

Evaluating Modular Dialogue System for Form Filling Using Large Language Models

Sherzod Hakimov¹, Yan Weiser¹ and David Schlangen^{1,2}

¹Computational Linguistics, Department of Linguistics
University of Potsdam, Germany

²German Research Center for Artificial Intelligence (DFKI), Berlin, Germany
firstname.lastname@uni-potsdam.de

Abstract

This paper introduces a novel approach to form-filling and dialogue system evaluation by leveraging Large Language Models (LLMs). The proposed method establishes a setup wherein multiple modules collaborate on addressing the form-filling task. The dialogue system is constructed on top of LLMs, focusing on defining specific roles for individual modules. We show that using multiple independent sub-modules working cooperatively on this task can improve performance and handle the typical constraints of using LLMs, such as context limitations. The study involves testing the modular setup on four selected forms of varying topics and lengths, employing commercial and open-access LLMs. The experimental results demonstrate that the modular setup consistently outperforms the baseline, showcasing the effectiveness of this approach. Furthermore, our findings reveal that open-access models perform comparably to commercial models for the specified task.

1 Introduction

Dialogue evaluation stands as a critical discipline within Natural Language Processing (NLP), gaining heightened significance with the emergence of large language models (LLM). The introduction of recent commercial and open-access models has transformed the landscape by enabling building dialogue applications where such models are fine-tuned to follow instructions (Ouyang et al., 2022). As these models showcase impressive capabilities in generating coherent and contextually relevant responses to the given prompts, evaluating their performance on multi-turn interactions (dialogue settings) requires a deeper look that goes beyond conventional metrics (Hudecek and Dusek, 2023b). Recent advancements in the field suggested incorporating LLMs into self-contained *modules* that have certain task objectives (instructed by prompting (Brown et al., 2020)) and placed in a simu-

lated environment for benchmarking or frameworks for developing agents with LLMs (Chalamalasetti et al., 2023; Qiao et al., 2023; Wu et al., 2023; Li et al., 2023; Zhou et al., 2023).

Form filling is one of the traditional tasks for conversational interfaces, and a whole markup scheme (VoiceXML) has been designed around it (McTear et al., 2016). In this paper, we want to explore how the capabilities of modern LLMs as “linguistically programmed linguistic processors” can address this task. We propose the idea of building a modular dialogue system for the special purpose of form-filling. Typical form filling involves a user answering questions sequentially until all required fields are completed. Our proposed idea is to decompose a larger task of form filling into smaller sub-tasks and assign a specific module to them. Each module is realized through prompting a general purpose LLM and is responsible for solving only the assigned sub-task. We do not program each module in a traditional programming language but instead use the “programming through prompting” approach based on LLMs. Such a division of the overall task into multiple sub-tasks includes benefits such as handling the context window size limit since providing the complete form and all interactions in a single prompt text might lead to reaching the token limits of current LLMs (Ratner et al., 2023). Another dimension is to rank various LLMs for their instruction-following abilities, such as form filling and its sub-tasks.

The overview of how a modular dialogue system works for the form-filling task is illustrated in Figure 1. One module (chunk generator) is responsible for going through all fields and finding commonalities among them to create chunks. Such commonalities can refer to questions that can be asked together, e.g., place of birth and date of birth, which can be asked in a single question, leading to shorter dialogues. Such a way of filling out the form reduces the number of required turns for the

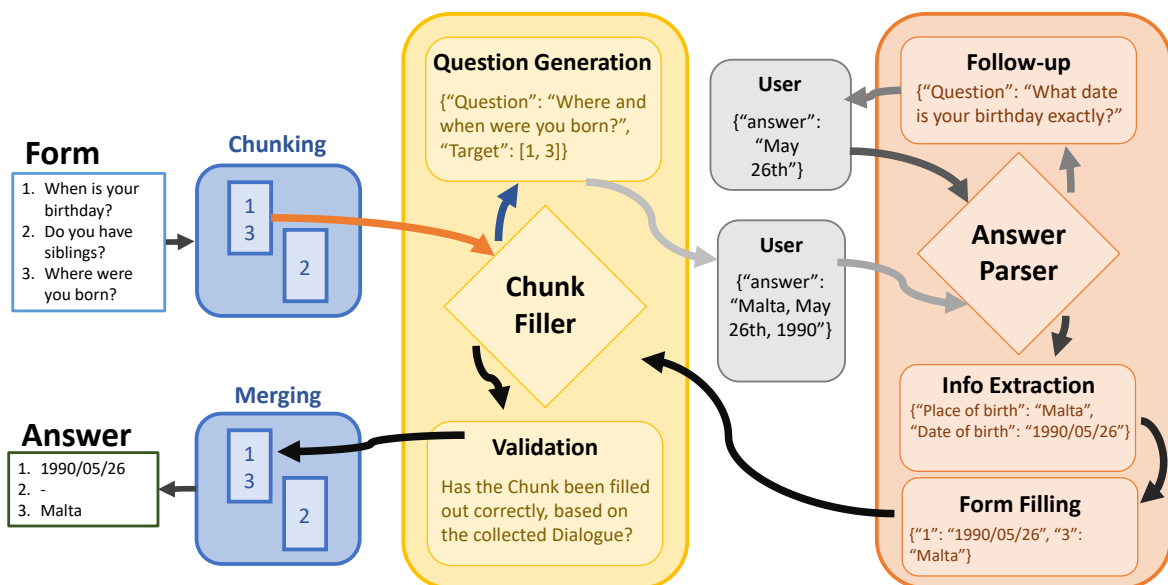


Figure 1: Overview of the modular approach for form filling by prompting LLMs. Each module is responsible for a specific sub-task.

user while keeping the provided answers semantically related. Once the question is asked and the user answers it, the module responsible for *answer parsing* matches the question to the answer. If the provided answers suffice, then the module responsible for *information extraction* applies its role to extract the required information for each form field in the selected chunk. The process goes on in this manner until the required fields in the form are processed. The user can be prompted to provide an answer again if a particular criterion is not met, e.g., the answer parser detects an issue with the provided answer.

We compare the proposed method with a baseline setup that uses two modules that do multiple sub-tasks simultaneously. We evaluate commercial and open-weight LLMs for their performance in realizing these modules. We also evaluate (in simulation) the whole system and show that it has specific desirable properties, such as leading to fewer turns (shorter dialogues), which is due to the fact that certain fields can be grouped together rather than processing the form in sequential order. We present here the first proof-of-concept where the main functionality is simulated. We selected four existing forms from different categories that include various fields such as free-form text field, multiple choice, date, number, single-choice. The test with real users remains to be done in the future.

Our main contributions can be summarized as follows:

- evaluation of dialogue systems for the task of form-filling
- modular setup that allows delegating the task to multiple sub-modules to solve it collectively
- extensive experimental evaluation where we compare commercial LLMs with open-weight ones for their instruction-following capabilities in a dialogue setting

The source code of the approach is shared publicly¹.

2 Related Work

Recent studies by Chalamalasetti et al. (2023); Qiao et al. (2023); Wu et al. (2023) focused on the idea of defining *modular components* on top of large language models (LLM) and simulating them on certain environments for their dialogue capabilities. The recent paradigm in pre-trained LLMs is that they are also fine-tuned to follow instructions (Ouyang et al., 2022), essential to building dialogue systems. Wu et al. (2023) proposed to

¹https://github.com/clp-research/modular_form_filling_with_llm

benchmark LLMs as agents on six games such as Rock-Paper-Scissors, Tower of Hanoi, Minecraft, etc. Each game features a unique setting and environment. Their benchmarking framework allows comparison of commercial LLMs (e.g. GPT-4) against open-weight alternatives. Qiao et al. (2023) followed a similar idea by proposing the evaluation of LLMs through goal-driven conversational games. One of the games is called ask-guess, where two players (prompted LLMs) cooperate to guess the target word by engaging in a multi-turn dialogue. Chalamalasetti et al. (2023) proposed the combination of multiple word games such as taboo, wordle, and reference games where the task is to describe the given image to the partner. Their approach is based on defining such games as dialogue games where each participating player (prompted LLMs) follows specific instructions given by the game master.

Regarding the idea of defining any task as a modular approach, Lu et al. (2023) proposed the approach called *Chameleon*, where multiple components are glued together. Their approach is built on top of an LLM, acting as an agent with access to tools such as web search and optical character recognition to extract text from images, etc. One of the main contributions of our paper is the idea of dividing the task into multiple sub-modules where Ratner et al. (2023) has proposed a similar notion of creating chunks of long context.

2.1 Dialogue Evaluation

When evaluating dialogue systems, most studies rely on human evaluation since the quality of dialogue is a subjective measurement (Quinderé et al., 2013). This qualitative assessment is, therefore, costly. There are also ways to assign quantitative measures to dialogue systems that work primarily for task-oriented dialogue systems. Gasic et al. (2008) introduce a scoring function for their probabilistic system that rewards picking up new beliefs and punishes every step taken to encourage shorter interactions. A recent survey by Deriu et al. (2021) on evaluation methods for dialogue systems identified five attributes that an evaluation strategy should follow: automatic, repeatable, correlated to human judgment, differentiate between different dialogue systems, and explainable. Mehri and Eskénazi (2020) suggested a scoring function that leverages LLMs to generate scores. Hudecek and Dusek (2023a) in their recent work evaluated

their dialogue system, which is based on LLM for dialogue state tracking and domain detection, on task-oriented settings by looking at automatic metrics such as task success as well as human judgments. Another evolving recent direction is to employ LLMs judging generated responses of another LLM. Chan et al. (2023) proposed *ChatEval* where their setup composed of LLMs discuss and evaluate the quality of generated responses.

3 Modular Dialogue Setup

Our proposed methodology is to build a dialogue system capable of leading another participant to fill out any form. The dialogue system is built on top of multiple sub-modules acting on their restricted task descriptions. A module, in this context, is built by prompting an LLM with the defined task description and is expected to follow the instructions in the task description. We divide modules into two distinct groups: interactive & task-performing modules. Next, we explain each in detail.

3.1 Interactive Modules

These modules are instructed to handle the current part of the task and decide on the next step. They can instruct others to perform multiple small sub-tasks, collect the outputs, and then decide what remaining part of the sub-task to focus on next.

Dialogue Manager: It sits on the top of all to control the flow of the dialogue and keep track of the state. Similar to typical dialogue state tracking in traditional dialogue systems. Its main goal for the form-filling task is to slide the forms into pieces to reduce the context and handle it one piece at a time. It has three options to choose form: 1) *form chunks*, 2) *fill chunk*, and 3) *stop*. Creating chunks is dividing the form into chunks by grouping similar questions. Fill chunk is the process handled by another module (explained below) to fill out the grouped questions. The stop process is executed once all selected chunks and their corresponding questions are answered. It is also invoked when needed to exit from endless loops if encountered.

Chunk Filler: It receives a certain chunk and all the fields from the form that are grouped under it. It can perform three actions: 1) *question generation*, 2) *answer validation*, 3) *stop*. Question generation is the process of generating a question that encompasses some fields (or all) that are part of a selected chunk of the form. It is called as long as there are fields for which an answer has not been provided

yet. Answer validation is the process of checking the validity of the provided answer with respect to the fields that are asked in the question. Stop essentially ends the module’s actions and returns the filled fields that are part of the given chunk.

Answer Parser: It is invoked after each interaction to parse the given answer. It can perform three actions: 1) *information extraction*, 2) *follow-up question*, 3) *repeated question*. Information extraction is the process of matching the fields in the selected chunk with the provided answer. For instance, for questions such as “What is your place of birth and date?” an answer such as “I was born in Malta on May 26th 1990” is provided. The fields in the chunk can be “birthplace” and “birth date”. The task of the information extraction step is to extract which part of the text corresponds to which form field and return those values normalized, e.g., following certain date formatting. The follow-up question is the process of asking another question if a certain field can not be filled with the provided answer. The repeated question is used when the assignment is completely missed or a clarification question is asked.

3.2 Task-performing Modules

These modules are instructed to do one specific task and return the outputs in a certain format. The interactive modules described in Section 3.1 employ task-performing modules to handle certain actions.

Question Extractor: It takes a single field of the form as input and generates a single sentence that summarizes the field. It is called by the Dialogue Manager.

Question Grouping: Using the generated summaries for each field, it generates groups based on the summaries and assigns them a unique name. We refer to these groups as *chunks*. It is called by the Dialogue Manager.

Grouping Validator: By utilizing the summaries and the generated groups, it validates whether each field (its summary) is part of exactly one group (chunk). If needed, it returns revised groups and corresponding fields. The returned groups of fields are used as the final form for the rest of the dialogue. It is called by the Dialogue Manager.

Question Generator: It takes an entire chunk (groups of fields) and generates a single question that encapsulates some fields. It is not necessary to include all fields at once. It is called as long as

some fields exist in a chunk for which an answer has not been provided yet. It is called by the Chunk Filler.

Follow-up Question Generator: It is instructed to generate another question in case a certain field of a chunk can not be filled based on the provided answer. It is called by the Answer Parser.

Repeated Question Generator: It is instructed to generate another question in case the provided answer is marked as not satisfactory. Using the previous turns in the dialogue and the current fields in a chunk and generates another question. It is called by the Answer Parser.

Information Extractor: It is instructed to match the fields of interest for which a question has been generated, to the provided answer. It returns key-value pairs where keys correspond to form fields and values are snippets from text that are related to them. It is called by the Answer Parser.

Form Filler: It takes the extracted information, the generated question, and fills out the field in the form for which an answer has been provided. The output is returned back to the Chunk Filler.

Answer Validator: It is instructed to check if a certain field has been filled correctly regarding the answer provided for it. Certain fields can be marked as empty if the provided answer does not match the summary of the field. It returns the validated fields back to the Chunk Filler.

All outputs generated by any module are done in JSON format. The full prompts for each of them are available in in Figure 4 and 5.

4 Experimental Setup

In this section, we provide details about forms that are used for the experiments, compared baseline dialogue system and LLMs used as the basis for realizing the modular dialogue setup.

4.1 Forms

All experiments are based on using existing forms that were chosen from four different topics. Each selected form already existed and publicly accessible. The choice of forms is based on one hand to include variety in terms of terminology used in different domains, and on the other hand to include forms with varying number of fields (between 14-52). Additionally, the forms include different types of field such single choice, multiple choice, free form and set of predefined data types such as date and number. We chose the following forms:

- SS5 Form - Application for a Social Security Card: it includes 39 fields focusing on the applicant’s personal information and history with social security. It includes single-choice, multiple-choice, date, number, and free-form text fields.
- EPA Form - Report Environmental Violations: it includes 16 fields where most are optional.
- MED Form - Confidential Medical History: it includes 52 fields where many fields correspond to the medical domain. It was chosen to check whether such domain-specific forms can be handled the same way as others.
- INV Form - Invention Disclosure: it includes 14 fields with multiple-choice, single-choice, and free-form text fields.

The full forms with their fields are provided in Appendix 9.

4.2 Evaluation Metrics

We use the following quantitative metrics to assess whether the form-filling task has been completed and to what degree. We defined the following three metrics:

Task-success: a weighted sum of correctly filled required and optional fields. It measures whether the evaluated setups miss to fill out any field in forms. It can be calculated as follows:

$$\text{Success} = \frac{R_{req} + w * R_{opt}}{1 + w} \quad (1)$$

where R_{req} is the ratio of required fields that are filled correctly, R_{opt} is the ratio of optional fields that are filled correctly, and w is a parameter that assigns how impactful the ratio of optional fields is. It is set to 0.2, which means that most of the weight is assigned to getting the required fields correctly filled and certain reward is also assigned for filling optional fields.

Task-efficiency: it is computed based on the number of interactions required to complete a certain form. The form filling is assumed to be performed efficiently when the interactions are less than the number of fields in a form and repetition of some questions does not occur. The idea behind having less number of interactions than available fields is that to enable filling multiple fields in a single turn. It can be achieved by the *question grouping* module that groups similar fields together,

also called as *chunks*. For instance, the questions related to someone’s birth date, place and location can be asked in a single turn instead of three turns. It is computed as follows:

$$\text{Efficiency} = \frac{1}{\max(1, \frac{2T}{L}) + \frac{Rep}{L}} \quad (2)$$

where T is the number of turns, L is the number of fields in a form, and Rep is the number of repetitions that have occurred during form filling.

Score: Finally, we combine these two metrics to get a value that is the harmonic mean of efficiency and success. It allows ranking of evaluated models with a single metric that combines both task success and efficiency.

$$\text{Score} = \frac{2}{\frac{1}{\text{Success}} + \frac{1}{\text{Efficiency}}} \quad (3)$$

4.3 Evaluated Models

We selected models based on the availability of larger models from open-source community and included one model from commercial side as well. We used the following models that have been evaluated for each chosen form:

- *gpt-3.5-turbo* is a commercial model from OpenAI².
- *sheep-duck-llama-2* an open-weight model available on HuggingFace. It is a fine-tuned version of Llama2 70B (Touvron et al., 2023)³.
- *Openbuddy-llama2-70b*: it is another open-weight model that is fine-tuned version of Llama2 70B⁴.

4.4 User Simulation

To provide various answers based on certain user profiles, we simulated user answers by prompting LLM (Zhang et al., 2018; Lin et al., 2022). The answer to each given question was generated based on the selected user profile. Here is the list of used profile descriptions: *a banker living in Downtown New York, a fisherman living on the west coast of the USA, a politician and lawmaker, an actress*

²<https://openai.com/blog/chatgpt>

³<https://huggingface.co/Riivid/sheep-duck-llama-2>

⁴<https://huggingface.co/OpenBuddy/openbuddy-llama2-70b-v13.2>

Model	Form	Task-success	Task-efficiency	Score
gpt-3.5-turbo	<i>SS5</i>	×	×	×
	<i>EPA</i>	0.83	0.72	0.77
	<i>MED</i>	×	×	×
	<i>INV</i>	0.83	1.00	0.91
sheep_duck_llama2	<i>SS5</i>	×	×	×
	<i>EPA</i>	×	×	×
	<i>MED</i>	×	×	×
	<i>INV</i>	0.38	1.00	0.55
OpenBuddy-70B	<i>SS5</i>	×	×	×
	<i>EPA</i>	×	×	×
	<i>MED</i>	×	×	×
	<i>INV</i>	×	×	×

Table 1: Results of runs in the baseline setup. Models were tested 5 times. × indicates the metrics cannot be calculated since the run resulted in failed instruction following for the form filling.

Model	Form	Task-success	Task-efficiency	Score
gpt-3.5-turbo	<i>SS5</i>	0.84	0.99	0.91
	<i>EPA</i>	0.91	0.56	0.69
	<i>MED</i>	0.93	0.60	0.73
	<i>INV</i>	1.00	0.92	0.96
sheep_duck_llama2	<i>SS5</i>	0.46	0.88	0.60
	<i>EPA</i>	0.80	0.95	0.87
	<i>MED</i>	0.87	0.99	0.93
	<i>INV</i>	0.95	0.83	0.86
OpenBuddy-70B	<i>SS5</i>	0.83	0.97	0.89
	<i>EPA</i>	0.96	0.46	0.62
	<i>MED</i>	0.94	0.98	0.96
	<i>INV</i>	0.91	1.00	0.95

Table 2: Results of runs in the modular setup. Models were tested 10 times. The best *Score* value for each form is highlighted in bold.

who moved from France to Los Angeles, a beetroot farmer with German roots and an office job, an olympic swimmer, a brain surgeon, a female soccer player, a social media influencer, a university student.

The user profiles for each run during the experiments are chosen randomly. The respective prompts for the generation of user answers using the selected profile is given below.

Imagine you are [Insert 1] and you are filling out a form referring to [Insert 3].

Come up with a generic yet plausible answer to the question. Provide your answer in the following format:
{"answer": ""}

QUESTION:
[Insert 2]

4.5 Baseline Setup

We selected the following as a baseline to compare against the proposed modular setup. The baseline system works by taking alternating turns. In the first turn, the module uses the form in order to generate a question to ask the user. In the next turn, the answer to the question generated in the last turn is used to fill out the form’s fields. After each step of filling out the form, it can stop the process and return it to its current state. Since the first module takes the entire form into context, it quickly runs into limitations regarding its input the context limits. Due to this constraint, some longer forms (for example, the MED form) cannot be run on all models in this arrangement. To combat context size limitations, fields that have been filled out are removed from the form to fit the remaining fields

into the context window. The prompt templates for both turns are given in Figure 3.

5 Results

Baseline Setup: The results for the evaluated models are available in Table 1. Many runs on the selected forms stopped the dialogue system since instructions were not followed. Thus, many rows in the table do not include corresponding values for the metrics. The results indicate that only *gpt-3.5-turbo* is able to complete *EPA* and *INV* forms using the baseline setup. *Sheep-duck-llama2* is able to complete the *INV* form while *OpenBuddy-70B* cannot follow instructions for all four forms. These findings indicate that the form-filling using the selected two sub-modules is not adequate for the task of form-filling in a dialogue setup.

Modular Setup: The results for are available in Table 2. Compared to the baseline setup, all three models are able to process the forms without any interruptions caused by instructions not being followed. All models achieved high task-success except for the *sheep-duck-llama2* on the *SS5* form. It indicates that the division into sub-tasks is better for instruction following since each module is responsible for the limited context of the form. Thus not running into the issue of exceeding the context token limit size. The dialogue transcripts of the *gpt-3.5-turbo* for *SS5* and *MED* forms are available in Figure 10 and the transcripts for the *INV* and *EPA* forms in Figure 11.

Commercial vs. Open-weight Models: The average of results across all forms and metrics is given in Figure 2a. *gpt-3.5-turbo* achieves the best performance in *task-success* while *openbuddy-70B* being the second-best with very close outcome. In terms of *task-efficiency*, *sheep-duck-llama2* is the best in this category, while the other two models struggled with the *EPA* form the most. Overall, we can observe that open-weight models are comparable in many ways to the selected commercial model (*gpt-3.5*) for the selected task.

Shorter Dialogues: Figure 2b shows the average number of fields in the selected forms and the number of turns it took for each model. It can be seen that the models need fewer turns in a dialogue than the total number of fields in the forms. It can be explained by the fact that certain fields in the form are merged into a single question. For instance, the scores of the models for the *INV* form are higher than other forms (0.96, 0.86, 0.95 for

gpt-3.5, *sheep-duck-llama2*, *Openbuddy*, respectively). The form includes 14 fields, but on average, all models take only eight turns to achieve reasonably high performance. Similar observations can be seen for the *MED* and *SS5* forms. The number of turns for the *EPA* form indicates that the models need at least the same number of turns as the available fields or even longer in some cases.

Based on these outcomes from the experiments, we can conclude that **form filling can be utilized as a task to compare dialogue systems built on LLMs** (commercial or open-access) for their instruction-following abilities (Chalamalasetti et al., 2023; Qiao et al., 2023; Wu et al., 2023), **modular setup yields better performance across all metrics** when compared with the baseline setup.

6 Conclusion

In this paper, we presented a study of evaluating dialogue systems built on top of LLMs with the task of form filling. The dialogue system is built by defining a prompt text that includes instructions on handling the given input. Specifically, we focus on building such a dialogue system on LLMs by defining multiple sub-modules that are assigned a specific role to solve. Our modular setup has been tested on the selected four forms from various topics and lengths by employing commercial and open-access LLMs. Our experimental results suggest that the modular setup outperforms the baseline. Another important finding is that open-access models are on par with the commercial model for the selected task. In future work, we plan to extend the modular dialogue system into multiple similar tasks, compare more commercial and open-access models, and perform the study that involves users filling out the forms to compare against the proposed method.

References

- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, and et al. 2020. [Language models are few-shot learners](#). In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.
- Kranti Chalamalasetti, Jana Götze, Sherzod Hakimov, Brielen Madureira, Philipp Sadler, and David Schlangen. 2023. [clembench: Using game play to evaluate chat-optimized language models as conversational agents](#). In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language*

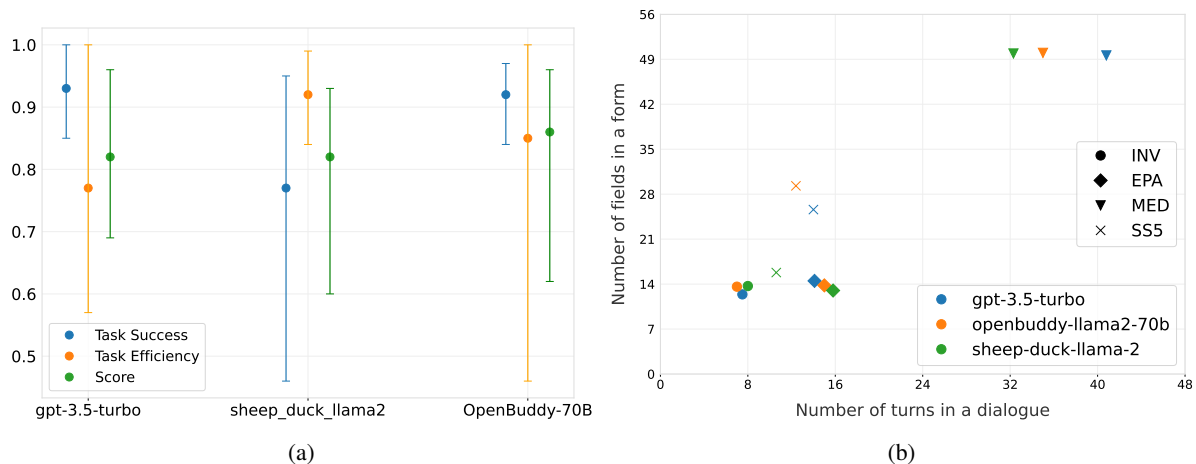


Figure 2: **Left:** comparison of evaluated models on the modular setup averages across all forms, **Right:** an average over the number of turns taken and the number of fields filled out per form per model.

Processing, pages 11174–11219, Singapore. Association for Computational Linguistics.

Chi-Min Chan, Weize Chen, Yusheng Su, Jianxuan Yu, Wei Xue, Shanghang Zhang, Jie Fu, and Zhiyuan Liu. 2023. [Chateval: Towards better llm-based evaluators through multi-agent debate](#). *CoRR*, abs/2308.07201.

Jan Deriu, Álvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. [Survey on evaluation methods for dialogue systems](#). *Artif. Intell. Rev.*, 54(1):755–810.

Milica Gasic, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, Kai Yu, and Steve J. Young. 2008. [Training and evaluation of the HIS POMDP dialogue system in noise](#). In *Proceedings of the SIGDIAL 2008 Workshop, The 9th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 19-20 June 2008, Ohio State University, Columbus, Ohio, USA*, pages 112–119. The Association for Computer Linguistics.

Vojtech Hudecek and Ondrej Dusek. 2023a. [Are large language models all you need for task-oriented dialogue?](#) In *Proceedings of the 24th Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2023, Prague, Czechia, September 11 - 15, 2023*, pages 216–228. Association for Computational Linguistics.

Vojtech Hudecek and Ondrej Dusek. 2023b. [Are llms all you need for task-oriented dialogue?](#) *CoRR*, abs/2304.06556.

Jiatong Li, Rui Li, and Qi Liu. 2023. [Beyond static datasets: A deep interaction approach to llm evaluation](#).

Hsien-Chin Lin, Christian Geishauer, Shutong Feng, Nurul Lubis, Carel van Niekerk, Michael Heck, and Milica Gasic. 2022. [Gentus: Simulating user behaviour and language in task-oriented dialogues with generative transformers](#). In *Proceedings of the 23rd*

Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGDIAL 2022, Edinburgh, UK, 07-09 September 2022, pages 270–282. Association for Computational Linguistics.

Pan Lu, Baolin Peng, Hao Cheng, Michel Galley, Kai-Wei Chang, Ying Nian Wu, Song-Chun Zhu, and Jianfeng Gao. 2023. [Chameleon: Plug-and-play compositional reasoning with large language models](#). *CoRR*, abs/2304.09842.

Michael McTear, Zoraida Callejas, and David Griol. 2016. *The Conversational Interface: Talking to Smart Devices*. Springer, Cham.

Shikib Mehri and Maxine Eskénazi. 2020. [Unsupervised evaluation of interactive dialog with dialogpt](#). In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, SIGdial 2020, 1st virtual meeting, July 1-3, 2020*, pages 225–235. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022. [Training language models to follow instructions with human feedback](#). In *NeurIPS*.

Dan Qiao, Chenfei Wu, Yaobo Liang, Juntao Li, and Nan Duan. 2023. [Gameeval: Evaluating llms on conversational games](#). *CoRR*, abs/2308.10032.

Marcelo Quinderé, Luís Seabra Lopes, and António J. S. Teixeira. 2013. [Evaluation of a dialogue manager for a mobile robot](#). In *IEEE International Symposium on Robot and Human Interactive Communication, IEEE RO-MAN 2013, Gyeongju, South Korea, August 26-29, 2013*, pages 126–132. IEEE.

Nir Ratner, Yoav Levine, Yonatan Belinkov, Ori Ram, Inbal Magar, Omri Abend, Ehud Karpas, Amnon

Shashua, Kevin Leyton-Brown, and Yoav Shoham. 2023. [Parallel context windows for large language models](#). In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 6383–6402. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, and et al. 2023. [Llama 2: Open foundation and fine-tuned chat models](#). *CoRR*, abs/2307.09288.

Yue Wu, Xuan Tang, Tom M. Mitchell, and Yuanzhi Li. 2023. [Smartplay : A benchmark for llms as intelligent agents](#). *CoRR*, abs/2310.01557.

Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. 2018. [Personalizing dialogue agents: I have a dog, do you have pets too?](#) In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 2204–2213. Association for Computational Linguistics.

Xuhui Zhou, Hao Zhu, Leena Mathur, Ruohong Zhang, Haoifei Yu, Zhengyang Qi, Louis-Philippe Morency, Yonatan Bisk, Daniel Fried, Graham Neubig, and Maarten Sap. 2023. [Sotopia: Interactive evaluation for social intelligence in language agents](#).

7 Module Prompts

The prompt templates for the baseline setup are given in Figure 3. The prompt templates for the modules are given in Figure 4 and 5.

8 Dialogue Transcripts

The dialogue transcripts of the *gpt-3.5-turbo* for all forms are available in Figure 10 and 11.

9 Forms

All four chose forms with their fields are given in Figure 6, 7, 8, and 9 for SS5 - Application for a Social Security Form⁵, EPA – Report Environmental Violations⁶, MED - Confidential Medical History⁷, and INV - Invention Disclosure⁸, respectively.

⁵<https://www.ssa.gov/forms/ss-5.pdf>

⁶<https://echo.epa.gov/report-environmental-violations>

⁷<https://www.england.nhs.uk/south/wp-content/uploads/sites/6/2019/11/gwh-medical-history-forms.pdf>

⁸https://researchprotections.appstate.edu/sites/researchprotections.appstate.edu/files/interactive%20ip_asu_invention%20and%20discovery%20disclosure%20form.pdf

TEMPLATE 7.0.1

Question Generator

Please generate a question. The answer to the question will be used to fill out fields of the given FORM. Try to ask about multiple fields in a single question if they are related. Give all possible Options if they are limited by the FORM. Only fields with an "answer" attribute can be filled out. Your output should be in the following format: "question": ""

FORM: [Insert 1]

Form Filler

Given the following ANSWER, you can do one of the following actions:
1. follow_up: Ask another question to get more information.
2. fill_form: use the answer to fill out fields in the form.
3. stop: stops the filling of the form. Only call this once all required fields have been filled.
Choose the appropriate option and output your choice in the following format: "next action": "".

ANSWER: [Insert 1]

Figure 3: Prompt templates used for baseline setup.

TEMPLATE 7.0.2

Dialogue Manager

You are managing the filling of a form by deciding what action needs to be taken. The progress can be seen in the CURRENT STATE. You can choose from the following options:

1. `form_chunks`: call this at the very beginning to split the form into workable chunks.
2. `fill_chunk`: work on a chunk of the form and fill out its fields. specify an empty chunk to work on.
3. `stop`: Stops the process. Call this once all chunks have been validated.

Specify what next action should be. Your output needs to have the following format: `"next action": "", "chunk to work on": null`

CURRENT STATE: [State]

Chunk Filler

You are managing the filling of a form. Based on the CURRENT STATE, decide which of the following actions should be taken:

1. `"question_generation"`: Ask a question in order to fill out empty fields.
2. `"fill_validation"`: validate fields that have been answered.
3. `"stop"`: stops the filling of this form, call this once all fields are validated.

The CURRENT STATE gives you information about every field in the form. Fields can be `"empty"`, `"answered"` or `"validated"`. Only return the filled out RETURN FORM and nothing else. The output needs to have the format of the RETURN FORM.

RETURN FORM: `"next action": ""`

CURRENT STATE: [State]

TEMPLATE 7.0.3

Answer Parser

The given DIALOGUE has the goal to fill out the given FORM. You need to decide what Option should be taken next. Your options for next actions are:

1. `"information_extraction"`: Choose this action if the given answer contains all the necessary information.
2. `"follow_up_question"`: Choose this if the given answer is invalid or only covers part of the necessary information.
3. `"repeat_question"`: Choose this if the User asked for clarification or did not answer the question at all.

Give your output in the following format: `"next action": ""`

DIALOGUE: [Insert 1]

INFORMATION TO EXTRACT: [Insert 3]

FORM: [Insert 2]

Question Extractor

You are given a section from a form in JSON format and are asked to summarize it into a single sentence. The section might contain multiple fields, all of which need to be part of the summary sentence. Be sure to mention if the field in the section relies on another field.

Your output should follow the following format: `"summary": ""`
GIVEN SECTION: [Insert]

Question Grouping

You are given a number of summaries which need to be grouped by their semantic commonalities. The summaries are given in the following format `"name": "summary"`. Each summary needs to be grouped and each group contains at most 5 summaries. Choose group names based on the semantic commonalities of the summaries. The output has the following format:

`"first group name": list of summary names in that group e.g. ["name of summary 3", "name of summary 2"], "second group name": list of its members e.g. ["name of summary 5"]`. Only output the groups with their members without any explanation or additional information.

SUMMARIES: [Insert]

Figure 4: Prompt templates used for interactive and two task-performing modules

TEMPLATE 7.0.4

Grouping Validator

Given a number of summaries and a grouping of these summaries based on semantic commonalities, decide if there is a better grouping of the given summaries. If the given grouping is already the best grouping of the summaries, return the given grouping and nothing else. If there is a better way to group the summaries return the changed grouping in the same format as the original one, and nothing else, instead. Some guidelines: Each group can contain up to 5 summaries, each question needs to be grouped, group names need to be based on a common aspect of the summaries. Be sure to follow these guidelines and do not explain your answer.
SUMMARIES: [Insert 1]
GROUPING: [Insert 2]

Question Generator

Please generate a question. The answer to the question will be used to fill out empty fields of the given FORM. Try to ask about multiple fields in a single question if they are related. Give all possible Options if they are limited by the FORM. Only fields with an "answer" attribute can be filled out. Only ask for fields where the answer has not been given. An empty string as an answer to a field that is not required means that this field is to be left empty. Also specify which fields the question is targeted towards by listing the field names. Your output must have the following format: "question": "", "fields": []
FORM: [Insert 1]

Follow-up Question Generator

You are given a short dialogue and information which needs to be asked for. You are also given the form section, which the dialogue is based on. It was determined that the form can not be filled out based on the dialogue. Generate a new question in which you state what is considered a valid answer. Your output should be in the following format: "question": ""
DIALOGUE: [Insert 1]
INFORMATION TO EXTRACT: [Insert 2]
FORM SECTION: [Insert 3]

TEMPLATE 7.0.5

Repeated Question Generator

You are given a short dialogue and the last answer did not match what was expected. You are also given the form section, which the dialogue is based on. Your task is to write a new question that asks for the same information but also specifies what was wrong with the previous answer and nudges the User towards giving a better answer. Only return the filled out return form and nothing else.
RETURN FORM: "new question": ""
DIALOGUE: [Insert 1]
INFORMATION TO BE EXTRACTED: [Insert 2]
FORM SECTION: [Insert 3]

Information Extractor

You are given a short DIALOGUE. Extract information as facts about the topic discussed in the given DIALOGUE. The information needs to be stored as key, value pairs of strings of text and no other forms of data structures. Your output should have the following format: "key": "value", ... for example an acceptable output would be "day of the week": "monday", "car": "corvette"
DIALOGUE: [Insert 2]

Form Filler

You are given information and a section of a form in json format. You are also given the fields on which to work. If an optional field is to be left empty, write an empty string as the answer. Return the entire form section after filling out fields that have been answered.
FORM SECTION: [Insert 1]
INFORMATION: [Insert 2]
FIELDS TO WORK ON: [Insert 3]

Answer Validator

You are given a FORM SECTION and some INFORMATION. Please make sure that the FORM SECTION has been filled out correctly, based on the given INFORMATION. Make any changes necessary to the FORM SECTION (if any) and return the entire FORM SECTION including your adjustments.
FORM SECTION: [Insert 1]
INFORMATION: [Insert 2]

Figure 5: Prompt templates used for task-performing modules

Social Security SS-5 Form

1. Name

1.1. Name to be shown on card.

1.1.1. First Name (text, required)

1.1.2. Full middle name (text, optional)

1.1.3. Last Name (text, required)

1.2. Full name at birth

Info

These fields need to be filled out if the full name at birth is different than the Name on the Card

1.2.1. First Name (text, required)

1.2.2. Full middle name (text, optional)

1.2.3. Last Name (text, required)

2. Social security number previously assigned to the person listed in item 1 (text, required)

3. Place of birth

3.1. City (text, required)

3.2. State or foreign country (text, required)

4. Date of birth (text, required)

5. Citizenship (single-choice, required)

Options

U.S. Citizen, Legal Alien Allowed to work, Legal Alien not allowed to work, other

6. Ethnicity – Are you Hispanic or Latino? (single-choice, optional)

Options

Yes, No

7. Race (multi-choice, required)

Options

Native Hawaiian, Alaska Native, Asian, American Indian, Black/African American, Other Pacific Islander, White

8. Sex (single-choice, required)

Options

Male, Female

9. Mother

9.1. Mother's Name at birth

9.1.1. First name (text, required)

9.1.2. Full middle name (text, optional)

9.1.3. Last name (text, required)

9.2. Mother's social security number (text, optional)

10. Father

10.1. Father's name at birth

10.1.1. First name (text, required)

10.1.2. Full middle name (text, optional)

10.1.3. Last name (text, required)

10.2. Father's social security number (text, optional)

11. Has the person listed in item 1 or anyone acting on his/her behalf ever filed for or received a social security number card before? (single-choice, required)

Info

If yes, answer 12-13 else skip to 14.

Figure 6: SS5 - Application for a Social Security Form

Environmental Violation Report Form

1. Suspected Violator's Name (text, required)
2. Suspected Violation Location (text, required)
3. Suspected Violation City (text, required)
4. Suspected Violation State (text, required)
5. Suspected Violation ZIP Code (text, required)
6. Responsible Party (single-choice, required)

Options

Individual, Company, Government/Military, Unknown

7. Is the suspected Violation still occurring? (single-choice, required)

Options

Yes, No

8. Date of incident (text, required)

Info

Enter Date in DD.MM.YYYY format.

9. Is this an emergency? (single-choice, required)

Options

Yes, No

10. Intention (single-choice, required)

Options

Accidental, Intentional, Unknown

11. Violation Method (single-choice, required)

Options

Release, Dump/Buried, Spill, Spray, Fill, Falsified

12. Affected Subject(s) (single-choice, required)

Options

Land, Water, Air, Worker, Documents

13. Violation Description (text, required)

Info

Include a detailed description of the violation. For example, gas drilling, drum dumping etc. If necessary, include specific directions.

14. Reporter Contact Information

Info

You are not required to provide your contact information, but the EPA might want to reach out to you for additional information.

- 14.1. Your Name (text, optional)
- 14.2. Your Email (text, optional)
- 14.3. Your Phone Number (text, optional)

Figure 7: EPA – Report Environmental Violations Form

Medical Form

1. **Your Name**
 - 1.1. Title (text, optional)
 - 1.2. Surname (text, required)
 - 1.3. First Name (text, required)
2. **Your date of birth (text, required)**
3. **Your Sex (single-choice, required)**

Options
Male, Female
4. **Your Address (text, required)**
5. **Your Postcode (text, required)**
6. **Your Occupation (text, required)**
7. **Your home telephone number (text, optional)**
8. **Your mobile number (text, required)**
9. **Your emergency contact**
 - 9.1. Name (text, required)
 - 9.2. Phone Number (text, required)
 - 9.3. Relationship to you (text, required)
10. **Your best interest contact**
 - 10.1. Name (text, required)
 - 10.2. Phone number (text, required)
 - 10.3. Relationship to you (text, required)
11. **Your Doctor's details**
 - 11.1. Doctor's name (text, required)
 - 11.2. Doctor's Phone number (text, required)
 - 11.3. Doctor's Address (text, required)
 - 11.4. Doctor's Postcode (text, required)
12. **Do you weigh (single-choice, required)**

Options
Less than 21 stone (133kg), Between 21 & 35 stone (133-222kg), more than 35 stone (222kg)
13. **Do you have (multi-choice, required)**

Options
Hearing Loss?, Sight Loss?, Mobility Problems?, None of the above
14. **How many units of alcohol do you drink per week? (text, required)**

Info
A unit is half a pint of lager, a single measure of spirits or a small glass of wine.
15. **Do you smoke tobacco products? (yes/no, required)**
16. **If you smoke tobacco products, how many daily? (text, optional)**
17. **If you don't smoke tobacco products, have you smoked in the past? (yes/no, optional)**
18. **Do you chew tobacco, pan or use gutkha? (yes/no, required)**
19. **If you do not chew tobacco, pan or use gutkha, have you done so in the past? (yes/no, optional)**
20. **Are you currently Receiving treatment from a doctor, hospital or clinic? (yes/no, required)**
21. **Are you currently Taking any prescribed medicines? (yes/no, required)**

Info
This includes tablets, inhalers, injections, contraceptives and ointments.

Figure 8: MED - Confidential Medical History Form

Invention Disclosure Form

1. **What is the Title of the Invention? (text, required)**
2. **What category does the invention fall into? (single-choice, required)**

Options

Nano-Technology, Computational and Efficiency Enhancers, Biotechnology and Agro-medicine, Health Care/Medical Devices, Software, Manufacturing/Process Enhancers, other

Info

if you choose other, please answer the following question.

3. **What category would you assign to the invention? (text, optional)**
4. **Choose all categories that apply to this invention. (multi-choice, required)**

Options

New Process, New Device, New Product, New User of an existing Process/Product, New Composition of Matter, Improvement to an existing Process/Product

5. **When was this invention conceived? (text, required)**
6. **Describe how this invention came to be. (text, required)**
7. **Have working prototypes, product apparatus or compositions, etc. been made and tested? (single-choice, required)**

Options

Yes, No

8. **Give a detailed description of the Invention. (text, required)**
9. **Describe the particular problem the invention seeks to solve. (text, required)**
10. **What existing technologies or products solve or attempt to solve the same or similar problems? (text, required)**
11. **What advantage(s) does this invention provide over existing technologies or products? (text, required)**
12. **What novel and/or unusual features distinguish this invention from existing technologies or products? (text, required)**
13. **What are possible uses for the invention? (text, required)**

Info

List all possible commercial applications of the invention. In addition to immediate applications, are there other uses that might be realized in the future (i.e., other contemplated forms of the invention or alternate aspects/uses)?

14. **What research plans do you have for further development of the invention? (text, required)**

Info

Are plans in progress? Scheduled? Are plans dependent upon commercial or federal sponsorship?

Figure 9: INV - Invention Disclosure Form



Figure 10: Dialogue transcripts of modular setup based on *gpt-3.5-turbo* on *SS5* and *MED* forms

```

CURRENT STATE: {"fields": {"Is the suspected Violation still occurring?": "empty"}}
Chunk Filler:
{"next action": "question_generation"}
FORM: {"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["yes", "no"], "answer": null}}
Question Generator:
{"question": "Is the suspected Violation still occurring?", "fields": ["Is the suspected Violation still occurring?"]}
QUESTION: {"question": "Is the suspected Violation still occurring?", "fields": ["Is the suspected Violation still occurring?"]}
User:
{"answer": "Yes, the suspected violation is still occurring."}
DIALOGUE: [{"Assistant": "Is the suspected Violation still occurring?"}, {"User": "Yes, the suspected violation is still occurring."}] FORM: {"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["yes", "no"], "answer": null}}
Answer Parser:
{"next action": "information_extraction"}
DIALOGUE: [{"Assistant": "Is the suspected Violation still occurring?"}, {"User": "Yes, the suspected violation is still occurring."}]
Information Extractor:
{"suspected violation": "still occurring"}
FORM SECTION: {"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["yes", "no"], "answer": null}} INFORMATION: {"suspected violation": "still occurring"} FIELDS TO WORK ON: ["Is the suspected Violation still occurring?"]
Form Filler:
{"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["no"], "answer": "yes"}}
CURRENT STATE: {"fields": {"Is the suspected Violation still occurring?": "answered"}}
Chunk Filler:
{"next action": "answer_validation"}
FORM SECTION: {"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["yes", "no"], "answer": "yes"}} INFORMATION: {"suspected violation": "still occurring"}
Answer Validator:
{"Is the suspected Violation still occurring?": {"type": "checkbox", "required": true, "options": ["yes", "no"], "answer": "yes"}}
CURRENT STATE: {"fields": {"Is the suspected Violation still occurring?": "validated"}}

```

(a) EPA Form

```

CURRENT STATE: {"chunks": {"Title of the Invention": "empty", "Category of the Invention": "empty", "Conception of the Invention": "empty", "Prototypes and Testing": "empty", "Description of the Invention": "empty", "Problem the Invention Seeks to Solve": "empty", "Group 1": "empty", "Group 2": "empty"}, "last action": ["form_chunks"]}
Dialogue Manager:
{"next action": "fill_chunk", "chunk to work on": "Title of the Invention"}
CURRENT STATE: {"fields": {"What is the title of the Invention?": "empty"}, "last action": [], "last question": null}
Chunk Filler:
{"next action": "question_generation"}
FORM: {"What is the title of the Invention?": {"type": "text-field", "required": true, "answer": null}}
Question Generator:
{"question": "What is the title of the Invention?", "fields": ["What is the title of the Invention?"]}
QUESTION: {"question": "What is the title of the Invention?", "fields": ["What is the title of the Invention?"]}
User:
{"answer": "The WaveTracker"}
DIALOGUE: [{"Assistant": "What is the title of the Invention?"}, {"User": "The WaveTracker"}] INFORMATION TO EXTRACT: ["What is the title of the Invention?"] FORM: {"What is the title of the Invention?": {"type": "text-field", "required": true, "answer": null}}
Answer Parser:
{"next action": "information_extraction"}
DIALOGUE: [{"Assistant": "What is the title of the Invention?"}, {"User": "The WaveTracker"}]
Information Extractor:
{"title of the Invention": "The WaveTracker"}
FORM SECTION: {"What is the title of the Invention?": {"type": "text-field", "required": true, "answer": null}} INFORMATION: {"title of the Invention": "The WaveTracker"} FIELDS TO WORK ON: ["What is the title of the Invention?"]
Form Filler:
{"What is the title of the Invention?": {"type": "text-field", "required": true, "answer": "The WaveTracker"}}
CURRENT STATE: {"fields": {"What is the title of the Invention?": "answered"}, "last action": ["question_generation"], "last question": "What is the title of the Invention?"}

```

(b) INV Form

Figure 11: Dialogue transcripts of the modular setup based on *gpt-3.5-turbo* on *EPA* and *INV* forms