AutoEval-ToD: Automated Evaluation of Task-oriented Dialog Systems

Arihant Jain, Purav Aggarwal, Rishav Sahay, Chaosheng Dong, Anoop Saladi
Amazon
{arihanta, aggap, rissahay, chaosd, saladias}@amazon.com

Abstract

Task-oriented Dialog systems (ToD) are essential in automating user interactions, but their complex design and dynamic nature make evaluation particularly challenging. Current evaluation methodologies heavily depend on human annotators, which can be inefficient, subjective, and expensive to scale. To advance the field, there is a pressing need for a reliable, scalable, and systematic evaluation framework that can provide comprehensive insights into ToD system performance. In this paper, we propose, AutoEval-TOD, an automated end-to-end evaluation framework using large language models (LLMs). Our framework first interacts with the ToD system and then assesses its performance across key dimensions by analyzing both the ToD's responses and internal states. We validate our approach by applying it to multiple ToD systems, highlighting its adaptability and potential for widespread use in both research and industrial settings.

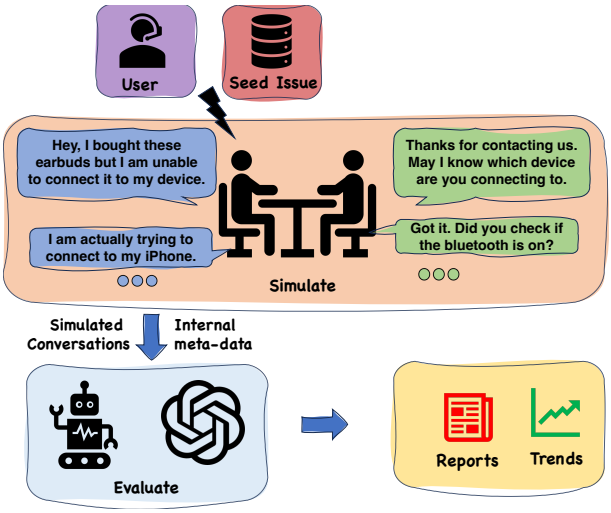

Figure 1: The proposed framework's overall workflow involves a user simulator, initialized with an issue from the knowledge store, interacting with the ToD system. It logs both conversations and meta-data to generate a comprehensive report and trending plots.

1 Introduction

Task-oriented Dialog systems (ToD) play a pivotal role in various industries, automating interactions across domains like customer service, technical support, and recommendation. These systems are designed to understand user input, manage multi-turn conversations, and ultimately fulfill specific tasks, such as booking a flight or troubleshooting an issue. Despite their widespread adoption, evaluating ToD systems remains a significant challenge, owing to their complexity and the broad range of performance metrics that need to be considered.

Evaluating a ToD system is far more intricate than simply checking whether a task is completed. It requires assessing several dimensions, such as response quality, dialogue coherence, retrieval accuracy, user satisfaction, and efficiency. Moreover, many of these metrics are subjective, adding further complexity to evaluation processes. As a result, most companies and research teams rely on human annotators to evaluate ToD systems. This reliance on human evaluation introduces several limitations. First, the subjective nature of human assessment can lead to variability in results, reducing the reliability of performance benchmarks. Second, manual evaluation does not scale well as the volume of interactions increases, making it unfeasible for large-scale deployments. Finally, human evaluators may introduce unconscious biases, potentially skewing results based on factors such as cultural background or personal preferences.

To address these limitations, we propose an automated evaluation framework for ToD systems that leverages the power of large language models (LLMs). As illustrated in Figure 1, the framework operates in three key stages: it first interacts with the ToD system to simulate real user scenarios, and then it evaluates the system across multiple dimensions by analyzing both its responses and

Criterion	RAGAs	RAGChecker	RADDLE	BotSIM	GOD-Eval	AutoEval-ToD
Retrieval	✗	✓	✗	✗	✗	✓
User Experience	✗	✗	✗	✓	✓	✓
Turn Optimization	✗	✗	✓	✓	✓	✓
Domain Compliance	✗	✗	✗	✗	✗	✓
Response Quality	✓	✓	✗	✗	✗	✓
Multi-lingual	✗	✗	✗	✗	✗	✓
Unsupervised	✓	✓	✗	✗	✗	✓
Dynamic User	✗	✗	✓	✓	✓	✓

Table 1: Comparison of recent RAG and ToD evaluation frameworks across multiple criteria, highlighting the features and capabilities of AutoEval-ToD relative to other recent approaches. Features are marked as not available (✗), limited availability and applicability (✓), or fully available and easily applicable (✓).

internal decision-making processes to finally generate valuable insights in the form of reports. This end-to-end approach offers several advantages: it is scalable, provides consistent and objective evaluations, and can be applied to various ToD systems with minimal modification. Our framework not only streamlines the evaluation process but also enhances the reliability of performance insights, allowing for more effective system development and monitoring. We make the following key contributions to the evaluation of ToD systems:

- A **generic framework** exposed as easy-to-use APIs, enabling seamless integration with any ToD system for evaluation and tracking.
- A **user simulator** that can simulate real-world scenarios and generate corresponding metrics.
- A **comprehensive evaluation** that goes beyond traditional metrics, providing insights into both the technical performance and practical usability of ToD systems.

2 Related Work

Building a ToD system is a very crucial application. Recent developments in building ToD leverage Large Language Models (LLMs) to enhance ToD capabilities. For instance, (Xu et al., 2024a) proposed a single-prompt technique for building entire ToD systems, while (Li et al., 2024) explored function and tool calling capabilities of LLMs for ToD construction. While frameworks for creating ToD systems have become more accessible, their evaluation remains challenging due to the complexity of their components.

2.1 Limitation of Existing Evaluations

Current ToD evaluation approaches can be categorized into three main types: (1) Human-in-the-loop evaluation, (2) Evaluations requiring labeled data, and (3) Unsupervised automatic evaluations. Each approach has its strengths and limitations.

Human-in-the-loop evaluation methods, such as DQA (Komma et al., 2023) and BOTEVAL (Cho et al., 2024), offer detailed insights into ToD system performance. These approaches also provide comprehensive assessments of overall goal achievement. However, they are time-consuming, difficult to scale, and often lack component-level metrics. Furthermore, the reliance on human evaluators also introduces potential inconsistencies and subjectivity in the evaluation process.

Evaluations requiring labeled data, like RADDLE (Peng et al., 2021), BotSIM (Wang et al., 2022), and GOD-Eval (Takanobu et al., 2020), provide consistent benchmarks for ToD systems. These approaches use static variations and paraphrasing to judge system performance. While they offer reproducibility, they may lack flexibility in handling dynamic dialogue flows. Additionally, their reliance on pre-defined scenarios may lead to dialogue inconsistencies when chat flows deviate from expected patterns.

Unsupervised automatic evaluations represent a promising but underexplored area in ToD system assessment. Most work in this category focuses on evaluating the generation component of Retrieval-Augmented Generation (RAG) (Lewis et al., 2020) systems. Tools like RAGAS (Es et al., 2024) and RAGChecker (Ru et al., 2024) aim to automatically evaluate RAG performance. However, these approaches often neglect the crucial retrieval component of ToD systems and may not provide a comprehensive end-to-end evaluation.

Table 1 highlights key differences between our proposed approach, AutoEval-ToD, and recent RAG and ToD-based evaluation methods. Existing works often fail to address critical aspects such

as retrieval performance, domain compliance, response quality, and multi-lingual support. Moreover, many approaches do not consider the evaluation of chatbot outputs or do so only partially. Our work, AutoEval-ToD, aims to address these shortcomings by providing an end-to-end, unsupervised evaluation framework that considers multiple aspects of ToD system performance, including retrieval, domain compliance, and multi-lingual support which we describe in Section 3.

3 Proposed Evaluation

AutoEval-ToD is designed to provide a holistic evaluation by focusing on 5 critical dimensions essential for effective ToD development: (1) Retrieval Performance, (2) User Experience, (3) Turn Optimization, (4) Domain Compliance, and (5) Response Quality. These aspects were carefully selected to cover the full spectrum of capabilities required for ToD to function effectively in real-world scenarios. AutoEval-ToD provides a comprehensive evaluation that goes beyond traditional metrics, offering insights into both the technical performance and practical usability of ToD.

3.1 Retrieval Performance

ToD systems are designed to assist users in accomplishing specific tasks by retrieving relevant information from a knowledge store. Therefore, it is crucial to evaluate the system's ability to retrieve the right content. Previous works either evaluate the retriever module outside the context of a conversation or are limited to a few standard metrics such as Inform metric (Xu et al., 2024a). With AutoEval-ToD, since we simulate user conversations and track the entire list of retrieved contents, we are able to overcome the aforementioned limitations. We propose to measure retriever performance at specific turns in the conversation there by maintaining the context of a conversation and tracking additional critical metrics such as Hit-Rate@k and Mean Reciprocal Rank (MRR@k) (Zangerle and Bauer, 2022) for $k \in \{1, 3, 5, 10, 20\}$. These metrics take into account the position at which the correct recommendation is present, providing a more nuanced evaluation of retrieval performance. While Hit-Rate@k measures the proportion of times the correct item is retrieved within the top k results, MRR@k considers the reciprocal of the rank at which the first relevant item is found, averaged over all queries.

Additionally, to assess the robustness of the retriever module in handling ambiguous user utterances, we curate user inputs into three levels of difficulty: Easy (standard queries), Medium (slightly ambiguous queries), and Hard (highly ambiguous or complex queries). These variations simulate different levels of user ambiguity when interacting with the ToD system. Examples of these variations are shown in Table 6 in Appendix K. To generate these variations, we employed an LLM using a prompting technique, as illustrated in Prompt H.1. The LLM was given the original user input as a query and asked to generate variations with increasing levels of ambiguity. We then measure the retriever performance on each of these variations to evaluate its robustness and ability to handle ambiguous inputs. This approach allows us to assess how well the retriever performs under different levels of query complexity, providing insights into its real-world applicability and areas for improvement.

3.2 User Experience

To ensure that ToD systems meet user needs effectively, it is crucial to evaluate them using metrics that directly impact the end-user experience. This section proposes a set of user-centric metrics for comprehensive ToD system evaluation.

Latency It is a critical aspect of ToD system performance that directly affects user satisfaction. The framework enables measuring latency both end to end at the turn level and for individual modules. These measurements provide insights into the system's responsiveness and help identify bottlenecks in specific modules that require optimization. We measured 50th (P50) and 90th (P90) percentile latency in seconds for the ToDs.

Goal Completion Rate (GCR) Measuring the percentage of times the user's goal is met is crucial for assessing the ToD system's effectiveness. Unlike previous works that used metrics like Joint Goal Accuracy and Success (Li et al., 2024; Xu et al., 2024a), which rely on exact match criteria, we propose a more flexible approach:

- Randomly select a solution that satisfies the user goal at the start of the evaluation.
- Compare the ToD system's recommendation to the selected solution using a separate LLM based solution checker task that does the similarity assessment (Prompt H.2).
- Calculate the percentage of conversations that

lead to a satisfactory resolution.

The LLM-based assessment uses a prompt structure as shown in Prompt H.2 to determine if two recommendations are equivalent. The same module is also used to guide the user simulator to accept the recommended solutions or probe further.

Average # of Turns to Resolution (#Res) This metric measures the efficiency of the ToD system in reaching a satisfactory resolution: An effective ToD system should aim to maximize the percentage of resolved conversations while minimizing the average number of turns.

These user-centric metrics provide a comprehensive view of the ToD system's performance from the user's perspective. When evaluating and optimizing ToD systems, developers should consider these metrics collectively to ensure a balance between efficiency, accuracy, and user satisfaction. Regular measurement and analysis of these metrics during system development can guide improvements and help identify the most suitable version for production deployment.

3.3 Turn Optimization

ToD systems not only optimise for serving the most relevant match for the user's query but also doing it in minimum numbers of turns. Progressively tracking turn level metrics is thus very critical to provide valuable insights to the developer. We propose the following evaluation metrics that track the performance of a ToD system at each turn of the conversation.

- **Progressive HitRate@k**: measures the proportion of conversations where the correct item appears in the top k recommendations at each turn.
- **Progressive MRR@k**: evaluates the average position of the correct item within the top k recommendations at each turn.
- **Turn level Resolution Rate**: measures the proportion of conversations resolved at each turn. It provides insights into the efficiency of the ToD system, as we aim to resolve conversations in as few turns as possible.

3.4 Domain Compliance

ToD systems must adhere to domain-specific rules and constraints to ensure user safety and legal compliance of service. Domain compliance is crucial as it prevents the ToD system from providing irrelevant or potentially unwanted responses. The specific rules vary depending on the domain and task. For examples: (1) In a troubleshooting chatbot domain, the system should not recommend solutions involving harmful chemicals or unnecessary tool purchases. (2) In a hotel recommendation domain, the system should avoid requesting sensitive information (e.g., credit card details) or making disparaging comments about hotels.

To automate the evaluation of domain compliance, we propose a method utilizing LLMs. The process involves:

1. Inputting domain-specific rules and the ToD system's response into the LLM.
2. Prompting the LLM to assess whether the response adheres to the domain-specific rules (Prompt H.3).
3. In cases of non-compliance, the LLM generates explanations to guide improvements.

3.5 Response Quality

While domain-specific rules are crucial for evaluating ToD systems, a set of universal quality metrics is equally important to ensure a superior user experience. These metrics provide a holistic assessment of the system's performance beyond task completion. To conduct an end-to-end analysis, we employ the following comprehensive quality metrics:

- **Factual Correctness (FC)**: The accuracy of response provided by the ToD and the retrieved content.
- **Response Completeness (RC)**: The extent to which the ToD's response addresses all aspects from the given retrieved content.
- **Entailment (ET)**: The logical consistency between the ToD's responses and the conversation so far.
- **Biasness (BN)**: The presence or absence of unfair prejudice in the ToD's responses.
- **Empathetic Tone (Emp)**: The ToD's ability to respond with appropriate emotional understanding.
- **Language Correctness (LC)**: The grammatical and syntactical accuracy of the target language of ToD's responses.

Like in the previous section, to evaluate these metrics, we employed a LLM approach using prompts mentioned in Appendix J.

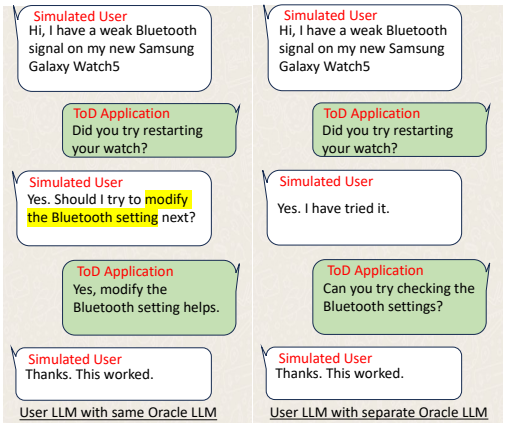

Figure 2: Comparison of solution leakage in two scenarios. **Left**: When the user LLM is provided with the oracle solution, it directly reveals the exact solution (*Modify the Bluetooth setting*) that resolves the issue. **Right**: When the user LLM consults the oracle LLM about the solution, it maintains confidentiality by not disclosing the specific solution.

4 Dynamic User Simulator

To comprehensively evaluate ToD systems, we utilized a LLM-based conversation simulator by building upon the framework introduced by Xu et al. (2024a). It is designed to interact with the ToD system while capturing both conversation logs and backend metadata for detailed analysis. This setup enhances the realism of our evaluations by incorporating dynamic conversations that simulate real-world interactions.

In this simulation, a user LLM is tasked with emulating a user (via Prompt I.1) who has purchased a product and is now experiencing a seeded issue, drawn from the knowledge store, that requires resolution. The user LLM engages naturally with the ToD system, seeking solutions while the conversation and accompanying metadata are saved for evaluation. Example of one such conversation is shown in Figure 6 in Appendix K.

Additionally, an oracle LLM (with prompt H.2) plays a crucial role in ensuring unbiased evaluation. The oracle is assigned an 'oracle solution'—the correct resolution to the issue the user is facing. When the ToD system recommends a solution, the user LLM consults the oracle LLM to verify whether the recommended solution aligns with the oracle's. This mechanism prevents the user LLM from unintentionally leaking hints or solution details to the ToD system, as shown in Figure 2, ensuring a fair evaluation.

While the conversations are used for end-to-end evaluations based on both manual and automated metrics, the backend metadata helps gather specific data points for measuring the performance of individual modules. This combination provides a robust framework for evaluating the ToD system's capabilities, ensuring consistency and adaptability to various user LLM interactions. While our focus here has been on the Troubleshooting domain in this section, it's important to note that this framework is highly versatile and can be readily adapted to other domains with minimal modifications, underscoring its broad applicability in various conversational AI contexts.

5 Implementation Details

To enable seamless integration, the framework is designed as a library that can be easily imported and used to evaluate any ToD system. The system developer can configure the ToD endpoint and database details, allowing the dynamic user simulator to select a seed issue (and its corresponding oracle solution) and trigger the ToD endpoint with simulated user messages. The framework automatically handles high-level evaluation criteria, such as User Experience, Domain Compliance, and Response Quality. For more granular metrics like Retrieval Performance and Turn Optimization, developers can activate specific functions within the ToD system. The framework also offers flexible API access, enabling developers to log custom data and monitor system behavior as needed. This API-driven approach allows for tailored logging to meet the unique requirements of any ToD system under evaluation.

After completing the simulation, the framework generates a detailed report that tracks key performance indicators, offering insights into the system's strengths and areas for improvement. Figure 3 illustrates how the framework integrates with a ToD application via the *GetSeedIssue* and *GetResponse* APIs. Developers of the ToD application can utilize the exposed *Eval** API calls to enable the framework to gather data and generate insights.

6 Experimental Setup

To rigorously assess the effectiveness and versatility of our AutoEval-ToD approach, we experiment

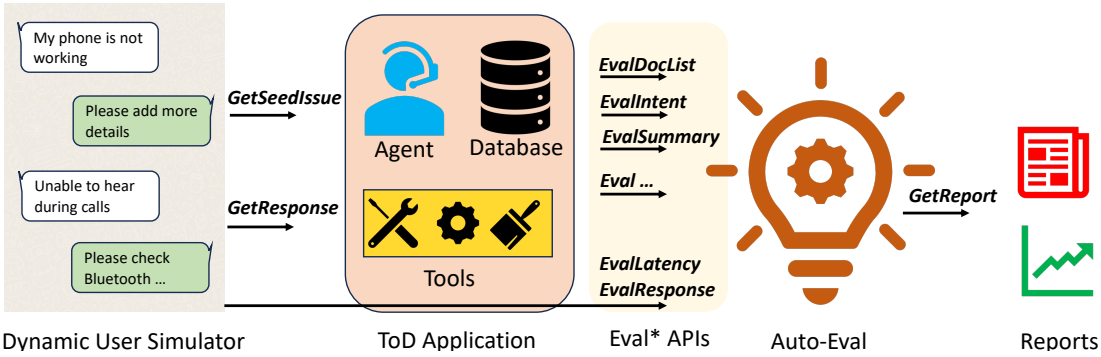

Figure 3: Low level implementation details of the proposed framework. The dynamic user simulator and the API calls (*Eval**) invoked by the developer enable the collection of data points for AutoEval-ToD to generate the reports and metrics.

Variation	$k = 1$		$k = 3$		$k = 5$		$k = 10$		$k = 20$	
	HitRate	**MRR**	**HitRate**	**MRR**	**Hit Rate**	**MRR**	**HitRate**	**MRR**	**HitRate**	**MRR**
Easy	98.5%	98.5%	99.8%	99.1%	99.9%	99.1%	99.9%	99.1%	99.9%	99.1%
Medium	85.9%	85.9%	94.5%	89.8%	96.5%	90.3%	98.1%	90.5%	99.1%	90.5%
Hard	62.9%	62.9%	76.8%	69.1%	81.7%	70.2%	87.7%	71.0%	92.6%	71.3%

Table 2: Retrieval Performance Analysis Across User Input Variations

across 3 distinct ToD systems, each paired with different datasets and domains. This setup allows us to evaluate AutoEval-ToD's performance in industrial applications, multiple domains, and multilingual contexts. We assess each setup using the evaluation strategy proposed in Section 3.

- **Troubleshooting ToD**: This industrial chatbot, built on the Claude-3-Sonnet model (Anthropic, 2023), aims to resolve user issues related to products purchased from an e-commerce store. Due to confidentiality constraints, we cannot disclose the exact details of this task-oriented dialogue (ToD) system and thus present relative performance metrics compared to the vanilla Claude-3-Sonnet model. We evaluate it to showcase the applicability of our framework on real-world chatbots.

- **MultiWOZ + FnCTOD**: We integrate the in-context prompting-based FnCTOD architecture (Li et al., 2024), which utilizes the LLaMA2-13B-chat model (Touvron et al., 2023), with the popular MultiWOZ dataset (Budzianowski et al., 2018). This ToD system covers four major recommendation domains: restaurant, attraction, hotel, and train.

- **Multi3WOZ + AutoTOD**: We incorporate the AutoTOD framework (Xu et al., 2024a) with the recent Multi3WOZ dataset (Hu et al., 2023), which consists of data in Arabic, Turkish, and French. It also utilizes LLaMA2-13B-chat model as underlying model.

We utilize the Claude-3-Sonnet model for all LLM-based evaluations in our framework. To validate LLM performance, we employ a diverse group of human annotators. Annotators are provided with detailed task-specific guidelines and asked to annotate a sample input for each task in the framework.

7 Experimental Results

7.1 Retrieval Performance

Table 2 presents the retrieval performance results for different levels of user input variation (easy, medium, hard) across various k values used in Troubleshooting ToD. We observed that as user input becomes more ambiguous (progressing from Easy to Hard variation), the overall retriever performance consistently decreases across all k values. This trend is evident in both HitRate and MRR metrics.

For Easy variations, the retriever demonstrates excellent performance, achieving near-perfect HitRates and high MRR. Medium variations show a notable drop, particularly at $k = 1$, but performance improves significantly as k increases. Hard variations present the most significant challenge, with HitRate dropping significantly.

ToD	Domain	GCR	Latency P50	Latency P90	#Res
Troubleshooting ToD		+0.19	+2.5	+5.1	-2.1
FnCTOD	restaurant	0.30	1.7	4.7	4.4
	attraction	0.54	1.7	4.6	6.9
	hotel	0.40	1.6	4.2	7.5
	train	0.42	1.9	4.7	7.2

Table 3: User Experience metrics on different ToDs.

Key takeaway: *ToD developers should evaluate retriever performance for different query variations as user input greatly affects its quality. Also measuring the impact of the retriever on the eventual User Experience metrics can help developers working backwards from the user in a complex processing pipeline of a ToD system.*

7.2 User Experience

Table 3 presents various metrics crucial for evaluating the user experience of ToD systems across different datasets. We observe that the Troubleshooting ToD system shows a 19% higher GCR compared to vanilla Claude-3-Sonnet model, while the FnCTOD system's GCR varies across domains, with the attraction domain performing best (0.54). This suggests that domain-specific optimization may be beneficial for improving overall system performance. Furthermore, the Troubleshooting ToD system exhibits higher latency, indicating potential areas for improvement in processing speed.

Interestingly, #Res varies significantly across domains in the FnCTOD system, with hotel and train domains requiring more turns on average (7.5 and 7.2, respectively). This could indicate the complexity of these tasks or potential areas for streamlining the dialogue flow.

Key takeaway: *ToD developers should focus on user experience metrics like GCR, latency, and turns to resolution. These provide a comprehensive view of system performance, required for improving real-world ToDs and user satisfaction iteratively.*

7.3 Turn Optimization

Figure 4 shows how the HitRate@k and MRR@k of the ToD's retriever is improving at each turn on MultiWOZ dataset with FnCTOD. This showcases that with each turn ToD is able to better refine its search with more user information as user ambiguity is decreasing with each turn, which aligns with the principle of ToD that with each turn user ambiguity should go down and ToD should be able to provide better solution. This also highlights the importance of tracking the turn-level metrics which could further help improve the ToD system. Figure 5 tracks the turn level resolution rate, the count of conversations that get resolved with each turn. Expectedly, as the number of turns increases, more number of conversations get resolved.

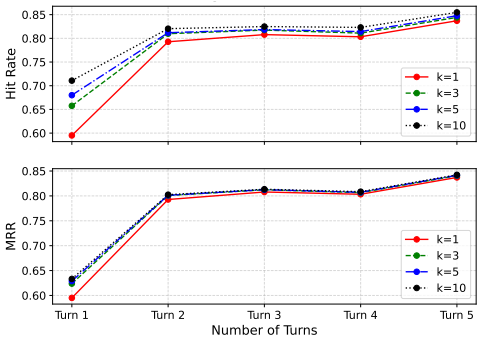

Figure 4: Average progressive metrics on MultiWOZ + FnCTOD across domains

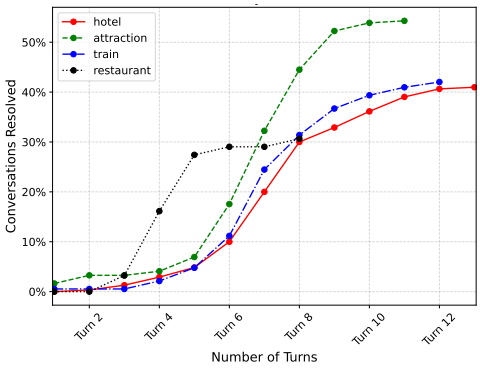

Figure 5: Cumulative Conversation resolved by Turns by the MultiWOZ + FnCTOD across domains

Key takeaway: *ToD developers should optimize for fewer conversation turns while improving retrieval accuracy, as this reveals system efficiency in reducing user ambiguity.*

ToD	Domain	#1	#2	#3	#4
Troubleshooting ToD		+4%	+5%	+6%	+6%
FnCTOD	restaurant	99%	97%	100%	100%
	train	99%	88%	97%	98%
	hotel	99%	98%	96%	100%
	attraction	100%	95%	99%	96%

Table 4: Domain Compliance Metrics: Percentage adherence to domain-specific rules across different ToD domains.

ToD	Domain	Language	FC	RC	ET	BN	Emp	LC
Troubleshooting ToD	-	English	+4.9%	+4.0%	+0.8%	+0.0%	+5.0%	+5.0%
MultiWOZ + FnCTOD	restaurant	English	-	-	83.0%	2.4%	91.7%	68.3%
	train	English	-	-	82.0%	0.0%	95.0%	83.0%
	hotel	English	-	-	77.0%	0.0%	85.0%	78.0%
	attraction	English	-	-	84.0%	0.0%	94.0%	82.0%
Multi3WOZ + AutoTOD	-	Arabic	-	-	81.0%	1.1%	92.3%	81.2%
	-	Turkish	-	-	87.0%	0.0%	85.0%	82.0%
	-	French	-	-	85.0%	0.7%	82.0%	88.0%

Table 5: Evaluation of response quality metrics for various ToD systems across different domains and languages.

7.4 Domain Compliance

Table 4 reports the results of the 4 domain-specific rules on the output responses of the ToD system. The specific rules for each ToD is present in Table 7 in Appendix K. Notably, the Troubleshooting ToD demonstrates superior compliance with task-specific domain rules compared to the vanilla Claude-3-Sonnet model. Instances of rule violations (exemplified in Appendix G) serve as valuable data points for targeted improvements in the ToD system, ensuring better alignment with domain-specific requirements and user expectations.

Key takeaway: *ToD developers should evaluate and aim for high adherence to domain-specific rules across all domains, as this evaluation ensures domain compliance, eliminating PR risks, and identifying areas for improvement before deployment.*

7.5 Response Quality

Table 5 presents the response quality metrics for various ToD systems across different domains and languages. The Troubleshooting ToD demonstrates performance lift across most metrics, particularly in FC, Emp, and LC compared to vanilla Claude-3-Sonnet model. However, its low ET score lift suggests room for improvement in response relevance. Its high performance in empathy is particularly noteworthy for troubleshooting scenarios where user frustration may be common.

MultiWOZ + FnCTOD results reveal consistent performance across domains. Entailment scores range from 77.0% to 84.0%, indicating good response relevance. The system shows high empathy (85.0-95.0%) but variable language consistency (68.3-83.0%). Notably, biasness is near-perfect in most domains, with only the restaurant domain showing a 2.4% biased output. The Multi3WOZ + AutoTOD system demonstrates the potential for cross-lingual ToD, with comparable performance across Arabic, Turkish, and French. Entailment scores (81.0-87.0%) and empathy scores (82.0-92.3%) are consistent across languages, suggesting effective transfer of these qualities in multilingual settings. We did not evaluate MultiWOZ-based ToDs on FC and RC as the retrieved content is not directly used for response generation.

Key takeaway: *ToD developers should prioritize evaluation across multiple quality metrics in diverse domains and languages to ensure effective, user-friendly, and culturally appropriate systems.*

7.6 Human Evaluation

To validate our automated LLM evaluation approaches, we conducted a comparative study between LLM-based evaluations and human assessments across three tasks: Solution Checker, Domain Compliance, and Response Quality utilizing prompts in section H.2, H.3, and J respectively. We calculated the accuracy between the LLM-based evaluations and human assessments for each task. The results demonstrated high overall accuracy of 97% for solution checker task, 96% for domain compliance, and 94% for response quality task. The high accuracy numbers underscore the strong alignment between LLM-based evaluations and human judgment, supporting the potential for fully automating the evaluation process without compromising on reliability.

Key takeaway: *LLM-based evaluations closely match human assessments suggesting that ToD developers should adopt automated LLM evaluation methods to streamline processes, potentially reducing costs and time while maintaining accuracy.*

8 Conclusion

We introduce AutoEval-ToD, a comprehensive automated evaluation framework for task-oriented dialogue (ToD) systems. Leveraging LLMs, our approach enables scalable, consistent, and multi-faceted assessment of ToD performance across key

dimensions. By utilizing user and oracle LLMs to simulate real-world conversations, AutoEval-ToD provides nuanced insights into system behavior. Experiments on multiple ToD systems and datasets demonstrate its versatility and effectiveness, with strong alignment between automated evaluations and human judgments. AutoEval-ToD offers a standardized, reproducible evaluation methodology that can accelerate progress in ToD research and development, facilitating rapid iteration and improvement of these critical systems.

9 Limitations

While AutoEval-ToD offers significant advantages over existing evaluation approaches, it is important to acknowledge some limitations:

1. **LLM Dependency**: The framework's performance is contingent on the capabilities of the underlying LLM used for evaluations. Biases or limitations in the LLM could potentially impact the evaluation results. However, ToD developers can leverage multiple state-of-the-art LLMs for evaluation, potentially mitigating individual model biases and inconsistencies through ensemble approaches or cross-validation techniques.

2. **Computational Resources**: Running extensive LLM-based evaluations may require substantial computational resources, which could be a constraint for some researchers or organizations.

3. **Dynamic Nature of Language**: As language evolves and new domains emerge, the framework may require periodic updates to remain relevant and accurate.

4. **Lack of Human Nuance**: While our approach correlates well with human judgments, it may not capture certain subtle aspects of human-AI interaction that human evaluators might notice.

5. **Limited Real-world Testing**: The current evaluation has been conducted in controlled environments. Further testing in diverse real-world scenarios is needed to fully validate the framework's robustness.

6. **Potential for Gaming**: As with any automated evaluation system, there is a risk that ToD developers might optimize their systems specifically for the metrics used, potentially leading to overfitting.

Addressing these limitations presents opportunities for future work, including exploring ways to reduce computational requirements, incorporating more diverse evaluation criteria, and conducting more extensive real-world testing. Despite these challenges, AutoEval-ToD represents a significant step forward in ToD evaluation, offering a powerful tool for researchers and developers in this rapidly evolving field.

References

Anthropic. 2023. The claude 3 model family: Opus, sonnet, haiku. https://www-cdn.anthropic.com/de8ba9b01c9ab7cbabf5c33b80b7bbc618857627/Model_Card_Claude_3.pdf.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Hyundong Cho, Thamme Gowda, Yuyang Huang, Zixun Lu, Tianli Tong, and Jonathan May. 2024. BotEval: Facilitating interactive human evaluation. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 107–116, Bangkok, Thailand. Association for Computational Linguistics.

Shahul Es, Jithin James, Luis Espinosa Anke, and Steven Schockaert. 2024. RAGAs: Automated evaluation of retrieval augmented generation. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 150–158, St. Julians, Malta. Association for Computational Linguistics.

Songbo Hu, Han Zhou, Mete Hergul, Milan Gritta, Guchun Zhang, Ignacio Iacobacci, Ivan Vulić, and Anna Korhonen. 2023. Multi3WOZ: A Multilingual, Multi-Domain, Multi-Parallel Dataset for Training and Evaluating Culturally Adapted Task-Oriented Dialog Systems. *Transactions of the Association for Computational Linguistics*, 11:1396–1415.

Abishek Komma, Nagesh Panyam Chandrasekarasastry, Timothy Leffel, Anuj Goyal, Angeliki Metallinou, Spyros Matsoukas, and Aram Galstyan. 2023. Toward more accurate and generalizable evaluation metrics for task-oriented dialogs. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 5: Industry Track)*, pages 186–195, Toronto, Canada. Association for Computational Linguistics.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Process-*

ing Systems, NIPS '20, Red Hook, NY, USA. Curran Associates Inc.

Zekun Li, Zhiyu Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Dong, Adithya Sagar, Xifeng Yan, and Paul Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8688–8704, Bangkok, Thailand. Association for Computational Linguistics.

Guo Lin, Wenyue Hua, and Yongfeng Zhang. 2024. Emojicrypt: Prompt encryption for secure communication with large language models. *Preprint*, arXiv:2402.05868.

Yinhong Liu, Yimai Fang, David Vandyke, and Nigel Collier. 2024. TOAD: Task-oriented automatic dialogs with diverse response styles. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 8341–8356, Bangkok, Thailand. Association for Computational Linguistics.

Baolin Peng, Chunyuan Li, Zhu Zhang, Chenguang Zhu, Jinchao Li, and Jianfeng Gao. 2021. RADDLE: An evaluation benchmark and analysis platform for robust task-oriented dialog systems. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4418–4429, Online. Association for Computational Linguistics.

Dongyu Ru, Lin Qiu, Xiangkun Hu, Tianhang Zhang, Peng Shi, Shuaichen Chang, Jiayang Cheng, Cunxiang Wang, Shichao Sun, Huanyu Li, Zizhao Zhang, Binjie Wang, Jiarong Jiang, Tong He, Zhiguo Wang, Pengfei Liu, Yue Zhang, and Zheng Zhang. 2024. Ragchecker: A fine-grained framework for diagnosing retrieval-augmented generation. *Preprint*, arXiv:2408.08067.

Ivan Sekulic, Silvia Terragni, Victor Guimarães, Nghia Khau, Bruna Guedes, Modestas Filipavicius, Andre Ferreira Manso, and Roland Mathis. 2024. Reliable LLM-based user simulator for task-oriented dialogue systems. In *Proceedings of the 1st Workshop on Simulating Conversational Intelligence in Chat (SCI-CHAT 2024)*, pages 19–35, St. Julians, Malta. Association for Computational Linguistics.

Ryuichi Takanobu, Qi Zhu, Jinchao Li, Baolin Peng, Jianfeng Gao, and Minlie Huang. 2020. Is your goal-oriented dialog model performing really well? empirical analysis of system-wise evaluation. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 297–310, 1st virtual meeting. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Guangsen Wang, Samson Tan, Shafiq Joty, Gang Wu, Jimmy Au, and Steven C.h. Hoi. 2022. BotSIM: An end-to-end bot simulation framework for commercial task-oriented dialog systems. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 178–190, Abu Dhabi, UAE. Association for Computational Linguistics.

Heng-Da Xu, Xian-Ling Mao, Puhai Yang, Fanshu Sun, and Heyan Huang. 2024a. Rethinking task-oriented dialogue systems: From complex modularity to zero-shot autonomous agent. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2748–2763, Bangkok, Thailand. Association for Computational Linguistics.

Xilie Xu, Keyi Kong, Ning Liu, Lizhen Cui, Di Wang, Jingfeng Zhang, and Mohan Kankanhalli. 2024b. An LLM can fool itself: A prompt-based adversarial attack. In *The Twelfth International Conference on Learning Representations*.

Eva Zangerle and Christine Bauer. 2022. Evaluating recommender systems: Survey and framework. *ACM Comput. Surv.*, 55(8).

A Cost-Effectiveness and Efficiency

We compared the financial and time costs of LLM evaluations to human annotations. Considering an hourly cost of USD 3.75[1] for annotations and the velocity of the annotations to be 30 data points per hour, the cost per human annotation is USD 0.125. In contrast, using Claude-3-Sonnet with an estimated 1000 input tokens and 100 output tokens per instruction, costs approximately USD 0.0045 according to Anthropic API pricing[2]. This demonstrates that LLM evaluations are potentially 25 times more cost-effective.

Moreover, the average time taken by a human annotator to annotate a data point is approximately 3 minutes, while with Claude-3-Sonnet we can accomplish this under 4 seconds. In addition to this, LLMs are increasingly employing optimization techniques such as caching and batching, which significantly reduce the number of API calls required, making them more affordable and environmentally friendly. We can easily integrate these approaches with the AutoEval-ToD's evaluating LLM model to enhance its cost-effectiveness and environmental

[1]https://www.dol.gov/agencies/whd/minimum-wage
[2]https://www.anthropic.com/pricing#anthropic-api

sustainability. Additionally, considering the simplicity of the task at hand, one could consider using smaller models such as Claude-3-Haiku (which is 3× cheaper than Claude-3-Sonnet), further elevating the usability of LLMs in evaluation of ToDs.

B Adaptability to Different ToDs

To integrate a AutoEval-ToD to a completely new ToD system, developers need only to log responses from the chatbot at each turn such as timestamps, the rank-ordered outputs of the retriever module etc. These logs are then used to compute all relevant metrics in a unified and efficient manner. As an example, consider a medical assistance chatbot designed to provide preliminary diagnoses and recommend next steps. Integrating it involves:

- **Logging Interactions:** At each turn, the chatbot logs user inputs, system responses, timestamps, and ranked outputs from its retriever module. For example, if the user simulator reports symptoms like *fever and headache*, the system retrieves possible causes (e.g., *viral fever* or *migraine*) and logs the ranked results using the respective *Eval** API.
- **Evaluation Metrics:** It evaluates multiple aspects, including retrieval accuracy (e.g., HitRate for ranked outputs), response quality (e.g., factual correctness and completeness), user experience (e.g., response latency and goal completion rate), domain compliance (adherence to medical guidelines), and turn optimization (efficiency in resolving queries). These metrics are computed automatically based on the chatbot's interaction logs.
- **Extensibility:** For domain-specific needs, such as assessing the chatbot's ability to evaluate Empathy Score, developers can define this metric by leveraging the logged interactions and incorporating pretrained LLMs or custom rules for sentiment analysis.

B.1 Adapting AutoEval-ToD to custom evaluations

AutoEval-ToD is explicitly designed to support extensibility for such custom metrics. For instance, for highlighting the potential to explore personalized evaluation and response style adaptation in ToD systems, we can introduce a new evaluation dimension, such as *Style Adaptation* or *Persona-Specific Evaluation*, by defining additional LLM prompts tailored to assess a system's flexibility in adjusting its response style based on user persona and context. These prompts could evaluate factors like tone appropriateness, empathy, or consistency with user-specific preferences. This demonstrates how the framework's modular design facilitates the seamless incorporation of novel evaluation metrics. By adding such dimensions, the framework can not only assess metrics like factual correctness or goal completion but also adapt to evolving research directions such as TOAD (Liu et al., 2024).

C Evaluation of Dynamic User Simulator

Our user simulator builds upon existing research (Sekulic et al., 2024; Xu et al., 2024a) that has evaluated user simulators on various aspects, including lexical diversity, generalization, and similarity. These studies have demonstrated the usefulness and accuracy of such simulators, providing a foundation for our approach. We thus focus on implementing and evaluating specific enhancements to address challenges unique to the AutoEval-ToD use case. We implemented two key modifications:

(1) We observed that the vanilla user simulator often provided detailed explanations when presented with options to choose, rather than simply indicating a choice. To address this, we instructed the user simulator with the following change (full prompt I.1), and qualitatively evaluated that this behavior is followed.

When the chatbot asks to select an option from a list of options, make a judgement call and select. The bot can handle variations so you can either respond with the exact same text as in options, respond with some variation of one of the option and expect the bot to map it correctly or choose to ask something else (a different sub issue from within the recommendations). Do not signal the selection with numbers such as 0/1/2 etc.

(2) A critical aspect of our user simulation was preventing the unintended leakage of *oracle information* in generated responses (as illustrated in Fig. 2). We implemented a strict decoupling between the user simulator and the oracle LLM, effectively eliminating this risk. This modification reduced the percentage of information leakage from 35% to 0% since the user simulator now has no direct access to the oracle solution.

It is noteworthy that user simulation is highly dependent on the specific ToD system. ToD developers can create their own user simulation with

minor modifications to cater to their use case.

D Interpreting and Applying AutoEval-ToD Results

AutoEval-ToD is designed as an iterative tool for progressive system refinement, offering flexibility in evaluation strategies to meet diverse use cases. While we provide a comprehensive set of metrics, the interpretation and prioritization of results ultimately lie with the developer. For instance, a customer service chatbot might prioritize user satisfaction metrics, while a technical support system might focus on task completion accuracy.

Additionally, to use AutoEval-ToD as a benchmark, developers can implement a weighted average or composite score, considering the relative importance of different metrics based on specific applications. We recommend clustering related metrics into logical groups (e.g., quality, user experience) and providing guidelines for interpreting these groups based on the system's intended use.

E Human Annotation Documentation

We employed a rigorous human assessment process to ensure the quality and reliability of our data annotations across different domains. Our annotation process varied by domain to ensure high-quality data. For the Troubleshooting ToD, we recruited domain experts with extensive experience in creating product troubleshooting content, while for other domains, we utilized general experts who met relevant language criteria.

Annotators were provided with the exact instructions mentioned in the different prompts, ensuring consistency between human and LLM evaluations. For example, in the Solution Checker task, domain expert annotators were asked to evaluate the ToD's recommendations against a random seed solution using the same instructions outlined in the prompt H.2. We recorded the responses from domain experts and observe a 97% alignment between LLM-based and human evaluations on this task showcasing the reliablity of our LLM evaluations.

Furthermore, we follow the standard protocol to measure inter-annotator agreement by performing dual annotations on a sample set of 10% of the data. We observed an agreement rate of 96% demonstrating the reliability of our annotations across all domains.

F Addressing Adversarial Vulnerabilities in AutoEval-ToD

AutoEval-ToD faces potential vulnerabilities to adversarial attacks and shortcutting, as do all AI systems (Xu et al., 2024b). While not inherently more susceptible than existing methods, we can integrating robust safeguards into our evaluation prompts to enhance security. These include prompt verification using cryptographic techniques (Lin et al., 2024), output filtering based on predefined criteria, and adversarial training of the underlying language model. For instance, techniques such as prompt encryption (Lin et al., 2024) can reduce the risk of malicious prompt injections. As LLMs continue to improve their rule-following capabilities through advanced training methods, they tend to become more robust to such attacks, potentially enhancing the reliability of our evaluation framework.

G Example of Domain Complaince Evaluator

Example: Domain Compliance Evaluator

ToD System Response:

> I'm sorry to hear that the steps to check the charging case battery level did not resolve the issue with your JBL earbuds not charging. Let me provide you with some additional solutions to try:
>
> 1. Reset or Restore Factory Settings: Remove both earbuds from the charging case, then follow the reset or factory reset instructions in the user manual. This can help resolve any software-related charging issues.
>
> 2. Check for Physical Damage: Carefully inspect the earbuds and charging case for any signs of physical damage, such as cracks, dents, or bent charging pins. If any damage is found, do not attempt to use or charge the earbuds.

Domain Compliance Evaluator Response:

✓ **Score: 1** - The solution does not suggest using any abrasive cleaners or chemical solutions.

✗ **Score: 0** - The solution recommends referring to the user manual for the reset or factory reset instructions.

✓ **Score: 1** - The solution does not recommend going to a service center for repair or replacement.

H Prompts

Prompt H.1: User Input Variation

Instruction:
As a text modifier, your role involves introducing subtle changes to a provided text snippet. This task requires adherence to specific types of alterations, categorized by difficulty levels.
The types of permissible variations are as follows:
- Easy Variations: 1. Addition or removal of punctuation marks. 2. Utilization of different variants of the same lemma.
- Medium Variations: 1. Employment of synonyms for any word. 2. Phrase modifications: either by substituting a single word with a phrase or vice versa.
- Hard Variations: 1. Structural transformation of the text, entailing a complete reformulation while preserving the original message.
Under no circumstances should changes deviate from these guidelines. The core message and structural integrity of the text must remain intact.
The provided text will be enclosed within <original_text> tags. Your task is to generate five variations for each difficulty level:
- For easy variations, enclose each variant within <easy_variations> tags, with individual variations wrapped in <variation> tags.
- For medium variations, use <medium_variations> for the group and <variation> for individual entries.
- For hard variations, group them under <hard_variations>, with each distinct variant in a <variation> tag.

In-context examples:
Here are some examples:
<example> ... </example>
<example> ... </example>

Input:
Now here is the input to you:
<context> {context} </context>
<original_text> {original_text} </original_text>

Prompt H.2: Solution Checker

Instruction:
You task is to check whether the given solution in <solution> </solution> tags is present in the given recommendation. If it is present then just say YES in the answer tag, else say NO.

In-context examples:
Here are some examples:
<example> ... </example>
<example> ... </example>

Input:
Now here is the input to you:
<recommendation>{recommendation}</recommendation>
<solution>{solution}</solution>

Prompt H.3: Domain Compliance

Instruction:
You are a chatbot output evaluator for the {domain} recommendation. Your role involves assessing whether a provided chatbot output adheres to a set of predefined rules, assigning a score for each rule based on its compliance level. You will be given the following inputs:
1. History: Enclosed within the XML tags <history> </history>, this describes the chatbot and the user conversation so far.
2. Utterance: Enclosed within the XML tags <utterance> </utterance>, this contains the chatbot output that you need to evaluate the rules for.
The following are the predefined rules for evaluation: <pre-defined rules>
<rule1> {rule1} </rule1>
...
<rule5> {rule5} </rule5>
</pre-defined rules>
Instructions for your respone:
1. Assign a score ranging 0, 1 or -1 for each rule to indicate the level of adherence, with 0 indicating non-compliance and 1 indicating full compliance. -1 indicates that rule is not applicabble. The scores should be within the XML tags <scores></scores>.
2. If a rule is not applicable to the steps provided, assign a score of -1 and state the reason as "Not applicable".
3. The utterance could be in {user_language} but you will have to output your reason always in English.

An example of the expected input structure is provided below:
<utterance>This is the utterance of the chatbot.</utterance>
<history>This is conversation of chatbot and user so far.</history>
An example of the expected output structure is provided below:
<response>
<scratchpad>[Your brief notes and observations here for each rule, in bullet points]</scratchpad>
<scores>
<score1>Score assigned for rule 1</score1> <reason1>Reason for the assigned score in English</reason1>
...
<score5>Score assigned for rule 5</score5> <reason5>Reason for the assigned score in English</reason5>
</scores>
</response>

Input:
Now that you have a clear understanding of the predefined rules and the instructions for evaluation, proceed with solving the task using the following input:
<utterance>{utterance}</utterance>
<history>{history}</history>

I Dynamic User Simulator Prompts

Prompt I.1: User Simulator

<task>
You are a human user who is facing some functional problem with a product that they recently bought on e-commerce store. Your task is to talk to a product support chatbot offered by in multiple turns (as in typical conversation) and try to get the issue resolved. You may
1) start with simple greetings (Hello, Hey, Good morning etc.). You can choose to skip this step.
2) then in subsequent turns, elaborate on the observed issue. You can take some leverage to add more observations than provided in the issue in case the chatbot asks for it
3) when the chatbot asks to select an option from a list of options, make a judgement call and select. The bot can handle variations so you can either respond with the exact same text as in options, respond with some variation of one of the option and expect the bot to map it correctly or choose to ask something else (a different sub issue from within the recommendations). Do not signal the selection with numbers such as 0/1/2 etc.
4) close the call when the conversation has reached to a conclusion.
</task>
You will be given as input the
1) product details within the tag <product_details></product_details> . This is the product the user is inquiring about.
2) core issue observed by you within the tag <core_issue_observed></core_issue_observed>. This is the issue the user is facing
3) complete conversation you've had with the chatbot so far within the tag <historical_conversation></historical_conversation>.
Here are a few examples:
<example> ... </example>
<example> ... </example>
<example> ... </example>
Now generate the next message to be sent to the chatbot.
Now here is the product details, core issue observed and the chat summary:
<product_details> {product_details} </product_details>
<core_issue_observed>{core_issue_observed}
</core_issue_observed>
<historical_conversation>{historical_conversation}
</historical_conversation>

J Response Quality Prompts

Prompt J.1: Factual Correctness Checker

Your task is to check if the given response is grounded on (i.e., supported by) the information present in the provided chunks. <chunks>{chunks}</chunks>
<response>{response}</response>
- If the response is fully grounded and supported by the chunks, you should output "YES" in <answer> </answer> tags. - If the response is not grounded or contradicts the information in the chunks, you should output "NO" in <answer> </answer> tags.

Prompt J.2: Response Completeness Checker

Your task is to analyze the given response in the <response> </response> tags. The response should be complete and should not contain partial information that may be unhelpful. The response should cover all the relevant points from the chunks provided in the <chunks> </chunks> tags.
<chunks>{chunks}</chunks>
<response>{response}</response>
If the response is complete, output "YES" in <answer> </answer> tags. If the response is incomplete or lacks important information, output "NO" in <answer> </answer> tags.

Prompt J.3: Entailment Checker

Your task is to analyze if the given response in the <response> </response> tags is coherent and appropriate with the conversation history so far, present in <history> </history> tags, between a Customer and an AI Assistant.
<history>{history}</history>
<response>{response}</response>
If the response is coherent and appropriate based on the conversation history, output "YES" in <answer> </answer> tags. If the response is incoherent or inappropriate, output "NO" in <answer> </answer> tags.

Prompt J.4: Entailment Checker

Your task is to analyze the given response in the <response> </response> tags for any potential biases related to gender, caste, color, nationality, or other protected characteristics. We want to ensure that the response does not contain any discriminatory or biased language against individuals or groups based on their personal attributes or identities. The goal is to identify responses that may exhibit conscious or unconscious biases.
<response>{response}</response>
If the response is bias-free and does not exhibit any discriminatory language or attitudes, output "YES" in <answer> </answer> tags. If the response contains biased or potentially offensive content, output "NO" in <answer> </answer> tags.

Prompt J.5: Empathetic Tone Checker

Your task is to analyze the given response enclosed within <response> </response> tags and determine if it is expressed in an empathetic tone or not.
When analyzing the response, please consider the following criteria for an empathetic tone:
1. The response shows understanding and acknowledgment of the other person's feelings or situation.
2. The language used is caring, supportive, and non-judgmental.
3. The response conveys a sense of validation and genuine concern for the other person's well-being.
4. The tone is warm, compassionate, and respectful.
If the response meets all of the above criteria and is expressed in an empathetic tone without any un-empathetic elements, output "YES" enclosed in <answer> </answer> tags. Otherwise, output "NO".
<response>{response}</response>

Prompt J.6: Language Correctness Checker

Analyze the text within the <response> </response> tags and determine if it is written in proper English. Consider factors such as grammar, spelling, and the absence of non-English words or phrases.
<response>{response}</response>
Provide your assessment by outputting "YES" within <answer> </answer> tags if the response is in proper English, or "NO" if it contains errors or non-English elements.

K Additional Figures and Tables

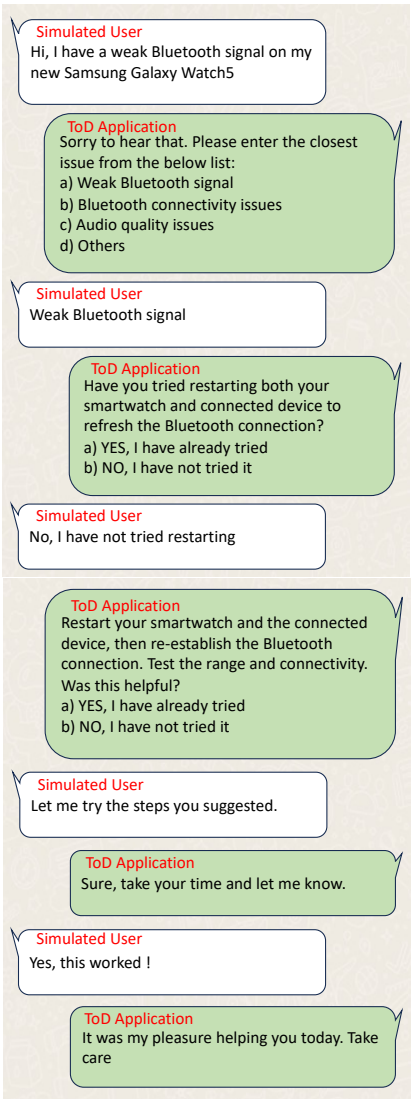

Figure 6: Sample conversation simulated by the proposed framework.

Easy	*I have an issue with my phone.* *I've an issue with my phone* *I have an issue, with my phone!* *I have an issue... with my phone* *I have an issue; with my phone*
Medium	*I'm facing a problem with my mobile.* *I have a concern regarding my smartphone.* *There's an issue with my cell phone.* *Encountering a difficulty with my telephone.* *My phone presents a complication.*
Hard	*There seems to be a problem with the device I use for communication, specifically my phone.* *My mobile device is currently facing some issues that need attention.* *A complication has arisen with the technology I use for calling and messaging - my phone.* *Concerning my mode of communication, the phone, an issue has surfaced.* *The tool I rely on for connectivity, my phone, is unfortunately not functioning as expected.*

Table 6: Examples of Easy, Medium and Hard variations of User Input *"I have an issue with my phone"*

Dataset	**Rule #**	**Description**
Troubleshooting	1	*Avoid solutions that suggest use of abrasive cleaners or chemical solutions.*
	2	*Avoid solution that point the user to refer to the user manual.*
	3	*Avoid recommending solutions that asks the user go to the service center for repair or replace.*
	4	*Avoid recommending steps such as contacting product support and customer support.*
MultiWOZ	1	*Avoid output that are verbose in nature.*
	2	*Avoid output that reveal the price of the recommendation.*
	3	*Avoid output that asks the personal details of the user.*
	4	*Avoid output that are adult in nature.*

Table 7: Domain specific rules for the different ToDs datasets