Interpreting User Requests in the Context of Natural Language Standing Instructions

Nikita Moghe[γ*] **Patrick Xia**[Φ] **Jacob Andreas**[Φ]
Jason Eisner[Φ] **Benjamin Van Durme**[Φ] **Harsh Jhamtani**[Φ]
[γ]School of Informatics, University of Edinburgh
[Φ]Microsoft Semantic Machines
nikita.moghe@ed.ac.uk hjhamtani@microsoft.com

Abstract

Users of natural language interfaces, frequently powered by Large Language Models (LLMs), must often repeat their full set of preferences each time they make a similar request. We describe an approach to LLM-based dialogue modeling in which persistent user constraints and preferences – collectively termed *standing instructions* – are provided as additional context for such interfaces. For example, when a user states *I'm hungry*, a previously expressed preference for Persian food can be automatically added to the LLM prompt, influencing the search for relevant restaurants. We develop NLSI, a language-to-program dataset consisting of over 2.4K English dialogues spanning 17 domains, in which each dialogue is paired with a user profile (a set of user-specific standing instructions) and corresponding structured representations (a sequence of API calls). A key challenge in NLSI is to identify which subset of the standing instructions is applicable to a given dialogue. NLSI contains diverse phenomena, from simple preferences to interdependent instructions such as triggering a hotel search whenever the user is booking tickets to an event. We conduct experiments on NLSI using prompting with large language models and various retrieval approaches, achieving a maximum of 46% exact match on API prediction. Our results demonstrate the challenges in identifying the relevant standing instructions and their interpretation into API calls[1].

1 Introduction

Large language models (LLMs) such as GPT-3 (Brown et al., 2020), GPT-4 (OpenAI, 2023), and LLaMa 2 (Touvron et al., 2023) are increasingly being used with tools and APIs (Schick et al., 2023; Qin et al., 2023) to provide additional functionality

Figure 1: Parsing an utterance into a structured output, in the presence of a *user-specific* set of *standing instructions*. A model for the task needs to identify (explicitly or implicitly) the subset of instructions applicable to the utterance and interpret the utterance into API calls.

to users. For example, ChatGPT allows several external plugins such as OpenTable for searching restaurants or Expedia for booking travel.[2] These applications must learn to identify which service the user is seeking while respecting preferences across diverse domains that are unique to each user. Understanding such preferences can aid in personalising the user experience by providing tailored responses, increased accuracy in recommendations and saving user time. However, in most cases, users must verbalise their preferences in detail during the interaction, including for repeated requests.

Past work has explored learning preferences from user-system interactions over time (Micarelli et al., 2007; Salemi et al., 2023). These preferences can be hard to learn while also requiring significant amounts of training data. Further, these learnt preferences are *implicit* and usually cannot be interpreted or edited by the user.

*Work done while interning at Microsoft
[1]Code: https://github.com/nikitacs16/nlsi
Data: https://huggingface.co/datasets/nikitam/nlsi

[2]https://openai.com/blog/chatgpt-plugins

Findings of the Association for Computational Linguistics: NAACL 2024, pages 4043–4060
June 16-21, 2024

We propose incorporating personalised *standing instructions* explicitly as additional context while interpreting a user's requests. Standing instructions are user-provided natural language statements to change or prescribe system behaviour under certain circumstances. For example, in Figure 1, the user wishes to look for some nearby restaurants. In the absence of standing instructions, the user might have to interact for multiple turns with the system to arrive at their preferred restaurant cuisine and location. By looking up the relevant standing instructions for restaurants, the system can directly search for *Persian restaurants* in *San Leandro*, saving the user's time as well as providing customised/localised recommendations. Explicit natural language instructions are also both controllable and interpretable. A user can inspect and edit their standing instructions, especially for preferences that change over time. Further, the generated outputs can be directly linked to the relevant standing instructions, improving the user's trust in the system (Liu et al., 2023).

Our work is related to Gupta et al. (2022), which conditions a dialogue model's response on a set of developer guidelines. Their work focuses on controlling response generation in open-domain dialogue systems with a focus on reducing toxicity and enhancing safety. More recently, commercial LLM providers have introduced System Prompts[3]/Custom Instructions[4]/Preamble[5] which have an option to include guidelines at the beginning of every conversation to improve response generation. However, not much is known about how it operates, and no evaluations of its usage have been documented, especially in the task-oriented setting.

This work makes the following contributions: **(i)** We systematically study the incorporation of standing instructions in a task-oriented setup. We develop and introduce **NLSI** (Natural Language Standing Instructions), an English-language dataset in which every example consists of a conversation between the user and a dialogue agent, accompanied by a collection of standing instructions (*user profile*) and a sequence of API calls reflecting user intents. **(ii)** We investigate six reasoning types for using standing instructions that range from a single instruction for a specific attribute to more complex situations such as the user proposing multiple preferences for same aspect, *etc.* These reasoning types introduce challenges pertaining to subset selection of relevant standing instructions and then inferring the structured API calls and their arguments. These include instructions that specify a single preference to more complex ones that involve multi-hop, cross-domain, and conflict reasoning. **(iii)** We use this dataset to benchmark a variety of methods involving the selection and interpretation of user utterances in the presence of standing instructions. We observe that our LLM-based methods are far from perfect, raising new challenges in retrieval, reasoning, and semantic parsing.

[3] https://docs.anthropic.com/claude/docs/how-to-use-system-prompts

[4] https://openai.com/blog/custom-instructions-for-chatgpt

[5] https://txt.cohere.com/chatbot-chat-endpoint/

2 Task Overview

We are interested in translating a user utterance into a sequence of API calls in the context of user-specific standing instructions (Figure 1). Consider a conversational context x, which consists of dialogue history between the user and the system (if any) and the user's current utterance. We assume a user profile u consisting of a sequence of natural language instructions $u_1, u_2, ...u_M$. In this setting, instruction following consists of a *selection* task (which obtains a set of standing instructions z from the user profile u that are relevant to x) followed by an *interpretation* task (which predicts API calls y based on the conversational context and the relevant subset of standing instructions z). We assume access to a schema s that lists the valid API method names and their keyword arguments (slots). Formally, an agent of this kind is described by a generative model:

$$z \sim p(\cdot \mid x, u)$$
$$y \sim p(\cdot \mid x, z, s)$$

3 Dataset: NLSI

Existing related datasets have focused on generating safer responses in open-domain dialogue via natural language guidelines (Gupta et al., 2022) or looked at personalised text generation by conditioning on a set of past user-written documents like emails or reviews (Salemi et al., 2023). Similarly, Madaan et al. (2022) improved response generation on user feedback on past conversations to assist new users on tasks such as ethical reasoning and word scrambling. Joshi et al. (2017); Irfan et al.

	PLAIN	MULTIHOP	MULTIPREFERENCE
Relevant Standing Instructions (z)	>I always go to Santa Rosa if I'm looking for Movies. >I like fantasy movies the best.	>If I'm looking for a flight, American Airlines is my go-to. >If I'm flying American Airlines, check for Economy seating class.	>If I ask for Events, my preferred event type is Music. >When the event type is Music, search for Blues as the category. >Search for the event name Greensky Bluegrass if the category is Blues. >If I ask for Events, my preferred event type is Sports.
Conversation (x)	*User*: I want to go out to watch a movie, please help me find a good one.	*User:* Can you get on and get me a round trip ticket? *Agent:* Where will you go? Where are you coming from? *User:* I'm going to SFO from New York City.	*User:* My schedule is free today and I plan to go to an event in Seattle, WA. I want to look for events in that area.
API calls (y)	`GetMovies( genre="fantasy", location="Santa Rosa")`	`GetFlights( destination="SFO", origin="New york", airlines="American Airlines", seating_class="Economy")`	`GetEvents( city="Seattle, WA", event_type="Music", category="Blues", event_name="Greensky Bluegrass") GetEvents( city="Seattle, WA", event_type="Sports")`

Table 1: Some examples from NLSI. User profile is not shown for brevity. (1) In PLAIN, the instructions usually represent a domain matching problem. (2) In MULTIHOP, note that the seating class attribute *Economy* is dependent on choosing the instruction with *American Airlines*. (3) For the example for MULTIPREFERENCE, as there are two preferences for the same attribute `event_type`, there are two separate API calls. Further, the API call with `event_type` *Music* has additional attributes. Additional examples are provided in Table 5 in Appendix A.

(2021) focus on incorporating personalisation in task-oriented dialogue with a small set (<5) of preferences. Due to the lack of comprehensive datasets that study the use of natural language standing instructions in a language-to-program setup, we created NLSI. Our dataset covers multiple domains like airline booking or finding events. Each domain has an associated API.

3.1 Reasoning Types

In the context of standing instructions, various types of reasoning might be needed to predict API calls. Following a single standing instruction may be easier than composing and reasoning over several instructions. Furthermore, reasoning across several instructions in the same domain, like booking hotels, may be easier than across domains. Thus, to enable comparisons at different difficulties, we designated six reasoning types for NLSI. While these are not exhaustive, they allow us to systematically study a range of situations ranging from simple domain matching to more complex reasoning (see examples in Table 1):

NONEAPPLICABLE For these examples, no standing instructions from the user profile are required for interpreting the user's utterance ($z = \emptyset$).

PLAIN These examples use the standing instructions directly: each argument can be predicted from a single standing instruction. All the relevant standing instructions, z, belong to the same domain.

MULTIHOP These examples contain at least one standing instruction in z that is relevant to the dialogue x only due to the presence of another standing instruction in z. These are of the form "if A then B" and "if B then C", where A, B, and C are slot names from the same domain. For example, in Table 1, choosing seating_class as *economy* is dependent on choosing airlines as *American Airlines*. Such examples test multi-hop reasoning abilities of the model.

MULTIDOMAIN These examples are like MULTIHOP except that there is an instruction in z that links two domains. These examples typically involve triggering API(s) from an additional domain while being consistent on any shared arguments such as location. For example, the user might request searching for Hotels when looking for places to visit (Travel). These example types require the identification of standing instructions relevant to either domain as well as sharing any common attributes, like location or date, across the domains. These examples challenge multi-domain understanding in addition to multi-hop reasoning.

SGD	Action	NLSI
User: Can you get on and get me a round trip ticket?	use as dialogue	**Dialogue:** User: Can you get on and get me a round trip ticket?
Agent: Where will you go? Where you coming from?	use as dialogue	Agent: Where will you go? Where you coming from?
User:I'm going to SFO from New York City.	use as dialogue, use as parameters	User: I'm going to SFO from New York City.
Agent: When are you leaving? When will you return?	discard	**Standing Instructions:**
User: I need to get back on the 14th. I really insist on getting American Airlines tickets. I have mile advntage with them. I'm taking off on Sunday this week.	convert to standing instruction	If I'm looking for a flight, American Airlines is my go-to
Agent: You're in luck, there's an American Airlines flight that takes off at 8:50 pm. You'll return leaving at 8:55 pm. You'll only pay $203 for everything.	discard	
User: Ok, just make sure I get the best economy deal	convert to standing instruction, dependent on the previous one	If I'm flying American Airlines, check for Economy seating class
		API Call:
Agent: Ok to be clear: 1 ticket from New York going to San Francisco on American Airlines at 8:50 pm on March 3rd, economy. You'll return boarding at 8:55 pm on March 14th.	discard this and future turns	`GetFlights( destination="SFO", origin="New York City", airlines="American Airlines ", seating_class="economy")`

Table 2: Converting an example from SGD dataset (Rastogi et al., 2020) into NLSI format. We show a per utterance decision process to obtain the dialogue, standing instructions, and parameters for the API call. We exclude parameters that cannot be converted into standing instructions. We exclude utterances not relevant to the creation of standing instructions.

MULTIPREFERENCE These examples contain standing instructions catering towards multiple preferences for the same attribute. The interpretation task for such examples requires placing multiple API calls respecting the different constraints. For example, searching for *Music* or *Sports* when looking for an event type.

CONFLICT These examples include instructions in the profile u that conflict with the information in the user utterance in the dialogue x. The model should gracefully handle such situations and give preference to the user's request.

Examples can contain standing instructions demonstrating multiple reasoning types. In NLSI, we associate each example with a single type as based on the above ordering - a type occurring later in the above ordering gets precedence.

3.2 Dataset Creation

We constructed NLSI by extending Schema Guided Dataset (SGD, Rastogi et al., 2020). SGD consists of multi-turn conversations across 20 domains like airlines or restaurants. We chose SGD because the dialogues in that dataset include natural and rich conversations and the accompanying annotations make it possible to construct the ground truth API labels. The process outlined below intends to repurpose an existing dataset for studying the selection and interpretation tasks. In a real-world setting, a user might provide explicit preferences through another interface, or else such preferences would be inferred from the user's continuous interaction with the system. We briefly discuss the dataset creation below and provide details in Appendix A.

Extracting standing instructions: We first identified which slots within the SGD schema can be translated into standing instructions based on the slot descriptions provided in the original dataset. For example, `theatre_name` is inclined to be a persistent user preference unlike `movie_title` or `date` which are likely to change with every interaction.

Each conversation in SGD originated from a sequence of actions that a user or agent should take alternately. For example, the second conversation in Table 1 was based on a template sequence like `Inform(airline_ticket)` $\rightarrow$ `Request(origin, dest)` $\rightarrow$ `Inform(origin, dest)` $\rightarrow$ `Offer(airlines)` $\rightarrow$ `Confirm(airlines), Request(seating_class)`. These sequences were then specialized by binding the variables, and the resulting sequence was written as a dialogue by a crowd worker that constituted this SGD example. We reverse-engineer the original SGD creation process to construct the standing instructions for NLSI.

To convert an SGD dialogue to an NLSI dia-

logue with standing instructions, we retained the first one or three turns as the conversational context x, and converted the remaining turns into the relevant standing instructions z. See an illustration in Table 2. Continuing our example, the natural language turns that specified `airlines="American Airlines"`, `seating_class="Economy"` were converted to standing instructions. We excluded information from any turns that could not be converted into a standing instruction - see the sixth utterance in the table.

We start with templated instructions for different scenarios in an if-then format akin to the work in Gupta et al. (2022). To convert these templated instructions into natural language, we use GPT-3 to paraphrase the templated instructions and obtain diverse instructions. We list the prompts to obtain these paraphrases in Appendix A.

Forming user profiles: The above process provides us with the *relevant* standing instructions z for the given example from SGD, but these are only part of the full user profile u. A user will have additional preferences that are not relevant to the given example. To emulate this, for the given example, we create u by augmenting z with M randomly sampled instructions from other examples. These "distractor" instructions are sampled from domains unrelated to the current domain(s).

API calls: The outputs of the interpretation task are API calls y, in line with the recent works of integrating LLMs with tools and plugins (Schick et al., 2023; Qin et al., 2023). The API calls are of the format `GetDomain(slot_1=value_1, slot_2=value_2)`. The argument names and values are derived from annotations in the SGD examples, which are either mentioned in the user's utterance or inferred in the standing instructions.

Dataset Statistics: We construct a balanced test set based on the different reasoning types: 340 per reasoning type, leading to a total of 2040 examples across 17 domains. The train set contains at most 10 examples per domain with a minimum of five examples per reasoning type, for a total of 150 examples. The remaining examples form the development set (251). There are 10.4 ± 3.0 instructions in a user profile (min: 3, max: 22) and there are 2.1 ± 1.7 relevant standing instructions per example in the dataset (min: 0, max: 10). There are 17 function calls corresponding to the 17 domains.

4 Methods

Given the recent success of using LLMs to generate outputs in structured prediction tasks (Roy et al., 2023; Schick et al., 2023; Heck et al., 2023), we use an LLM-based method to interpret a user utterance into a structured API call. We use in-context learning (Dong et al., 2023) by providing K demonstrations, where K is tuned on the dev set. These demonstrations are obtained by retrieving examples from the training set that are most similar to the current dialogue of the test example using the BM25 similarity measure (Robertson et al., 1994) as in Rubin et al. (2022); Roy et al. (2023). The examples are arranged in a best-first order. We describe the different paradigms (Figure 2) used for the interpretation task by selecting the instructions implicitly (DIRECT Interpretation), jointly (SELECT-AND-INTERPRET) or explicitly (SELECT-THEN-INTERPRET).

4.1 Direct Interpretation

In the **DIRECT** method, we do not have any explicit selection of standing instructions from the user profile, and directly interpret the dialogue context into API calls. The input to the LLM (Figure 2) consists of (i) instructions about the interpretation task including the information about using standing instructions, (ii) the schema of the dataset (list of functions and arguments that can be used when generating API calls) s, (iii) user profile u, (iv) user's dialogue x, and (v) API calls y. Of these, (iii)-(v) are repeated for every demonstration example and the test example only consists of the user profile and the dialogue. We also include the list of categorical slots and their categories as well as a list of boolean slots while describing the schema. This method is similar to the commercial usage of System Prompts. This setup allows us to evaluate the ability of implicit selection of the relevant standing instructions for the interpretation task.

4.2 Joint Selection and Interpretation

Inspired by the effectiveness of techniques like *Chain-of-Thought* prompting (Wei et al., 2022) across several tasks (Chu et al., 2023), we also treat the direct interpretation task with a two-step approach: generate the relevant standing instructions $z \subseteq u$ and then generate the corresponding API calls y. Such explicit selection can enhance the transparency of the system by exposing the relevant subset of instructions to the user (Liu et al.,

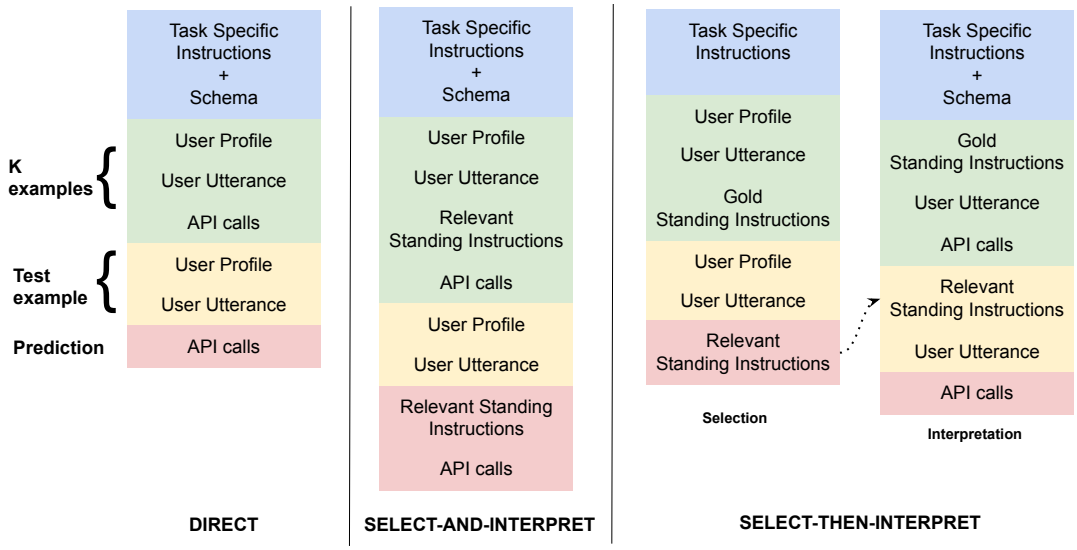

Figure 2: Illustration of different prompting methods. The blocks in red are the expected output generation and every other block is part of the input. The green bits are repeated K times, providing K demonstrations for in-context learning. DIRECT Interpretation conditions the generation of API calls on the user profile and user utterance. SELECT-AND-INTERPRET requires the generation of the appropriate standing instructions based on user profile and user utterance followed by API generation. SELECT-THEN-INTERPRET receives the predicted standing instructions from a separate Selection Model (see left) in addition to the user utterance and then generates the API calls. The selection step only generates the standing instructions based on the user profile and the user utterance.

2023). To implement the method, the input prompt to the LLM is modified such that the demonstrations include the set of all standing instructions u, the relevant standing instructions z, and then the API calls y (Figure 2). We refer to this method as **SELECT-AND-INTERPRET**.

4.3 Selection Then Interpretation

Here we treat selection and interpretation with two separate models. The interpretation model is similar to the one described for DIRECT, except that instead of user profile, the relevant standing instructions are used directly. By decoupling the selection task from the interpretation task, we can explore popular methods of information retrieval for selection. As the user profile size increases, and the instructions no longer fit into the prompt, a separate selection step can be convenient. We now describe various approaches for the selection step.

ORACLE: The selection step simply returns the true z. This setup measures the standalone performance of the interpretation task when given the correct standing instructions.

BM25: The selection step sets z to the N instructions from the user profile u that are most similar to the dialogue x using BM25 (Robertson et al., 1994), where N is tuned on the dev set. To compute the corpus statistics for BM25, each instruction in u is considered a document, and as is each standing instruction from the train examples.

CONTRIEVER: As above, replace BM25 with cosine similarity. The dialogue x and each standing instruction in u is embedded into $\mathbb{R}^{768}$ with a pretrained sentence encoder, CONTRIEVER (Izacard et al., 2022). Both BM25 and CONTRIEVER have been used as baselines in similar past work (Gupta et al., 2022; Salemi et al., 2023).

ICL: We also experiment with using LLMs for the selection task. The fixed input prompt to the LLM consists of instructions for the selection task, followed by exactly six demonstrations, each consisting of a dialogue x, user profile u, and relevant standing instructions z and then the test example (see Figure 2, Selection). We randomly sampled the six demonstrations from the training set, one per reasoning type, and used the same demonstrations for all the test examples.

ICL-DYNAMIC: Similar to ICL, except that now K demonstrations are dynamically retrieved from the train split by using the ones that are similar to the dialogue in the current example through BM25.

MULTI-PASS: In our preliminary experiments with LLM-based selection methods, we observed that the LLMs consistently missed a subset of relevant instructions in the MULTIHOP and MULTIDOMAIN reasoning types. We propose running the selection step multiple times to add these missing instructions. We use the standing instructions selected in the first pass of the selection process from ICL as part of the prompt to perform a new selection step. We instruct the model to find the standing instructions that are missing from the current selection set. Though the process can be iterated across multiple steps, we found the best results with only one additional round of selection.

5 Experiments

We benchmark the dataset on the above methods to explain the various challenges on the benchmark. We used GPT-3.5 (`text-davinci-003`), GPT-4 as the base LLMs from GPT family. We use LLaMA 2 (7B) for the selection task and CodeLLaMA 2 (7B) for the interpretation task from the LLaMA 2 family (Touvron et al., 2023).

5.1 Evaluation

For both selection and interpretation tasks, we report exact match and sample F1 score. The exact match for interpretation task is 1 when every function call and its arguments equal to the ground truth. We treat function_name-argument_name-argument_value as triples when computing F1 similar to the evaluation in dialogue state tracking (Dey et al., 2022). For the selection task, an exact match is when the set of predicted instructions is equal to the ground truth set of instructions. We post-process the outputs for both the tasks (see Appendix B), e.g. we exclude any predicted instructions not present in the user profile.

5.2 Results

We report the results for the different methods in Table 3. Overall, across all the methods, using GPT-4 as the base LLM has better results.

Within the different ways of incorporating the selection task with the interpretation task, we find that DIRECT interpretation gives the best result (as per EM), closely followed by the SELECT-AND-INTERPRET and then ICL when using GPT-3.5 and LLaMA 2. This trend shifts for GPT-4 where MULTI-PASS has the best results followed by ICL and DIRECT. Despite the success of chain-of-thought methods in tasks like mathematical reasoning (Wei et al., 2022) and multi-hop question answering (Yoran et al., 2023), we find that generating for selection and then generating API call within the same prompt may not be suitable for incorporating standing instructions.

We also experimented with fine-tuning smaller pre-trained models like RoBERTa (Liu et al., 2019) and CodeT5 (Wang et al., 2021) for the selection and interpretation task respectively. The selection task has EM/F1 results as 54.3/64.4. The interpretation task only reaches 7.6/37.3 suggesting that smaller models will require inclusion of techniques beyond fine-tuning such as cross-attention between the schema and the standing instructions, use of data augmentation *etc*. See Appendix C.2 for more details.

Models struggle to effectively incorporate standing instructions The best-performing configuration across all the methods only has an exact match of 46%. Considering the ORACLE method has an exact match of 58.5%, there is a considerable gap in performance. Incorporating standing instructions to interpret the user's context is not a trivial problem and would require approaches beyond the listed prompting methods. Even with the gold standing instructions in ORACLE, the models fail to achieve perfect exact match for interpretation, which shows the difficulty of the interpretation task. We attribute this to the examples in our dataset that require understanding from different contexts - standing instructions, list of valid APIs, and the current dialogue. Further, the relevance of standing instructions can be dependent on each other. This may explain why we found that standard retrieval approaches fail at this task. Our findings align with the observations made in other tasks that find the retrieval of some form of context from a separate memory to be challenging (Weir et al., 2023; Majumder et al., 2023).

Comparison across selection methods We find that LLM-based selection methods surpass traditional methods based on lexical statistics and embedding similarity for the GPT family as also seen in Sun et al. (2023). Further, the gap between the ORACLE setting in the selection module and the best-performing configuration is substantial on both exact match and F1, suggesting that selecting

Method	GPT-3.5				GPT-4				LLaMA 2 (7B)			
	Selection		Interpretation		Selection		Interpretation		Selection		Interpretation	
	EM↑	F1↑	EM↑	F1↑	EM↑	F1↑	EM↑	F1↑	EM↑	F1↑	EM↑	F1↑
DIRECT	N/A	N/A	**32.0**	**66.4**	N/A	N/A	42.0	67.9	N/A	N/A	**15.1**	**47.8**
SELECT-AND-INTERPRET	25.9	50.3	28.0	65.9	46.5	67.6	40.2	73.2	12.0	26.2	15.0	47.7
SELECT-THEN-INTERPRET												
BM25	17.3	19.3	11.2	39.7	17.3	19.3	11.8	40.8	**17.3**	19.3	7.8	30.9
CONTRIEVER	14.6	51.5	17.2	57.5	14.6	51.5	25.4	62.7	14.6	**51.5**	9.3	40.6
ICL	**33.5**	48.1	24.7	61.6	65.9	67.7	44.7	75.5	6.1	23.9	3.7	22.9
ICL-DYNAMIC	29.0	32.2	19.5	54.9	60.1	61.3	40.7	73.4	12.6	21.2	7.4	29.6
MULTI-PASS	24.3	**52.1**	20.6	57.2	**68.5**	**70.2**	**46.0**	**76.6**	8	14.3	5.3	22.0
ORACLE	N/A	N/A	55.9	82.8	N/A	N/A	58.5	84.1	N/A	N/A	36.5	68.7

Table 3: Results of the different methods on the NLSI dataset for the interpretation task and selection task evaluated on sample F1 and Exact Match (EM) by using different base LLMs from GPT and LLaMA families (LLaMA 2 (7B) for selection and CodeLLaMA 2 (7B) for interpretation). DIRECT has the highest score on exact match followed by SELECT-AND-INTERPRET for GPT-3.5 and LLaMA 2 (7B) while MULTI-PASS is best followed by ICL for GPT-4. For the selection task, LLM based models are better for GPT models while LLaMA 2 struggles on this task.

Type	ORACLE	DIRECT	JOINT	ICL	ICL-D	MULTI-P
NONEAPPLICABLE	68.2	57.3	48.8	61.4	62.6	61.1
PLAIN	77.9	67.6	70.5	69.7	65.0	70.8
MULTIHOP	65.5	56.4	47.3	59.1	57.9	60.2
MULTIPREFERENCE	55.8	24.1	32.6	42.6	38.2	44.7
MULTIDOMAIN	30.9	16.1	12.6	12.0	07.6	14.4
CONFLICT	70.2	35.0	32.0	33.5	22.3	34.4

Table 4: Per reasoning type exact match on the interpretation task (GPT-4). JOINT is SELECT-AND-INTERPRET, ICL-D is ICL-DYNAMIC and MULTI-P is MULTI-PASS. All the methods find PLAIN easiest while struggling at MULTIDOMAIN. There is no consistent winning method.

the relevant standing instructions explicitly from the user profile in the context of the conversation is itself challenging. This is most reflected in the LLaMA 2 (7B) results where the selection task has results worse than the BM25 and CONTRIEVER.

Over time, we envision the capability to add new standing instructions to user profiles, which might exceed the prompt's capacity. We anticipate that our benchmark can be useful for evaluating interesting questions in LLMs augmented with external memory (Lewis et al., 2020). Further, decoupling the selection step would provide more transparency, as it would allow users to see their individual standing instructions that influenced the generated output (Liu et al., 2023)

5.3 Results by reasoning type

We break down the examples by reasoning type in Table 4 with GPT-4 and investigate the accuracy of different methods (See Appendix C for remaining results). We observe that different methods display varying trends across different reasoning types and there is no one consistent *winner* among these methods. We find that PLAIN is the easiest reasoning type for all the methods, suggesting that LLMs do have the capacity to follow simple standing instructions. The methods perform worse on more complex MULTIDOMAIN examples (<17%) or MULTIPREFERENCE examples. These examples require sharing arguments across multiple domains, following individual standing instructions under respective domains, and reasoning across different standing instructions. Also, MULTI-PASS has improvement over MULTIDOMAIN and MULTIPREFERENCE suggesting that another round of selection can benefit the reasoning types where complex reasoning over the instructions is required.

5.4 Qualitative Analysis

We annotate 100 erroneous examples each from the DIRECT and ICL from GPT-3.5 with the most prominent error (See Table 9 for examples). Common errors include the hallucination of variables (Example 1) and missing arguments (Example 3) while generating API calls. For MULTIPREFERENCE, some predictions exclude the second API call. Further, if one of the repeating arguments has a standing instruction dependent on its value, the model does not include this conditional dependence when generating the API call (Example 2). For MULTIDOMAIN, some predictions exclude API calls from the remaining domains (Example 3). For DIRECT, overgeneration of API calls is common. The model is likely to confuse demonstrations from PLAIN with MULTIDOMAIN. Another possible reason is that the model incorrectly considers many irrelevant instructions in the profile while generating the API calls. For ICL, missing and incorrectly predicted standing instructions from the selection step produce erroneous arguments in the API calls.

6 Related Work

NL guidelines: Gupta et al. (2022) collected and released a dataset of NL guidelines that govern the safe response generation in dialogue systems. Compared to theirs, we showcase a more challenging retrieval setup: we have more applicable instructions on average, with rich phenomena such as MULTIHOP or MULTIPREFERENCE. Moreover, we are concerned with generating structured representations as a more complex final task.

Irfan et al. (2021) consider a variant of standing instructions in a barista setting where the instructions consist of the favourite drink and snack of the corresponding user. Similarly, Joshi et al. (2017) provide a user profile consisting of age, gender, and favourite food item structured as a dictionary to enhance the style of response generation that is appropriate to the selected attributes. Both these works use toy scenarios (Weston et al., 2015a), are single-domain, contain < 5 attributes for personalisation, and use non pretrained LSTM-based sequence-to-sequence methods (Weston et al., 2015b) for benchmarking. Our work offers more diverse scenarios, domains (17), and attributes (150). Our instructions are more complex than maintaining user preferences in a key-value format. We also explore the complexity of selecting relevant standing instructions often requiring multi-domain and multi-hop reasoning. More recently, commercial LLM providers also offer guidelines to enhance personalisation similar to the notion of standing instructions but lacks a reported systematic evaluation (See Appendix C).

The use of declarative NL specifications has been explored in past work. For example, Ye et al. (2023) use an LLM to generate a declarative task specification, coupled with an off-the-shelf automated theorem prover to derive the final answer. Weir et al. (2023) discuss methods to generate user-NPC dialogues based on game quest specifications. Constitutional AI (Bai et al., 2022) identifies whether some model response violates a given rule, and then revises the response accordingly.

Closely related to the use of standing instructions is also learning from feedback (Labutov et al., 2018; Tandon et al., 2022; Madaan et al., 2022), where the goal is to maintain a memory of user-provided feedback and use it to augment the knowledge used by question-answering models at test time. Analogously, standing instructions can also be seen as a form of memory.

Personalisation: Personalisation in dialogue has been extensively studied (Li et al. (2016); Zhang et al. (2018); Majumder et al. (2020); *inter-alia*) where the personality traits are provided through NL statements. However, all these works focus on providing a persona to the bot to generate more engaging responses rather than assisting the users in completing their request.

In a broader sense, learning from preferences has been fundamental to improving user experience. These include personalised review generation (Li et al., 2020), personalised search results through collaborative filtering (Micarelli et al., 2007) or leveraging a profile of user interests (Speretta and Gauch, 2005). Salemi et al. (2023) explored personalised text generation with LLMs on tasks such as article generation given past articles authored by the user. Our work provides incorporation of preferences explicitly through standing instructions. Such explicit mention will aid in better understanding of the generated result.

7 Conclusion

We proposed the use of standing instructions - a set of natural language statements that contain the user's preferences to aid the interpretation of the user's requests. To facilitate this, we created NLSI, a language-to-program dataset based on SGD. This enabled us to explore two tasks: standing instruction selection and interpretation task of generating API calls which are conditioned on the selected instructions and conversational context. We experimented with several methods for the selection and interpretation tasks.

Our results show that while LLMs are capable of incorporating standing instructions as an additional context to an extent, their usage of standing instructions is far from perfect. The models struggled to select the instructions in the user profile that were relevant to the given dialogue, which in turn affected the interpretation task. Moreover, as reasoning types become more intricate and involve complex reasoning or interactions among the respective standing instructions, the interpretation of these instructions becomes increasingly challenging for these methods. This calls for the development of new approaches in incorporating standing instructions, reasoning-based retrieval, and memory-augmented representations.

Ethics Statement

Our dataset is based on SGD (Rastogi et al., 2020) which consists of fictional conversations. The real world named entities such as restaurant names for the dataset were sampled from Freebase while date/times were sampled synthetically. No human names or any personal information is present in the dataset. Our task involves API call generation in a constratined setup which generally does not produce harmful or toxic responses.

Limitations

Our task setup is limited to generating API calls for the current turn. In an ideal scenario, the LLM or the service should also display the results in a user-friendly format, like natural language or Markdown, and perhaps confirm with the user before executing the call. Our dataset is not accompanied by the results from respective API calls or replies from the system due to the unavailability of results from the base dataset. The different reasoning types in our dataset are not exhaustive and future work could look into expanding them. The number of APIs in the dataset is 17 that currently fits in the prompt. In future iterations, as the number of APIs will increase beyond the prompt length, we would need to incorporate techniques from Qin et al. (2023); Ye et al. (2024) as an additional step to select the right APIs.

As our dataset is derived from an existing academic task-oriented dialogue dataset, it is useful for testing methods, but we caution readers that real-world services will include more complex standing instructions, domains, and user scenarios. The standing instructions were derived from templates and then adequately paraphrased. Despite this, we find it to be a challenging and non-trivial benchmark as evident in our results section Further, preferences stated explicitly by a human user would likely take a wider range of natural language forms. Preferences deduced from the user's past history might take a non-linguistic form, as in recommendation systems; they might be uncertain or soft constraints that cannot be passed directly as arguments to simple search APIs.

Acknowledgements

We thank Tom Hosking, Matthias Lindemann, and Katya Taktasheva for their feedback. We thank the anonymous reviewers for their useful suggestions.

References

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional AI: Harmlessness from AI feedback. *Computing Research Repository*, arXiv:2212.08073.

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.

Zheng Chu, Jingchang Chen, Qianglong Chen, Weijiang Yu, Tao He, Haotian Wang, Weihua Peng, Ming Liu, Bing Qin, and Ting Liu. 2023. A survey of chain of thought reasoning: Advances, frontiers and future. *Computing Research Repository*, arXiv:2309.15402.

Suvodip Dey, Ramamohan Kummara, and Maunendra Desarkar. 2022. Towards fair evaluation of dialogue state tracking by flexible incorporation of turn-level performances. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 318–324, Dublin, Ireland. Association for Computational Linguistics.

Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Lei Li, and Zhifang Sui. 2023. A survey on in-context learning. *Computing Research Repository*, arXiv:2301.00234.

Prakhar Gupta, Yang Liu, Di Jin, Behnam Hedayatnia, Spandana Gella, Sijia Liu, Patrick Lange, Julia Hirschberg, and Dilek Hakkani-Tur. 2022. Dialguide: Aligning dialogue model behavior with developer guidelines. *arXiv preprint arXiv:2212.10557*.

Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-chin Lin, Carel van Niekerk, and Milica Gasic. 2023. ChatGPT for zero-shot dialogue state tracking: A solution or an opportunity? In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 936–950, Toronto, Canada. Association for Computational Linguistics.

Bahar Irfan, Mehdi Hellou, and Tony Belpaeme. 2021. Coffee with a hint of data: Towards using data-driven approaches in personalised long-term interactions. *Frontiers in Robotics and AI*, 8.

Gautier Izacard, Mathilde Caron, Lucas Hosseini, Sebastian Riedel, Piotr Bojanowski, Armand Joulin, and Edouard Grave. 2022. Unsupervised dense information retrieval with contrastive learning. *Trans. Mach. Learn. Res.*, 2022.

Chaitanya K. Joshi, Fei Mi, and Boi Faltings. 2017. Personalization in goal-oriented dialog. *NeurIPS 2017 Conversational AI Workshop*.

Igor Labutov, Bishan Yang, and Tom Mitchell. 2018. Learning to learn semantic parsers from natural language supervision. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1676–1690, Brussels, Belgium. Association for Computational Linguistics.

Patrick S. H. Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. 2020. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Jiwei Li, Michel Galley, Chris Brockett, Georgios Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 994–1003, Berlin, Germany. Association for Computational Linguistics.

Junyi Li, Siqing Li, Wayne Xin Zhao, Gaole He, Zhicheng Wei, Nicholas Jing Yuan, and Ji-Rong Wen. 2020. Knowledge-enhanced personalized review generation with capsule graph neural network. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 735–744.

Yang Liu, Yuanshun Yao, Jean-Francois Ton, Xiaoying Zhang, Ruocheng Guo, Hao Cheng, Yegor Klochkov, Muhammad Faaiz Taufiq, and Hang Li. 2023. Trustworthy llms: a survey and guideline for evaluating large language models' alignment. *Computing Research Repository*, arXiv:2308.05374.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized bert pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

Aman Madaan, Niket Tandon, Peter Clark, and Yiming Yang. 2022. Memory-assisted prompt editing to improve gpt-3 after deployment. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 2833–2861.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? You probably enjoy nature: Persona-grounded dialog with commonsense expansions. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 9194–9206.

Bodhisattwa Prasad Majumder, Bhavana Dalvi Mishra, Peter Jansen, Oyvind Tafjord, Niket Tandon, Li Zhang, Chris Callison-Burch, and Peter Clark. 2023. CLIN: A continually learning language agent for rapid task adaptation and generalization. *Computing Research Repository*, arXiv:2310.10134.

Alessandro Micarelli, Fabio Gasparetti, Filippo Sciarrone, and Susan Gauch. 2007. Personalized search on the World Wide Web. In Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl, editors, *The Adaptive Web: Methods and Strategies of Web Personalization*, volume 4321 of *Lecture Notes in Computer Science*, pages 195–230. Springer.

OpenAI. 2023. GPT-4 technical report. *Computing Research Repository*, arXiv:2303.08774.

Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2023. ToolLLM: Facilitating large language models to master 16000+ real-world apis. *Computing Research Repository*, arXiv:2307.16789.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for SQuAD. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789, Melbourne, Australia. Association for Computational Linguistics.

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.

Stephen E. Robertson, Steve Walker, Susan Jones, Micheline Hancock-Beaulieu, and Mike Gatford. 1994. Okapi at TREC-3. In *Text Retrieval Conference*.

Subhro Roy, Sam Thomson, Tongfei Chen, Richard Shin, Adam Pauls, Jason Eisner, and Benjamin Van Durme. 2023. BenchCLAMP: A benchmark for evaluating language models on syntactic and semantic parsing. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Ohad Rubin, Jonathan Herzig, and Jonathan Berant. 2022. Learning to retrieve prompts for in-context learning. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2655–2671, Seattle, United States. Association for Computational Linguistics.

Alireza Salemi, Sheshera Mysore, Michael Bendersky, and Hamed Zamani. 2023. LaMP: When large language models meet personalization. *Computing Research Repository*, arXiv:2304.11406.

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 *Thirty-seventh Conference on Neural Information Processing Systems*.

Mirco Speretta and Susan Gauch. 2005. Personalized search based on user search histories. *The 2005 IEEE/WIC/ACM International Conference on Web Intelligence (WI'05)*, pages 622–628.

Weiwei Sun, Lingyong Yan, Xinyu Ma, Shuaiqiang Wang, Pengjie Ren, Zhumin Chen, Dawei Yin, and Zhaochun Ren. 2023. Is ChatGPT good at search? investigating large language models as re-ranking agents. *Computing Research Repository*, arXiv:2304.09542.

Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352, Seattle, United States. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and fine-tuned chat models. *Computing Research Repository*, arXiv:2307.09288.

Yue Wang, Weishi Wang, Shafiq Joty, and Steven C.H. Hoi. 2021. CodeT5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *NeurIPS*.

Nathaniel Weir, Ryan Thomas, Randolph D'Amore, Kellie Hill, Benjamin Van Durme, and Harsh Jhamtani. 2023. Ontologically faithful generation of non-player character dialogues. *Computing Research Repository*, arXiv:2212.10618.

Jason Weston, Antoine Bordes, Sumit Chopra, and Tomas Mikolov. 2015a. Towards ai-complete question answering: A set of prerequisite toy tasks. *Computing Research Repository*, arXiv:1502.05698.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015b. Memory networks. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Junjie Ye, Yilong Wu, Songyang Gao, Caishuang Huang, Sixian Li, Guanyu Li, Xiaoran Fan, Qi Zhang, Tao Gui, and Xuanjing Huang. 2024. RoTBench: A multi-level benchmark for evaluating the robustness of large language models in tool learning. *Computing Research Repository*, arXiv:2401.08326.

Xi Ye, Qiaochu Chen, Isil Dillig, and Greg Durrett. 2023. SatLM: Satisfiability-aided language models using declarative prompting. In *The 3rd Workshop on Mathematical Reasoning and AI at NeurIPS'23*.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. *Computing Research Repository*, arXiv:2304.13007.

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 (Volume 1: Long Papers)*, pages 2204–2213, Melbourne, Australia. Association for Computational Linguistics.

A Dataset Construction Details

We provide further details about dataset construction.

Forming examples for different reasoning types: We do not need to extract any standing instructions z for examples in NONEAPPLICABLE. For examples in PLAIN, each (domain, slot, value) triple was extracted and written in natural language via an if-then template and appropriately paraphrased. Since each slot is independent of each other, this set of instructions form z. MULTIHOP examples were formed by creating a hierarchy of slots associated with the same domain like *seating_class* is dependent on *airlines*. If the subsequent dialogue states contained the same dependent slots, then that example was categorized as a MULTIHOP example, where the primary slot value was obtained from the dialogue or one of the standing instructions. MULTIDOMAIN examples were dialogues from SGD that were inherently multi-domain because they required API calls from different domains. These reasoning types were created through a deterministic process based on the existing SGD data.

MULTIPREFERENCE examples were formed by duplicating one of the ground truth standing instructions from PLAIN, MULTIHOP and MULTIDOMAIN, and substituting an argument value with another relevant entity. Meanwhile, CONFLICT examples were formed with examples from PLAIN or MULTIHOP. We added information that conflicts with the gold standing instruction like asking for *Mexican* restaurants when the standing instruction is about preference for *Italian* restaurants. We provide examples for the remaining reasoning types in Table 5.

Sampling instructions for user profile: We drew M instructions uniformly from the range $[3, 12]$. In particular, we drew the distractor instructions before splitting the dataset into train/dev/test, so training examples were constructed with some distractors sourced from the test set. Given this dataset, however, our experiments followed the usual protocol of holding out the test set while constructing our systems.

Post-processing: We also included several rounds of post-processing on the dataset to remove undesirable or unrealistic situations that arise either through the noise in the base dataset or our extraction process. We removed examples with domain mismatches in case of MULTIDOMAIN such as requesting music which is followed by a request for bus ticket booking. We unified domains such as *Restaurant_1*, *Restaurant_3* as *Restaurants*. *Restaurant_2* was renamed as *HouseStays*. We also deduplicated the slot names under these domains like *location* and *area* was converted to *area*. Similarly, the *Services* domain was expanded as *Salons*, *Doctors*, and *Dentists* instead. All the examples were constructed only from the domains and examples available in the training set of SGD. In addition to removing domains whose combination doesn't make sense in the MULTIDOMAIN reasoning type, we also remove MULTIDOMAIN examples which do not have any attributes for the second domain.

The instructions obtained through the above deterministic process were templated. For paraphrasing the templated instructions, we prompted GPT-3 to generate paraphrases with three distinct prompts to promote diversity.

Prompt 1: `Write a colloquial paraphrase for the given sentences. Refrain from using if then format`

Prompt 2: `Reword the following in your own words. Keep the same meaning. Change the sentence structure to exclude if then format:`

Prompt 3: `Reword the following in your own words. Keep the same meaning. Make the sentences sound like instructions or commands.`
`Change the sentence structure to exclude if-then format. If the sentence starts with "If I ask for xyz", also reword that xyz part.`

We replace the templated standing instruction randomly with one of the paraphrases leading to 4097 unique instructions across the dataset.

B Experiment Details

B.1 Setup

For the selection experiments involving BM25 and Contriever, N was varied from 1 to 10 and chosen according to the best exact match on the dev set (N=4 for BM25, N=2 for CONTRIEVER). For LLMs, the K for demonstrations was varied among {3,5,8}, with K=5 being best for ICL-DYNAMIC and other interpretation tasks. For the MULTIPASS experiments, we varied K for three additional rounds and found that providing one additional pass had the best results on the development set. We use temperature of 0 while decoding from the LLMs unless specified otherwise. We use LLaMA 2 7B[6]

[6] https://huggingface.co/meta-llama/Llama-2-7b-hf

	CONFLICT	NONEAPPLICABLE	MULTIDOMAIN
User Profile (u)	>When I request Restaurants, I prefer Italian cuisine. >If I'm looking for a doctor, I'd rather have a General Practitioner. >If I'm opening a bank account, I want it to be a savings account. >I'd like to get a Doctor in San Rafael if I can. ...	>Request Restaurants with Filipino cuisine as my preference. >Request Music by Iggy Azalea as my preferred artist. >If I'm looking to go to the movies, my go-to theatre is Airport Stadium Cinemas. >If I'm looking for a flight, my go-to airline is Alaska Airlines. >Request Events, specifically Sports events.	>When I request Movies, I typically enjoy ones that are comedic. >My first choice when requesting Travel is Vegas >When it comes to Hotels, I prefer ones that are rated 1-star. >My go-to theater for Movies is AMC Bay Street. >If I'm looking into Travel, I should also check out Hotels >I'd like my travel to be kid-friendly. ...
Relevant Standing Instructions (z)	>I'd like to get a Doctor in San Rafael if I can.	None	>My first choice when requesting Travel is Vegas >If I'm looking into Travel, I should also check out Hotels. >When it comes to Hotels, I prefer ones that are rated 1-star. I'd like my travel to be kid-friendly.
Conversation (x)	*User*: I need to find a Gynecologist	*User*: Can you help me find some attractions to see? *Agent:* Where should I look? *User*: How about in KL?	*User*: User: Any good tourist traps out there?

API calls (y)	`GetDoctors(`
	`  type="Gynecologist",`
	`  location="San Rafael")`
		`GetTravel(`
		`  location="KL")`
			`GetTravel(`
			`  good_for_kids="True"`
			`  location="Vegas")`
			`GetHotels(`
			`  average_rating="1",`
			`  location="Vegas")`

Table 5: Some examples from NLSI. (1) In CONFLICT, user requests for an attribute that is against the standing instructions ("Gynecologist" v/s "General Practionier"). (2) In NONEAPPLICABLE, the user makes a request which is not affected by the standing instructions. (3) In MULTIDOMAIN, the examples contain an instruction which requires invoking a hotel search for the same location when user requests for places to visit.

for the selection experiments. As our API calls are similar to the python syntax of a function, we use CodeLLamA 2 7B, which is instruction fine-tuned, [7] for the interpretation experiments. We also found CodeLLaMA 2 (7B) had better results than LLaMA 2 (7B) for the interpretation task on the validation set. We use 2 24GB GPUs, batch size of 1, full precision models for the these experiments. It takes approx 48 hours to make a pass over the entire test set.

For evaluation, all the outputs were converted to lowercase and double quotes were unified to a fixed unicode. Using "vs" and "versus" was unified to "versus". The models were not penalised if they produced *subcategory* instead of *event_type* arising due to the noise in the base dataset. For the interpretation evaluation, the API calls were converted to function_name-slot-value triples per slot-value per API call. In the case of examples multiple API calls, the models had a tendency to include every attribute in a single API call instead of separate API calls. To penalise this in the exact match, if the number of predicted API calls was not equal to the number of ground truth API calls the model received an exact match of 0.

[7] https://huggingface.co/codellama/CodeLlama-7b-Instruct-hf

B.2 Prompts

We shall now list the prompts used in our experiments.

B.2.1 Selection Task

For the selection tasks, the prompt is described in Figure 3. For the MULTI-PASS experiments, an additional instruction was added to the prompt "If some instructions are missing from the current set, generate those instructions under Remaining Applicable Standing Instructions". The test example consists of "Applicable Standing Instructions" from the previous iteration and "Remaining Applicable Standing Instructions" is appended with every demonstration.

B.2.2 Interpretation Task

We describe the prompt in Figure 4 used for Direct Interpretation and SELECTION-THEN-INTERPRETATION methods. The set of standing instructions will vary depending on the type of experiment. For JOINT SELECTION AND INTERPRE-

```
Standing instructions allow a user to add preferences or requirements that an agent would like to consider when generating its responses.
The user's current utterance in the dialogue has priority over standing instructions.
For the given dialogue, which of the following standing instructions are applicable? If no standing instructions are applicable, then generate "None".

Standing Instructions:
<demonstration standing instructions>

Dialogue:
<demonstration dialogue>

Applicable Standing Instructions:
<demonstration applicable standing instructions>
<EOS>

Standing Instructions:
<test standing instructions>

Dialogue:
<test dialogue>
```

Figure 3: Prompt for the ICL Selection task. The number of examples and the type of examples will vary according to the experiment

TATION, the prompt includes an additional sentence "For the following dialogue, first generate the appropriate applicable standing instructions from the user profile and then generate API calls based on the dialogue and the selected standing instructions." between "Standing instructions allow you to add preferences or requirements that an agent would like to consider when generating the parser." and "The user's current utterance in the dialogue has priority over standing instructions.". The demonstration and test example format look as Figure 5.

```
Dialogue:
<demonstration dialogue>

User Profile:
<demonstration standing instructions>

Applicable Standing Instructions
<applicable demonstration standing instructions>

API Calls:
<demonstration api calls>
<EOS>

Dialogue:
<test dialogue>

User Profile:
<test standing instructions>
```

Figure 5: Demonstration and test example format for Select-And-Interpret experiments

C Additional Results

C.1 Dependence on paraphrasing

We experiment with five different random seeds for the dataset creation, creating five different versions of the dataset. We evaluate the DIRECT method on the LLAMA-2 model for the development set. The average exact match across these datasets is 15.1±0.7 suggesting only small variance.

Selection Method	Interpretation Training Data	EM	F1
QA	User Profile	11.2	43.0
QA	Applicable	12.2	42.4
Oracle	User Profile	13.2	47.3
Oracle	Applicable	15.5	50.1

Table 6: Interpretation task scores when fine-tuned with User Profile and Applicable standing instructions respectively for the interpretation task while using "Oracle" or standing instructions obtained from a fine-tuned QA model (based on RoBERTa). Fine-tuned models struggle at the interpretation task and a model trained with applicable standing instructions is better.

C.2 Fine-tuning experiments

We fine-tune smaller pre-trained models to benchmark them on the NLSI dataset.

Selection Task: We start with trained extractive question-answering system that uses RoBERTa-base (Liu et al., 2019) as the encoder and SQuAD 2.0 (Rajpurkar et al., 2018) as the training dataset. [8] In our setup, the dialogue forms the paragraph, and ["yes"] and ["no"] are appended to the start of the dialogue. The question is "Is the standing instruction X applicable" and if the predicted answer is "yes", the respective instruction X is selected. This process is repeated for every instruction in the user profile. We further fine-tune this question-answering model by converting every example in the training set into such a format.

Interpretation Task: We fine-tune a code-specific pre-trained model, namely CodeT5 (220 M) (Wang et al., 2021) on NLSI dataset. As this is a Sequence-to-Sequence model, the input consists of the dialogue concatenated with the instructions

[8] https://huggingface.co/deepset/roberta-base-squad2

Type	ORACLE	DIRECT	JOINT	ICL-D	ICL	MULTI-P
NONEAPPLICABLE	65.3	45.9	37.9	54.4	58.5	29.4
PLAIN	80.3	56.2	56.5	41.8	28.5	36.5
MULTIHOP	65.3	41.8	34.1	27.6	19.1	34.1
MULTIPREFERENCE	40.0	11.5	11.5	8.8	4.1	9.7
MULTIDOMAIN	23.2	3.5	3.2	0.6	0.3	1.2
CONFLICT	70.3	34.1	26.2	17.1	6.8	14.7

Table 7: Per reasoning type exact match on the interpretation task (GPT-3.5). ICL-D is ICL-DYNAMIC and MULTI-P is MULTI-PASS. All the methods find PLAIN easiest and struggle on MULTIDOMAIN. Different methods show different trends without a consistent winner.

Type	ORACLE	DIRECT	JOINT	ICL	ICL-D	MULTI-P
NONEAPPLICABLE	45	24.4	23.8	4.1	27.9	17.6
PLAIN	62.1	36.2	37.1	8.8	7.4	5.3
MULTIHOP	48.2	17.1	17.4	1.5	1.5	2.9
MULTIPREFERENCE	19.4	5.3	4.4	0.9	1.5	0.6
MULTIDOMAIN	3.2	1.2	0.6	0.3	0.3	0.0
CONFLICT	48.8	8.2	7.4	7.4	6.5	5.8

Table 8: Per reasoning type exact match on the interpretation task (LLaMA 2). JOINT is SELECT-AND-INTERPRET, ICL-D is ICL-DYNAMIC and MULTI-P is MULTI-PASS. All the methods find PLAIN easiest while struggling at MULTIDOMAIN. There is no consistent winning method.

from the user profile and the output consists of the API calls. This is similar to the DIRECT method discussed in Section 4. To simulate the SELECT-THEN-INTERPRET paradigm, we design two interpretation models, one using all the standing instructions from the user profile and the other using the applicable standing instructions only (Applicable). **Results**: The stand-alone selection task leads to an Exact Match/F1 score of 54.3/64.4 which provides a strong baseline result. The DIRECT interpretation results in 7.6/37.3 indicative of a need for better interpretation models. The results for SELECT-THEN-INTERPRET with smaller models are reported in Table 6. We find that SELECT-THEN-INTERPRET has improved results over DIRECT unlike some of the LLM results. We further find that using applicable standing instructions during the training of the interpreter leads to better results. Even with oracle instructions and interpreter trained with applicable instructions, the interpretation task has poor capabilities.

C.3 Scenario Type results for GPT-3.5 and LLaMA 2

We report the results by reasoning type for experiments using base LLM as GPT-3.5 in Table 7 and LLaMA 2 in Table 8. The trends are similar to the trends discussed in Section 5.3.

C.4 OpenAI's Custom Instructions

OpenAI also recently reported the introduction of custom instructions[9] that allow the users to add requirements or preferences that ChatGPT should consider when generating the responses. This is similar to our notion of standing instructions. To test the effectiveness of this feature (free version), we use the instructions from the user profile as "custom instructions". We pose the API generation task as a standalone task and hope for the model to directly incorporate the standing instructions from the custom instructions. We also use the **ICL** setup to provide examples about the task as discussed in Section 4.3. As this effort requires manual copy-pasting of examples, we randomly selected and evaluated 17 examples per type, amounting to 102 test examples. While not directly comparable with Table 3, the exact match for the interpretation task on this subset is 15.6 and the slot F1 score is 45.5. Thus, the model does not necessarily incorporate the correct custom instructions every time. It is prone to copying arguments from the demonstration example as well as hallucinating the arguments and their values. For some examples, the model is prone to over-generation of API calls and other unrelated text. We remark that due to the opacity of the "custom instructions" UI, we do not know the prompt or exact LLM API call and we have observed nondeterminism in the outputs.

C.5 Qualtitative Analysis

We list some examples exhibiting prominent errors discussed in Section 5.4 in Table 9.

[9] https://openai.com/blog/custom-instructions-for-chatgpt

Reasoning Type	Conversation	Gold SI	Gold API	Predicted API	Error
PLAIN	*User*: I am looking for a place of interest to visit.	>If I'm looking to go somewhere, my top choice is New York City.	`GetTravel(category="Tourist Attraction", location="New York City)`	`GetTravel(category="Place of Interest", location="New York City")` `GetMusic(genre="Country")` `GetMedia(genre="Drama", directed_by="Qui Sheng")` ...	Hallucination
MULTI PREFERENCE	*User*: I fancy watching a movie at home this evening	>When I request Media, my favorite type is adventure. >*If you're looking for an adventure movie, search for one directed by Joel Coen*. >When I request Media, my favorite type is biographical.	`GetMedia(directed_by="Joel Coen", genre="adventure")` `GetMedia(genre="biographical")`	`GetMedia(genre="adventure"),` `GetMedia(genre="biographical")`	Skipped Instruction
MULTI DOMAIN	*User*: I'm looking for some interesting attractions to visit, specifically a Museum. Can you help me find any?	>I prefer London as my destination when I travel. >If I'm looking into Travel, I should also check out Hotels. >I prefer to stay in hotels that have a two-star rating when I am making reservations.	`GetTravel(category="Museum", location="London")` `GetHotels(average_rating="2", location="London")`	`GetTravel(category="Museum")`	Missing API Prediction

Table 9: Examples of prominent errors across the DIRECT and ICL methods (with GPT-3). The incorrectly predicted or missing arguments and function calls are marked in red. The DIRECT models tend to produce several unrelated API calls (first example). Both DIRECT and ICL have a tendency to miss an argument that is only dependent on one of the attributes in MULTIPREFERENCE, in this case missing the director *Joel Coen*. Majority of predictions in MULTIDOMAIN fail at generating the API calls for the second domain.

```
You are designing a parser that takes in a user utterance and some standing instructions and outputs a set of API calls.
Every API call consist of "GetX" where X is domain name and uses slot names listed below as arguments.
We list the domain name followed by the list of possible slot names. Some slot names can be categorical or boolean
The values for the arguments can come from the user's dialogue or standing instructions. If the user requests a slot name and no value is found, use "?".
If the user requests dontcare, use value as "any".
Standing instructions allow you to add preferences or requirements that you'd like to consider when generating the parser.
If standing instructions are applicable across multiple domains, place an API call per situation per domain.
If some of the applicable standing instructions have instructions of similar type, place multiple API calls respecting the standing instructions.
If some slots are applicable across several domains, generate the respective slot names for the respective domains.

Schema:
Banks: recipient_account_name, amount, recipient_account_type
Buses: origin, departure_date, fare_type, transfers, price, group_size, destination, destination_station_name, origin_station_name, departure_time
Events: event_name, city, category, event_location, number_of_tickets, time, address_of_location, date, venue_address, event_type
Flights: origin, inbound_arrival_time, is_redeye, outbound_departure_time, outbound_arrival_time, inbound_departure_time, return_date, airlines,
seating_class, refundable, number_stops, destination_airport, departure_date, fare, destination, passengers, origin_airport
Homes: pets_allowed, visit_date, address, property_name, rent, number_of_baths, area, number_of_beds, furnished, phone_number
Hotels: has_wifi, average_rating, check_out_date, price, pets_welcome, number_of_days, location, check_in_date, phone_number,
number_of_rooms, street_address, hotel_name
HouseStays: rating, phone_number, has_laundry_service, check_out_date, total_price, check_in_date, address, number_of_adults, where_to
Media: title, directed_by, subtitles, genre
Movies: theater_name, movie_name, price, show_date, location, show_time, number_of_tickets, genre, show_type, street_address
Music: song_name, year, album, artist, genre, playback_device
RentalCars: dropoff_date, pickup_time, pickup_city, pickup_date, total_price, car_type, car_name, pickup_location
Restaurants: price_range, restaurant_name, city, has_live_music, serves_alcohol, time, date, phone_number, cuisine, street_address, party_size
Salons: is_unisex, average_rating, city, appointment_date, appointment_time, stylist_name, phone_number, street_address
Dentists: dentist_name, phone_number, offers_cosmetic_services, city, appointment_date, appointment_time, address
Doctors: doctor_name, city, average_rating, appointment_date, appointment_time, type, phone_number, street_address
Travel: good_for_kids, category, attraction_name, location, phone_number, free_entry
Weather: city, temperature, date, precipitation, humidity, wind


Further, following slots have categorical values:
recipient_account_type: checking, savings
fare_type: Economy, Economy extra, Flexible
(Events) category: Place of Worship, Theme Park, Museum, Historical Landmark, Park, Tourist Attraction, Sports Venue, Shopping Area,
Performing Arts Venue, Nature Preserve
event_type: Music, Sports
seating_class: Economy, Premium Economy, Business, First Class
refundable: True, False
airlines: United Airlines, American Airlines, Delta Airlines, Southwest Airlines, Alaska Airlines, British Airways, Air Canada, Air France
show_type: regular, 3d, imax
playback_device: TV, kitchen speaker, bedroom speaker
(Doctors) type: Gynecologist, ENT Specialist, Ophthalmologist, General Practitioner, Dermatologist
car_type: Compact, Standard, Full-size
price_range: inexpensive, moderate, expensive, very expensive


Further, following slots are boolean:
has_wifi, pets_allowed, subtitles, offers_cosmetic_services, has_laundry_service, is_unisex,

good_for_kids, has_live_music, pets_welcome, serves_alcohol, is_redeye, furnished, free_entry


Dialogue:
<demonstration dialogue>

Standing Instructions:
<demonstration instructions>

API Calls:
<demonstration api calls>
<EOS>

Dialogue:
<test dialogue>

Standing Instructions:
<test instructions>

API Calls:
```

Figure 4: Prompt used for interpretation experiments. We include the template for demonstration examples and test examples in this figure. Note the demonstration examples will be repeated based on the number of demonstration examples used