Exploring Usability Issues in Instruction-Based and Schema-Based Authoring of Task-Oriented Dialogue Agents

Amogh Mannekote
amogh.mannekote@ufl.edu
University of Florida
United States

Mehmet Celepkolu
mckolu@ufl.edu
University of Florida
United States

Joseph B. Wiggins
jbwiggi3@ufl.edu
University of Florida
United States

Kristy Elizabeth Boyer
keboyer@ufl.edu
University of Florida
United States

ABSTRACT

Platforms such as Google DialogFlow and Amazon Lex have enabled easier development of conversational agents. The standard approach to training these agents involve collecting and annotating in-domain data in the form of labelled utterances. However, obtaining in-domain data for training machine learning models remains a bottleneck. Schema-based dialogue, which involves laying out a structured representation of the flow of a "typical" dialogue, and prompt-based methods, which involve writing instructions in natural language to large language models such as GPT-3, are promising ways to tackle this problem. However, usability issues when translating these methods into practice are less explored. Our study takes a first step towards addressing this gap by having 23 students who had finished a graduate-level course on spoken dialogue systems report their experiences as they defined structured schemas and composed instruction-based prompts for two task-oriented dialogue scenarios. Through inductive coding and subsequent thematic analysis of the survey data, we explored users' authoring experiences with schema and prompt-based methods. The findings provide insights for future data collection and authoring tool design for dialogue systems.

CCS CONCEPTS

• **Human-centered computing → User studies**; **Usability testing**; • **Computing methodologies → Discourse, dialogue and pragmatics**.

KEYWORDS

dialogue systems; schema-based dialogue; zero-shot prompting; user studies;

ACM Reference Format:
Amogh Mannekote, Mehmet Celepkolu, Joseph B. Wiggins, and Kristy Elizabeth Boyer. 2023. Exploring Usability Issues in Instruction-Based and Schema-Based Authoring of Task-Oriented Dialogue Agents. In *ACM conference on Conversational User Interfaces (CUI '23), July 19–21, 2023, Eindhoven, Netherlands.* ACM, New York, NY, USA, 6 pages. https://doi.org/10.1145/3571884.3604310

1 INTRODUCTION

Large language models like T5 [19] and GPT-3 [3] have significantly streamlined the development of task-oriented dialogue systems. However, two main challenges hinder the widespread adoption of conversational agents in new domains: 1) the requirement of machine learning expertise for training or fine-tuning language models on specific dialogue tasks, and 2) the limited availability of labeled in-domain training data [4].

Commercial platforms such as Google DialogFlow and Amazon Lex attempt to address the first challenge by offering user-friendly development interfaces that cater to machine learning non-experts. However, these platforms generally rely on fragmented pipeline architectures, which may compromise the full potential of the resulting dialogue system. Furthermore, their performance is heavily dependent on the presence of substantial in-domain training data [22].

In response to the second challenge of data scarcity, data-efficient machine learning approaches that require only a handful of demonstrations (termed "few-shot learning") have emerged [23]. Until recently, few-shot learning methods primarily focused on learning models for *individual* components within a dialogue system's pipeline, such as natural language understanding [18], dialogue state tracking [5], and response generation [18]. The special case of few-shot learning where the model requires zero demonstrations is called "zero-shot learning [21]." In contrast to the approaches that seek to develop *-shot models in a piecemeal fashion for each module, this study examines two general-purpose paradigms that allows designers to specify the *overall* behavior of dialogue agents that have gained traction in recent years: structured dialogue schemas and instruction-based natural language prompts.

1.1 Dialogue Schemas

A dialogue schema in the context of task-oriented dialogue is a structured representation of the conversation flow, and can capture the key entities (also known as 'slots'), their relationships, and the potential paths a user can take to accomplish a specific task or goal in the dialogue. For instance, in the context of a hotel-reservation

CUI '23, July 19–21, 2023, Eindhoven, Netherlands
© 2023 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-0014-9/23/07.
https://doi.org/10.1145/3571884.3604310

dialogue, the slots could include the number of people to book the room for, the duration of stay, and the check-in date. Schema-based approaches facilitate knowledge transfer across previously unseen domains using a minimal number of training examples [1, 16, 20]. The primary focus of current research in this area is model development [6, 11, 12, 14, 17]. However, to truly extend the applicability of dialogue schemas beyond research settings and into broader development environments "in the wild" [10], it is crucial to address the challenges of transforming these schema formats into practical and user-friendly interfaces.

1.2 Instruction-Based Prompts

The rapid advancements in large language models' capacity to interpret natural language instructions have sparked a significant increase in the use of instruction-based prompts [15] for executing various NLP tasks in a zero-shot manner [13]. While prompts can adopt diverse forms, including in-context examples, employing natural language instructions is particularly fitting for developing dialogue agents. This approach mirrors how humans learn and are instructed to communicate in new domains, such as when a human customer service representative is trained to handle support for a novel product.

Both schemas and instruction-based prompts serve as potent tools for democratizing the creation of task-oriented dialogue systems by significantly lowering the technical barrier associated with developing such an agent. To accomplish this goal, it is imperative for researchers to delve deeper into the usability issues associated with these formats, going beyond the exclusive focus on model performance metrics. As a preliminary effort towards gaining valuable insights, we conducted an exploratory research study to answer the following research question: *What are the usability trade-offs and issues involved in authoring a task-oriented dialogue agent using schemas and instructional prompts?* To answer this research question, we introduced two hypothetical dialogue scenarios in a taxi ride-booking domain. We presented both scenarios to our participants and asked them to author dialogue agents for each scenario using both a schema and an instruction-based prompt.

2 METHODS

To explore our research question, we conducted an IRB-approved study involving 23 participants by asking them to author task-oriented dialogue agents using both schema-based and prompt-based interfaces and asked them to report their experiences. The study focused exclusively on participants' self-reported experiences, which were gathered through post-surveys, to analyze the usability of schema-based and prompt-based authoring methods for dialogue systems. Although the study produced artifacts in the form of written instructions and constructed schemas, we chose not to conduct an in-depth analysis of these because the primary evaluation criterion for authored instructions or schemas should be the model performance on dialogue tasks. However, during the study period, both schema-based and prompt-based dialogue models were in their nascent stages, rendering performance assessments premature.

2.1 Study Procedure

We provided participants with two dialogue scenarios. For each scenario, they first defined their dialogue agent using an instruction-based prompt. Then, they defined the agent using a dialogue schema of their choosing (Procedural or Declarative, both described in Section 2.3). In total, each participant carried out *four* authoring activities in a session. The study was conducted remotely via Zoom (the study was hosted as a web application) and was constrained to a 90-minute time-frame; however, participants could complete their activity ahead of schedule and conclude the session.

2.2 Dialogue Scenarios

Each participant developed task-oriented dialogue agents for *two* scenarios in the taxi ride-booking domain. Each scenario required participants to create a dialogue agent using a structured dialogue schema as well as an instruction-based prompt. The scenarios are briefly described below (the full text of the scenario descriptions are given in Appendix A).

Scenario 1: Adding Service Tiers to an Existing Ride-Booking Flow. In this scenario, participants were informed that the taxi service provider intended to offer three different service tiers (*XL*, *Share*, and *Regular*) to their customers. The goal was to adapt an existing dialogue flow for booking a taxi ride by adding a new slot that captured and utilized the customer's preferred ride tier.

Scenario 2: Integrating a Ride Cancellation Feature. In this scenario, participants were asked to incorporate a new functionality into the existing taxi ride-booking system, allowing users to cancel previously booked taxi rides. When processing a ride cancellation request, the system needs to follow a series of steps: 1) obtain the customer's name and phone number 2) make an API call to fetch the Booking ID and any associated cancellation fees, 3) notify the customer of any applicable cancellation charges, 4) get the customer's confirmation to proceed with the cancellation, and 5) perform a second API call to finalize the cancellation process.

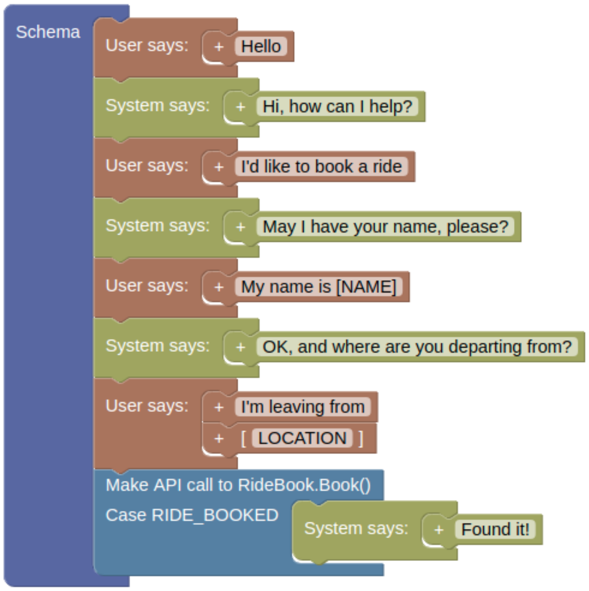
2.3 Schema Formats

In this study, we investigated the usability of two prominent schema formats, both of which are grounded in established benchmark datasets for schema-based dialogue systems. We refer to these formats as the Procedural and Declarative formats.

Initially, both formats were represented as raw JSON objects, which can be difficult and error-prone to edit. To address this issue, we developed two block-based programming interfaces[1], each corresponding to one of the schema formats, providing our participants with a more accessible authoring interface.

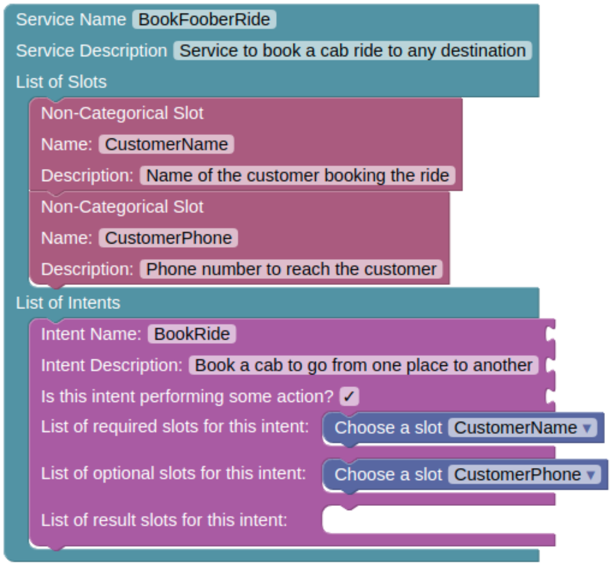
Procedural Schema. The Procedural schema, derived from the STAR dataset by Mosig et al. [16], represents a dialogue domain as a directed graph similar to a flowchart. It consists of nodes representing user utterances, system responses, or backend service calls. Nodes linked to user utterances or system responses are connected to example utterances. Figure 1 shows a section of the block-based interface for the Procedural schema.

[1] These block-based interfaces were created utilizing Blockly (https://developers.google.com/blockly).



Schema
User says: + Hello
System says: + Hi, how can I help?
User says: + I'd like to book a ride
System says: + May I have your name, please?
User says: + My name is [NAME]
System says: + OK, and where are you departing from?
User says: + I'm leaving from
+ [LOCATION]
Make API call to RideBook.Book()
Case RIDE_BOOKED
System says: + Found it!

Figure 1: A sample snippet of the Procedural schema, which is based on the STAR dataset from Mosig et al. [16].



Service Name BookFooberRide
Service Description Service to book a cab ride to any destination
List of Slots
Non-Categorical Slot
Name: CustomerName
Description: Name of the customer booking the ride
Non-Categorical Slot
Name: CustomerPhone
Description: Phone number to reach the customer
List of Intents
Intent Name: BookRide
Intent Description: Book a cab to go from one place to another
Is this intent performing some action? ✓
List of required slots for this intent: Choose a slot CustomerName ▾
List of optional slots for this intent: Choose a slot CustomerPhone ▾
List of result slots for this intent:

Figure 2: A sample snippet of the Declarative schema, which is based on the Schema-Guided Dialogue (SGD) dataset from Rastogi et al. [20].

Declarative Schema. The Declarative format, based on the Schema-Guided Dialogue (SGD) dataset by Rastogi et al. [20], aims to improve model generalization by linking intents and slots in the dialogue system's ontology to succinct natural language descriptions. Figure 2 presents an example of the block-based interface designed for the Declarative schema format.

2.4 Participants

Our study aimed to select participants familiar with dialogue system concepts, specifically intents and slots. We recruited 23 undergraduate and graduate students (18 male, 5 female) who completed an spoken dialogue systems course. The course covered concepts related to the linguistics of conversation, dialogue interface design, components of a standard dialogue systems, and fundamental machine learning and NLP concepts. As part of their final evaluation, the course required students to create their own dialogue agents throughout the semester and live-test it with their peers. The participants were recruited through a course discussion board announcement. Participants earned 2% course credit for their involvement in the study.

2.5 Instruments

Pre-Survey. At the outset of the study, participants completed a pre-survey in which they reported basic demographic information to self-assess their proficiency in three areas: (1) creating virtual assistants using platforms such as Google DialogFlow, (2) general programming abilities, and (3) understanding of machine learning concepts. Subsequently, participants watched two brief video tutorials that demonstrated how to use two custom block-based interfaces to define Declarative and Procedural schemas respectively. The block-based interfaces were designed by us to offer participants a user-friendly way of defining the schemas. We describe the interfaces in detail in Section 2.3.

Post-Survey. Upon completion of the developmental tasks (described below), participants filled out a post-survey that consisted of: 1) a questionnaire designed to appraise the usability of the block-based interfaces for schema definition, utilizing the System Usability Scale [2] and 2) a survey asking for participants' open-ended feedback on their experiences of creating schemas and crafting instruction-based prompts. The complete set of questions asked in the latter part of the survey is shown in Table 1.

1. Which schema did you choose for the first scenario of adding the tiers? (Procedural-based or Declarative-based)? Why did you pick that one over the other?
2. Which schema did you choose for the second scenario of adding the cancellation capability? (Procedural-based or Declarative-based)? Why did you pick that one over the other?
3. How would you compare writing the instructions to building a schema for the first scenario (adding a tier)?
4. How would you compare writing the instructions to building a schema for the second scenario (adding the cancellation capability)?

Table 1: The post-survey questionnaire included the above four questions to elicit open-ended feedback from participants about the authoring interfaces.

2.6 Data Analysis

We used an inductive coding process [8] to analyze the open-ended responses from the post-survey. First, two researchers conducted open coding on all of the transcripts. Then, the primary coder, in consultation with another researcher, iteratively derived three themes from participants' open-ended responses.

3 RESULTS

Out of 23 participant-defined block-based programs, 18 were successfully compiled into schemas, while 5 had minor issues like orphaned blocks or improper text formatting within the blocks. Based on the System Usability Scale results, we found that participants were generally satisfied with the block-based interface for both scenarios. There was no statistical difference between the reported SUS scores for the two scenarios ($p = 0.20$). Table 2 shows the details of the t-test performed.

To analyze the responses to the open-ended survey items, we first grouped the response-units based on the interface that it pertains to, and then identified individual themes under each of them (as described in Section 2.6). In total, we derived three themes from inductively coding the participants' responses: 1) Cognitive Scaffolding 2) Effort, and 3) Precision and Expressivity. We elaborate each of these in detail below.

Table 2: Result of a paired t-test between SUS scores for the two scenarios of 1) including a service tier (modifying an existing schema), and 2) adding a ride cancellation feature (creating one from scratch).

Scenario 1 Mean (SD)	Scenario 2 Mean (SD)	p-value
74.18 (11.67)	69.70 (17.99)	0.20

3.1 Cognitive Scaffolding

When developing a dialogue agent for a real-world task, the key decisions that need to be made include the types of information that needs to be collected from the end-user, the set of supported intents, and other factors such as phrasing of the utterances. A dialogue system authoring interface can offer the dialogue system designer various kinds of scaffolding to reduce the cognitive effort in translating their understanding of the dialogue into a working agent.

Instruction-based Prompts. Seven participants reported that a blank text-field fell short in helping them think through the specific pieces of information that needed to go into their instructions. Of particular note was the concern around the lack of scaffolding provided by a blank text-field in thinking through fine-grained details of the dialogue flow. For example, P18 reported, "*building a schema . . . is definitely a more comprehensive and effective way, since you will get lost while writing a long paragraph of text but still miss important information.*"

Procedural Schema. The majority of participants (13 out of 23) appreciated the resemblance of Procedural schemas with an actual conversation, stating that it allowed them to play out a conversation in their head. For example, according to P6, "*. . . there are multiple things which would be considered only when you go through a live conversation which the Procedural schema definitely helps with.*" In addition, P6 also found Procedural schema "*visually appealing.*"

Declarative Schema. Three participants (P2, P11, P10) attributed their aversion to the Declarative schema to the abstract nature of thinking about the dialogue flow in terms of intents and slots. However, two participants who did prefer the Declarative schemas (P12, P18) said that thinking about the dialogue flow in a declarative way freed them up from having to worry about the specific order in which the slots had to be requested as well as the natural language phrasing of the utterances. In this regard P12 said that the Declarative schema offered "*more flexibility regarding the way information is obtained.*"

3.2 Effort

This theme encompasses findings related to both the duration of time as well as the cognitive effort involved in authoring an agent.

Instruction-based Prompts. Perceptions about the effort involved in writing instruction-based prompts differed depending on which of the two scenarios was in question. Four participants (P1, P3, P15, P18) reported that writing instructions for Scenario 1 was quite straightforward since they only had to mention the three tiers. However, Scenario 2 was seen as significantly more complex and participants felt that unambiguously writing instructions putting significant thought into it. Even then, many felt it was "*long-winded*".

Procedural Schema. Although the amount of manual effort was not reported as an issue when it came to Scenario 1, five participants stated that defining a Procedural schema for Scenario 2 (in which they had to define a complete dialogue flow from the ground up) was tedious. For example, P3 responded by saying, "*Adding the cancellation capability via schema was fairly straightforward with the Procedural schema, but it was time consuming and a bit tedious.*"

Declarative Schema. Three participants reported the Declarative schema to be less labor-intensive than the Procedural schema. For example, P2 said that the Declarative schema was "*less cumbersome", while P18 said that it was "simpler and higher-level than the Procedural schema.*"

3.3 Precision and Expressivity

Having fine-grained control over minute details of the dialogue flow is an important factor in a dialogue authoring format. Under this theme, we report the participants' responses on the level of precision and expressivity that each authoring format provided them. Participants made no direct comments about the expressivity (or the lack thereof) of the Procedural schema.

Instruction-based Prompts. Six participants felt that natural language instructions inherently left room for ambiguity and misinterpretation, particularly in Scenario 2. This can be summed up well in the following response from P11: "*I think the cancellation process had more structure (must do A before B) whereas selecting the tier is far less structured of an interaction, making it easier to describe in words succinctly.*"

Declarative Schema. Two contrasting perspectives emerged concerning the expressivity of the Declarative schema. The first viewpoint (P2, P4, P10, P11) considered the inability to create complex dialogue flow patterns, such as loops and conditional branches, as a limitation on the schema's expressivity. On the other hand, the second perspective (P8, P9, P12, P18) argued that delegating control logic to the underlying model was a more sensible approach, as manually crafting dialogue flow could potentially lead to brittleness.

4 DESIGN IMPLICATIONS

These findings suggest a few general design implications for the development of authoring tools for defining task-oriented dialogue agents.

Combining Strengths of Different Interfaces. Our findings suggest that while instruction-based prompts often fall short in helping participants think through the specific pieces of information needed, the Procedural schema excelled at it. At the same time, schemas (particularly the Declarative schema) ran the risk of being seen as complex and *code-like* compared to writing instructions in English. For example, the Procedural schema could be used when authoring an agent for a brand new scenario from scratch, while the Declarative schema can be used to make edits to an existing agent.

Reducing Effort in Dialogue Authoring. According to our findings, participants' reported effort for writing instruction-based prompts and creating schemas varied depending on the complexity of the task. In order to create a more user-friendly authoring experience, it is important to design interfaces that minimize cognitive and manual effort, particularly for complex tasks. This could include features such as drag-and-drop components, reusable templates, or visual aids.

Model Transparency. People are known to adapt their language based on their audience's interpretative capability [7]. This is evident from the participants' mixed interpretations of the level of explicitness and specificity required while writing instructions. In the case of schemas, too, we observed similar trends. Therefore, the choice of authoring formats should not be considered in isolation, but rather, in tandem with the capabilities of the model interpreting it.

5 LIMITATIONS AND FUTURE WORK

This preliminary study has several limitations, which can inform future research directions. First, while we focused on instruction-based prompts, providing a handful of "in-context" examples is another widely used prompting technique in tandem with large language models [3, 9]. A user study comparing instruction-based and example-based prompts can shed light on the pros and cons of these two prompting methods. Second, participants in our study were asked to author the agent before encountering any real-world examples of dialogues. However, in the real-world, building dialogue systems is not a one-time activity, but an iterative process where the agent's training data and behavior is constantly modified even post-deployment as more conversation logs become available. Finally, our study was conducted before the public release of a powerful instruction-tuned model such as GPT-3.5, so schema and prompt creation were not connected to an actual model. It would be beneficial to perform a similar experiment using a working model, enabling a more thorough evaluation of schema and prompt authoring in practical contexts.

6 CONCLUSION

This study critically evaluated the user experience associated with three distinct dialogue system authoring interfaces: Instruction-based Prompts, Procedural Schema, and Declarative Schema. Participants reported varying levels of cognitive load, precision, and comprehensibility across these interfaces. Instruction-based prompts were seen as easily approachable yet lacking in detailed guidance, leading to potential omissions and ambiguities. Procedural schema resonated with the participants due to its conversational mirroring, but was viewed as labor-intensive for intricate scenarios. Declarative schemas, despite offering flexibility and higher-level simplicity, were perceived as abstract and occasionally insufficient for complex dialogue flows. These findings underline the need for a user-oriented approach in the development of dialogue authoring tools, amalgamating the strengths of each interface to achieve a balance between simplicity, flexibility, and depth. We anticipate that these insights will spark discourse in the conversational user interface and dialogue system research communities, emphasizing the importance of usability in designing developer tools and formulating benchmark datasets.

REFERENCES

[1] Dan Bohus and Alexander I. Rudnicky. 2009. The RavenClaw dialog management framework: Architecture and systems. *Computer Speech & Language* 23, 3 (July 2009), 332–361. https://doi.org/10.1016/j.csl.2008.10.001

[2] John Brooke. 1996. SUS - A quick and dirty usability scale. *Usability evaluation in industry* 189, 3 (1996), 8.

[3] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language Models are Few-Shot Learners. *arXiv:2005.14165 [cs]* (July 2020). http://arxiv.org/abs/2005.14165 arXiv: 2005.14165.

[4] Julia Cambre and Chinmay Kulkarni. 2020. Methods and Tools for Prototyping Voice Interfaces. In *Proceedings of the 2nd Conference on Conversational User Interfaces* (Bilbao, Spain) *(CUI '20)*. Association for Computing Machinery, New York, NY, USA, Article 43, 4 pages. https://doi.org/10.1145/3405755.3406148

[5] Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-Shot Transfer Learning with Synthesized Data for Multi-Domain Dialogue State Tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, Online, 122–132. https://doi.org/10.18653/v1/2020.acl-main.12

[6] Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-Guided Multi-Domain Dialogue State Tracking with Graph Attention Neural Networks. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (April 2020), 7521–7528. https://doi.org/10.1609/aaai.v34i05.6250

[7] Herbert H. Clark and Gregory L. Murphy. 1982. Audience Design in Meaning and Reference. In *Advances in Psychology*, Jean-François Le Ny and Walter Kintsch (Eds.). Language and Comprehension, Vol. 9. North-Holland, 287–299. https://doi.org/10.1016/S0166-4115(09)60059-5

[8] Jennifer Fereday and Eimear Muir-Cochrane. 2006. Demonstrating Rigor Using Thematic Analysis: A Hybrid Approach of Inductive and Deductive Coding and Theme Development. *International Journal of Qualitative Methods* 5, 1 (2006), 80–92. https://doi.org/10.1177/160940690600500107 arXiv:https://doi.org/10.1177/160940690600500107

[9] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361* (2020).

[10] Young-Bum Kim, Dongchan Kim, Anjishnu Kumar, and Ruhi Sarikaya. 2018. Efficient Large-Scale Neural Domain Classification with Personalized Attention. In *Proceedings of the 56th Annual Meeting of the Association for Computational*

Linguistics (Volume 1: Long Papers). Association for Computational Linguistics, Melbourne, Australia, 2214–2224. https://doi.org/10.18653/v1/P18-1206

[11] Miao Li, Haoqi Xiong, and Yunbo Cao. 2020. The SPPD System for Schema Guided Dialogue State Tracking Challenge. *arXiv:2006.09035 [cs]* (June 2020). http://arxiv.org/abs/2006.09035 arXiv: 2006.09035.

[12] Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue State-Tracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, Online, 5640–5648. https://doi.org/10.18653/v1/2021.naacl-main.448

[13] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *arXiv:2107.13586 [cs]* (July 2021). http://arxiv.org/abs/2107.13586 arXiv: 2107.13586.

[14] Shikib Mehri and Maxine Eskenazi. 2021. Schema-Guided Paradigm for Zero-Shot Dialog. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*. Association for Computational Linguistics, Singapore and Online, 499–508. https://aclanthology.org/2021.sigdial-1.52

[15] 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)*. Association for Computational Linguistics, Dublin, Ireland, 3470–3487. https://doi.org/10.18653/v1/2022.acl-long.244

[16] Johannes E. M. Mosig, Shikib Mehri, and Thomas Kober. 2020. STAR: A Schema-Guided Dialog Dataset for Transfer Learning. *arXiv:2010.11853 [cs]* (Oct. 2020). http://arxiv.org/abs/2010.11853 arXiv: 2010.11853.

[17] Vahid Noroozi, Yang Zhang, Evelina Bakhturina, and Tomasz Kornuta. 2020. A Fast and Robust BERT-based Dialogue State Tracker for Schema-Guided Dialogue Dataset. *arXiv:2008.12335 [cs, stat]* (Aug. 2020). http://arxiv.org/abs/2008.12335 arXiv: 2008.12335.

[18] Baolin Peng, Chenguang Zhu, Chunyuan Li, Xiujun Li, Jinchao Li, Michael Zeng, and Jianfeng Gao. 2020. Few-shot Natural Language Generation for Task-Oriented Dialog. In *Findings of the Association for Computational Linguistics: EMNLP 2020*. Association for Computational Linguistics, Online, 172–182. https://doi.org/10.18653/v1/2020.findings-emnlp.17

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *J. Mach. Learn. Res.* 21, 1, Article 140 (jan 2020), 67 pages.

[20] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards Scalable Multi-Domain Conversational Agents: The Schema-Guided Dialogue Dataset. *Proceedings of the AAAI Conference on Artificial Intelligence* 34, 05 (April 2020), 8689–8696. https://doi.org/10.1609/aaai.v34i05.6394 Number: 05.

[21] Bernardino Romera-Paredes and Philip Torr. 2015. An embarrassingly simple approach to zero-shot learning. In *International conference on machine learning*. PMLR, 2152–2161.

[22] Daniel Rough and Benjamin Cowan. 2020. Don't Believe The Hype! White Lies of Conversational User Interface Creation Tools. In *Proceedings of the 2nd Conference on Conversational User Interfaces* (Bilbao, Spain) *(CUI '20)*. Association for Computing Machinery, New York, NY, USA, Article 17, 3 pages. https://doi.org/10.1145/3405755.3406140

[23] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* 53, 3 (2020), 1–34.

A DESCRIPTIONS PROVIDED FOR THE SCENARIOS

Participants were directed to write natural language instructions for both scenarios using the text in Section A.1. The texts in Sections A.2 and A.3 were used to describe the dialogue scenarios to the participants.

A.1 Directions Given to the Participants

Your job is now to write instructions (similar to, say, how you would write an email to a human) to an agent directing it to ask customers which tier they want to book a ride in during every call. Make sure that your instructions can be clearly understood.

A.2 Scenario 1

Foober's management has recently introduced three tiers for rides ("XL", "Share", and "Regular") instead of having just a single tier. Your CEO wants your customer service agents to ask customers which tier they want to book a ride in during every call.

The way that your customer service agents work is by filling out a form with the customer's details (such as their name, phone number, pick-up and destination locations, etc.). The form was created by your company's developers. Now they have included a new field for the tier.

A.3 Scenario 2

Foober drivers are increasingly irate that their customers are no longer present at the location when they arrive. They feel that this is partly due to Foober not offering their customers an option to cancel the cab if needed. Foober's management has instructed you to allow callers to cancel the cab if they need to. However, to prevent riders from cancelling last minute, Foober has decided to charge a cancellation fee of $10.00 if the rider does not cancel within the first 5 minutes of booking.

The engineering backend team has already developed TWO forms for the agents to fill out when creating a cancellation request.

The first form allows the agent to enter the customer details and check whether they have a ride already booked and if so, it will return the Booking ID and whether it will cost them anything to cancel the ride.

The second form allows the agent to actually cancel the ride using the Booking ID.