LLM-based Frameworks for API Argument Filling in Task-Oriented Conversational Systems

Jisoo Mok[1*] **Mohammad Kachuee**[2] **Shuyang Dai**[2] **Shayan Ray**[2]
Tara Taghavi[2] **Sungroh Yoon**[1,4†]
[1] Department of ECE, Seoul National University [2] Amazon
[4] Interdisciplinary Program in Artificial Intelligence, Seoul National University

Abstract

Task-orientated conversational agents interact with users and assist them via leveraging external APIs. A typical task-oriented conversational system can be broken down into three phases: external API selection, argument filling, and response generation. The focus of our work is the task of argument filling, which is in charge of accurately providing arguments required by the selected API. Upon comprehending the dialogue history and the pre-defined API schema, the argument filling task is expected to provide the external API with the necessary information to generate a desirable agent action. In this paper, we study the application of Large Language Models (LLMs) for the problem of API argument filling task. Our initial investigation reveals that LLMs require an additional grounding process to successfully perform argument filling, inspiring us to design training and prompting frameworks to ground their responses. Our experimental results demonstrate that when paired with proposed techniques, the argument filling performance of LLMs noticeably improves, paving a new way toward building an automated argument filling framework.

1 Introduction

Task-oriented conversational systems, illustrated in Figure 1, largely consist of three processes: external API selection, argument filling, and response generation (Hosseini-Asl et al., 2020). The API selection phase selects which one from the pre-defined pool of APIs must be called to complete the user request. Once the appropriate external API to carry out the user request has been selected, the argument filling phase must reliably identify and provide correct arguments to the API by faithfully following the API schema and dialogue history. An API schema, an example of which is also demonstrated in Figure 1, is typically assumed to be given as a part of the API and includes required arguments and their types. Therefore, the API schema and dialogue history provide sufficient information for the conversational agent to identify which arguments are necessary to complete the API call. Lastly, the response generation phase, as the name suggests, returns an appropriate response to the user based on the API output.

The user dissatisfaction in argument filling mainly stems from the conversational agent being incapable of adhering to the API schema and dialogue history. The erroneous arguments that digress away from the API schema are considered "Syntax Errors", and hallucinated responses that deviate from the user utterances are considered "Hallucinations." In Figure 2, we provide examples of each error type that occurs when performing argument filling for the "Hair Appointment" API.

Large Language Models (LLMs) trained with instructions have recently been garnering much attention as a promising model for enabling human-like and safe user-agent interactions in open-domain conversations (Ouyang et al., 2022; Wang et al., 2022b). The aim of this paper is to explore whether the strength of LLMs can be harnessed specifically for the purpose of argument filling in task-oriented conversational systems. To construct an LLM-backed framework for argument filling, their outputs must strictly follow and stay faithful to the pre-defined API schema and user utterances, a process commonly known as “grounding." Our initial zero-shot performance evaluation of LLMs of various sizes reveals that LLM-generated responses suffer severely from both syntax errors and hallucinations, necessitating the development of additional techniques to appropriately ground their responses for the task of our interest.

We investigate two separate and unique avenues to tackle the problem of grounding for open- and

* Work done while interning at Amazon (magicshop1118@snu.ac.kr)
† Corresponding Authors

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track), pages 419–426

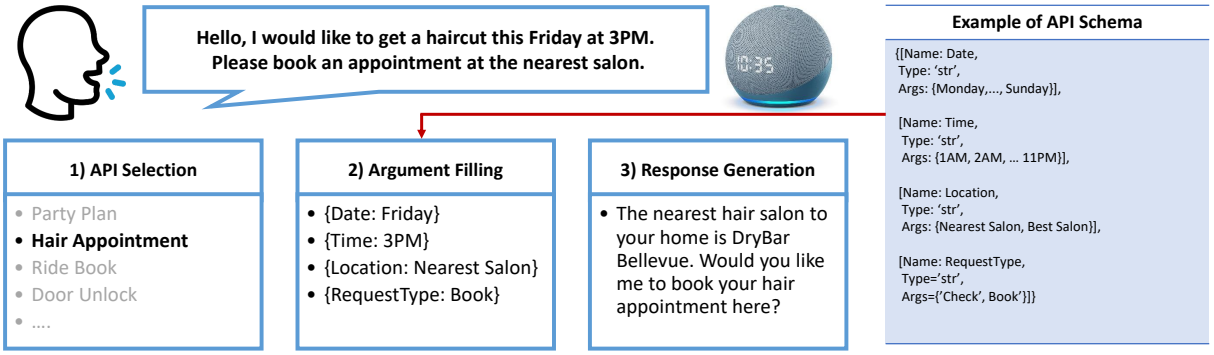

Figure 1: An overview of how a task-oriented conversational agent might complete a user's request to book a haircut. To begin with, the agent selects the "Hair Appointment" API from the list of available APIs. An example of the pre-defined API schema associated with the "Hair Appointment" API is given on the far right side. Following API selection, the argument filling step utilizes the API schema and dialogue history to identify arguments to complete the API call. Finally, the agent responds to the user with the utterances produced in the response generation step.

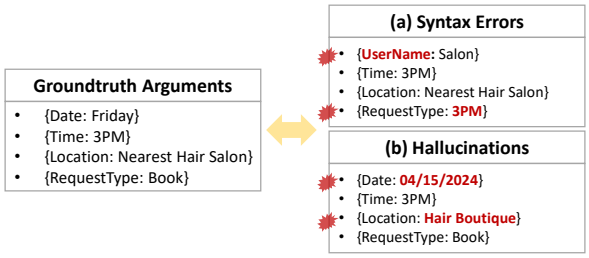

Figure 2: Examples of two potential errors that can arise in argument filling. (a) Syntax errors refer to those that digress away from the pre-defined API schema. (b) Hallucinations correspond to those that deviate from the user intention and utterances.

closed-sourced LLMs. On one hand, for open-sourced LLMs, *e.g.,* LLAMA-v1-7B, we propose a two-step instruction-tuning framework that is comprised of supervised fine-tuning (SFT) and rejection sampling (RS). Our experimental results show that utilizing the proposed instruction-tuning framework noticeably outperforms the naïve SFT baseline. On the other hand, in the case of closed-sourced LLMs whose weights are not directly accessible, we demonstrate that their performance can be improved by replacing the plain prompt design with a "multi-step prompting" scheme. Our contributions can be summarized as follows:

- This is the first work to explore the utilization of LLMs for argument filling in task-oriented conversational agents. Our results demonstrate that when paired with a proper grounding process, LLMs can offer a simpler and more autonomous alternative to conventional approaches in argument filling.
- For open-sourced LLMs, we propose a cohesive training pipeline to ground their behaviors. The proposed training pipeline consists of two phases: model bootstrapping via supervised fine-tuning and additional fine-tuning with model-generated outputs, which have undergone rejection sampling through a custom reward function. For closed-sourced LLMs, we explore an advanced prompting technique that is more fine-grained and informative.
- We provide substantial experimental results to demonstrate the effectiveness of the proposed approaches. Notably, the LLAMA-v1-7B model fine-tuned using the proposed instruction-tuning pipeline outperforms strong zero-shot baselines obtained by prompting significantly larger LLMs.

2 Related Works

2.1 Language Models for Task-oriented Dialogues

Utilization of pre-trained Language Models for Task-oriented Dialogues (ToD) was pioneered by Zhang et al. (2019) and Peng et al. (2021). Kulhánek et al. (2021) and Lin et al. (2020) improved the basic ToD modeling approaches by employing contrastive state training and belief state differences, respectively. Other works (Pandey et al., 2018; Cai et al., 2019; Nekvinda and Dušek, 2022) proposed to combine generative models with retrieval-based approaches. While Hudeček and Dušek (2023) perform zero-shot evaluation of various LLMs for ToD modeling, to the best of our knowledge, this is the first work to exploit and instruction-tune LLMs in the billion-parameter regime for argument filling in ToD systems.

```
You are a conversational Agent interacting with the API and dialog history:
### API:
{'Name': 'ride_book',
'Description': 'ride_book',
'Args': [name=Price, type=int, min=5, max=50, is_required=True,
       name=AllowsChanges, type=str, choices=['True', 'False'], is_required=True,
       name=MinutesTillPickup, description=Minutes until pickup, type=int, min=5, max=30, is_required=True,
       name=ServiceProvider, description=Service Provider, type=str, choices=['Uber', 'Lyft', 'Taxi'], is_required=True, …]
### Dialog history:
[Past Dialogue]
[AGENT] I found a Uber ride for you from 'Craig and Center' to 'Airport' for 36 credits. Should I book that for you?
[USER] That sounds good.
[SYSTEM] ride_book() <- {'CustomerName': 'Alexis', 'DepartureLocation': 'Craig and Center', 'ArrivalLocation': 'Airport',
'RequestType': 'Book', 'ServiceProvider': '["Uber","Lyft"]'}
[API] ride_book() -> {'APIName': 'ride_book', 'Message': 'Ride booked.'}
[AGENT] I have booked your ride.
[USER] I just remembered that last time Mark drove me he got lost and I missed an appointment. He isn't my driver, is her?
[AGENT] Your driver is Sirius.
[USER] I forgot my friend wanted to meet me at BrewLab cafe at Hospital not Airport. Can you change the destination to the hospital?
Generate the inputs to the API: ride_book() <-
Sample Ground-truth Argument: {Price: 15, AllowsChanges: True, MinutesTillPickup: 5, ServiceProvider: Lyft}
```

Figure 3: Abbreviated illustration of the default prompt template that includes API description and dialogue history. We also provide an example of a ground-truth argument, which is pre-processed to follow a dictionary-like format.

2.2 Large Language Models and Instruction-tuning

The introduction of Transformer-based architectures heralded the beginning of large and incredibly capable models for Natural Language Processing (NLP) (Vaswani et al., 2017). Transformer-based language models with several billions of parameters, such as GPT-3 (Brown et al., 2020) and OPT (Zhang et al., 2022), have shown unprecedented zero- and few-shot performance across diverse NLP tasks. The generalization capability of these so-called Large Language Models (LLMs) was further improved by training them via instruction-tuning (Goldwasser and Roth, 2014) with in-context instructions. The promising results obtained by instruction-tuning inspired the development of large instruction-paired datasets, such as NaturalInstructions-v1 (NI-v1) (Mishra et al., 2022) and SuperNaturalInstructions (Wang et al., 2022a). The remarkable performance of general-purpose instruction-tuned models inspired the development of more domain-specific models. Examples of such models include: InstructUIE (Wang et al., 2023) for information extraction, CoEDIT (Raheja et al., 2023) for writing, ChatDoctor (Yunxiang et al., 2023) for medical purposes, and Goat (Liu and Low, 2023) for mathematics.

3 Proposed Methodology

3.1 Prompt Design

To guarantee experimental consistency across different models and datasets, we first design a common prompt template for argument filling. An example of the default prompt template, which includes a short instruction, the pre-defined API schema, and dialogue history up to the specified API call, is provided in Figure 3. This prompt template is used for both in-context instruction tuning and evaluation processes and remains fixed across all of our experiments unless stated otherwise.

3.2 Instruction-tuning Framework for Open-sourced LLMs

Phase I. Model Bootstrapping via Supervised Fine-tuning We first bootstrap the LLM's responses on argument filling prompts, so that its generative behavior can be controlled to output the arguments in a dictionary format, as illustrated in Figure 3. Following the conventional fine-tuning scheme, we fine-tune the LLM using the cross entropy loss. Once the bootstrapping phase is completed, we propose to augment the train dataset using model-generated outputs. In the next section, we define a custom reward function that is employed to score and select generated samples to be included in the additional fine-tuning phase.

Phase II. Rejection Sampling with Custom Reward Function "Rejection Sampling" commonly refers to the process of identifying desirable model-generated outputs that are capable of further improving the performance on the target task. Therefore, the success of rejection sampling is heavily contingent on the definition of the reward function that can accurately reflect the usefulness of model-generated outputs. To define the custom reward function for argument filling, we first categorize potential sources of error into: non-existent key (NK), missing key (MK), schema-grounded but incorrect

value (SV), and hallucinated value (HV). The key and value here refer to the corresponding components of the key-value pairs of the model-generated arguments, which have been bootstrapped to follow a dictionary-like format. A detailed description of each error type is provided below:

- **Non-existent Key (NK):** The generated key is not provided as a part of the pre-defined schema.
- **Missing Key (MK):** The model-generated arguments are missing an expected key that is required by the pre-defined schema.
- **Schema-grounded but Incorrect Value (SV):** The generated value follows the pre-defined schema but deviates from the dialogue history, resulting in an incorrectly identified argument.
- **Hallucinated Value (HV):** The generated value does not follow the pre-defined schema, and hence, it is incorrect by definition.

The total number of errors in a model-generated output can be computed through a simple summation of all 4 error types: $N_{\text{Error}} = N_{\text{NK}} + N_{\text{MK}} + N_{\text{SV}} + N_{\text{HV}}$. The error rate can then be defined as: $N_{\text{Error}}/N_{\text{Total}}$, where N_{Total} denotes the total number of keys and values in the ground-truth argument. This error rate is normalized between -1 and 1 to obtain the final reward value following the equation: $R = 1 - 2 * N_{\text{Error}}/N_{\text{Total}}$.

After the LLM has been bootstrapped on the argument filling datasets, we sample K number of outputs from the model and score the generated outputs using the above reward function. We only select outputs that yield positive reward to augment the train dataset. With the newly added instances mixed in the train dataset, we perform one additional epoch of supervised fine-tuning.

There exist two expected advantages of incorporating rejection-sampled model outputs. First, utilizing the model outputs filtered with the custom reward function allows us to effectively augment the train dataset with desirable instances without the need to collect additional data points to avoid overfitting. Second, we expect that incorporating these outputs will improve the fine-tuned LLM's robustness to noisy data points it may encounter at test-time. Even if the model-generated outputs yield positive reward, they will inevitably be noisier than the curated train dataset with ground-truth labels. Therefore, the LLM that has been exposed to noisier data points in the rejection sampling phase will exhibit a higher degree of robustness and generalization performance.

3.3 Multi-step Prompting Scheme for Closed-sourced LLMs

It is infeasible to fine-tune LLMs whose design and weights are not released to the public. Therefore, we additionally explore a more fine-grained and informative prompting method to complement larger LLMs. The default prompt design as described in Figure 3 asks the model to identify required arguments and extract appropriate information to fill them all at the same time. For multi-step prompting with hints, we instead prompt the model to identify and fill one argument at a time. By using this more targeted prompt design, we are providing the LLM with additional information about required slots and effectively restricting its generative behavior to prevent its digression from the pre-defined schema and dialogue history.

4 Experimental Set-up

4.1 Datasets and Models

4.1.1 Datasets

We primarily use STAR (Mosig et al., 2020) and SGD (Rastogi et al., 2020) datasets as test beds to validate our approach.

- **STAR:** is a collection of realistic, task-oriented dialogues that includes 5,820 dialogues that span 24 tasks and 13 domains. The schemas in the STAR dataset are similar to "task specifications," which contain information about the ideal dialogue flow for each task.
- **SGD:** is a rich, fully-annotated dataset, which contains more than 22,000 dialogues that encompass 20 domains, ranging from banks to travels and weather. The comprehensive annotation that includes schema representation makes it a flexible and convenient dataset to investigate not only argument filling but also other components of task-oriented conversational systems.

We verify the competitiveness of proposed approaches under both in- and out-of-domain scenarios. Under the in-domain scenario, train and test dialogues are sampled from the same set of domains, while under the out-of-domain scenario, the test dialogues contain domains that were not observed during the training process. To create an in-domain benchmark, we randomly split the entire dataset into train and test datasets, such that domains are evenly represented across the two. For out-of-domain evaluation, we purposefully curate the test dataset, such that no explicit or semantic

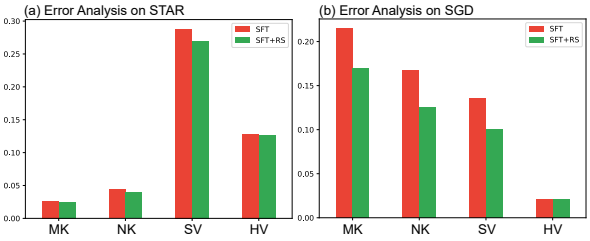

Figure 4: Analyses of four different error rates on (a) STAR and (b) SGD **in-domain** benchmarks.

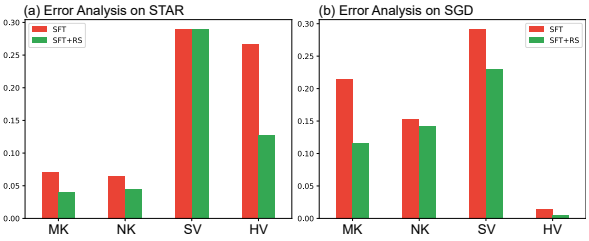

Figure 5: Analyses of four different error rates on (a) STAR and (b) SGD **out-of-domain** benchmarks.

overlap exist between tasks in the train dataset and those in the test dataset.

4.1.2 Models

• **LLAMA-7B (Touvron et al., 2023):** is a state-of-the-art foundational LLM released by Meta AI. While the LLAMA models of various sizes have been open-sourced, we primarily utilize LLAMA-v1-7B model for fine-tuning experiments.

• **ChatGPT**[1]**:** is widely regarded as one of the most powerful LLMs; its release is perceived to be a significant milestone in the evolution of conversational AI systems. Because the model weights have not been open-sourced, we rely on OpenAI's ChatGPT API for evaluation.

4.2 Libraries and Hyperparameters

We utilize the Huggingface (Wolf et al., 2019) library for implementation and training of models. All experiments are executed on NVIDIA V100 GPU with 32GB RAM. The following set of hyperparameters is used for the supervised fine-tuning phase: batch size of 8, Adam optimizer with initial learning rate of 0.00002, weight decay of 0.1, and constant learning rate scheduling. We run the supervised fine-tuning phase for 5 epochs before performing rejection sampling. As mentioned in Section 3.2, we perform additional fine-tuning with rejection-sampled data for only one additional epoch. All hyperparameters remain unchanged from the supervised fine-tuning phase.

4.3 Compared Approaches

• **Zero-shot**: is the most naïve baseline obtained by prompting the pre-trained LLMs with the prompt design provided in Figure 3. The pre-trained LLMs are used as is without undergoing additional fine-tuning on task-oriented dialogue datasets.

• **Multi-Step**: replaces the naïve prompting process with the multi-step prompting scheme in Section 3.3. Since multi-step prompting only improves the model at inference time, the pre-trained LLM is again used with no alterations.

• **Supervised Fine-tuning** (*-sft*): is a baseline obtained by instruction-tuning the LLM on fully-labeled train set of task-oriented dialogue datasets following the Phase I process in Section 3.2.

• **Supervised Fine-tuning + Our Rejection Sampling** (*-sft-rs*): trains the fine-tuned LLM on additional model-generated data that have been selected according to the proposed reward for rejection sampling (Phase II of Section 3.2).

4.4 Metrics

• **BLEU:** (Papineni et al., 2002) quantifies the semantic similarity between model-generated and reference sentence pairs. Its close alignment with human perception of generation quality and low computational cost make BLEU a particularly compelling metric for automatic evaluation of Natural Language Processing (NLP) systems.

• **Fuzzy Matching:** is adopted to quantify the argument filling accuracy. We employ fuzzy match, instead of exact match, such that minor typos and capitalization, which should not determine the quality of the generated outputs, do not influence the performance metric.

• **F-1 Score:** takes into account both the character-level precision and recall of predicted arguments. F-1 score is a preferred choice of metric over accuracy when evaluating datasets with significant class imbalances (i.e., the number of test samples per API is unevenly distributed).

5 Results

5.1 In-Domain Results

The results obtained on STAR and SGD datasets under the in-domain evaluation setting are reported in Table 1. The suffixes *-sft* and *-sft-rs* are used to denote models that have been trained only with

[1] https://openai.com/blog/chatgpt

Methods	Models	SGD			STAR		
		BLEU	FM	F-1	BLEU	FM	F-1
Zero-shot	LLAMA-v1-7B	0.0104	5.2852	0.0472	——		
	ChatGPT	0.4578	44.5853	0.4802	0.2127	26.0679	0.2094
Multi-step Prompting	ChatGPT	0.4578	44.5853	0.4802	0.2127	26.0679	0.2094
Instruction-tuned	LLAMA-v1-7B-*sft*	0.7802	91.1299	0.7718	0.3418	58.19	0.3209
	LLAMA-v1-7B-*sft-rs*	0.8003	91.6462	0.7834	0.3734	62.7669	0.3605

Table 1: Comparison of different models and training/prompting methods under the **in-domain** evaluation setting. LLAMA-v1-7B-*sft-rs* clearly outperforms all other baselines, showing the efficacy of the proposed training scheme.

Methods	Models	SGD			STAR		
		BLEU	FM	F-1	BLEU	FM	F-1
Zero-shot	LLAMA-v1-7B	0.0118	5.4612	0.0456	——		
	ChatGPT	0.2460	35.9156	0.3701	0.2045	33.5571	0.2357
Multi-step Prompting	ChatGPT	0.3166	48.7200	0.4212	0.2281	33.7857	0.2672
Instruction-tuned	LLAMA-v1-7B-*sft*	0.6972	86.3976	0.6642	0.2512	54.7000	0.2330
	LLAMA-v1-7B-*sft-rs*	0.7705	90.6652	0.7608	0.3511	65.0714	0.3200

Table 2: Comparison of different models and training/prompting methods under the **out-of-domain** evaluation setting. The results are generally consistent with those obtained under the in-domain setting.

supervised fine-tuning and with supervised fine-tuning and rejection sampling, respectively. Multi-step prompting that provides additional hints successfully improves the performance of the ChatGPT models. More importantly, we observe that the LLAMA-v1-7B model that has been trained with the proposed instruction-tuning pipeline with rejection sampling (LLAMA-v1-7B-*sft-rs*) obtains the best performance across all metrics on both datasets. This result clearly demonstrates that with our training framework, relatively smaller and light-weight LLMs can outperform larger ones. Furthermore, the superiority of LLAMA-v1-7B-*sft-rs* to LLAMA-v1-*sft* provides strong support for incorporating rejection-sampled data to effectively improve the performance of fine-tuning with less training budget. Lastly, we note that a larger degree of performance improvement is observed on the SGD dataset, which has a wider variety of tasks and thus can be considered more difficult.

5.2 Out-of-Domain Results

To simulate an out-of-domain test scenario, we deliberately create a train-test split, such that there is no explicit or implicit task domain overlap between the train and test set The results obtained under the out-of-domain evaluation setting are reported in Table 2. In general, the out-of-domain evaluation results show similar tendencies to the in-domain results. While the proposed instruction-tuning framework and multi-step prompting successfully improve the performance of open-sourced and closed-sourced LLMs, respectively, they both experience slight performance degradation when compared to the in-domain evaluation results.

5.3 Error Analyses

We analyze sources of error in outputs generated by LLAMA-v1-7B-*sft-rs* to identify room for improvement. In Figures 4 and 5, we compare the four error rates, as defined in Section 3.2, in LLAMA-v1-7B-*sft* and LLAMA-v1-7B-*sft-rs* models. Training the LLAMA-v1-7B model with SFT + RS reduces all four error rates, and the rate of hallucinated value errors is particularly low compared to other errors. This analytical result implies that once grounded, the LLM mostly ceases to hallucinate and remains close to the API schema and dialogue history provided as a part of the prompt template.

6 Conclusion

This paper explored and uncovered the powerfulness of leveraging LLMs to automate the argument filling process, a core component in task-oriented conversational systems. The strong experimental results indicate that the proposed methods, used in conjunction with open- or closed-source LLMs, are effective for restricting the LLM's generative behavior, specifically for argument filling.

Acknowledgements

This work was supported in part by the Institute of Information & Communications Technology Planning & Evaluation (IITP) and the National Research Foundation of Korea (NRF) grants funded by the Korean government (MSIT) [No. 2021-0-01343, No. 2022-0-00959, Artificial Intelligence Graduate School Program (Seoul National University), No. 2022R1A3B1077720] and the BK21 FOUR program of the Education and Research Program for Future ICT Pioneers, Seoul National University in 2024.

Limitations and Potential Risks

One limitation of our work is that proposed frameworks are validated only on one open- or closed-sourced model. In addition, while LLMs are quite capable of completing the argument filling task, the inference time for LLMs may still be longer than many of smaller, more targeted language models. Accelerating LLM inferencing, however, is outside the scope of our work.

Reliance on closed-sourced LLMs could pose unforeseen risks since the backbone model could be altered without notice. Even if significant changes are made to the design and weights of the closed-sourced models, there is no way for us to know what those alterations are. This complete black-box nature of closed-sourced LLMs may make it an undesirable choice of backbone model. Therefore, we conjecture that utilizing a targeted decoding scheme that can further enforce the LLM to follow specific parts of the prompt template could assist in reducing schema-related errors.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Deng Cai, Yan Wang, Wei Bi, Zhaopeng Tu, Xiaojiang Liu, and Shuming Shi. 2019. Retrieval-guided dialogue response generation via a matching-to-generation framework. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1866–1875.

Dan Goldwasser and Dan Roth. 2014. Learning from natural instructions. *Machine learning*, 94(2):205–232.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Vojtěch Hudeček and Ondřej Dušek. 2023. Are llms all you need for task-oriented dialogue? *arXiv preprint arXiv:2304.06556*.

Jonáš Kulhánek, Vojtěch Hudeček, Tomáš Nekvinda, and Ondřej Dušek. 2021. Augpt: Auxiliary tasks and data augmentation for end-to-end dialogue with pre-trained language models. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 198–210.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405.

Tiedong Liu and Bryan Kian Hsiang Low. 2023. Goat: Fine-tuned llama outperforms gpt-4 on arithmetic tasks. *arXiv preprint arXiv:2305.14201*.

Swaroop Mishra, Daniel Khashabi, Chitta Baral, and Hannaneh Hajishirzi. 2022. Cross-task generalization via natural language crowdsourcing instructions. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3470–3487.

Johannes EM Mosig, Shikib Mehri, and Thomas Kober. 2020. Star: A schema-guided dialog dataset for transfer learning. *arXiv preprint arXiv:2010.11853*.

Tomáš Nekvinda and Ondřej Dušek. 2022. Aargh! end-to-end retrieval-generation for task-oriented dialog. *arXiv preprint arXiv:2209.03632*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 35:27730–27744.

Gaurav Pandey, Danish Contractor, Vineet Kumar, and Sachindra Joshi. 2018. Exemplar encoder-decoder for neural conversation generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1329–1338.

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.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. Soloist: Building task bots at scale with transfer learning and machine teaching. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Vipul Raheja, Dhruv Kumar, Ryan Koo, and Dongyeop Kang. 2023. Coedit: Text editing by task-specific instruction tuning. *arXiv preprint arXiv:2305.09857*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Xiao Wang, Weikang Zhou, Can Zu, Han Xia, Tianze Chen, Yuansen Zhang, Rui Zheng, Junjie Ye, Qi Zhang, Tao Gui, et al. 2023. Instructuie: Multi-task instruction tuning for unified information extraction. *arXiv preprint arXiv:2304.08085*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Anjana Arunkumar, Arjun Ashok, Arut Selvan Dhanasekaran, Atharva Naik, David Stap, et al. 2022a. Benchmarking generalization via in-context instructions on 1,600+ language tasks. *arXiv preprint arXiv:2204.07705*.

Yizhong Wang, Swaroop Mishra, Pegah Alipoormolabashi, Yeganeh Kordi, Amirreza Mirzaei, Atharva Naik, Arjun Ashok, Arut Selvan Dhanasekaran, Anjana Arunkumar, David Stap, et al. 2022b. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 5085–5109.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*.

Li Yunxiang, Li Zihan, Zhang Kai, Dan Ruilong, and Zhang You. 2023. Chatdoctor: A medical chat model fine-tuned on llama model using medical domain knowledge. *arXiv preprint arXiv:2303.14070*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Yizhe Zhang, Siqi Sun, Michel Galley, Yen-Chun Chen, Chris Brockett, Xiang Gao, Jianfeng Gao, Jingjing Liu, and Bill Dolan. 2019. Dialogpt: Large-scale generative pre-training for conversational response generation. *arXiv preprint arXiv:1911.00536*.