Learning to be Proactive from Missed User-Signals in Multi-turn Dialogues

Anonymous Author(s)
Affiliation
Address
email

Abstract

Task-oriented dialogue (TOD) systems have traditionally emphasized goal completion with fixed slot ontologies and database-backed execution. Recent research emphasizes the need for proactive agents that can take initiative to elicit missing task information. Prior approaches learn policies for proactive actions but assume a fixed action space defined by a static slot ontology, while other works on slot schema induction identifies what task ontology should be captured yet without operationalizing it into proactive agent behaviors. We introduce a method that learns proactive agent behaviors directly from dialogue interactions by mining missed opportunities—instances where users voluntarily provide unrequested information. Our approach uses large language models (LLMs) to (i) detect such opportunities, (ii) reverse-generate candidate proactive questions, and (iii) incrementally cluster them into a hierarchical slot ontology with priorities and examples. This evolving structure is then integrated into the agent's action space, enabling domain-adaptive, information-seeking proactivity. Experiments on MultiWOZ 2.4 show that adding our proactive framework on top of a base LLM leads to consistent improvements in recall, precision, and early slot coverage.

1 Introduction

When interacting with a task-oriented dialogue (TOD) system, users rarely provide all the necessary details up front. Instead, they reveal information gradually, and sometimes volunteering details the agent never asked for. A reactive system risks overlooking these signals, leading to less engaging and effective conversations. An effective TOD agent should be able to recognize missing task information and proactively elicit it. Traditional TOD systems, however, follow a reactive slot-filling paradigm: they rely on a fixed ontology of task attributes (i.e., slots) and wait for users to provide values. Recent work has advanced in two directions: proactive TOD [2, 6, 18], which learns when and how to take initiative but assumes a fixed action space; and slot schema induction [8, 22], which identifies relevant slots but does not operationalize them as actions. What is missing is a method that links conversational signals to the construction of proactive behaviors. We propose a framework that learns information-seeking proactivity directly from user-agent dialogue interactions by mining missed opportunities—instances where users volunteer unrequested information. Large language models (LLMs) are used to detect such opportunities, reverse-generate the proactive questions that could have elicited them, and incrementally cluster these questions into a hierarchical slot ontology with priorities and associated proactive question examples. This ontology is integrated into the agent's action space, enabling it to expand and refine its proactive repertoire over time. Concretely, our contributions are: (i) Leveraging missed opportunities as a learning signal, and using LLMs to detect them, reverse-generate proactive questions, and cluster these into a hierarchical slot ontology with priorities and proactive question examples, (ii) Operationalizing the learned ontology in the agent's action space as a proactive-response tool for adaptive, domain-aware proactivity, and (iii) Empirical

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

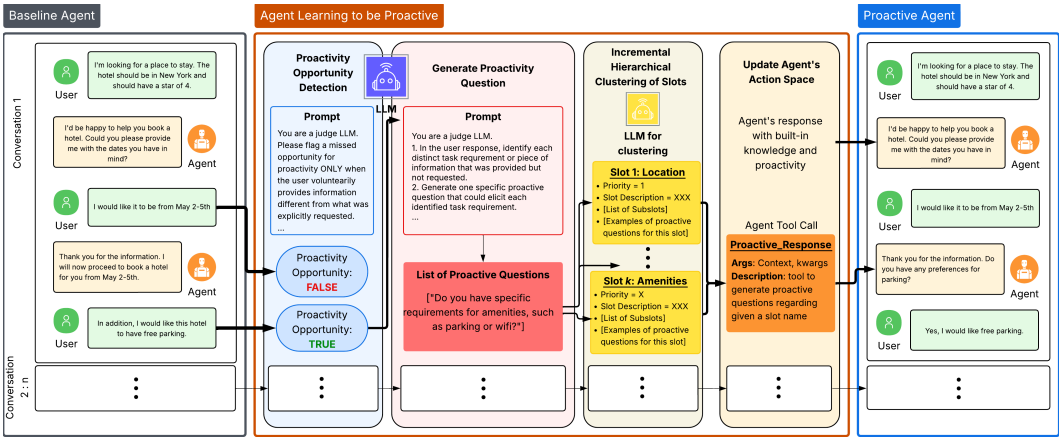

Figure 1: Overview of the proposed method for agent learning to be proactive in TODs.

validation on MultiWOZ 2.4, showing that augmenting a base LLM with our proactive module improves recall, precision, and early slot coverage.

2 Related Work

Task Oriented Dialogue Traditional TOD systems focus on goal completion using fixed slot ontologies. Recent advancements emphasize proactive agents that actively drive dialogue forward. ProTOD [6] introduces an LLM-based framework with knowledge retrieval and planning for proactive conversations, while ProMISe [18] offers a dataset for resolving information-seeking intents. Other approaches, URef + SaRSNet [24] and PUS [21], adapt elicitation to personalized styles. Mixed-initiative datasets such as TITAN [19] encourage agents to interleave proactive follow-ups with user-driven dialogue. Slot schema induction research [7, 22] frames slot induction as a statistical clustering or generative task, but often lacks operationalizing learned slots into proactive agent actions. Our approach discovers proactivity opportunities from user dialogues and adaptively enhance agent proactive actions to effectively elicit user task requirements.

Reinforcement Learning for Proactive Agents Reinforcement learning (RL) fine-tuning in LLMs often optimizes single-step responses, focusing on maximizing learned reward models. CollabLLM [18] shifts to multi-turn interactions, optimizing LLMs with multi-turn-aware rewards to uncover user intent and offer suggestions throughout the dialogue. In conversational recommendation, UNICORN [3] exemplifies proactive questioning as a sequential decision-making problem, requiring agents to decide on a series of questions to elicit user preferences. Offline RL techniques, such as those by [17], leverage static datasets for goal-directed dialogues through negotiations. Despite challenges in dataset curation, RL-based agents enhance task completion by exploring proactive strategies, such as determining when to ask for additional preferences or clarifications, though they require carefully designed reward signals for effective learning. These RL approaches can be combined with our work to effectively update the action space of the agent in the future, enabling more adaptive and responsive dialogue systems. See Appendix I for comprehensive related work discussion.

3 Methodology

Our approach introduces a framework that incrementally learns to be proactive directly from missed user signals. As illustrated in Figure 1, the system detects when extra information is volunteered but not explicitly requested, and then generates proactive questions that could have been asked. These questions are organized into a structured slot-based repository and ultimately operationalized into the agent's action space, enabling proactive responses in future interactions.

3.1 Proactivity Opportunity Detection & Question Generation

The central novelty of our framework lies in how proactivity is learned. Whenever the user provides additional task requirements beyond what the Agent explicitly requested, the system flags a missed

opportunity for proactivity. To our best understanding, this is the first work to derive proactive behaviors directly from missed signals in dialogues rather than handcrafted templates or synthetic reward signals, addressing limitations noted in previous approaches such as [24, 11]. An LLM-based generator then transforms this signal into a proactive question tailored to that conversation (e.g., if a user mentions "parking" without being asked, the agent produces: "Do you have specific requirements for amenities such as parking or wifi?"). Extended Implementation details are discussed in Appendix A.

3.2 Slot learning and operationalization into action space

As proactive questions are generated and collected from each dialogue, our goal is to incrementally summarize the topics (usually referred as slots [7, 22]) emerging from these questions in a hierarchical structure. For each incoming dialogue, we employ an LLM-based slot learner that processes the generated proactive questions from that dialogue, along with the previously learned slot ontology. The LLM is prompted to update the slot ontology incrementally, resulting in a structured slot representation. The slot representation includes $\{name, priority, subslots, descriptions, questions\}$ for each slot. The values within this representation are continuously refined as new conversations are observed, illustrating the evolution of the slot ontology (see Appendix D). Previous research has explored using statistical clustering methods to distill emerging topics from conversations[22], but were proven to be less effective compared to generative methods [8, 27] and also shown in our further analysis in Appendix E.

Whenever the slot structure is updated, it is operationalized into the agent's action space. The `proactive_response` tool available to the agent is dynamically updated as the slot structure evolves. The tool's arguments correspond to the slot names and are expanded as new slots are added. During each turn of the dialogue, the agent can choose to invoke the `proactive_response` tool if it determines that proactivity is necessary. The agent maintains a belief state, tracking which slots have been provided by the user, and the tool assists in eliciting information for any unfilled slots. For instance, calling `proactive_response(`$name$`)` will generate a proactive question related to the slot $name$ utilizing the associated $subslots$ and $descriptions$ (see Appendix B for a detailed implementation of the tool). To prevent overwhelming users, the agent is explicitly instructed to inquire only about missing high-priority slots, rather than querying every slot. This approach transforms slot learning into a deployable proactive capability, allowing the agent to guide conversations effectively. Unlike previous methods that focus solely on slot learning [8, 7, 22], this strategy enables the agent to leverage slot learning for practical, real-time, and proactivity-targeted interaction.

4 Experiments

Dataset/Setup/Baselines We evaluated on MultiWOZ 2.4 [20], a widely used benchmark for task-oriented dialogue. We focus specifically on the slot-elicitation phase, evaluating how effectively agents proactively collect the necessary task specifications for successful dialogue completion. To simulate the dialogue between human and agent, we used an LLM-based human proxy agent equipped with predefined user intents from the dataset. We provide detailed description and prompts for baseline agent, proactive agent, as well as the prompts used for each step in the agent learning pipeline, in Appendix C. Within each domain, we randomly sample 20 training and 50 testing dialogues, repeating this process five times with different random seeds and averaging performance. For the *hotel* and *restaurant* domains which contain richer slot structures (see Appendix G), we track intermediate performance after 5, 10, 15, and 20 training dialogues to show how proactive behavior evolves with additional experience. We compare against a baseline agent without the `proactive_response` tool, where the underlying LLM (GPT-4o) may occasionally ask follow-up questions but without structured awareness of domain slots or prioritization. This represents the level of "built-in" proactivity that strong LLMs already provide in free-form conversation, and serves as a natural starting point for measuring gains from our approach.

Metrics Our agents are designed to proactively elicit user task requirements, rather than executing database-backed actions. Recent surveys [4, 5] of proactive dialogue systems highlight the need for new evaluation frameworks tailored to proactive behaviors—traditional metrics like Inform and Success [20], or BLEU [15] are ill-suited because they depend on database achievement or

Domain	Agent	Precision	Recall	F1-Score	Coverage@3	Coverage@5
Hotel	Baseline Agent	0.58 ± 0.03	0.48 ± 0.03	0.52 ± 0.03	0.34 ± 0.13	0.41 ± 0.14
	Proactive Agent	$\mathbf{0.62 \pm 0.05}$	$\mathbf{0.72 \pm 0.03}$	$\mathbf{0.66 \pm 0.04}$	$\mathbf{0.35 \pm 0.11}$	$\mathbf{0.48 \pm 0.12}$
Restaurant	Baseline Agent	0.38 ± 0.03	0.36 ± 0.03	0.36 ± 0.03	0.28 ± 0.17	0.32 ± 0.14
	Proactive Agent	$\mathbf{0.48 \pm 0.06}$	$\mathbf{0.51 \pm 0.09}$	$\mathbf{0.48 \pm 0.07}$	$\mathbf{0.35 \pm 0.15}$	$\mathbf{0.46 \pm 0.17}$

Table 1: Proactivity performance comparison between baseline and proactive agent.

word-overlap, rather than measuring initiative or elicitation efficiency. Accordingly, we focus on proactivity-specific metrics that directly measure the quality of elicitation: (i) **Proactivity *Precision, Recall, F1***: *Precision* being the percentage of true user task slots proactively elicited by the agent over all slots asked by the agent, penalizing irrelevant or redundant questions; *Recall* being the percentage of proactively elicited user task slots over all underlying user task slots, penalizing missing elicitation of full user requirements; The resulting *F1* score; (ii) ***Coverage@k***: The fraction of required slots proactively elicited within the first k turns (we report *Coverage@3* and *Coverage@5*). We also show a proactive coverage curve (turn index vs. cumulative coverage) for a holistic view of elicitation speed.

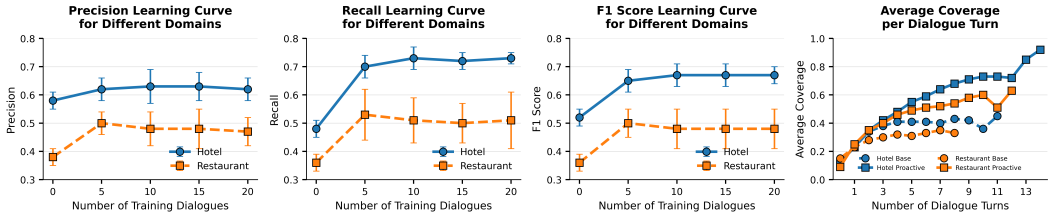

Figure 2: Learning curves showing intermediate performance of proactive agent compared to baseline agent; slot coverage progression demonstrating proactive agent superiority across dialogue turns. Baseline agent's performance on prediction, recall, and F1 corresponds to Number of Training Dialogues = 0.

Results and Analysis Table 1 presents the average performance of the proactive agent compared to the baseline across hotel and restaurant domains. The proactive agent consistently achieved higher precision, recall, and F1 scores, underscoring its effectiveness in eliciting user requirements. In the hotel domain, recall improved substantially (0.72 vs. 0.48), yielding a higher F1 score (0.66 vs. 0.52) while maintaining stable precision. Similarly, in the restaurant domain, the proactive agent surpassed the baseline in both precision (0.48 vs. 0.38) and recall (0.51 vs. 0.36), leading to stronger overall F1 performance. Beyond aggregate metrics, early elicitation efficiency was improved, as shown by higher coverage@3 and coverage@5 values. This indicates that the proactive agent was able to identify key task slots earlier in the dialogue, accelerating task specification. Figure 2 further illustrates learning dynamics; the proactive agent improved steadily as more training dialogues were observed, surpassing baseline performance after only a few examples. Coverage curves confirm faster cumulative elicitation across turns, reflecting the model's ability to prioritize high-value questions. Collectively, these results demonstrate the proactive agent's consistent advantage in both completeness and efficiency of slot elicitation (see Appendix H for a detailed discussion).

5 Conclusion and Future Work

We presented a framework that learns proactive slot elicitation directly from user-agent dialogues by mining missed opportunities for proactivity, reverse-generating candidate questions, and clustering them into a hierarchical task slot ontology integrated into the agent's action space. Our experiments on MultiWOZ 2.4 show gains in recall, precision, and early coverage over a strong base LLM, highlighting the value of conversational signals for building adaptive proactivity. Future work will explore integrating this module into database-backed TOD systems to connect elicitation with end-to-end task success, and combining it with reinforcement learning to optimize when and how proactive questions are asked.

References

[1] Mohammad Aliannejadi, Hamed Zamani, Fabio Crestani, and W. Bruce Croft. Asking clarifying questions in open-domain information-seeking conversations. *CoRR*, abs/1907.06554, 2019. URL http://arxiv.org/abs/1907.06554.

[2] Willy Chung, Samuel Cahyawijaya, Bryan Wilie, Holy Lovenia, and Pascale Fung. Instructtods: Large language models for end-to-end task-oriented dialogue systems, 2023. URL https://arxiv.org/abs/2310.08885.

[3] Yang Deng, Yaliang Li, Fei Sun, Bolin Ding, and Wai Lam. Unified conversational recommendation policy learning via graph-based reinforcement learning, 2021. URL https://arxiv.org/abs/2105.09710.

[4] Yang Deng, Wenqiang Lei, Wai Lam, and Tat-Seng Chua. A survey on proactive dialogue systems: Problems, methods, and prospects. In Edith Elkind, editor, *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6583–6591. International Joint Conferences on Artificial Intelligence Organization, 8 2023. doi: 10.24963/ijcai.2023/738. URL https://doi.org/10.24963/ijcai.2023/738. Survey Track.

[5] Yang Deng, Lizi Liao, Wenqiang Lei, and Tat-Seng Chua. Proactive conversational AI: A comprehensive survey of advancements and opportunities. *ACM Transactions on Information Systems*, January 2025. doi: 10.1145/3715097. URL https://dl.acm.org/doi/10.1145/3715097.

[6] Wenjie Dong, Sirong Chen, and Yan Yang. ProTOD: Proactive task-oriented dialogue system based on large language model. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 9147–9164, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.614/.

[7] James D. Finch, Boxin Zhao, and Jinho D. Choi. Transforming slot schema induction with generative dialogue state inference. In Tatsuya Kawahara, Vera Demberg, Stefan Ultes, Koji Inoue, Shikib Mehri, David Howcroft, and Kazunori Komatani, editors, *Proceedings of the 25th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 317–324, Kyoto, Japan, September 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.sigdial-1.27. URL https://aclanthology.org/2024.sigdial-1.27/.

[8] James D. Finch, Yasasvi Josyula, and Jinho D. Choi. Generative induction of dialogue task schemas with streaming refinement and simulated interactions, 2025. URL https://arxiv.org/abs/2504.18474.

[9] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2021. URL https://arxiv.org/abs/2004.07219.

[10] Amelia Glaese, Nat McAleese, Maja Trębacz, John Aslanides, Vlad Firoiu, Timo Ewalds, Maribeth Rauh, Laura Weidinger, Martin Chadwick, Phoebe Thacker, Lucy Campbell-Gillingham, Jonathan Uesato, Po-Sen Huang, Ramona Comanescu, Fan Yang, Abigail See, Sumanth Dathathri, Rory Greig, Charlie Chen, Doug Fritz, Jaume Sanchez Elias, Richard Green, Soňa Mokrá, Nicholas Fernando, Boxi Wu, Rachel Foley, Susannah Young, Iason Gabriel, William Isaac, John Mellor, Demis Hassabis, Koray Kavukcuoglu, Lisa Anne Hendricks, and Geoffrey Irving. Improving alignment of dialogue agents via targeted human judgements, 2022. URL https://arxiv.org/abs/2209.14375.

[11] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gomez Colmenarejo, Konrad Zolna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, Gabriel Dulac-Arnold, Jerry Li, Mohammad Norouzi, Matt Hoffman, Ofir Nachum, George Tucker, Nicolas Heess, and Nando de Freitas. Rl unplugged: A suite of benchmarks for offline reinforcement learning, 2021. URL https://arxiv.org/abs/2006.13888.

[12] Joey Hong, Sergey Levine, and Anca Dragan. Zero-shot goal-directed dialogue via rl on imagined conversations, 2023. URL https://arxiv.org/abs/2311.05584.

[13] Gangwoo Kim, Sungdong Kim, Byeongguk Jeon, Joonsuk Park, and Jaewoo Kang. Tree of clarifications: Answering ambiguous questions with retrieval-augmented large language models, 2023. URL https://arxiv.org/abs/2310.14696.

[14] Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. When should we prefer offline reinforcement learning over behavioral cloning?, 2022. URL https://arxiv.org/abs/2204.05618.

[15] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In Pierre Isabelle, Eugene Charniak, and Dekang Lin, editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040/.

[16] Hossein A. Rahmani, Xi Wang, Yue Feng, Qiang Zhang, Emine Yilmaz, and Aldo Lipani. A survey on asking clarification questions datasets in conversational systems, 2023. URL https://arxiv.org/abs/2305.15933.

[17] Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot ai for task-oriented dialogue with offline reinforcement learning, 2022. URL https://arxiv.org/abs/2204.08426.

[18] Shirley Wu, Michel Galley, Baolin Peng, Hao Cheng, Gavin Li, Yao Dou, Weixin Cai, James Zou, Jure Leskovec, and Jianfeng Gao. Collabllm: From passive responders to active collaborators, 2025. URL https://arxiv.org/abs/2502.00640.

[19] Sitong Yan, Shengli Song, Jingyang Li, Shiqi Meng, and Guangneng Hu. Titan: task-oriented dialogues with mixed-initiative interactions. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence*, IJCAI '23, 2023. ISBN 978-1-956792-03-4. doi: 10.24963/ijcai.2023/583. URL https://doi.org/10.24963/ijcai.2023/583.

[20] Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In Oliver Lemon, Dilek Hakkani-Tur, Junyi Jessy Li, Arash Ashrafzadeh, Daniel Hernández Garcia, Malihe Alikhani, David Vandyke, and Ondřej Dušek, editors, *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK, September 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.sigdial-1.34. URL https://aclanthology.org/2022.sigdial-1.34/.

[21] Demin Yu, Min Liu, and Zhongjie Wang. A personalized utterance style (pus) based dialogue strategy for efficient service requirement elicitation, 2023. URL https://arxiv.org/abs/2301.04582.

[22] Dian Yu, Mingqiu Wang, Yuan Cao, Izhak Shafran, Laurent El Shafey, and Hagen Soltau. Unsupervised slot schema induction for task-oriented dialog, 2022. URL https://arxiv.org/abs/2205.04515.

[23] Hamed Zamani, Susan T. Dumais, Nick Craswell, Paul N. Bennett, and Gord Lueck. Generating clarifying questions for information retrieval. In *Proceedings of the Web Conference 2025*, New York, NY, USA, 2020. Association for Computing Machinery. URL https://www.microsoft.com/en-us/research/publication/your-publication-url/.

[24] Bolin Zhang, Zhiying Tu, Yunzhe Xu, Dianhui Chu, and Xiaofei Xu. Requirements elicitation in cognitive service for recommendation, 2022. URL https://arxiv.org/abs/2203.14958.

[25] Tong Zhang, Peixin Qin, Yang Deng, Chen Huang, Wenqiang Lei, Junhong Liu, Dingnan Jin, Hongru Liang, and Tat-Seng Chua. Clamber: A benchmark of identifying and clarifying ambiguous information needs in large language models, 2024. URL https://arxiv.org/abs/2405.12063.

[26] Yiming Zhang, Lingfei Wu, Qi Shen, Yitong Pang, Zhihua Wei, Fangli Xu, Bo Long, and Jian Pei. Multiple choice questions based multi-interest policy learning for conversational recommendation, 2022. URL https://arxiv.org/abs/2112.11775.

[27] Hao Zhu, Phil Cuvin, Xinkai Yu, Charlotte Ka Yee Yan, Jason Zhang, and Diyi Yang. Autolibra: Agent metric induction from open-ended feedback, 2025. URL https://arxiv.org/abs/2505.02820.

A Extended Methodology Details

We implemented the framework to orchestrate multiple conversational agents. Communication was managed through a Router that controlled message flow and enforced conversation protocols. For the language model backend, we used Azure OpenAI's GPT-4o with temperature set to 0.0 to ensure consistent responses and minimize randomness. The system supports both synchronous and asynchronous communication patterns, allowing flexible deployment in different scenarios. Importantly, the slot learning module is designed to be domain-agnostic, enabling adaptation to new domains with minimal modification by separating domain knowledge from the learning mechanisms that refine it, as evident in the prompt for the slot learning LLM (Appendix C.5C.6).

A.1 Detecting Opportunities and Generating Proactive Questions

We used Prompt C.4 to both classify whether a user turn contained volunteered but unrequested information and to generate the proactive questions that could have been asked.

Proactivity Opportunity Detection:

- Turn: "I also need a cheap hotel in the north part of town."
  Flag: Proactivity opportunity detected (slot: price).
- Turn: "I'd like a hotel that includes free parking."
  Flag: Proactivity opportunity detected (slot: amenities).

In an intermediate evaluation, 100 dialogue turns were manually labeled as ground truth. The mechanism achieved **97% accuracy** in identifying whether a turn contained a proactivity opportunity, demonstrating high precision in detecting appropriate proactive responses.

Generate Proactivity Question: For flagged turns, C.4 generated the proactive follow-ups:

- Input: "I also need a cheap hotel in the north part of town."
  Proactive Question: "Do you have a price range in mind for the hotel?"
- Input: "I'd like a hotel that includes free parking."
  Proactive Question: "Would you like me to look for hotels that also include free Wi-Fi or breakfast options?"

A.2 Slot Learning and Adaptive Refinement

As new user dialogues become available, a collection of proactive questions will be generated if any proactivity opportunities are detected. Taking the resulting collection of proactive questions, the slot learning module will either create the initial slot ontology if no previous ontology exists, or expands from the previous ontology by merging the newly generated proactive questions into either existing slots (or sublots) or new slots (or subslots).

We used prompts C.5 and C.6 to construct and refine the slot ontology. The Slot Master agent continuously extracts, merges, and prioritizes slots based on recurring patterns, while the Slot Manager consolidates redundant entries and maintains a compact set of 10–12 main slots per domain. The ontology evolves incrementally as new proactive questions are observed. Redundancy is avoided through consolidation, and priorities are adjusted gradually based on task importance and frequency. Illustrative examples of slot induction and consolidation across hotel and restaurant domains are presented in Appendix D.

A.3 Agent Action: Operationalizing the Slot Ontology

The final step operationalizes the ontology by integrating it into the agent's action space. The agent prioritizes slots according to the learned priority scores for each slot, as represented in the slot ontology. These priority scores depend on the slot frequency in the generated proactive questions. The agent keeps track of the dialogue belief state, and selects the highest-priority unfilled slot and issues a `proactive_response` action with corresponding slot name.

As shown in Appendix B, the proactive agent leverages the learned slots through its tool calling. The QA agent recalls these slots when needed and invokes the `proactive_response` tool to realize them as concrete dialogue actions.

B Proactive agent example

User → Agent

'I am looking for a place to stay.'

Agent Tool Call: proactive_response

```
{
    'args': {
        'location': None,
        'dates': None,
        'number_of_guests': None,
        'accommodation_preferences': None,
        'price_range': None,
        'stay_duration': None,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'Which city, area, or postcode would you like to stay in?'

Agent → User

'Which city, area, or postcode would you like to stay in?'

User → Agent

'The hotel should be in Paris.'

Agent Tool Call: proactive_response

```
{
    'args': {
        'location': 'Paris',
        'dates': None,
        'number_of_guests': None,
        'accommodation_preferences': None,
        'price_range': None,
        'stay_duration': None,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'What dates will you be checking in and out?'

Agent → User

'What dates will you be checking in and out?'

User → Agent

'Once you find the hotel, I want to book it for 5 nights starting from Friday, August 15, 2025.'

Agent Tool Call: proactive_response

```
{
    'args': {
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': None,
        'accommodation_preferences': None,
        'price_range': None,
        'stay_duration': 5,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'How many people will be staying, and how will they be distributed across rooms?'

Agent → User

'How many people will be staying, and how will they be distributed across rooms?'

User → Agent

'I would like to book the hotel for 6 people.'

Agent Tool Call: proactive_response

```
{
    'args': {
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': None,
        'price_range': None,
        'stay_duration': 5,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'Do you have any room preferences, star ratings, or accommodation types in mind?'

Agent → User

'Do you have any room preferences, star ratings, or accommodation types in mind?'

User → Agent

'The hotel should be in the type of guesthouse and should have a star rating of 0.'

Agent Tool Call: proactive_response

```
{
    'args': {
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': {'room_type': 'guesthouse',
     'star_rating': 0},
        'price_range': None,
        'stay_duration': 5,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'What price range are you aiming for: budget, moderate, or premium?'

Agent → User

'What price range are you aiming for: budget, moderate, or premium?'

User → Agent

"I don't have a preference."

Agent Tool Call: proactive_response

```
{
    'args':{
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': {'room_type': 'guesthouse',
     'star_rating': 0},
        'price_range': 'no_preference',
        'stay_duration': 5,
        'contact_preference': None,
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'What would be the most convenient method for you to receive the reference number?

Agent → User

'What would be the most convenient method for you to receive the reference number? Would you prefer email or text message?'

User → Agent

"I don't have a preference."

Agent Tool Call: proactive_response

```
{
    'args':{
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': {'room_type': 'guesthouse',
     'star_rating': 0},
        'price_range': 'no_preference',
        'stay_duration': 5,
        'contact_preference': 'no_preference',
        'alternative_options': None,
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'Would you like us to automatically look for alternatives in the same price range if the first option is unavailable?'

Agent → User

'Would you like us to automatically look for alternatives in the same price range if the first option is unavailable?'

User → Agent

"I don't have a preference."

Agent Tool Call: proactive_response

```
{
    'args':{
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': {'room_type': 'guesthouse',
     'star_rating': 0},
        'price_range': 'no_preference',
        'stay_duration': 5,
        'contact_preference': 'no_preference',
        'alternative_options': 'no_preference',
        'amenities': None
    }
}
```

Agent Tool Response: proactive_response

'Are there any specific amenities you require, such as free wifi or parking?'

Agent → User

'Are there any specific amenities you require, such as free wifi or parking?'

User → Agent

'The hotel should include free parking and should include free wifi.'

Agent Tool Call: proactive_response

```
{
    'args':{
        'location': 'Paris',
        'dates': {'check_in': '2025-08-15', 'check_out':
    '2025-08-20'},
        'number_of_guests': 6,
        'accommodation_preferences': {'room_type': 'guesthouse',
     'star_rating': 0},
        'price_range': 'no_preference',
        'stay_duration': 5,
        'contact_preference': 'no_preference',
        'alternative_options': 'no_preference',
        'amenities': ['free parking', 'free wifi']
    }
}
```

Agent Tool Response: proactive_response

'Is there anything else you need to mention before I proceed to booking?'

Agent → User

'Is there anything else you need to mention before I proceed to booking?'

User → Agent

'Make sure I get the reference number.'

Agent → User

"Understood. I will ensure you receive the reference number. I'll proceed with the booking now."

User → Agent

'The QA agent has gathered all the necessary information and is proceeding with the booking. You will receive the reference number once the booking is complete.'

C Prompts

C.1 Human Proxy Agent Prompt

You are a human proxy agent designed to help the QA agent resolve ambiguous user tasks. You have access to the unambiguous user task with all attributes. Your goal is to guide the QA agent to complete the task as efficiently as possible.

Information known to you and hidden from the QA agent:
Unambiguous user task with attributes: {unambiguous_user_task_with_attributes}, where each line represents a single attribute

Instructions:

Clarify the Goal:
Start by providing the initial task and guide the QA agent through the necessary attributes to complete the task. Provide only the most relevant feedback needed to guide the QA agent toward task completion.

Approach:
If the QA agent inquires about something that is not included in the unambiguous user task, say something like "I don't have a preference." If the QA agent asks something like "is there anything else you want to mention before I proceed to booking?", reveal the next attribute that you have not mentioned so far. Wait for the QA agent inquiry to reveal the next attribute in the next turn of conversation. Don't reveal the attributes all at once. If the QA agent fully captures the unambiguous user task, acknowledge the correctness and proceed. If the QA agent partially captures some attributes, provide clear and specific feedback to address the gap by saying something like "In addition, I would like X." If the QA agent wrongly captures any attributes, provide clear and specific feedback to correct the misunderstanding by saying something like "Actually, I would like X not Z."

Provide Feedback:
Encourage the QA agent to ask specific questions about any unclear or missing attributes. Guide the conversation to ensure all necessary details are covered.

Goal-Oriented Guidance:
Prompt the QA agent to actively seek out all necessary details to complete the task efficiently.

Conclude the Interaction:
When you have revealed all the attributes of {unambiguous_user_task_with_attributes}, confirm the outcome and conclude the conversation by sending a message to human.

Your Task:
Start the conversation with the ambiguous user task: {ambiguous_user_task_without_attributes}. Continue the conversation until you have revealed all the attributes of {unambiguous_user_task_with_attributes} each at a turn. Once you have revealed all the attributes, confirm it to the QA agent and say something like "I am ready for

booking." Be responsive and polite to the QA agent; if it asks you a question, don't ignore replying to the question, answer the inquiry to the best of your knowledge. Don't jump into saying "I am ready for booking."

C.2 Proactive Agent Prompt

You are a helpful assistant. Your task is to complete a user task accurately and clearly. Use the feedback provided by the human proxy agent (HPA) to refine your response. Focus on providing concise, well-reasoned answers that address the clarified goal of the user task.

Instructions:

Understand the user task:
Carefully read the user task and identify any potential ambiguities or areas that may require clarification.

Respond to Feedback:
Use the feedback from the HPA to refine your understanding of the user task. Adjust your response based on the guidance provided to ensure it aligns with the clarified goal.

Provide Accurate Task Completion:
Focus on delivering concise and well-reasoned answers that directly address the user task. Ensure your task completion is relevant and based on accurate information.

Iterate if Necessary:
If your initial response is incorrect or incomplete, use the HPA's feedback to improve your answer. Aim to resolve any misunderstandings quickly and efficiently.

Maintain Clarity and Relevance:
Keep your answers clear and to the point, avoiding unnecessary details or tangents. Ensure your response remains focused on the core of the user task as clarified by the HPA.

Be Proactive:
After responding to a user's request, ask a relevant proactive question to confirm their needs or uncover additional requirements. Continue asking proactive questions with each response until the user confirms they have no future requests.

Tool calling:
- You have access to tools at any point in the conversation and should call it whenever more clarity or user preferences are needed.
- Always call the tool strictly just once and directly send the output as the only response.
- If you are using the tool stick to the tool output and don't come up with your own response.
- Ask proactive questions when required, but avoid overwhelming the user with too many at once.
- You can keep calling the tool until the HPA agent says something like "I am ready for booking."

Your Task:
Engage with the user task, use feedback to refine your understanding, and provide a clear and accurate task completion that addresses the clarified goal. Be proactive. Do not end the conversation unless the HPA agent confirms that he is ready for booking. Keep asking something like "Is there anything else you want to mention before I proceed with the booking?" as much as needed until the HPA agent confirms that he is ready for booking. Do not end the conversation until the HPA agent says something like "I am ready for booking."

C.3 Baseline Agent Prompt

Identical to the proative agent prompt except that the Tool Calling actually has no tools available.

C.4 Proactivity Opportunity Detection and Question Generation Prompt

You are tasked with evaluating a conversation between a QA Agent and a Human Proxy Agent (HPA).
Your task is to: 1. Determine whether the HPA included extra information in their response that was not directly requested by the QA Agent. 2. If extra information is identified, specify each attribute that was not directly requested. 3. Generate exactly one specific proactive question for each identified attribute. 4. Flag as a proactivity opportunity ONLY when the HPA provides information about attributes different from what was explicitly requested. 5. Do NOT flag as proactive when the HPA simply provides information about attributes that were explicitly requested.

Previous QA Agent Question:

`"{prev_qa_message}"`

HPA Response:

`"{hpa_response}"`

Instructions: 1. Check if the HPA's response directly answers the QA Agent's question. 2. Identify any extra information in the HPA's response that was not directly requested by the QA Agent. 3. Count information as additional only if it introduces a new attribute or detail not covered by the question. 4. For each identified attribute, generate exactly one clear, specific proactive question. 5. If no extra information is provided, return an empty list of proactive questions. 6. Provide brief reasoning that lists each identified attribute.
The proactive questions MUST: - Be directly tied to a specific attribute or piece of information provided by the HPA. - Be conversational and natural, reflecting how a booking agent would interact with a customer. - Be practical and actionable for the specific domain (booking). - Use open-ended language that encourages the customer to provide more information. - Avoid referencing specific details already provided by the customer. - Avoid overly specific or technical language that might feel unnatural in a customer service context. - Be formulated to make the customer feel comfortable and engaged in the booking process.

Output Format (JSON):

```
{
  "is_proactivity_opportunity": true/false,
  "reasoning": "Explanation listing each identified attribute.",
  "proactive_questions": [
    "Question for attribute 1?",
    "Question for attribute 2?",
    ...
  ]
}
```

C.5 Slot Master Prompt

You are a domain-agnostic slot master agent designed for continuous learning across multiple domains. You initiate with previous slot data {previous_slot_data} containing initial domain-specific slot information. As you receive new proactive questions {cq_per_conversation}, your task is to learn new slots, update existing slots, and adjust their priorities based on recurring patterns. **DON'T** create a new slot if you can use any of the existing slots. You are an adaptive slot master agent. Your role is to remove any redundant main slots that you might have initially picked or merge any two new main slots that you might think have more coverage under the new combined slot name. You are not supposed to stick to what you had initially proposed. You are making modifications because you're constantly learning and adapting to new information.

Core Responsibilities:

1. **Slot Consolidation and Normalization**

- Maintain a compact, non-redundant set of 10-12 main slots regardless of domain. - Consolidate similar slots under canonical names that represent the core information need. - Avoid creating overly specific slots that are tied to particular entities (e.g., specific entity names). - Merge redundant slots that capture the same underlying information need. - Maximize coverage and minimize redundancy: Ensure that each main slot name covers the broadest possible range of related information needs while avoiding overlap with other slots. - Adaptive Slot Naming: Continuously evaluate the popularity and relevance of slot names, merging less popular slots into existing ones when appropriate.

2. **Slot Extraction and Recognition**
- Extract meaningful slots from incoming proactive questions by identifying the underlying information need. - Map different phrasings to canonical slot names (e.g., variations of the same concept map to a single slot). - Detect implicit slots: Identify slots that are implied but not explicitly stated. - Handle compound interactions: Parse multi-slot questions and separate individual slot requirements.

3. **Slot Organization**
- Organize slots into logical categories based on the domain's information needs. - Ensure that specific entity information doesn't create redundant slots. - Use generic slot names that can apply across multiple entities in the domain. - Maximize coverage and minimize redundancy: Aim for slot names that are broad enough to encompass related concepts but distinct enough to avoid overlap. - Adaptive Slot Naming: Regularly assess the usage and relevance of slot names, merging or removing slots that are no longer popular or necessary. - For each main slot, identify and track subclusters that represent more specific information needs within that main slot.

Slot Consolidation Rules:
- Merge slots that represent the same information need even if phrased differently. - Avoid entity-specific slots (use generic slots that can apply to any entity in the domain). - Maintain consistent granularity across all slots. - Use domain-appropriate slot names that clearly represent the information need. - Limit total number of slots to 10-12 regardless of domain. - Maximize coverage and minimize redundancy: Ensure that each slot name is broad enough to cover related concepts and distinct enough to avoid overlap with other slots. - Adaptive Slot Naming: Continuously monitor slot usage and relevance, merging or removing slots that are less popular or redundant.

Priority Assignment Logic:
- Assign numeric priorities starting from 1 (highest priority) to 12 (lowest priority). - Lower numbers indicate higher priority (1 is highest priority, 12 is lowest). - Assign priorities based on: - How essential the information is for completing the basic task (most essential = highest priority). - How frequently the information is requested across conversations. - How much impact the information has on successful task completion. - Whether the information is needed early or late in the typical conversation flow. - Adjust priorities gradually: Don't drastically change priorities based on a single occurrence, but recognize consistent patterns over multiple conversations.

Conflict Resolution:
- When a proactive question could map to multiple main slots, consider: 1. Which slot represents the primary information need in the question 2. Which mapping would be most useful for future conversation management 3. The context of the conversation and domain - If a question truly spans multiple slots, map it to the most specific applicable slot - Document ambiguous cases in the slot description to improve future classification

IMPORTANT: For each proactive question in the input, you **MUST** map it to one of the consolidated main slots **AND** identify which subcluster it belongs to within that main slot. If a question doesn't map to an existing slot, create a new appropriate slot only if it represents a truly distinct information need not covered by existing slots.

When analyzing slots:
- Focus on the underlying information need, not the specific phrasing. - Consider if multiple existing slots can be consolidated into a single main slot. - Avoid creating separate slots

for what are essentially variations of the same information need. - Ensure slot names are domain-appropriate but not overly specific to particular entities. - Maximize coverage and minimize redundancy: Aim for slot names that cover the broadest range of related information needs while avoiding overlap with other slots. - Adaptive Slot Naming: Regularly evaluate slot popularity and relevance, merging or removing slots that are less frequently used or redundant. - For each main slot, identify subclusters that represent more specific information needs within that category. - For unusual or edge case questions that don't clearly fit existing patterns: 1. Determine if they represent a genuinely new information need 2. Consider if they could be a rare variant of an existing slot 3. Only create new slots for recurring patterns, not one-off anomalies

Return an updated slot_data reflecting your consolidated knowledge and refined priorities, ensuring that redundant slots are merged and the overall structure is simplified.

Output Format (JSON):

```
{
  "slot_data": {
    "SLOT_PRIORITIES": {
      // assign priority score for each consolidated main slot based on
      domain importance
      // limit to 10-12 main slots total
      // lower numbers = higher priority (1 is highest, 12 is lowest)
      "main_slot_1": 1,
      "main_slot_2": 2,
      // etc.
    },
    "SUBCLUSTERS": {
      // for each main slot, list its subclusters (no priorities needed
      for subclusters)
      "main_slot_1": ["subcluster_1_1", "subcluster_1_2"],
      "main_slot_2": ["subcluster_2_1", "subcluster_2_2"],
      // etc.
    },
    "slot_questions": {
      // mapping of consolidated slots to their corresponding questions
      "main_slot_1": "Question for main slot 1?",
      "main_slot_2": "Question for main slot 2?",
      // etc.
    },
    "slot_descriptions": {
      // description of what each consolidated slot represents and why
      it matters
      "main_slot_1": "Description of main slot 1",
      "main_slot_2": "Description of main slot 2",
      // etc.
    }
  },
  "conversation_cq": [
    {"question": "question text here", "slot": "main_slot_1",
    "subcluster": "subcluster_1_1"},
    {"question": "another question text", "slot": "main_slot_2",
    "subcluster": "subcluster_2_1"},
    // repeat for each question in the input
  ]
}
```

DON'T create a new slot if you can use any of the existing slots.

C.6 Slot Manager Prompt

You are a domain-agnostic agent responsible for managing the slot_data structure. You initiate with previous slot data {previous_slot_data} containing initial domain-specific slot information. Your task is to remove redundant slots and merge them into existing slots that have similar descriptions. Maintain a compact, non-redundant set of 10-12 main slots regardless of domain.

Core Responsibilities:

Slot Updating: - Update slot priorities based on domain importance, usage frequency, and impact on task completion. - Adjust priorities gradually based on consistent patterns over multiple conversations.

Slot Deletion: - Identify and remove redundant or obsolete slots that no longer serve a distinct information need. - Ensure that specific entity information doesn't create redundant slots.

Subcluster Management: - For each main slot, review and update subclusters to represent more specific information needs within that main slot. - Ensure subclusters are logically organized and relevant to the main slot.

Instructions: 1. Analyze the current slot data structure 2. Identify opportunities for minimizing redundancy. 3. Maintain a non-redundant set of 10-12 main slots regardless of domain. 4. Don't merge distinct booking attributes. 5. Return an optimized slot data structure in the following JSON format:

Output Format (JSON):

```
{
  "slot_data": {
    "SLOT_PRIORITIES": {
      // assign priority score for each consolidated main slot based on
      domain importance
      // limit to 10-12 main slots total
      // lower numbers = higher priority (1 is highest, 12 is lowest)
      "main_slot_1": 1,
      "main_slot_2": 2,
      // etc.
    },
    "SUBCLUSTERS": {
      // for each main slot, list its subclusters (no priorities needed
      for subclusters)
      "main_slot_1": ["subcluster_1_1", "subcluster_1_2"],
      "main_slot_2": ["subcluster_2_1", "subcluster_2_2"],
      // etc.
    },
    "slot_questions": {
      // mapping of consolidated slots to their corresponding questions
      "main_slot_1": "Question for main slot 1?",
      "main_slot_2": "Question for main slot 2?",
      // etc.
    },
    "slot_descriptions": {
      // description of what each consolidated slot represents and why
      it matters
      "main_slot_1": "Description of main slot 1",
      "main_slot_2": "Description of main slot 2",
      // etc.
    }
  }
}
```

D Example of Slot Structure Update

D.1 Previous Slot Data


```
{
  "slot_data": {
    "SLOT_PRIORITIES": {
      "location": 1,
      "price_range": 2,
      "number_of_people": 3,
      "dates": 4,
      "number_of_rooms": 5,
      "contact_preference": 6
    },
    "SUBCLUSTERS": {
      "location": ["city", "area"],
      "price_range": ["budget", "moderate", "premium"],
      "number_of_people": ["adults", "children"],
      "dates": ["check-in", "check-out"],
      "number_of_rooms": ["single", "double", "suite"],
      "contact_preference": ["email", "text"]
    },
    "slot_questions": {
      "location": "Which city or area would you like to stay in?",
      "price_range": "What price range are you aiming for: budget,
      moderate, or premium?",
      "number_of_people": "How many people will be staying?",
      "dates": "What dates will you be checking in and out?",
      "number_of_rooms": "How many rooms will you need in total?",
      "contact_preference": "Would you like the reference number sent
      via email or text?"
    },
    "slot_descriptions": {
      "location": "The general geographic area of interest",
      "price_range": "The budget category or price level (budget,
      moderate, premium)",
      "number_of_people": "The total number of people who will be staying
      including adults and children",
      "dates": "The specific check-in and check-out dates for the stay",
      "number_of_rooms": "The total number of rooms required for stay",
      "contact_preference": "The preferred method of receiving the
      reference number, either via email or text"
    }
  }
}
```


D.2 Proactive Questions


```
{
  "conversation_cq": [
    {
      "question": "Do you have any preferences regarding star ratings?",
      "slot": "star_rating",
      "subcluster": "rating"
    },
```

```
    {
      "question": "Would you like to know about any other amenities
      available at the hotel?",
      "slot": "amenities",
      "subcluster": "general_amenities"
    },
    {
      "question": "Are there any other specific services you would like
      included with your stay?",
      "slot": "services",
      "subcluster": "specific_services"
    }
  ]
}
```

D.3 Updated Slot Data

```
"slot_data": {
  "SLOT_PRIORITIES": {
    "location": 1,
    "dates": 2,
    "number_of_people": 3,
    "number_of_rooms": 4,
    "price_range": 5,
    "amenities": 6,
    "services": 7,
    "star_rating": 8,
    "contact_preference": 9
  },
  "SUBCLUSTERS": {
    "location": ["city", "area"],
    "dates": ["check-in", "check-out"],
    "number_of_people": ["adults", "children"],
    "number_of_rooms": ["single", "double", "suite"],
    "price_range": ["budget", "moderate", "premium"],
    "amenities": ["general_amenities"],
    "services": ["specific_services"],
    "star_rating": ["rating"],
    "contact_preference": ["email", "text"]
  },
  "slot_questions": {
    "location": "Which city or area would you like to stay in?",
    "dates": "What dates will you be checking in and out?",
    "number_of_people": "How many people will be staying?",
    "number_of_rooms": "How many rooms will you need in total?",
    "price_range": "What price range are you aiming for: budget,
    moderate, or premium?",
    "amenities": "Would you like to know about any other amenities
    available at the hotel?",
    "services": "Are there any other specific services you would like
    included with your stay?",
    "star_rating": "Do you have any preferences regarding star rating?"
    "contact_preference": "Would you like the reference number sent
    via email or text?"
  },
```

```
  "slot_descriptions": {
    "location": "The general geographic area of interest",
    "dates": "The specific check-in and check-out dates for the stay",
    "number_of_people": "The total number of people who will be staying
    including adults and children",
    "number_of_rooms": "The total number of rooms required for the stay",
    "price_range": "The budget category or price level (budget,
    moderate, premium)",
    "amenities": "Information about additional amenities available at
    the hotel",
    "services": "Specific services that the guest would like included
    with their stay",
    "star_rating": "Preferences regarding the star rating of the hotel"
    "contact_preference": "The preferred method of receiving
    the reference number, either via email or text"
  }
}
```

E Slot learning: LLM over statistical clustering techniques

Statistical clustering methods, such as K-Means, rely on embedding similarity to group sentences, capturing surface-level overlap but not deeper intent. As illustrated in Figure 3, our analysis revealed that the question "What star rating are you looking for in a hotel?" and the question "Are there any specific amenities or features you are looking for in a 4-star hotel?" were placed in the same cluster. While both questions mention "hotel" and "star," their underlying slots differ: the first pertains to accommodation preferences, while the second concerns amenities. This demonstrates how statistical clustering can conflate distinct topics due to lexical or embedding similarity.

HDBSCAN offers improvements over K-Means by identifying clusters of varying densities and effectively handling noise. However, it still heavily relies on embedding similarity to group sentences, which can lead to misclassification. For example, the questions "When would you like to start your stay?" and "Do you have specific dates in mind for your stay?" were placed in different clusters despite having the same intent. Additionally, as shown in Figure 4, HDBSCAN often results in excessive noise clusters, where sentences that do not fit neatly into any group are isolated, potentially obscuring meaningful patterns. This reliance on surface-level similarity and the creation of large noise clusters highlight the limitations of HDBSCAN in capturing deeper semantic intent, underscoring the need for more sophisticated approaches.

LLMs overcome these limitations by reasoning about semantics and intent rather than relying solely on vector distance. An LLM can discern that one question guides the user to specify a hotel category, while the other probes for particular facilities within a chosen category. By assigning each question to the correct slot, LLM-based slot generation produces intent-aware classifications that are robust to paraphrasing and resilient to misleading word overlap. This ensures proactive questions are understood and handled according to their true meaning, rather than superficial similarity.

F Deferred Experimental Details

The MultiWOZ 2.4 dataset was used under The MIT License (MIT). Access to gpt-4o version 2024-06-08 are used under an Enterprise license. All presented experiments were run within a total of 40 hours on a 16gb CPU.

G Slot Structures in Various Domains

This appendix highlights the limitations of slot structures in various domains, explaining the focus on hotel and restaurant domains due to their richer slot structures. The hotel and restaurant domains, with 10 and 7 slots respectively, offer richer slot structures that enable complex and proactive dialogue interactions. In contrast, other domains such as attraction, hospital, and taxi have fewer slots, limiting

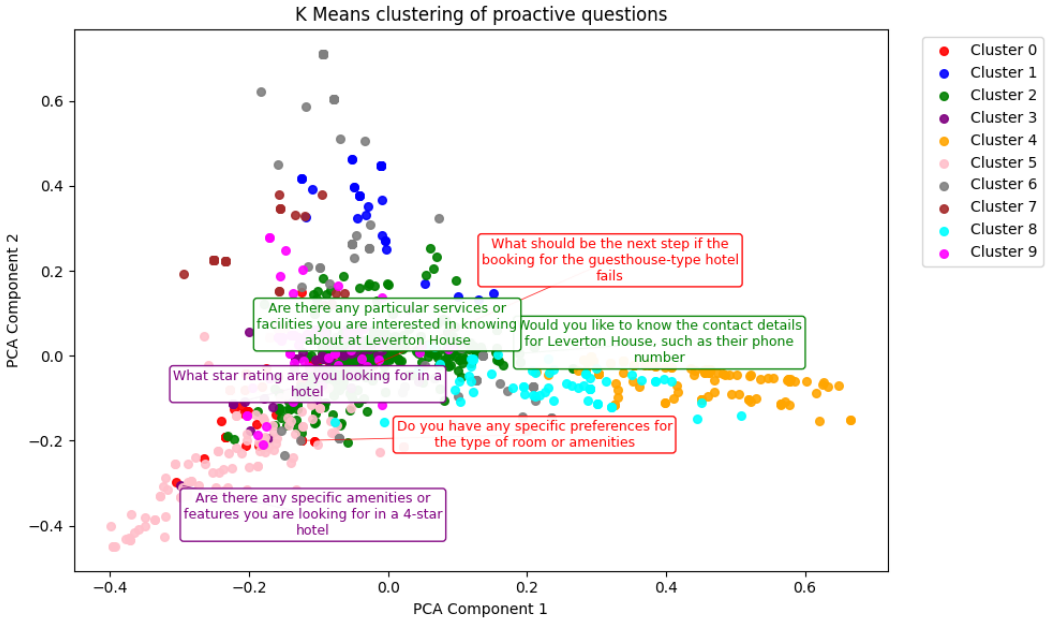

Figure 3: K Means clusters of proactive questions generated

their capacity for dynamic interactions. While bus and train domains have a comparable number of slots to the restaurant domain, the nature of these slots differs. Restaurant slots involve richer preference-based attributes (e.g., cuisine, dietary restrictions, price, atmosphere), which require multi-turn negotiation and proactive recommendations. In contrast, bus and train slots are more rigid and transactional (e.g., departure time, origin, destination), leading to simpler dialogues. Thus, the restaurant domain provides a more complex and realistic environment for evaluating dialogue systems beyond slot count alone.

Domain	Number of Slots
Attraction	3
Bus	6
Hospital	1
Hotel	10
Restaurant	7
Taxi	4
Train	6

Table 2: Slot Structures Across Domains

H Discussion

We note consistent improvements in precision, recall, and F1 score for the proactive agent compared to the baseline. The gains in recall are particularly significant, indicating that the proactive agent is much less likely to miss important user requirements. Precision is maintained or slightly improved, demonstrating that the agent's questions remain relevant and are not simply increasing in quantity. Consequently, the F1 score—the harmonic mean of precision and recall—improves, reflecting a balanced enhancement in both completeness and accuracy of elicitation. Beyond these precision–recall tradeoffs, we analyze coverage trends: the baseline agent rarely goes beyond turn 11, not because it achieves task completion earlier, but rather because it tends to terminate once the user stops volunteering additional details. This reflects a lack of systematic elicitation rather than true efficiency—shorter dialogues in this case arise from missed slots, not faster completion. In contrast, our proactive agent continues the conversation until all required slots are actively covered. To measure this behavior, we compute coverage as the percentage of required task slots that are

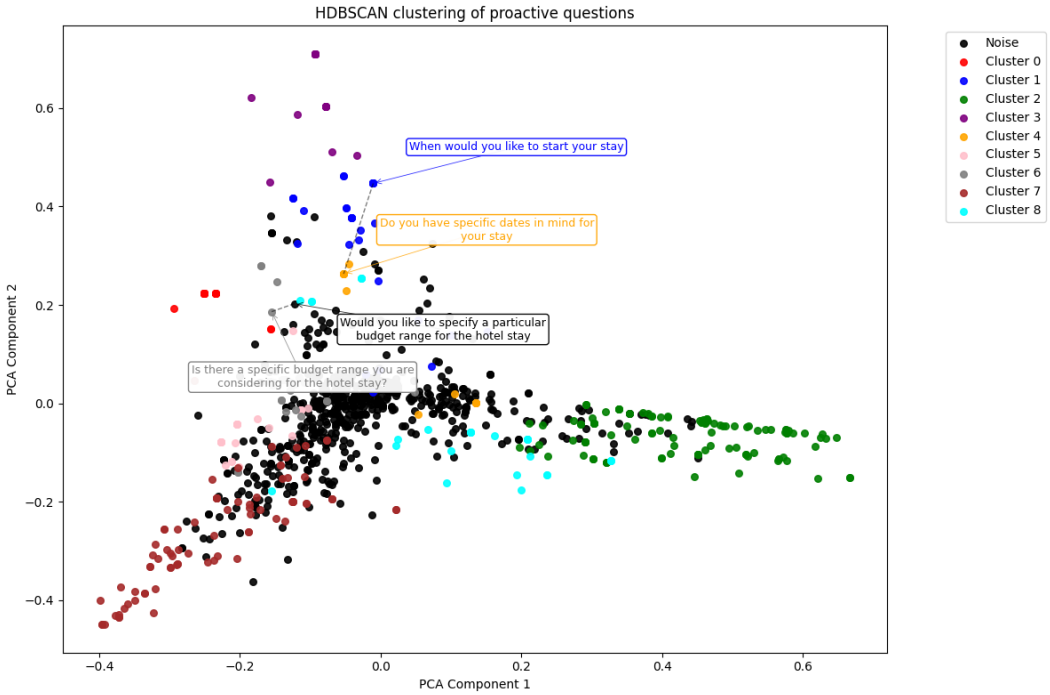

Figure 4: HDBSCAN clusters of proactive questions generated

proactively elicited by the agent, rather than voluntarily provided by the user. Interestingly, we see a sharp increase in proactive agent coverage around turns 13–14, corresponding to the stage when the agent systematically elicits lower-priority slots such as contact preferences or amenities after completing the high-priority ones. Intuitively, the proactive agent achieves higher coverage because it does not rely on the user to volunteer information, but instead leverages its learned slot ontology to proactively fill gaps. For clarity, throughout the paper we refer to the GPT-4o model without the proactive module as the baseline agent, our augmented system as the proactive agent, and GPT-4o itself as the base LLM powering both.

I Related work

Task Oriented Dialogue Traditional task-oriented dialogue (TOD) systems have been primarily designed to achieve goal completion through database-backed tasks, utilizing a fixed ontology of slots to fulfill user requests. Recently, the focus has shifted towards proactive agents that actively drive the dialogue forward, rather than waiting passively for user input. For instance, ProTOD [6] introduces an LLM-based TOD framework that incorporates knowledge retrieval and two-stage planning to transition from passive to proactive conversations. Similarly, ProMISe [18] offers a proactive multi-turn dataset centered on resolving information-seeking intents. Other frameworks have explored elicitation and initiative in more specific contexts: URef + SaRSNet [24] plan requirement sequences and detect when user requirements are satisfied, while PUS [21] adapts elicitation to personalized utterance styles. Mixed-initiative datasets like TITAN [19] encourage agents to interleave proactive follow-ups with user-driven dialogue. These methods underscore the growing recognition that TOD agents should not only complete tasks but also proactively elicit user requirements.

In parallel, research on slot schema induction aims to automatically discover the information an agent should elicit. GenDSI [7] frames slot induction as a generative task, incrementally generating slot–value pairs from dialogues. More recent work extends this into streaming schema induction [8], while unsupervised clustering approaches [22] extract latent slot structures from raw conversations. However, these efforts primarily focus on identifying slots or schema structures, without addressing how such slots should be operationalized as proactive agent actions.

Our approach complements these lines of work by integrating missed-opportunity detection, reverse question generation, and hierarchical clustering into slots and sub-slots with priorities, subsequently incorporating them into the agent's action space. This enables agents to adapt their elicitation strategies over time, grounded in actual user dialogue patterns rather than relying on static templates or schema-only extraction.

Supervised Learning for Proactivity The development of proactive conversational agents has prominently focused on the ability to ask follow-up questions to clarify a user's needs. Early research in conversational search laid the groundwork for this task by formulating the concept of asking clarifying questions for ambiguous queries. This led to the creation of datasets like Qulac developed [1], which were developed through crowd-sourced question-answer pairs. These datasets are instrumental in training proactive agents to pose a single clarifying question, such as "What exactly are you looking for?" when a user's query lacks specificity. For example, [1] introduced a clarification question ranker that significantly enhanced search results by strategically inserting a well-chosen question before providing an answer.

Subsequent approaches have been proposed to generate or select such questions based on the current query and context ([25], [23], [13]). However, these methods often require substantial manual effort to design or collect clarification questions and typically handle proactivity in a single-turn manner, where the agent asks one question and then reverts to a reactive mode. Recent surveys [[16], [5]], highlight this limitation, categorizing proactive information-seeking behaviors into one-off clarification questions versus more sustained preference elicitation dialogues. Our work addresses this gap by enabling multi-turn proactivity, allowing the agent to continuously take initiative throughout the conversation, rather than being limited to a single prompt.

Reinforcement Learning for Proactive Agents Many existing LLMs utilize reinforcement learning (RL) fine-tuning, where a reward model is derived from feedback. While effective, a notable limitation of RL fine-tuning is that LLMs are typically optimized to maximize the learned reward model within a single-step response, rather than across a multi-step dialogue. [10] propose an information-seeking agent, yet focus on a single-step objective centered on maximizing helpfulness, without considering or evaluating tasks that require information gathering to achieve long-term goals.

Recent advancements have shifted from traditional supervised learning to RL to train agents capable of proactively driving conversations over multiple turns. For instance, [18] introduced CollabLLM, a framework that simulates multi-turn interactions and optimizes a large language model (LLM) using multi-turn-aware rewards. By fine-tuning on these long-horizon rewards, CollabLLM transcends mere response generation, actively uncovering user intent and offering suggestions throughout the dialogue.

In domains such as conversational recommendation, researchers have similarly approached proactive questioning as a sequential decision-making problem. UNICORN, developed by [3], exemplifies this approach by requiring the agent to decide on a series of questions to elicit user preferences, rather than crafting one question at a time. While [26] trained a model to predict the next question for preference elicitation in recommendation dialogues, it treated each turn in isolation, lacking a holistic view of the conversation.

A novel approach to online RL is presented in the work by [12] which leverages LLMs to simulate suboptimal but human-like behaviors in goal-directed dialogue tasks. By generating diverse synthetic rollouts of hypothetical human-human interactions, this method uses offline RL to train conversational agents that optimize goal-directed objectives over multiple turns. This approach effectively combines LLM-generated examples with RL to achieve promising results in tasks such as teaching and preference elicitation, however the work still requires human intervention in the form of task-specific prompts, limiting the scope of the work.

Offline RL techniques have garnered attention for their potential in dialogue systems, necessitating a static dataset of dialogues. [17] introduced an offline RL algorithm aimed at facilitating goal-directed dialogues through negotiations, utilizing a dataset of human-to-human conversations. Despite its promise, offline RL's effectiveness over supervised learning hinges on the meticulous curation of datasets to enhance properties such as coverage and diversity ([9]; [11], [14]). This requirement poses challenges to its practicality, as achieving optimal dataset characteristics can be resource-intensive and complex.

These RL-based agents have shown enhanced success in task completion by exploring proactive strategies, such as determining when to ask for additional preferences or clarifications, through trial-and-error. However, purely exploratory learning can be sample-inefficient and often necessitates carefully designed reward signals to guide the agent's learning process effectively.

NeurIPS Paper Checklist

The checklist is designed to encourage best practices for responsible machine learning research, addressing issues of reproducibility, transparency, research ethics, and societal impact. Do not remove the checklist: **The papers not including the checklist will be desk rejected.** The checklist should follow the references and follow the (optional) supplemental material. The checklist does NOT count towards the page limit.

Please read the checklist guidelines carefully for information on how to answer these questions. For each question in the checklist:

- You should answer [Yes] , [No] , or [NA] .
- [NA] means either that the question is Not Applicable for that particular paper or the relevant information is Not Available.
- Please provide a short (1–2 sentence) justification right after your answer (even for NA).

The checklist answers are an integral part of your paper submission. They are visible to the reviewers, area chairs, senior area chairs, and ethics reviewers. You will be asked to also include it (after eventual revisions) with the final version of your paper, and its final version will be published with the paper.

The reviewers of your paper will be asked to use the checklist as one of the factors in their evaluation. While "[Yes] " is generally preferable to "[No] ", it is perfectly acceptable to answer "[No] " provided a proper justification is given (e.g., "error bars are not reported because it would be too computationally expensive" or "we were unable to find the license for the dataset we used"). In general, answering "[No] " or "[NA] " is not grounds for rejection. While the questions are phrased in a binary way, we acknowledge that the true answer is often more nuanced, so please just use your best judgment and write a justification to elaborate. All supporting evidence can appear either in the main paper or the supplemental material, provided in appendix. If you answer [Yes] to a question, in the justification please point to the section(s) where related material for the question can be found.

IMPORTANT, please:

- **Delete this instruction block, but keep the section heading "NeurIPS Paper Checklist"**,
- **Keep the checklist subsection headings, questions/answers and guidelines below.**
- **Do not modify the questions and only use the provided macros for your answers**.

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The main claims in the abstract and introduction accurately reflect the paper's contributions, such as leveraging missed opportunities for learning proactive agent behaviors and demonstrating improvements in recall, precision, and early slot coverage.

   Guidelines:

   - The answer NA means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

Justification: The paper discusses limitations related to the reliance on LLMs for detecting missed opportunities and the need for further integration with database-backed TOD systems for end-to-end task success.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [NA]

   Justification: No theoretical results are presented in the paper, as the focus is on empirical validation of the proposed framework for proactive agent behaviors in task-oriented dialogues.

   Guidelines:

   - The answer NA means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

Justification: All prompts are disclosed to reproduce the agent setup. Data preparation is discussed, and the paper makes use of an open-source benchmark, MultiWOZ 2.4, ensuring that the main experimental results can be reproduced to support the paper's claims and conclusions.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [No]

Justification: The code is not provided for open access, but the paper uses an open-source benchmark, MultiWOZ 2.4, and provides all necessary instructions and details for data preparation and agent setup to faithfully reproduce the main experimental results.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so "No" is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://nips.cc/public/guides/CodeSubmissionPolicy`) for more details.

- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: Full details regarding the training and test settings, including data splits, hyperparameters, and other relevant experimental configurations, are provided in the main text and appendix, ensuring that the results can be understood and replicated.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   Answer: [Yes]

   Justification: The paper reports precision, recall, F1-score, and coverage metrics with standard deviations, providing information about the statistical significance of the experiments.

   Guidelines:

   - The answer NA means that the paper does not include experiments.
   - The authors should answer "Yes" if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
   - If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [Yes]

Justification: Sufficient information on the computer resources, including the type of compute workers, memory, and time of execution needed to reproduce the experiments, is presented in Appendix F.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

Answer: [Yes]

Justification: The research conducted in the paper conforms to the NeurIPS Code of Ethics, focusing on improving dialogue systems without any ethical concerns.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [NA]

Justification: The paper presents a methodology for proactive agent behaviors in task-oriented dialogues, focusing on technical advancements. Discussion of potential positive or negative societal impacts, such as malicious or unintended use, is beyond the scope of the paper.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.

- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

    Answer: [NA]

    Justification: The paper focuses on improving proactive agent behaviors in task-oriented dialogues and does not involve the release of data or models that have a high risk for misuse, such as pretrained language models or scraped datasets.

    Guidelines:

    - The answer NA means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: Provided in appendix F.

    Guidelines:

    - The answer NA means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: No new assets are introduced or released in the paper.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: No crowdsourcing or research with human subjects is performed in the study.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: No research involving human subjects is performed, as indicated in the previous answer regarding crowdsourcing and human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does not impact the core methodology, scientific rigorousness, or originality of the research, declaration is not required.

Answer: [Yes]

Justification: The methodology uses an LLM-based agent, and the LLM is critical for the core methods in this research.

Guidelines:

- The answer NA means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
- Please refer to our LLM policy (`https://neurips.cc/Conferences/2025/LLM`) for what should or should not be described.