further instantiate our annotation pipeline on the PersonalWAB dataset, which captures personalized shopping recommendation interactions grounded in user personas, historical behavior, and

product metadata. Unlike CodeContest and MD, PersonalWAB explicitly models long-term user characteristics, making it well-suited for studying how personalization and proactivity interact in recommendation settings.

The dataset consists of four primary components: user profiles, user shopping history, user-issued instructions (queries), and a large catalog of products spanning multiple categories. Each interaction links a user persona and query to a recommended product, providing a rich context for extracting both preference-driven and constraint-driven dimensions. We follow the dataset's standard preprocessing pipeline, which includes consolidating user instructions, aligning them with corresponding user profiles, and filtering profiles to ensure balanced coverage across demographic attributes and occupations. All remaining profiles outside the curated subset are retained as a cold-start set for out-of-distribution evaluation.

Illustrative Interaction Structure. Each annotated instance in PersonalWAB (Table 6) is defined by three inputs: (i) a user persona capturing long-term preferences and traits, (ii) a user query expressing a shopping intent, and (iii) a recommended product with structured attributes such as category, price, and features. This structure allows the annotation process to distinguish between stable user preferences, situational needs, and product-side affordances.

Dimension Extraction Prompt. To extract interaction dimensions, we apply a curated annotation prompt that jointly reasons over the user persona, query, and recommended product. The prompt is designed to surface explicit preference signals as well as latent needs and constraints, while avoiding speculative inference beyond what is weakly supported by the input text. On the system side, extraction is restricted to properties of the recommended product itself.

```
You are an expert annotator analyzing
user-system interactions in shopping
recommendation tasks.
Extract all the explicitly stated and
latent dimensions from:
(1) User persona, (2) User query, (3)
Recommended product.

Objectives:
- Infer concise, latent, and
non-trivially stated aspects.
- Avoid speculation beyond weak,
text-grounded inference.
```

Component	Content
User Persona	Gender: Male; Age: 35–44; Occupation: Technician/Engineer; Price Sensitivity: Medium; Interaction Complexity: Medium; Shopping Interests: Electronics, Home Improvement, Health & Fitness; Brand Preferences: Apple, Sony, Under Armour, Duracell; Tone: Practical, straightforward, enthusiastic; Focus Aspects: Average rating, number of ratings, quality, durability
User Query	"Hey there! I'm looking for some durable athletic accessories or gear. Any recommendations for quality products with solid ratings? Eager to explore some options!"
Recommended Product	Title: Cramer Tuf-Skin Taping Base for Athletic Tape; Category: Athletic Tapes & Wraps; Store: Cramer; Average Rating: 4.6; Review Highlight: Improves tape durability, water-resistant, easy removal

Table 6: Example annotated interaction from the PersonalWAB dataset, illustrating the persona–query–product structure used for dimension extraction.

```
User-side axes:
goal intent, preference profile,
need-state signal, constraint
indicators,
missing-information signals, task
structure, contextual signals from
persona.

System-side axes:
product features, suitability signals,
attribute alignment, price signals,
brand alignment, recommendation
rationale, risk indicators.

Output format (JSON only):
{
"user_aspects": [
{"name": str, "value": str,
"justification": str}
],
"solution_aspects": [
{"name": str, "value": str,
"justification": str}
]
}

Justifications must cite evidence from
the input or be null if implicit.
```

A.2 Training Details for the Dimension Generating Agent

We fine-tune the Dimension Generating Agent (DGA) using the LLaMA-Factory framework (Zheng et al., 2024), which provides a unified interface for supervised fine-tuning

of large language models with parameter-efficient adaptations. We experiment with two instruction-tuned backbone models: Meta-Llama-3-8B-Instruct (Grattafiori et al., 2024) and Qwen-8B (Yang et al., 2025).

Fine-tuning Setup. The DGA is trained using supervised fine-tuning (SFT) with Low-Rank Adaptation (LoRA). We apply LoRA to all transformer layers with rank 8, allowing the model to adapt to dimension-level supervision while keeping the base model weights frozen. This design balances adaptation capacity and training efficiency.

Data and Input Formatting. Training is performed on the dimension-annotated datasets described in Appendix A, using instruction-style prompts formatted according to the target model's native template (e.g., LLaMA-3). Inputs are truncated to a maximum context length of 3248 tokens. Data preprocessing is parallelized across 16 workers to support scalable training.

Optimization and Training Configuration. We train for 7 epochs using a cosine learning rate schedule with a warmup ratio of 0.1. The learning rate is set to 1×10^{-4}, with an effective batch size of 8 achieved via gradient accumulation. Training is conducted in bfloat16 precision. Models are evaluated every 500 steps on held-out validation splits to monitor learning dynamics.

Inference Use. At inference time, the fine-tuned DGA is prompted to infer task-relevant dimensions for unseen interactions. Importantly, implicit dimensions are not directly supervised during training but are elicited at inference by conditioning on the learned dimension vocabulary and patterns captured through explicit supervision.

A.3 Prompt Specifications

This section documents the prompt templates used across all components of our framework. We provide the full prompts for the Dimension Generating Agent (DGA) and the Response Generating Agent (RGA), including their domain-specific instantiations. These prompts encode the calibration principles described in the main paper and are listed here verbatim to support reproducibility and transparency.

A.3.1 Dimension Generating Agent Prompts

Our DGA prompting strategy follows an expert-teacher analogy: a strong teacher does not merely echo what a student said, but uses prior experience with similar problems to identify what is *structurally missing* in the student's formulation. Concretely, the prompts ask the model to (i) extract what the user has made explicit, and (ii) surface non-redundant, task-relevant dimensions that are commonly required for successful resolution of similar tasks but are not currently stated. To prevent overreach, the prompts explicitly prohibit restating the user's content and prohibit providing solutions or advice, ensuring the output is a clean dimension inventory rather than a response. Across domains, the prompts enforce a strict JSON interface, enabling direct use as supervision and as a controllable intermediate representation in downstream components.

We provide the exact prompts used for CodeContest, PersonalWAB, and Medical Diagnosis below.

Code Contests

```
You are an expert coding problem analyst.
Your task is to analyze the user's coding
query, understand the underlying goal,
and identify both the dimensions they
have already expressed and the important
dimensions they have not yet mentioned.

Think carefully about how coding
tasks with similar goals are typically
specified and solved. More complete
or successful formulations often depend
on additional considerations that the
current user has not made explicit.
Your job is to surface these missing
dimensions, not to solve the problem.

You must not repeat or restate any
explicit information. You must not
provide solutions, advice, code, or
guidance. Focus only on identifying the
conceptual structure of the problem.

Your output contains two types of
dimensions:
————————————————
explicit_dimensions
————————————————
These are problem-solving dimensions
clearly grounded in the user query (e.g.,
stated goals, constraints, assumptions,
or task structure).

Each explicit dimension must include:
- name: concise conceptual label
- value: information directly stated or
implied in the query
- justification: why this dimension is
relevant to understanding or solving
tasks of this type

Extract all explicit dimensions that
meaningfully appear in the query.
————————————————
missed_dimensions
————————————————
These are important considerations
```

that the user has not mentioned but that typically matter when solving similar coding tasks. They should be conceptually diverse and non-redundant.

A missed dimension must:
- not appear in the query,
- not be safely inferable from explicit content,
- not be a generic personal detail (e.g., deadline, experience),
- represent a factor that affects correctness, robustness, efficiency, feasibility, resource use, or clarity.

Each missed dimension must include:
- name: conceptual label (e.g., edge-case category, constraint type)
- value: a plausible concrete instance or relevant detail
- justification: why this dimension matters and how its absence leaves the problem incomplete

You are encouraged to surface out-of-the-box dimensions that a typical user may never think to articulate. These should reveal deeper structural requirements, hidden dependencies, contextual contingencies, or evaluative axes that matter in realistic recommendation scenarios.
Extract as many missed dimensions as possible.

OUTPUT FORMAT (STRICT)

Return ONLY:

```
===START_JSON===
{
"explicit_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"},
...
],
"missed_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"},
...
]
}
===END_JSON===
```

Do not output anything outside the JSON markers.

INPUT

User_persona: <INSERT_PERSONA>

User_query: <INSERT_USER_QUERY>

Generate the JSON now.

PersonalWAB

You are an expert annotator analyzing user's query in personalized shopping and recommendation tasks. Your goal is to examine the user's query, infer their underlying intent, and identify both the dimensions they have explicitly expressed and the important dimensions they have not yet mentioned.

Think carefully about how shopping or preference-seeking tasks are typically formulated. More complete or successful recommendations often depend on additional considerations that the current user has not explicitly articulated. Your role is to surface these missing dimensions, not to solve the user's problem or recommend products.

You must not repeat or restate explicit information.
You must not generate suggestions, advice, or recommendations.
Focus only on identifying the conceptual structure of the user's query.

Your output MUST be valid JSON and nothing else.
You MUST start the JSON with: ===START_JSON===
You MUST end the JSON with: ===END_JSON===

No extra text, no explanations, no markdown, no trailing characters of any kind.

Your output contains two types of dimensions:

explicit_dimensions

These are dimensions clearly grounded in the user query (such as stated needs, constraints, preferences, context, or user motivations).

Each explicit dimension must include:
- name: concise conceptual label
- value: information directly stated or implied in the query
- justification: why this dimension is relevant for understanding shopping or preference-oriented tasks

Extract as many explicit dimensions as possible that meaningfully appear in the query.

missed_dimensions

These are important considerations not mentioned by the user but crucial for solving similar shopping or preference tasks. They should be conceptually diverse and non-redundant.

A missed dimension must:
- not appear in the query,
- not be safely inferable from the

explicit content,
- not be a generic personal detail (e.g., experience level),
- represent a factor that affects relevance, usefulness, ranking, personalization, feasibility, constraints, or decision quality.

Each missed dimension must include:
- name: conceptual label (e.g., budget constraints, lifecycle needs,product compatibility, preference specificity, situational factors)
- value: a plausible concrete instance or detail
- justification: why this dimension matters and how its absence leaves the preference or shopping need underspecified

Extract atleast 10 meaningful missed dimensions.

OUTPUT FORMAT (STRICT)

Return ONLY:

```
===START_JSON===
{
"explicit_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"}
],
"missed_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"}
]
}
===END_JSON===
```

Do not output anything outside the JSON markers.

INPUT

user_query: {user_query}

Generate the JSON now.

Medical Diagnosis

You are an expert clinical reasoning analyst. Your task is to study the patient_query, understand the underlying health concern, and identify both the dimensions the patient has already expressed and the important clinical considerations they have not yet mentioned.

Think carefully about how clinicians typically interpret and structure similar patient presentations. More complete or successful formulations of medical concerns often depend on additional contextual, symptom-based, or risk-related factors that the patient has not stated. Your job is to surface these missing dimensions, not to diagnose or recommend treatment.

You must not repeat or restate any explicit information. You must not provide medical advice. Focus only on exposing the underlying problem-structure: what is specified, and what remains unspecified.

1. explicit_dimensions

These are aspects clearly grounded in the patient's text, such as:
- reported symptoms,
- subjective interpretations of severity,
- concerns or questions,
- contextual details (timeline, triggers),
- emotional or informational states.

Each explicit dimension must include:
- name: concise clinical or contextual label
- value: the value directly stated or clearly implied in the patient's text
- justification: brief explanation of why this dimension is relevant for understanding similar medical presentations

Extract all meaningful explicit dimensions. Do not invent details.

2. missed_dimensions

These are important clinical considerations the patient has not mentioned but that clinicians typically explore when evaluating similar symptoms.
They should be conceptually diverse and non-redundant.

A missed dimension must:
- NOT appear in the patient's text,
- NOT be safely inferable,
- NOT be a generic personal detail unrelated to the complaint,
- reflect a clinically relevant factor affecting interpretation of symptoms, risk assessment, differential considerations, or need for follow-up.

Useful categories often include: - missing symptom qualifiers (duration, progression, distribution),
- unreported associated symptoms or red flags,
- gaps in exposure or risk factors,
- missing information about onset, triggers, or patterns, - relevant medical history elements,
- unasked clarifying questions clinicians typically use to assess

```
severity.

Each missed dimension must include:
- name: concise conceptual label (e.g., "Associated Symptoms", "Exposure History")
- value: a plausible clinically relevant instance (e.g., "fever", "recent contact","sudden vs gradual onset", "immune status")
- justification: why this dimension and value matter for interpreting the presentation and how their absence limits understanding

Produce a diverse, non-overlapping set of missed dimensions.
──────────────────
OUTPUT FORMAT (STRICT)
──────────────────
Return ONLY:

===START_JSON===
{
"explicit_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"},
...
],
"missed_dimensions": [
{"name": "string", "value": "string",
"justification":
"string"},
...
]
}
===END_JSON===

Do not output anything outside the JSON markers.
──────────────────
INPUT
──────────────────
patient_query: {patient_query}

Generate the JSON now.
```

A.4 Response Generating Agent Prompts

While the DGA surfaces *what is missing*, the Response Generating Agent (RGA) governs *how and when* those missing dimensions should be addressed. We design RGA prompts around an expert-editor analogy: an expert does not rewrite an answer from scratch, but carefully revises an existing response by filling gaps, correcting omissions, and calibrating tone and scope to the user's intent. The prompts therefore frame response generation as a *selective update* to a baseline response, explicitly conditioning on (i) the original user query, (ii) the existing system response, and (iii) the activated dimensions identified by the reranker.

Across domains, the prompts enforce three core principles. First, **initiative calibration**: the model must infer how proactive to be from the query itself rather than uniformly expanding the response. Second, **constraint-preserving repair**: the original response's structure and intent are preserved unless clarity, safety, or personalization requires otherwise. Third, **controlled uncertainty handling**: implicit dimensions are incorporated only when appropriate, with at most one clarifying question allowed to avoid unwarranted assumptions. Together, these constraints operationalize proactivity as a controlled, context-sensitive decision rather than verbosity. We provide the exact RGA prompts used for CodeContest, Medical Diagnosis, and PersonalWAB below.

Code Contest.

```
You are acting as a proactive coding assistant that reflects on an earlier response from a coding assistant.

Task:
Given a user query, an existing coding assistant response, and a set of missing aspects, generate an updated response that better supports the user by addressing relevant uncertainty and unmet informational needs.

Guiding principle:
The appropriate level of guidance depends on the patient's question.
You must infer how proactive to be from the query itself.

Initiative calibration:
If the query suggests confusion, failure, or blocked progress, expand the response to proactively cover important missing aspects.
If the query is narrow or explicitly limited, keep the response focused and minimally expanded.
If the query is narrow or clearly constrained, keep the response focused and minimally expanded.
If the query reflects frustration or ambiguity, broaden gently while avoiding assumptions about user skill or intent.

Handling missing aspects:
Explicit missing aspects should generally be addressed.
Implicit missing aspects represent possible knowledge gaps and should be included only when appropriate for this query.
If addressing an implicit aspect requires user-specific details, ask at most ONE clarifying question rather than speculating.
```

Response constraints:
Maintain a supportive, non-authoritative tone.
Avoid condescension, over-specification, or unnecessary implementation detail.
Preserve the original response's structure and intent unless safety or clarity requires change.
Prefer concise additions over complete rewrites.

Output constraints:
You MUST start with: ===START===
You MUST end with: ===END===
Output ONLY the final clinical assistant response.
No explanations,
no markdown, no trailing characters.

User query:
<user_query>

Existing coding assistant response:
<system_output>

Missing aspects (explicit missing + implicit knowledge gaps):
<missed_aspects>

Medical Diagnosis

You are acting as a clinical assistant that reflects on an existing response to a patient.

Task:
Given a patient query, an existing clinical assistant response, and a set of missing aspects, generate an updated response that better supports the patient by addressing relevant uncertainty and unmet informational needs.

Guiding principle:
The appropriate level of guidance depends on the patient's question.
You must infer how proactive to be from the query itself.

Initiative calibration:
If the query suggests urgency, safety risk, or possible harm, expand the response to proactively cover important missing aspects.
If the query is narrow or explicitly limited, keep the response focused and minimally expanded.
If the query signals uncertainty, curiosity, or lack of clarity, selectively surface helpful missing context.
If the query signals emotional strain or ambiguity, broaden gently while avoiding assumptions or diagnoses.

Handling missing aspects:
Explicit missing aspects should generally be addressed.
Implicit missing aspects represent possible knowledge gaps and should be included only when appropriate for this query.
If addressing an implicit aspect requires patient-specific details, ask at most ONE clarifying question rather than speculating.

Response constraints:
Maintain a supportive, non-authoritative clinical tone.
Avoid diagnosis, definitive medical claims, or alarming language.
Preserve the original response's structure and intent unless safety or clarity requires change.
Prefer concise additions over complete rewritesnd complete rewrites.

Output constraints:
You MUST start with: ===START===
You MUST end with: ===END===
Output ONLY the final clinical assistant response.
No explanations,
no markdown, no trailing characters.

Patient query:
<user_query>
Existing clinical assistant response:
<system_output>
Missing aspects (explicit missing + implicit knowledge gaps):
<missed_aspects>

PersonalWAB

You are acting as a proactive recommendation assistant that reflects on an earlier response from a recommendation assistant.

Task:
Given a user query, an existing recommendation assistant response, and a set of missing aspects, generate an updated response that better supports the user by addressing relevant uncertainty and unmet informational needs.

Guiding principle:
The appropriate level of guidance depends on the user's question.
You must infer how proactive to be from the query itself.

Initiative calibration:
If the query suggests indecision, dissatisfaction, or conflicting preferences, expand the response to proactively cover important missing aspects.
If the query is clearly scoped or asks for a specific type of suggestion, keep the response focused and minimally expanded.
If the query signals curiosity, vague goals, or exploratory intent, selectively surface relevant tradeoffs

```
or options.
If the query reflects overwhelm or uncertainty, broaden gently while avoiding assumptions about user preferences or constraints.

Handling missing aspects:
Explicit missing aspects should generally be addressed.
Implicit missing aspects represent possible preference gaps or context needs and should be included only when appropriate for this query.
If addressing an implicit aspect requires user-specific details, ask at most ONE clarifying question rather than speculating.

Response constraints:
Maintain a helpful, user-centered tone.
Avoid prescriptive, one-size-fits-all suggestions or rigid criteria.
Preserve the original response's structure and intent unless personalization or clarity requires change.
Prefer concise additions over complete rewrites.

Output constraints:
You MUST start with: ===START===
You MUST end with: ===END===
Output ONLY the final clinical assistant response.
No explanations,
no markdown, no trailing characters.

User query:
<user_query>

Existing recommendation assistant response:
<system_output>

Missing aspects (explicit missing + implicit knowledge gaps):
<missed_aspects>
```

A.4.1 Evaluation Prompt

Evaluating calibrated proactivity requires reasoning beyond surface-level correctness or fluency. The evaluator must infer user intent, identify implicit risks or missing considerations, and judge whether the assistant exercised the *appropriate amount of initiative* for the inferred domain. To this end, we use GPT-5 as an evaluation model due to its strong performance in instruction following, domain inference, and nuanced judgment across heterogeneous tasks. Importantly, GPT-5 is not used to generate responses in our framework; rather, it serves as a consistent and high-capacity evaluator to assess response quality under a unified rubric.

The evaluation prompt is designed to avoid common pitfalls in pairwise comparison. Responses are evaluated *independently*, without assuming which system is baseline or proactive, and without directly comparing responses against each other. This design prevents preference leakage and encourages absolute judgments grounded in the user's needs rather than relative differences. Proactivity is explicitly framed as *calibration*, not verbosity: evaluators are instructed to penalize both overreach (unwarranted assumptions, excessive warnings) and underreach (failure to address salient risks or gaps).

Each response is scored holistically on a 0–5 scale, considering intent alignment, anticipation of implicit needs, appropriateness of initiative, personalization, clarity, and missed opportunities. The requirement for concise, single-line justifications enforces disciplined reasoning and reduces evaluator drift, while the strict JSON output format ensures reliable downstream aggregation and analysis.

Evaluation Prompt

```
You are an expert evaluator of proactive and personalized AI assistants.
Your task is to independently evaluate two assistant responses to a user query with respect to calibrated proactivity and personalization.
You will be given:
- A user query
- Response A
- Response B

You must infer:
- The interaction domain (e.g., medical, shopping, coding, or other)
- The user's explicit needs
- Any important implicit considerations (e.g., risks, uncertainties, missing context, next steps)

Evaluation Principles:
- Evaluate each response independently. Do NOT compare them directly.
- Do NOT assume which response is baseline or proactive.
- Proactivity is not verbosity; it is taking the right amount of initiative given the user's intent and inferred domain.
- Penalize overreach (irrelevant warnings, excessive assumptions, unnecessary steps).
- Penalize underreach (ignoring important needs, risks, or uncertainties).

Score each response holistically on a 0-5 scale considering:
1. How well it addresses the user's stated intent
2. How well it anticipates and handles
```

```
unspoken but relevant needs
3. Appropriateness of initiative for the inferred domain
4. Personalization and contextual relevance
5. Clarity, tone, and overall usefulness
6. Missed opportunities or unnecessary intervention

Scoring Scale:
0 - Very poor
1 - Poor
2 - Weak
3 - Moderate
4 - Good
5 - Excellent

Output Requirements:
- Output MUST be valid JSON ONLY.
- Do NOT include explanations outside the JSON.
- Do NOT add extra fields.
- Justifications must be a single concise line each.

Respond ONLY in the following format:

{
"response_A_score": <integer 0-5>,
"response_B_score": <integer 0-5>,
"response_A_justification": "<single concise line>",
"response_B_justification": "<single concise line>"
}

User Query:
<INSERT USER QUERY HERE>
Response A:
<INSERT RESPONSE A HERE>
Response B:
<INSERT RESPONSE B HERE>
```

A.5 Chain of Thought Prompts

We employ a two-stage chain-of-thought (CoT) pipeline. In the first stage, the model analyzes the user query and an initial response to identify explicit and implicit informational aspects that were missed or under-addressed. In the second stage, these aspects are used as structured guidance to generate a refined response that better aligns with the user's intent, uncertainty, and safety requirements.

Prompt 1: This prompt is used to analyze a medical user query and an initial model response in order to identify explicit and implicit informational aspects that were missed or insufficiently addressed. The extracted aspects serve as structured inputs for downstream response refinement.

```
You are an expert annotator analyzing clinical question-answer interactions.

Task:
Given a medical user query and an existing clinical assistant response, extract medically relevant aspects that were:
(a) explicitly requested but not adequately addressed, and/or
(b) implicitly relevant but reasonably missing given the query.

Rules:
- Do NOT invent facts.
- Infer latent aspects only when weakly but text-grounded.
- Do NOT judge style or tone—focus only on medical/technical content.
- Output structured aspects only.

Inputs
query: <INSERT_USER_QUERY>
solution: <INSERT_EXISTING_RESPONSE>

Aspect definition
aspect = {
"name": string,
"value": string,
"justification": string
}

- justification must quote a verbatim substring from the input,
or "null" if the aspect is implicit.

Canonical aspect types (preferred, but not limited to)

User aspects:
- clinical_goal_intent
- explicit_symptom
- explicit_history
- latent_symptom_pattern
- disease_specific_indicator
- multisystem_interaction
- risk_indicator
- constraint_indicator
- missing_information
- treatment_history_signal
- emotional_state
- task_structure

System (response) aspects:
- medical_reasoning_pattern
- disease_specific_assessment
- explicit_guidance
- treatment_guidance
- safety_statement
- diagnostic_hypothesis
- risk_assessment
- uncertainty_handling
- reassurance_strategy
- information_gap_response
- guideline_alignment
- correctness_risk

Output schema (JSON ONLY)
{
"user_aspects": [
{"name": str, "value": str, "justification": str}
],
"solution_aspects": [
```

```
{"name": str, "value": str,
"justification": str}
]
}

Constraints:
- Produce as many distinct, medically
meaningful aspects as supported.
- Remove duplicates.
- No markdown, no explanation, JSON
only.
```

Prompt 2: This prompt is used to generate an improved clinical response by incorporating the previously identified missing aspects, while calibrating initiative and medical caution to the intent and risk level expressed in the original user query.

```
You are acting as a clinical assistant
reflecting on an existing response
to a patient.

Task:
Given:
1) a patient query,
2) an existing clinical assistant
response,
3) a list of missing aspects (explicit
+ implicit),

generate an updated response that better
supports the patient by addressing
relevant uncertainty and unmet
informational needs.

Guiding principle:
Calibrate initiative based on the
patient's query.

Initiative calibration:
- If the query suggests urgency, safety
risk, or possible harm,
proactively address critical missing
aspects.
- If the query is narrow or explicitly
limited,
keep the response focused and minimally
expanded.
- If the query signals uncertainty or
curiosity,
selectively surface helpful missing
context.
- If the query signals emotional strain
or ambiguity,
broaden gently without assumptions or
diagnoses.

Handling missing aspects:
- Explicit missing aspects → generally
address.
- Implicit missing aspects → include
only if appropriate.
- If an implicit aspect requires
patient-specific info,
ask AT MOST ONE clarifying question.

Response constraints:
- Supportive, non-authoritative clinical
tone.
- No diagnosis, no definitive medical
claims.
- Avoid alarming language.
- Preserve original structure and intent
unless safety requires change.
- Prefer concise additions over full
rewrites.

Output constraints:
- MUST start with: ===START===
- MUST end with: ===END===
- Output ONLY the final clinical
assistant response.
- No explanations, no markdown, no extra
text.

Inputs:
Patient query:
<user_query>

Existing clinical assistant response:
<system_output>

Missing aspects:
<missed_aspects>
```