

# A Few-shot Approach to Task-oriented Dialogue Enhanced with Chitchat

Armand Stricker and Patrick Paroubek

Université Paris-Saclay, CNRS, Laboratoire Interdisciplinaire des Sciences du Numérique  
91400, Orsay, France

{armand.stricker, patrick.paroubek}@lisn.upsaclay.fr

## Abstract

Large language models (LLMs) tuned for chat have recently been adopted for few-shot end-to-end task-oriented dialogue (TOD), with some success. To further assess this method, we conduct experiments on two, more complex, task-oriented benchmarks that integrate elements of chitchat into the conversation. We enhance a few-shot baseline by adding zero-shot chitchat detection and implementing function calling for dialogue state tracking (DST). We focus on this initial step in the TOD pipeline as errors due to added chitchat at this stage have a higher chance of impacting overall performance. We find that this prompting method shows increased resilience to mixed-mode inputs and our enhanced pipeline allows for natural inter-mode conversations, as assessed through human evaluation. Our findings also suggest that the performance gap between few-shot prompting and supervised task-specific models is narrowing.

## 1 Introduction

As chat-tuned LLMs continue to advance in fluency and instruction-following thanks to approaches leveraging human feedback (Bai et al., 2022), the prospect of developing a functional and conversational TOD system with a few dialogue examples becomes increasingly plausible. However, these models are predominantly evaluated using benchmarks characterized by short, explicit task-oriented requests, which do not fully reflect the complexity of more natural TOD interactions that might include situational details, or preliminary chitchat (Beaver et al., 2020; Gung et al., 2023; Young et al., 2022).

**Few-shot prompting for TOD** Recent studies have explored the potential of few-shot and zero-shot prompting approaches for DST (Saha et al., 2022) and end-to-end TOD systems. Madotto et al.

(2021) pioneer the exploration of few-shot prompting for DST, with limited results. Chen et al. (2023) adapt a meta-learning scheme for DST to stabilize the model’s ability to perform well under various prompts, using previous-generation LLMs suffering from limited context-lengths. Lesci et al. (2023) and Hu et al. (2022) frame DST as a table-based task, respectively inserting/deleting entries or generating SQL queries over tables, using few-shot prompting. For end-to-end TOD, Hudeček and Dusek (2023) and Zhang et al. (2023) propose chained prompting methods for constructing task bots with minimal human effort, highlighting DST as a critical first step. Indeed, errors in detecting the user’s constraints directly impact database search results and the subsequently generated responses.

In this paper, we focus on refining a few-shot, end-to-end approach (Hudeček and Dusek, 2023) for chitchat-enhanced TODs. The main enhancements we propose are adding a chitchat detection step, determining if a turn requires a chitchat or task-oriented response, and improving DST over the original approach, by casting it as a function call generation problem. Our enhancements aim to more effectively handle the conversational nuances arising from inter-mode contexts. With function calling, we capitalize on the extensive exposure of state-of-the-art LLMs to code during pre-training, assuming that such a prompt format is well-represented within the model’s initial training dataset and will generalize well in few-shot settings.

We evaluate our enhancements on two inter-mode benchmarks (Section 3.1). We compare our DST approach with two other DST prompt variants (Section 2.2) across both open- and closed-source LLMs. Our evaluation includes both automatic metrics and a human assessment, aiming to establish the effectiveness and robustness of a few-shot prompting approach in inter-mode dialogue.

## 2 Method

### 2.1 Few-Shot TOD Bot

We build upon Hudeček and Dusek 2023, further adapting their approach to be robust to inter-mode inputs. This baseline relies on three main prompts: one for **domain detection**, one for **DST** and one for **response generation**.

The dialogue state is progressively accumulated throughout the conversation and is utilized to retrieve entities the user may want to book from a database. At each turn, the LLM is prompted to extract the user’s constraints from their current request using a custom schema that employs colons and dashes (see **base** in Figure 2). To generate a response, the context, dialogue state and number of database results are concatenated into a prompt (as in Appendix A).

Additionally, to insert few-shot exemplars into DST and response prompts, a vector store is utilized to search and dynamically incorporate examples with similar contexts, thereby adapting the prompt to the current turn. In our implementation, we use only 10 dialogues from each domain of the MultiWOZ (Budzianowski et al., 2018) training set (< 1% of the available training dialogues) to create this vector database, following author recommendations.

### 2.2 Enhancements

**Function Calling for DST** Function calling is the ability for an LLM to interact with external APIs, databases or tools (Schick et al., 2023; Li et al., 2023). This is achieved by prompting the LLM to literally generate a function call such as `get_temp(loc='NYC')`. This approach is typically used to avoid hallucinated responses when precise, external knowledge or skills are needed. For example, *What’s the temperature in NYC?* requires access to a weather forecast API and the LLM should not attempt to respond to the query directly, as it might hallucinate the forecast based on its pretraining data.

In the context of MultiWOZ, once a domain has been selected (restaurant, taxi, hotel...), we apply a domain-specific function calling prompt. It includes a tailored function definition, formatted in JSON, detailing the name, type, and descriptions of the function’s parameters as shown in Figure 1. These parameters are in effect the possible constraints of the user which need to be determined before booking. Figure 2 depicts the function call

an LLM is expected to generate given a user request. We note that Li et al. (2024) very recently explore a similar approach for zero-shot DST, but do not evaluate it on inter-mode benchmarks.

As shown in Figure 2, we compare three prompting methods for DST. (1) Our **function** calling approach. (2) An **SQL** query generation method (Hu et al., 2022). In this framework, a user’s request is translated into an SQL query that is meant to interact with a table, containing MultiWOZ domain-specific entries. (3) An arbitrary **base** method (Hudeček and Dusek, 2023). For all approaches, the generated text is parsed to extract slots and values for DST evaluation. All prompts are shown in Appendix A.

For reference, we also **fine-tune** an LLM for end-to-end TOD (Hosseini-Asl et al., 2020). When training, we use mixed batches of examples from each benchmark and use LoRA fine-tuning (Hu et al., 2021). Training details are in Appendix B.

```
{
  "name": "find_book_restaurant",
  "description": "Find a restaurant and book a table",
  "parameters": {
    "type": "object",
    "properties": {
      "pricerange": {
        "type": "string",
        "description": "Price range of the restaurant",
        "possible_values": ["cheap", "moderate", "expensive"],
        "default_value": None,
      },
      "area": {
        "type": "string",
        "description": "Area where the restaurant is located",
        "possible_values": ["north", "east", "west", "south", "centre"],
        "default_value": None,
      },
    },
  },
  "....."
}
```

Figure 1: Function definition for restaurant booking.

**Chitchat Detection** Prior to the original domain detection prompt, we add a chitchat detection prompt, that determines whether a turn expects a *task* or *chitchat* response. This allows the model to separately handle open-ended chitchat turns during the conversation. Chitchat detection and generation prompts are shown in Appendix A.

Our experimental code can be found on GitHub<sup>1</sup>.

## 3 Experimental Setup

### 3.1 Benchmarks

We experiment with two chitchat-enhanced variants of version 2.2 (Zang et al., 2020) of MultiWOZ (Budzianowski et al., 2018). The same dialogue IDs are used across evaluation comparisons.

**FusedChat** This dataset (Young et al., 2022) prepends and appends full chitchat exchanges to

<sup>1</sup><https://github.com/armandstrickernlp/FewShot-InterModeBot>

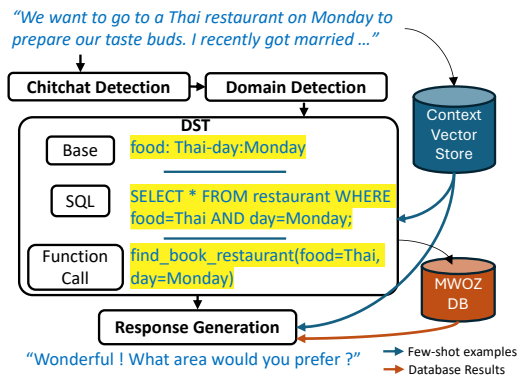


Figure 2: An overview of our augmented few-shot pipeline for chitchat-enhanced TOD. This illustrates the scenario where the chitchat detection prompt identifies a task-oriented request. We highlight the three DST prompting methods we consider along with their expected outputs (details in Section 2.2).

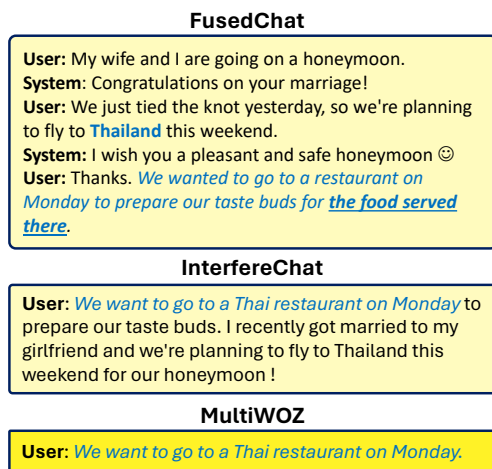


Figure 3: Side-by-side comparison of the same turn across each evaluated benchmark.

the original TODs. We focus on the subset that has *prefixed* exchanges, which adds complexity by rewriting initial TOD turns to include co-referring expressions such as *the food served there* in Figure 3. This challenges state tracking and turn detection as key task information is mixed into chitchat turns.

**InterfereChat** This dataset (Stricker and Paroubek, 2024) compresses the additional exchanges from FusedChat into a single user backstory, which is then embedded into a TOD request. Consequently, a single turn may be made up of a task request and additional situational details. This complexity makes turn type detection challenging, and may lead the model to overlook the user’s request and output a response which only focuses on the chatty details (Figure 3).

### 3.2 Model Zoo

We use both open- and closed-source LLMs to evaluate the impact of our enhancements.

- **Llama3-8B-Instruct**<sup>2</sup> and **Llama3-70B-Instruct**<sup>3</sup> are two recent, state-of-the-art LLMs, trained on 15 trillion tokens. **Llama2-13b-chat**<sup>4</sup> is an older variant of the Llama family, trained on 2 trillion tokens. We use these models in a few-shot setting only.
- **Gorilla-openfunctions-v2**<sup>5</sup> is a 7B model specifically fine-tuned for function calling. We use it exclusively for DST prompting in few- and zero-shot settings, while utilizing Llama3-8B-Instruct for remaining prompts.
- **GPT-3.5-turbo-0125**<sup>6</sup> is the only closed-source model we consider, given its cost-effectiveness. We test this model’s zero-shot function calling capability, as API calls to the model natively accept function definitions.

**Evaluation** We measure the impact of adapting the pipeline by performing an end-to-end evaluation, with the recommended evaluation toolkit<sup>7</sup>. Joint goal accuracy (**JGA**) and **Slot-F1** measure dialogue state predictions, with JGA counted as correct if all slots and values match the reference. **Success** rate assesses dialogue success overall, measuring whether the user’s desired goal was reached. See the MultiWOZ paper for more details. For response quality, we report **BLEU** (Papineni et al., 2002) and **BLEU-aug**, which measures BLEU only on responses which follow augmented turns from InterfereChat.

## 4 Results and Discussion

**Zero-shot chitchat detection** The results presented in Table 2 demonstrate that the selected LLMs generally perform well in turn classification across both benchmarks. Nevertheless, we note that turns which contain both useful task information and chitchat elements are challenging to classify as they appear in both benchmarks, with different labels (*chitchat* for FusedChat and *task* for InterfereChat). We see that different models exhibit varying classification biases. For instance, GPT-3.5-turbo tends to favor chitchat classification, as

<sup>2</sup><https://huggingface.co/meta-llama/Meta-Llama-3-8B-Instruct>

<sup>3</sup>[meta-llama/Meta-Llama-3-70B-Instruct](https://huggingface.co/meta-llama/Meta-Llama-3-70B-Instruct)

<sup>4</sup><https://huggingface.co/meta-llama/Llama-2-13b-chat-hf>

<sup>5</sup><https://huggingface.co/gorilla-llm/gorilla-openfunctions-v2>

<sup>6</sup><https://platform.openai.com/docs/models/gpt-3-5-turbo>

<sup>7</sup>[https://github.com/Tomiinek/MultiWOZ\\_Evaluation](https://github.com/Tomiinek/MultiWOZ_Evaluation)

Model	FusedChat				InterfereChat				MultiWOZ 2.2				JGA Drop (Fused)	
	BLEU	JGA	Slot-F1	Success	BLEU	BLEU-aug	JGA	Slot-F1	Success	BLEU	JGA	Slot-F1		Success
Supervised SotA	12.6 $\triangle$	58.4 $\triangle$	—	60 $\triangle$	22.0 $\blacktriangle$	25.0	64.0 $\blacktriangle$	—	73.2 $\blacktriangle$	19.90 $\blackstar$	60 $\diamond$	—	82 $\heartsuit$	—
Llama3-8B-Instruct <sub>finetune</sub>	20.55	74.8	92.52	73.4	17.28	80.87	76.78	92.92	74.8	20.56	76.84	93.03	76.2	2.04
Llama3-8B-Instruct <sub>func</sub>	4.94	<b>54.32*</b>	<b>74.95*</b>	31.6	4.39	<b>77.07</b>	<b>54.26*</b>	<b>77.46*</b>	<b>29.9</b>	5.20	<b>55.85*</b>	<b>78.73*</b>	<b>31.3</b>	<b>1.53</b>
Llama3-8B-Instruct <sub>sql</sub>	4.83	33.20	68.59	30.3	4.24	<b>77.07</b>	38.87	72.48	27.0	5.16	41.97	73.46	28.7	8.77
Llama3-8B-Instruct <sub>base</sub>	<b>5.53*</b>	36.58	67.65	<b>33.4</b>	<b>4.78*</b>	63.23	40.45	70.65	29.7	<b>5.80*</b>	39.40	71.90	29.9	2.82
Llama2-13b-chat <sub>func</sub>	2.27	<b>25.69*</b>	<b>56.46*</b>	<b>21.5</b>	<b>2.13</b>	43.44	<b>21.46*</b>	<b>58.15*</b>	<b>21.1</b>	2.32	<b>26.88*</b>	<b>60.08*</b>	<b>23.6</b>	1.19
Llama2-13b-chat <sub>sql</sub>	<b>2.48*</b>	17.88	49.18	19.1	2.12	46.35	13.34	52.55	20.9	2.32	17.17	54.33	21.7	<b>-0.71</b>
Llama2-13b-chat <sub>base</sub>	2.16	7.11	35.34	10.1	2.07	<b>49.70*</b>	11.17	45.13	13.5	<b>2.50</b>	15.13	49.20	12.29	8.02
Llama3-70B-Instruct <sub>func</sub>	6.52	<b>72.86*</b>	84.05	42.8	5.99	<b>68.73*</b>	<b>68.63*</b>	84.53	42.6	6.72	<b>72.35*</b>	85.0	45.9	<b>-0.51</b>
Llama3-70B-Instruct <sub>sql</sub>	6.50	51.05	<b>84.20</b>	45.9	6.01	51.62	50.40	85.21	44.5	6.67	52.67	85.51	45.1	1.62
Llama3-70B-Instruct <sub>base</sub>	<b>6.94*</b>	60.25	84.19	<b>47.5</b>	<b>6.58*</b>	65.80	61.60	<b>85.31</b>	<b>46.5</b>	<b>7.34*</b>	65.58	<b>86.32</b>	<b>48.6*</b>	5.33
Gorilla-v2 <i>few-shot DST</i>	4.91	54.86	78.52	38.7	4.48	77.07	56.16	79.95	33.8	5.36	59.46	81.44	33.8	4.6
Gorilla-v2 <i>0-shot DST</i>	5.07	61.07	66.64	33.0	4.51	31.93	60.25	69.52	34	5.36	64.54	71.24	33.2	3.47
GPT-3.5-turbo <i>0-shot DST</i>	6.36	51.08	71.01	39.7	5.51	22.40	51.0	72.22	41.0	6.46	58.61	74.52	36.2	7.53

Table 1: End-to-end evaluation with mean scores over 3 seeds. \* indicates statistical significance ( $p < 0.05$ , paired t-test) between best and second best values. For reference and context, rows in **yellow** show ours and previous supervised results:  $\triangle$ Young et al. (2022),  $\blacktriangle$ Stricker and Paroubek (2024),  $\blackstar$ Sun et al. (2023),  $\diamond$ Huang et al. (2023),  $\heartsuit$ Feng et al. (2023). Results in **blue** use few-shot prompting for DST with general purpose LLMs, which have not been explicitly adapted to the task of function calling. For comparison, rows in **green** use various models which have been tuned for function calling. Gorilla-v2 is used only for DST and relies on Llama3-8B-Instruct for response generation. Response generation is few-shot in all cases. Best results per Llama model size are in bold. We also show the relative drop in JGA between MultiWOZ and FusedChat results.

evidenced by its higher score on  $Acc_{fused}$ . In contrast, Llama3-8B-Instruct shows a propensity for classifying these turns as *task*-related, reflected in its higher  $Acc_{backstory}$  score. Llama3-70B-Instruct appears to strike a balance between these two tendencies. Overall we find this prompting approach to be functional, notably when it comes to interacting with the models in a live setting (Section 4). We plan nonetheless to refine this approach in future work.

Model	$Acc_{fused}$	$Acc_{interfere}$	$Acc_{backstory}$
Llama3-8B-Instruct	0.86	<b>0.98</b>	<b>0.91</b>
Llama2-13b-chat	0.85	0.81	0.45
Llama3-70B-Instruct	0.87	<b>0.98</b>	0.86
GPT-3.5-turbo	<b>0.89</b>	0.96	0.80

Table 2: Accuracy of zero-shot chitchat detection on both chitchat-enhanced benchmarks. *backstory* evaluates accuracy only on the InterfereChat turns that are augmented with contextual details, referred to as backstories.

**End-to-end evaluation** The results presented in Table 1 demonstrate that function calling consistently and generally outperforms other methods with statistical significance. This approach offers greater performance on inter-mode benchmarks and achieves the highest Joint Goal Accuracy across various model sizes. It is particularly noteworthy that Llama3-70B-Instruct surpasses previous supervised state-of-the-art JGA baselines on the benchmarks studied. We additionally find no statistically significant difference with Llama3-8B-Instruct<sub>finetune</sub> on the FusedChat benchmark. This indicates that this method is effective in retrieving task-information embedded in chitchat turns. We

notice similar Slot-F1 scores across benchmarks, showing the model can retrieve relevant information with all approaches, but performs best with function calls. This performance does come at a computational cost however, given the large scale of the model. Nonetheless, quantization strategies may be explored to improve latency and storage efficiency (Dettmers et al., 2024).

We observe that improved JGA does not necessarily translate to better task success, as no statistically significant difference is found in this regard.

BLEU-aug scores are generally high, mainly because the reference inter-mode responses from InterfereChat were generated with Llama2-70B-chat, a model from the same family. As we can see, GPT-3.5-turbo performs more poorly on this metric.

Llama3-8B achieves comparable and, in some cases, improved JGA and Slot-F1 scores compared to GPT-3.5-turbo, albeit requiring few-shot prompting instead of zero-shot. Gorilla-openfunctions-v2, a model specifically designed for function calling, performs better in a zero-shot setting than in a few-shot one on JGA, but not on Slot-F1. In the few-shot setting, its performance is on par with Llama3-8B.

The success of function calls with general LLMs can be attributed to two key factors: their widespread presence across various programming languages, which ensures their frequent occurrence in the code sections of LLM pre-training data, and their straightforward, easily producible syntax. These characteristics enable a more seamless conversion of user inputs into a structured format, where both the function name and its param-

eters closely align with the semantic content of the request. This semantic proximity suggests that chitchat enhancements may not be as disruptive as they might be with other approaches, provided they maintain coherence within the task context. In contrast, while SQL is also present in pre-training data, research by Tan et al. (2024) indicates that optimizing Text-to-SQL performance requires a more complex prompting strategy. This additional complexity may account for SQL’s comparatively lower performance in this scenario.

Model	Quality	Success	JGA	Clarify↓
Llama3-8B-Instruct	0.58	<b>90.0</b>	69.17	0.75
Llama3-70B-Instruct	<b>0.76</b>	<b>90.0</b>	<b>94.54</b>	<b>0.15</b>
GPT-3.5-turbo	0.70	85.0	76.25	0.65

Table 3: Human evaluation results. Quality is normalized ( $[0, 1]$  scale), success and JGA are percentages, and Clarify is the average number of reformulations needed per dialogue.

**Human evaluation** We also conduct a small in-house human evaluation, focusing on models capable of supporting the complete pipeline: Llama3-8B-Instruct, Llama3-70B-Instruct, and GPT-3.5-turbo with zero-shot DST. We randomly select 20 single-domain TOD goals from MultiWOZ, along with their corresponding backstories from InterferChat. Four NLP experts were tasked with achieving these goals while engaging in chitchat with the models, mimicking the inter-mode scenarios from our selected benchmarks.

During our initial pilot annotation, we observed that participants struggled to simultaneously focus on the task goal and the contextual chitchat details. To address this, we introduced a seed turn to initiate the conversation and streamline the task. This seed turn could either be a task request incorporating contextual details or an open-ended chitchat utterance. When presented with the latter, participants engaged solely in chitchat for up to three turns before transitioning to the task goal.

Post-interaction, participants evaluated the dialogue quality (“Was the system friendly and engaging? (1-5)”) and success (“Was the desired goal reached? (Yes/No)”). We normalized the quality scores to a  $[0, 1]$  scale. For successful dialogues, we tallied the number of queries that needed repeating. Additionally, we manually calculated JGA by examining the dialogue state at each turn in relation to the user’s request. We included dialogue states during chitchat turns as well.

Overall, we find the resulting inter-mode interactions to be quite natural and successful, more

than automatic metrics indicate. As Table 3 indicates, Llama3-70B-Instruct performs well in terms of JGA and success rate, with only the rare clarification needed. This model is also preferred in terms of quality of the interaction, with more engaging chitchat and TOD responses. The smaller Llama3 model also does quite well, with more successful interactions than GPT-3.5-turbo.

The models demonstrate several positive attributes in handling inter-mode dialogues. All models show an ability to recall and incorporate chitchat details from the beginning of the dialogue when concluding the interaction, a feature that annotators particularly appreciated. Additionally, even when the wrong turn type is predicted, responses often remain coherent and contextual, allowing for the conversation not to break down. State-tracking-wise, the Llama3 models succeed at following function descriptions, accurately using the specified possible values even when users employ synonyms. This precision is crucial for successful database searches. These capabilities contribute to more natural and context-aware conversations, enhancing the overall user experience.

Despite these strengths, several issues were identified. Firstly, responses to inter-mode turns often focused heavily on the task request while neglecting contextual details. A more significant concern is the hallucination of novel entities (such as restaurant names). This issue can mislead users into believing a task was successfully completed when it was not. Lastly, unlike Llama3 models, GPT-3.5-turbo tends to directly extract the user’s words when performing state tracking, potentially leading to database lookup errors (*expensive* vs. *pricey*). Interaction examples with qualitative analyses can be found in Appendix C).

## 5 Conclusion

We show that a few-shot TOD baseline can be successfully extended to handle inter-mode inputs. We find that casting DST as function call generation is robust and effective across various LLMs, substantially outperforming other few-shot state-tracking methods. Human evaluation results show the promising potential of an inter-mode few-shot TOD bot, overall successfully balancing chitchat and TOD within a single interaction. These findings suggest a significant advancement in simply and swiftly building more versatile and natural dialogue systems.

## 6 Acknowledgments

This work was granted access to the HPC resources of IDRIS under the allocation 20XX-AD011014510 made by GENCI.

## References

- Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.
- Ian Beaver, Cynthia Freeman, and Abdullah Mueen. 2020. Towards awareness of human relational strategies in virtual agents. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(03):2602–2610.
- 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.
- Derek Chen, Kun Qian, and Zhou Yu. 2023. Stabilized in-context learning with pre-trained language models for few shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 1551–1564, Dubrovnik, Croatia. Association for Computational Linguistics.
- Tim Dettmers, Ruslan A. Svirschevski, Vage Egiazarian, Denis Kuznedelev, Elias Frantar, Saleh Ashkboos, Alexander Borzunov, Torsten Hoefler, and Dan Alistarh. 2024. SpQR: A sparse-quantized representation for near-lossless LLM weight compression. In *The Twelfth International Conference on Learning Representations*.
- Yihao Feng, Shentao Yang, Shujian Zhang, Jianguo Zhang, Caiming Xiong, Mingyuan Zhou, and Huan Wang. 2023. Fantastic rewards and how to tame them: A case study on reward learning for task-oriented dialogue systems. *arXiv preprint arXiv:2302.10342*.
- James Gung, Emily Moeng, Wesley Rose, Arshit Gupta, Yi Zhang, and Saab Mansour. 2023. NatCS: Eliciting natural customer support dialogues. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9652–9677, Toronto, Canada. Association for Computational Linguistics.
- Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A Simple Language Model for Task-Oriented Dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191. Curran Associates, Inc.
- Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.
- Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.
- Tianjian Huang, Shaunak Ashish Halbe, Chinnadhurai Sankar, Pooyan Amini, Satwik Kottur, Alborz Geramifard, Meisam Razaviyayn, and Ahmad Beirami. 2023. Robustness through data augmentation loss consistency. *Transactions on Machine Learning Research*.
- Vojtěch Hudeček and Ondrej Dusek. 2023. Are large language models all you need for task-oriented dialogue? In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 216–228, Prague, Czechia. Association for Computational Linguistics.
- Pietro Lesci, Yoshinari Fujinuma, Momchil Hardalov, Chao Shang, Yassine Benajiba, and Lluís Marquez. 2023. Diable: Efficient dialogue state tracking as operations on tables. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 9697–9719, Toronto, Canada. Association for Computational Linguistics.
- Minghao Li, Yingxiu Zhao, Bowen Yu, Feifan Song, Hangyu Li, Haiyang Yu, Zhoujun Li, Fei Huang, and Yongbin Li. 2023. API-bank: A comprehensive benchmark for tool-augmented LLMs. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3102–3116, Singapore. Association for Computational Linguistics.
- Zekun Li, Zhiyu Zoey Chen, Mike Ross, Patrick Huber, Seungwhan Moon, Zhaojiang Lin, Xin Luna Dong, Adithya Sagar, Xifeng Yan, and Paul A. Crook. 2024. Large language models as zero-shot dialogue state tracker through function calling. *Preprint*, arXiv:2402.10466.
- Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *Preprint*, arXiv:2110.08118.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Debjoy Saha, Bishal Santra, and Pawan Goyal. 2022. [A study on prompt-based few-shot learning methods for belief state tracking in task-oriented dialog systems](#). *Preprint*, arXiv:2204.08167.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessi, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. [Toolformer: Language models can teach themselves to use tools](#). In *Advances in Neural Information Processing Systems*, volume 36, pages 68539–68551. Curran Associates, Inc.

Armand Stricker and Patrick Paroubek. 2024. [Chitchat as interference: Adding user backstories to task-oriented dialogues](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 3203–3214, Torino, Italy. ELRA and ICCL.

Haipeng Sun, Junwei Bao, Youzheng Wu, and Xiaodong He. 2023. [Mars: Modeling context & state representations with contrastive learning for end-to-end task-oriented dialog](#). In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 11139–11160, Toronto, Canada. Association for Computational Linguistics.

Zhao Tan, Xiping Liu, Qing Shu, Xi Li, Changxuan Wan, Dexi Liu, Qizhi Wan, and Guoqiong Liao. 2024. [Enhancing text-to-SQL capabilities of large language models through tailored promptings](#). In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6091–6109, Torino, Italy. ELRA and ICCL.

Tom Young, Frank Xing, Vlad Pandeale, Jinjie Ni, and Erik Cambria. 2022. Fusing task-oriented and open-domain dialogues in conversational agents. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 11622–11629.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. [MultiWOZ 2.2 : A dialogue dataset with additional annotation corrections and state tracking baselines](#). In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117, Online. Association for Computational Linguistics.

Xiaoying Zhang, Baolin Peng, Kun Li, Jingyan Zhou, and Helen Meng. 2023. [SGP-TOD: Building task bots effortlessly via schema-guided LLM prompting](#). In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 13348–13369, Singapore. Association for Computational Linguistics.

## A Prompt Templates

We depict the various prompts created for our experiments. See Figure 4 for function calling DST, Figure 5 for SQL DST, Figure 6 for baseline DST, Figure 7 for response generation, Figure 8 for chitchat

detection, Figure 9 for task-oriented domain detection and Figure 10 for chitchat generation.

```

System Message
You are a task-oriented conversational AI assistant that helps users to book restaurants. Use the function definition below to create a function call with the correct arguments for the user's booking.

{"name": "find_book_restaurant",
 "description": "Find a restaurant and book a table",
 "parameters": {
  "type": "object",
  "properties": {
    "pricerange": {
      "type": "string",
      "description": "Price range of the restaurant",
      "possible_values": ["cheap", "moderate", "expensive"],
      "default_value": None,
    },
    "area": {
      "type": "string",
      "description": "Area where the restaurant is located",
      "possible_values": ["north", "east", "west", "south", "centre"],
      "default_value": None,
    },
  },
}

-----
"call_example": "find_book_restaurant(pricerange=None, area=centre,
food='italian', name='pizza hut city centre', bookday='wednesday',
booktime='13:30', bookpeople=7)"
}

User Message
Output a function call with the correct function arguments given the customer's request. Make sure to follow the function definition. Focus only on the values mentioned in the last utterance.

-----
-----Example 0:
Context:
Customer: I'm looking for an Italian restaurant for Saturday.
Assistant: <<function>>find_book_train(bookday='Saturday', food='Italian')
-----

Now complete the following example:
Context:
Customer: Hello, I am looking for a restaurant in Cambridge. I believe it is called Golden Wok.

```

Figure 4: Function calling DST prompt for the restaurant domain. Few-shot exemplars can be optionally added. We add 5 in all our few-shot experiments. We find that adding a *default\_value* field and one to several call examples helps improve performance.

## B End-to-end Training Details

We use the SimpleToD framework (Hosseini-Asl et al., 2020) to train an end-to-end TOD system, while relying only on a single language model. Each training example is composed of a concatenated text sequence which includes a dialogue context, the dialogue state for the last user turn, the database state (number of available options), response dialogue acts, and a delexicalized natural language response. We show an input example in Figure 11. Each training batch contains examples from each benchmark in equal proportion. During inference, we pass only the dialogue context to the model. Once the dialogue state is generated, we interrupt generation to fetch the database state, concatenate it to the current output and then continue generating.

We train the language model using LoRA (Hu et al., 2021), using a rank  $r$  of 64. We set the scaling  $\alpha$  to  $2r$ , and target the key, query, value and output weight matrices of the self-attention module. This amounts to roughly 1% of total parameters being trained. We use a learning rate of  $5e-5$ . We

```

System Message
You are a task-oriented conversational AI assistant that helps users to book
restaurant. Using valid SQLite, answer the following multi-turn conversational
questions for the table provided below.

CREATE TABLE restaurant(
name text,
food text,
pricerange text CHECK (pricerange IN (dontcare, cheap, moderate, expensive)),
area text CHECK (area IN (centre, east, north, south, west)),
booktime text,
bookday text,
bookpeople int
)
/*
5 example rows:
SELECT * FROM restaurant LIMIT 5;
name food pricerange area booktime bookday bookpeople
pizza hut city centre italian dontcare centre 13:30 wednesday 7
the missing sock international moderate east dontcare dontcare 2
golden wok chinese moderate north 17:11 friday 4
cambridge chop house dontcare expensive center 08:43 monday 5
darrys cookhouse and wine shop modern european expensive center 11:20
saturday 8
*/

User Message
Write a valid SQL query to extract the information from the Table given the
customer's request. Make sure to end with a semicolon. Focus only on the
values mentioned in the last utterance.
-----
-----Example 0:
Context:
Customer: I'm looking for an Italian restaurant for Saturday.
Assistant: SELECT * FROM restaurant WHERE bookday = saturday AND food =
italian;
-----
Now complete the following example:
Context:
Customer: Hello, I am looking for a restaurant in Cambridge. I believe it is
called Golden Wok.

```

Figure 5: SQL DST prompt for the restaurant domain. Few-shot exemplars can be optionally added. We add 5 in all our few-shot experiments. We follow (Hu et al., 2022) for this approach.

train for up to 2 epochs with early stopping on a single 80Gb A100.

## C Interactive Evaluation

We show our dialogue interface in Figure 12 and a few example interactions: Table 4, Table 5, Table 6, Table 7, show successful and unsuccessful interactions with the various models.

```

User Message
Capture entity values from last utterance of the conversation
according to examples.
Focus only on the values mentioned in the last utterance.
Capture pair "entity:value" separated by colon and no spaces in
between.
Separate entity:value pairs by hyphens.
Values that should be captured are:
- "pricerange" that specifies the price range of the restaurant
(cheap/moderate/expensive)
- "area" that specifies the area where the restaurant is located
(north/east/west/south/centre)
- "food" that specifies the type of food the restaurant serves
- "name" that specifies the name of the restaurant
- "bookday" that specifies the day of the booking
- "booktime" that specifies the time of the booking
- "bookpeople" that specifies for how many people is the booking
made
Do not capture any other values!
If not specified, leave the value empty.
-----
-----Example 0:
Context:
Customer: I'm looking for an Italian restaurant for Saturday.
Assistant: bookday:'saturday'-food:'italian'
-----
Now complete the following example:
Context:
Customer: Hello, I am looking for a restaurant in Cambridge. I believe
it is called Golden Wok.

```

Figure 6: Baseline DST prompt for the restaurant domain. Few-shot exemplars can be optionally added. We add 5 in all our few-shot experiments. We follow (Hudeček and Dusek, 2023) for this approach. They do not use a system prompt in their implementation.

```

User Message
Definition: You are an assistant that helps people to book a restaurant.
You can search for a restaurant by area, food, or pricerange.
There is also a number of restaurants in the database currently
corresponding to the user's request.
If multiple restaurants are available, the Assistant should ask for
further preferences.
If you find a possible restaurant, the Assistant should provide
[restaurant_name], [restaurant_address], [restaurant_phone] or
[restaurant_postcode] if asked. Use these exact placeholders.
If no restaurants are available, the Assistant should ask for different
preferences.
Before booking a table, the Assistant should ask for the time and the
day of the booking and number of people. The Assistant should
provide [reference] when the booking has been made. Use these
exact placeholders.
**Always act as if booking is available.**
Write the Assistant response as a single line, based on the state and
the database. Act friendly and engaging.
-----
-----Example 0:
Context:
Customer: I'm looking for an Italian restaurant for Saturday.
State: {restaurant:{food: italian, bookday: saturday}}
database:restaurant: 19
Assistant: <response> What area would you prefer ? </response>
-----
Now complete the following example:
Context:
Customer: Hello, I am looking for a restaurant in Cambridge. I believe
it is called Golden Wok.
State: {restaurant: {name: Golden Wok }}
database:restaurant: 1
Assistant:

```

Figure 7: Response prompt for the restaurant domain. We use 5 few-shot examples for all experiments.



**System Message**  
 A user is using Cambridge's Towninfo Centre information assistant which can help users with information and bookings. These are tasks with certain specific domains.  
 Task domains include:  
 - train (booking train tickets)  
 - restaurant (finding and booking restaurants)  
 - hotel (finding and booking hotels)  
 - attraction (eg. "architecture", "sports", "entertainment", "cinema", "museum", "concert", "theatre"...)  
 - taxi (booking a taxi from one location to another)  
 You are an expert at determining if a User Turn contains task-related information or requests.

**User Message**  
 Use the Dialogue Context and the User Turn to classify the User Turn as "chitchat" or "task".  
 The criteria are:  
 If the User Turn contains:  
 - comments about personal life, opinions, or experiences  
 - casual comments about Cambridge or the domains (restaurants, trains, hotels, taxis, attractions)  
 then the User Turn is "chitchat".  
 If the User Turn contains  
 - an \*explicit\* request for information in a task domain (restaurant, train, hotel, attraction, taxi) or  
 - a request to perform an action in a task domain (restaurant, train, hotel, attraction, taxi) or  
 - an essential piece of information relevant to a task domain (restaurant, train, hotel, attraction, taxi)  
 then the User Turn is "task".

Dialogue Context:  
 User Turn:  
 Customer: I'm looking for an Italian restaurant for Saturday.

Respond with one word. Either "chitchat" or "task" between <label> and </label> tags.  
 Label:

Figure 8: Zero-shot chitchat detection prompt. This is meant to detect whether to perform chitchat or activate the few-shot TOD bot pipeline.

**System Message**  
 You are a conversational AI, capable of detecting the domain of a user request.  
 The only possible domains are :  
 - train (booking train tickets)  
 - restaurant (finding and booking restaurants)  
 - hotel (finding and booking hotels)  
 - attraction (eg. "architecture", "sports", "entertainment", "cinema", "museum", "concert", "theatre", "park", "church", "hotel", "nightclub", "swimming pool", "college", "concert hall", "boat", "historical landmark", "gallery", "shopping area", "nature preserve", "sports venue", "theme park")  
 - taxi (booking a taxi from one location to another)  
 Return only the domain name.

**User Message**  
 Select the appropriate domain given the customer's request.  
 Respond with only one word: the domain name.  
 It is very important you focus on the customer's current request to make your decision.  
 Context:  
 Customer: I'm looking for an Italian restaurant for Saturday.  
 Domain:

Figure 9: Zero-shot domain selection prompt. This prompts the LLM to choose one of the possible MultiWOZ evaluation domains.

**System Message**  
 You are a friendly conversational AI assistant. Your goal is to engage with the user in a friendly conversation.

**User Message**  
 Respond to the user with a short response. Focus on being friendly and engaging. Write the response in between <response> and </response> tags.  
 Context:  
 Customer: I love to play video games !  
 Response:

Figure 10: Zero-shot chitchat prompt. This prompts the LLM to engage in a brief chitchat conversation with the user, in the context of a TOD interaction.

```
<context> (...) <user> I have always
wanted to try the food at a British
restaurant. <system> Yes, they always
look so fancy! <user> Are there any
expensive ones ? </context>
<belief> restaurant food british,
restaurant pricerange expensive
</belief> <db> [db_state4] </db>
<action> restaurant inform area,
restaurant inform area, restaurant inform
area, restaurant inform choice, restaurant
select </action> <response> Of course!
We have [choice] of them in the [area],
[area], and [area]. What area would you
prefer? </response>
```

Figure 11: Training example for our end-to-end fine-tuned approach, based on SimpleToD (Hosseini-Asl et al., 2020).

Figure 12: Dialogue interface used for our human evaluation. Participants use the seed turn to start the conversation and then follow the steps below. They then move to the QA tab to evaluate the dialogue.

User	I've been toying around with my new virtual reality headset lately.	<b>State</b>
Assistant	That sounds like a lot of fun! What kind of games have you been trying out on your VR headset?	{}
User	Well, do you know Mario? It's a fancy plumber from Japan, but he's Italian and speaks English.	
Assistant	Oh, I know Mario! He's a classic character. Have you been playing a Mario game on your VR headset?	{}
User	Well, duh, yeah. You've asked me specifically about VR. It's a bit odd in VR because it's a platformer, but it's a Mario game, still fun...	
Assistant	That's awesome! Mario in VR must be a whole new experience. It's great that you're enjoying it even in a different format. Have you found any cool new features in the VR version?	{}
User	In VR you get to hug Luigi, that's pretty good, you can't do that on a standard screen.	
Assistant	That's so cool that you get to hug Luigi in VR! It must feel more immersive and interactive. It's those little details that make the experience special. Have you discovered any other fun surprises in the VR version?	{}
User	Well, can't hug Luigi all day, I'm taking a break. I could need a taxi to get to ruskin gallery. I'm thinking about leaving from saffron brasserie after dinner with colleagues up to 07:15.	
Assistant	81418747547 is the contact number, I have you in a ford.	{"taxi": { "departure": "saffron brasserie", "destination": "ruskin gallery", "leaveat": "19:15"}}
User	Perfect, thank you.	
Assistant	It sounds like you had a great time with Mario in VR! Enjoy your dinner at Saffron Brasserie and have a fantastic time at the Ruskin Gallery. If you need any more help or information, feel free to ask!	{"taxi": { "departure": "saffron brasserie", "destination": "ruskin gallery", "leaveat": "19:15"}}

Table 4: An interaction with GPT-3.5-turbo where the assistant successfully handles chitchat and task-oriented user inputs. It also correctly tracks that *after dinner..07:15* should be mapped to 19:15 and closes the dialogue with a contextual response, that refers back to the initial chitchat.

User	I need a taxi asap ! I was eating at Pizza Hut City Centre when two men started fighting outside. The guards arrived and managed to break apart the fight. I think I'm ready to go home	<b>State</b>
Assistant	What a scary experience! Don't worry, I'm here to help. What's your destination, and I'll get a taxi booked for you ASAP?	{ "taxi": { "departure": "Pizza Hut City Centre" } }
User	I need to get to the gonville hotel	
Assistant	I've booked a taxi for you, an Executive will come for you. Contact number is 01223 312343. Do you need any other help?	{ "taxi": { "departure": "Pizza Hut City Centre", "destination": "gonville hotel", "leaveat": "asap" } }
User	That'll do	
Assistant	Ah, you must be relieved to be getting out of there! I hope the rest of your evening gets better. Your taxi should arrive soon, and you'll be safely back at the Gonville Hotel in no time. If you need anything else, don't hesitate to ask. Otherwise, have a great night and try to put that scary incident behind you!	{ "taxi": { "departure": "Pizza Hut City Centre", "destination": "gonville hotel", "leaveat": "asap" } }

Table 5: An interaction with Llama3-70B-Instruct where the assistant successfully handles an inter-mode turn (1st turn) by reacting empathetically while moving the task forward. Interestingly, regarding the dialogue state, we notice the model tracks "asap" as a time the taxi should leave, which makes sense given the context but may not work with a database expecting time values.

User	My grandmother thinks she can cook better and has never allowed me to eat at an Italian restaurant. But now she's away on a trip, so my whole family is going to an Italian restaurant to find out.	<b>State</b>
Assistant	Oh wow, that sounds like a fun adventure! I hope you all have a blast trying out different dishes and discovering your own favorite Italian flavors. And who knows, maybe your grandmother will be surprised by your culinary discoveries when she gets back! What's your favorite Italian dish so far?	{ }
User	I love lasagna, like Garfield the cat haha Could you suggest an Italian restaurant in the south ?	
Assistant	Ahaha, Garfield is a great reference! Lasagna is a classic choice, and there are so many amazing Italian restaurants in the South. I'd be happy to help you find one! Can you tell me which city or state you're in? That way, I can give you some personalized recommendations.	{ }
User	I need an italian restaurant in the south of Cambridge please	
Assistant	Ahaha, Garfield is a great reference! I've found a lovely Italian restaurant in the south of Cambridge that serves delicious lasagna. Bella Vita is a popular spot, located at G4 Cambridge Leisure Park Clifton Way Cherry Hinton and can be reached at 01223323737. Would you like to book a table?	{ "restaurant": "bookpeople": "1", "area": "south", "food": "italian" }
User	Yes please can you book for 7 people at 2 pm on Friday ? I know, we're quite a large group...	
Assistant	I've booked you in at Bella Vita. It's a lovely Italian restaurant in the south of Cambridge that serves delicious lasagna. The table will be reserved for 15 minutes. Reference number is : 00000004.	{ "restaurant": "bookpeople": "7", "area": "south", "food": "italian", "bookday": "fri- day", "booktime": "14:00" },
User	okay thank you bye	
Assistant	It was my pleasure to help! I hope you and your family have a fantastic time at Bella Vita and enjoy every bite of your lasagna. Don't forget to take plenty of photos and share them with your grandmother when she gets back. Have a great day and buon appetito!	{ "restaurant": { "bookpeople": "7", "area": "south", "food": "italian", "bookday": "fri- day", "booktime": "14:00" } }

Table 6: An example of an unsuccessful dialogue with Llama3-8B-Instruct. Although it seems fine on the surface, the Bella Vita does not exist in the database, it is a hallucination from the model. Other than that, the dialogue state tracking is accurate and the chitchat is well-handled, with the model finishing the dialogue with a few contextual remarks and *buon appetito*, which is a nice touch. In turn 2, the model classifies the user input as *chitchat*, which is a mistake. However, the response is contextual and still allows for the task to move forward.

User	I love reading books. I find it interesting to read stories and get a peek into what the author is thinking.	<b>State</b>
Assistant	Aww, that's wonderful! There's something so special about getting lost in a good book, isn't there? I'm curious, what kind of books do you usually enjoy reading? Are you more into fiction, non-fiction, or a mix of both?	{}
User	I like books on art and meditation. i read a good one by Rick Rubin recently, have you heard of him ?	
Assistant	Ah, Rick Rubin is amazing! I've heard great things about his book. I'm a big fan of his work with the Red Hot Chili Peppers and other artists. Meditation and art are such great topics to explore in books - it's wonderful that you're interested in those areas. Have you found that reading about meditation has helped you in your own practice?	{}
User	A bit, I'm not as consistent with it as I should be though	
Assistant	Ah, don't be too hard on yourself! Consistency can be tough, but it's great that you're acknowledging the importance of meditation in your life. Maybe we can brainstorm some ways to make it more enjoyable and sustainable for you? Do you have a favorite meditation spot or a specific technique that you enjoy? I'm all ears!	{}
User	I like sitting in the park usually and listning to the sounds around me	
Assistant	Ah, that sounds lovely! Being in nature can be so grounding and calming. I'm sure the sounds of the park help you focus and relax. Have you ever tried incorporating guided meditations into your practice, or do you prefer to just sit and listen to your surroundings?	{},
User	Yeah I'm looking apfr a book on that. But I have to go to Stevenage to get it. There's a bookstore there.	
Assistant	The byard art is the 1 museum in the south and the entrance fee is free.	{"attraction": { "type": "museum", "area": "south" }}
User	Hmm can you book me a train to Stevenage from Cambridge this Saturday, so I can go to the bookstore ?	
Assistant	What a lovely conversation we've had! I'd be delighted to help you book a train to Stevenage from Cambridge this Saturday. There is a train that will depart at 05:21 and arrive at 20:15. Would you like me to book that for you?	{"attraction": { "type": "museum", "area": "south" }, "train": { "bookpeople": "1", "arriveby": "20:15", "day": "saturday", "destination": "stevenage" }}
User	I need one that arrives by 7:15 pm.	
Assistant	There are 10 trains matching your query. Shall I book one?	{"attraction": { "type": "museum", "area": "south" }, "train": { "bookpeople": "1", "arriveby": "19:15", "day": "saturday", "destination": "Stevenage", "departure": "Cambridge" }}

Table 7: An extract from an interaction with Llama3-8B-Instruct, where a slightly ambiguous user typo leads the model to incorrectly try to find a museum and start tracking the *attraction* domain. We notice though that this is not a problem for continuing with the task of booking a train which the model tracks accurately. We also find the chitchat quite convincing, with the model displaying good knowledge of the music producer Rick Rubin.