LUCID: LLM-Generated Utterances for Complex and Interesting Dialogues

Joe Stacey[1*] **Jianpeng Cheng**[2] **John Torr**[2] **Tristan Guigue**[2]
Joris Driesen[2] **Alexandru Coca**[3*] **Mark Gaynor**[2] **Anders Johannsen**[2]
[1]Imperial College London [2]Apple [3]University of Cambridge
j.stacey20@imperial.ac.uk, ac2123@cam.ac.uk
{jianpeng.cheng,jtorr,tguigue}@apple.com
{joris_driesen,mgaynor,ajohannsen}@apple.com

Abstract

Spurred by recent advances in Large Language Models (LLMs), virtual assistants are poised to take a leap forward in terms of their dialogue capabilities. Yet a major bottleneck to achieving genuinely transformative task-oriented dialogue capabilities remains the scarcity of high quality data. Existing datasets, while impressive in scale, have limited domain coverage and contain few genuinely challenging conversational phenomena; those which are present are typically unlabelled, making it difficult to assess the strengths and weaknesses of models without time-consuming and costly human evaluation. Moreover, creating high quality dialogue data has until now required considerable human input, limiting both the scale of these datasets and the ability to rapidly bootstrap data for a new target domain. We aim to overcome these issues with LUCID, a modularised and highly automated LLM-driven data generation system that produces realistic, diverse and challenging dialogues. We use LUCID to generate a seed dataset of 4,277 conversations across 100 intents to demonstrate its capabilities, with a human review finding consistently high quality labels in the generated data[1].

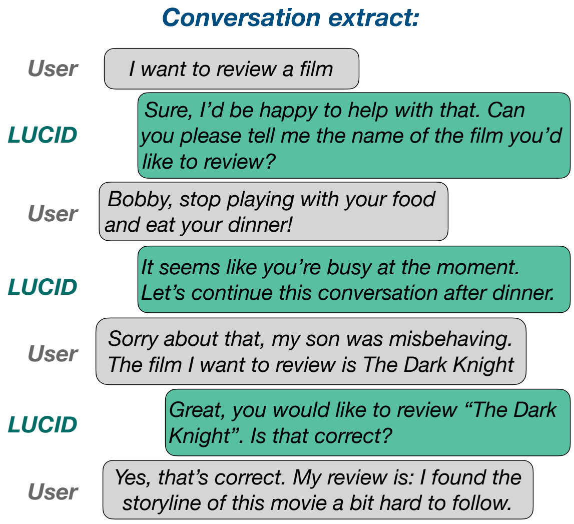

Figure 1: An extract of a LUCID conversation containing a challenging phenomenon. In this case, the second user response is most likely to be from an overheard conversation rather than providing the desired slot value.

1 Introduction

As AI virtual assistants become more sophisticated, there is an increasing need for dialogue datasets with more challenging conversational phenomena for both fine-tuning and evaluation. Existing datasets include multi-turn, multi-intent and multi-domain conversations (Rastogi et al., 2020; Budzianowski et al., 2018), in addition to multilingual datasets (Goel et al., 2023; FitzGerald et al., 2023; Hung et al., 2022; Li et al., 2021). However, in each case, the number of intents covered is relatively small. Moreover, the conversational phenomena included in these datasets are often limited in scope[2]. Additionally, current machine-to-machine data collection methods still involve varying degrees of human involvement, with humans paraphrasing machine generated templates into natural language, and/or manually crafting plausible sequences of intents as dialogue outlines (Shah et al., 2018; Rastogi et al., 2020).

To overcome these issues, we introduce LUCID, **L**LM-generated **U**tterances for **C**omplex and **I**nteresting **D**ialogues. LUCID is composed of a pipeline of modularised LLM calls that create realistic, accurate and complex data, allowing the data generation process to scale to more intents, slots and challenging conversational phenomena. LUCID involves automated intent generation, with a mock back-end[3] created for each intent. This mock

[*]Work undertaken while author was an intern at Apple

[1]Please see http://github.com/apple/ml-lucid-datagen for the data and project code

[2]The PRESTO dataset (Goel et al., 2023) does explicitly label specific conversational phenomena, but to the best of our knowledge it is unique in this respect

[3]The mock back-end converts intents into Python classes, which are then instantiated as objects

Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 4: Student Research Workshop), pages 56–74
June 18, 2024

back-end then interacts with LLM-based user and system agents, generating dialogues without the need for human annotation.

Ensuring data quality is a central challenge for a machine-to-machine generation process. We address this issue by breaking down the generation process into a pipeline of multiple, simpler LLM calls, thereby compartmentalising the data generation task into manageable steps that an LLM can consistently perform accurately. In addition, we use multiple LLM-based validators which discard conversations that *might* contain an issue. Our *if in doubt, discard* philosophy ensures a high quality standard for the data being created.

We release the data generation code to enable large scale data generation across different intents and domains, with the option of adding additional, complex conversational flows. We also provide training data, validation data, and two tests sets, a test set for seen intents, and an additional test set for unseen intents, allowing for convenient out-of-distribution evaluation.

2 Related Work

2.1 Task Oriented Dialogue Datasets

The most popular approach for creating dialogue datasets involves human-to-human interactions, with user annotators interacting with Wizard of Oz (WoZ) annotators (Budzianowski et al., 2018; El Asri et al., 2017; Zhu et al., 2020; Eric et al., 2017; Wen et al., 2017). While using human annotators can create diverse, large scale datasets, this is done at a considerable cost, with expert annotators required for accurate dialogue annotations. User annotators follow a generated conversation plan (Budzianowski et al., 2018; El Asri et al., 2017; Zhu et al., 2020), guiding their interactions with the WoZ agent. We find that even in a purely machine-to-machine setup, generating conversation plans for each dialogue remains an effective way to ensure conversational variety.

2.2 Automated Data Collection Methods

To reduce the workload of annotators, dataset collection is becoming increasingly automated. A popular approach is to generate conversation outlines, which are then converted into natural language by annotators (Shah et al., 2018; Rastogi et al., 2020; Lin et al., 2021) or using natural language templates (Bordes et al., 2017). As these conversation outlines are simulated based on hard-coded rules, this can limit the diversity of the user behaviour.

Human involvement in automated data generation includes ensuring the quality of the dataset, paraphrasing user and agent responses (Shah et al., 2018; Rastogi et al., 2020), providing semantic annotations (Goel et al., 2023; Budzianowski et al., 2018), outlining the sequences of user intents (Rastogi et al., 2020), and identifying out of scope or incoherent examples (Goel et al., 2023). We show that, with recent advances in language modelling (OpenAI, 2023; Ouyang et al., 2022), by reducing the data generation task into manageable steps, and using our *if in doubt, discard* validation methodology, it is now possible to achieve the same quality in an almost entirely machine-to-machine generation process. Parallel work by Liu et al. (2024) also introduces an automated method for generating task-oriented dialogue data. While we focus on the accurate labelling of challenging and diverse conversations, Liu et al. (2024) consider a variety of user personas with different styles of system responses.

See Appendix A for related work generating data with LLMs for other tasks.

3 Method

LUCID decomposes the data generation process into **14** individual LLM calls, described here as *stages*, creating manageable steps that LLMs can perform accurately. Alongside our *if in doubt, discard* validation, reducing the complexity of each LLM call helps to ensure the quality of our generated data. The data generation process consists of four main components (see Figure 2): the generation of intents (*stages 1-2*), a conversational planner (*stages 3-8*), turn-by-turn generation of conversations (*stages 9-12*), and our validation process (*stages 13 and 14*). The turn-by-turn data generation involves a User LLM agent interacting with a System LLM agent, which in turn interacts with a mock back-end created for each intent.

3.1 Intent Generation

Schema for each intent are generated by an LLM, using a short human-authored natural language description of the intent (*stage 1*)[4]. Using these descriptions, LUCID calls an LLM to generate the intent and slot names, as well as the data type of each slot and whether it is mandatory or optional. In total, 100 intents are generated across 13 do-

[4]Our code also allows intent schema to be created manually

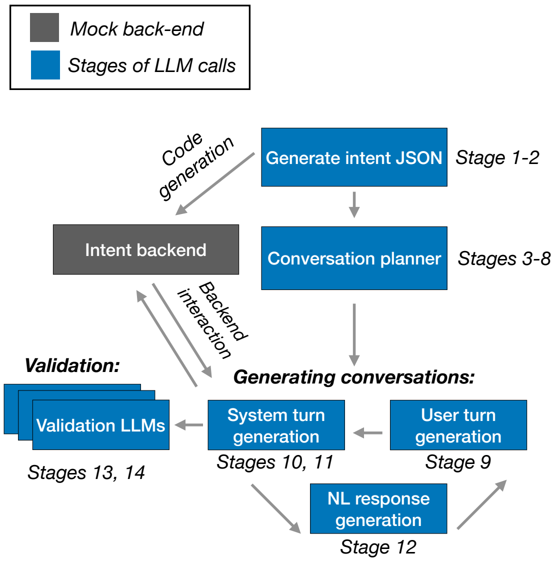

Figure 2: The stages in the LUCID data generation, generating intents (*stages 1-2*), planning conversations (*stages 3-8*), generating the conversations (*stages 9-12*) and validating the system predictions (*stages 13-14*).

mains (see Appendix H for a detailed breakdown of how transactional and query intents are generated).

The next stage (*stage 2*), involves generating plausible values for each slot. We use these slot values as a starting point for our conversation planner, helping to encourage varied conversations.

3.2 Conversation Planner

The conversation planner provides instructions that guide a user LLM agent down certain types of conversational flows. The planner specifies: 1) the sequence of intents, 2) the slot values for each intent, and 3) any complex conversational phenomena that must be included, specifying when and how these phenomena should be incorporated. This creates a plan that the user must adhere to, reducing the complexity of the data generation task, while also ensuring variety in the generated conversations. This plan is communicated to the User LLM at each turn through a series of conversation rules.

The planner also decides the sequence of intents that will be included in a conversation (*stage 3*). Depending on the primary intent used to start the conversation, the planner then decides which intents are likely to follow this intent, with the aim of creating both varied and realistic conversations. See Appendix B for further details about the planner.

3.2.1 Generating Slot Values

The slot values chosen by the planner substantially impact the conversations, and as a result, we have multiple, separate stages for generating the slot values (*stages 4, 5, 6 & 7*). This process involves updating the slot values to make sure these are realistic and coherent (*stage 4*), generating a reason why the user wants to perform any subsequent intents (*stage 5*), and generating slot values for the subsequent intents based on this justification (*stage 6*). Finally, an LLM updates the slot values across every intent in the conversation to ensure they are consistent and realistic when considered collectively (*stage 7*).

We additionally use an LLM to generate realistic entities to be returned after any queries (*stage 8*).

3.3 Generating Conversations

Conversations are generated turn-by-turn with a User LLM interacting with a System LLM, which in turn interacts (via Pythonic function calls and variable assignments) with a mock back-end for each intent. A Response LLM then communicates natural language responses back to the user. The user behaviour is governed by the conversational rules created by our planner, shaping the outcome of the conversations.

Conversations start with an utterance from the User LLM (*stage 9*), which is then interpreted and labelled by the System LLM (*stage 10*). Initially, no string slot values are predicted. These values are predicted in an additional stage (*stage 11*), where an LLM is instructed, where possible, to choose the string values from spans of the user utterance (avoiding hallucinations). The predicted semantic labels then interact with a mock back-end for each intent. The mock back-end then informs the System LLM about any missing slots or whether confirmation is required for the intent. Finally, the Response LLM responds back to the user (*stage 12*), requesting any additional slots or asking the user for confirmation.

3.4 LLM-based Validation

We implement an LLM-based validation process to ensure reliable and consistent labelling in the conversations, using our principle of *if in doubt, discard*. First, based on the observation that the system LLM is more uncertain about incorrect predictions, we repeat the system predictions twice (using a temperature value of 0.7), and abort the conversation if the three predictions are not identical (*stage*

13). Additional validation is then performed by another LLM (*stage 14*) which also labels the user requests, except this time with access to the conversation rules that the user is following. These predictions must also exactly match the original System LLM predictions, otherwise the conversation is aborted. The validation in stages 13 and 14 is performed before the string slot values are predicted, avoiding conversations being aborted when these slot values have slightly different phrasing. Further validation is described in Appendix E.

3.5 Introducing Additional Conversational Phenomena

To make interesting, diverse and challenging conversations, we introduce a wide range of conversational phenomena which are labelled automatically at a turn level (see Figure 4). These phenomena include *sarcastic* or *irrelevant* replies, or cases where the user is *overheard* in another conversation. LUCID also contains examples where a user corrects themselves, either within a turn (*in-turn correction*) or in a later turn (*correction*). Alternatively, a user may cancel an intent (*cancellation*) or delay confirming the intent until a future turn (*delay confirmation*). Our conversational phenomena also include cases where the virtual assistant requests a value for one slot, but the user responds about a different slot (*respond different slot*). Finally we also include ASR early end errors (*ASR-early end*), where the LLM produces truncated slot values where the user text is abruptly cut off. See Table 4 for the full distribution of these phenomena.

3.6 Annotation Scheme and our Mock Back-end

We apply a concise labelling system to track the states for each intent in a conversation. This labelling system follows a Pythonic syntax, with function calls used to initialise intents and entities when these are first mentioned, and attribute assignments used for any subsequent slot filling operations (see Figure 3). This schema-based, concise form of semantic labelling is highly convenient, and avoids the need for state tracking for each individual turn.

Our labelling involves four types of turns: 1) User turns; 2) System turns, labelling user intentions; 3) Signal turns, returned after our mock back-end processes the system command; and 4) Response turns, which are natural language responses back to the user. Further details on our labelling schema is provided in Appendix F.

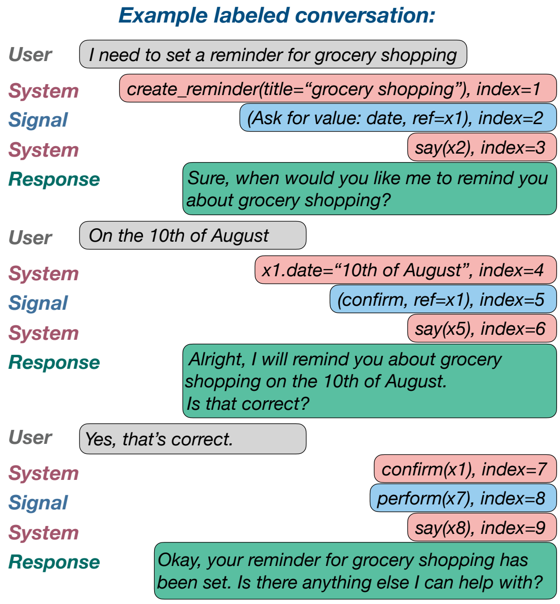

Figure 3: A (simplified) example labelled conversation. Each dialogue contains user, system, signal and response turns.

4 Analysis

4.1 Diversity of Slots and Intents

The generated LUCID data contains more intents and slots than existing task-oriented dialogue datasets (Table 1). Specifically, the dataset contains 100 intents, across 13 domains, with 501 different slots. While the SGD dataset contains more domains than LUCID, these domains are narrower in scope. For example, SGD includes separate domains for buses, taxis, flights and trains, while LUCID has a single transportation domain incorporating intents for each of these areas. The larger number of slots and intents in LUCID illustrates our ability to create diverse and challenging data using LUCID, despite generating a smaller dataset compared to SGD and MultiWOZ (see Table 5).

As the LUCID dataset was generated primarily to showcase the capabilities of the LUCID data generation system, others are free to use the LUCID system to generate much larger and even more complex datasets. This extensibility is what most clearly distinguishes LUCID from these other data generation efforts.

	# Domains	# Intents	Ints per Dom	# Slots	# Labelled Unhappy Paths.
PRESTO	-	34	-	303†	6
PRESTO-no dup	-	34	-	276†	6
SGD	**20**	88	4.4	365	0
SGD-no dup	**20**	46	2.3	240	0
MultiWOZ	7	11	1.6	35	0
LUCID	13	**100**	**7.7**	**501**	**9**

Table 1: Summary statistics of our dataset, displaying the number of domains and intents present, the number of intents per domain, the number of slots present, and the number of explicitly labelled conversational phenomena (unhappy paths). For PRESTO, we consider the 303 slots in English intents (†). Unlike Table 5, this table considers all splits in the dataset. Appendix I describes how duplicate slots and intents are removed for SGD and PRESTO.

4.2 Conversational Phenomena

LUCID contains a greater number of labelled conversational phenomena than existing dialogue datasets (Table 1). The recently released PRESTO dataset also contains turn-level annotated phenomena, labelling six types of unhappy paths (Goel et al., 2023). These unhappy paths include in-turn corrections, correcting actions, correcting slot values, code-mixing, disfluencies and cancellations. While half of these phenomena relate to corrections, this is the case for only two of our labelled phenomena, correcting slot values either in-turn or across turns. Instead, we focus on distinguishing between relevant, sensible user replies from cases where a virtual assistant should ask for clarification (rather than using the initial response to populate slot values).

4.3 Qualitative and Quantitative Analysis

We perform a qualitative analysis on the generated dataset (conducted by one of the paper authors) to thoroughly investigate the dataset quality and identify any issues. This included a manual review of 200 conversations in our dev set, which only highlighted two labelling errors (impacting only 1% of conversations). In comparison, Eric et al. (2020) identify annotation errors in 40% of turns in MultiWoz 2.0, demonstrating the relative quality of the LUCID system labels.

The two labelling errors identified in this review involved: 1) A user mentioning there will be no spoilers in a review, where LUCID correctly assigns the spoiler alert slot value as False, but additionally includes the text 'no spoilers in my review' as part of the review itself. 2) LUCID not recognising an in-turn correction by the user, mistakenly including all of the user text (including the correction itself) as part of a slot value. See Appendix C for further details. We additionally share a qualitative analysis of our data in Appendix D, highlighting specific areas where our method could be further improved. We include this analysis to further raise the bar for future LLM data generation efforts.

	Intent acc.	JGA
Test (seen):		
T5-Small	94.7	57.1
T5-Base	97.9	67.5
T5-Large	**98.7**	**69.0**
Test-OOD (unseen):		
T5-Small	95.3	22.0
T5-Base	97.6	42.2
T5-Large	**98.8**	**45.7**

Table 2: Results of our baseline model (with retrieval). Full results and evaluation metric descriptions are provided in Appendix G.

5 Baseline Results

We train T5 (Raffel et al., 2020) and Flan-T5 (Chung et al., 2022) baseline models on LUCID, evaluating on both our in-distribution and out-of-distribution test sets. When retrieving intent schemas, a Sentence-BERT (Reimers and Gurevych, 2019) model is used to encode the tool name and the last user utterance, choosing the tool with the highest cosine similarity.

As expected, the joint goal accuracy is considerably higher when evaluating on the seen test set compared to the unseen test set, with accuracy scores of 67.5% and 42.2% respectively for a T5-

base model (see Table 6). We also isolate the impact of the retrieval model, comparing three scenarios: 1) using our tool retrieval, 2) using an oracle tool retrieval, and 3) including no tools in the prompt. We find that the tool retrieval is not a major weakness of our baseline model (see Table 7). Finally, we evaluate our Flan-T5-base model on each different conversational phenomena (see Section 3.5), highlighting sarcasm, ASR-early end examples, and answering about a different slot as the most challenging phenomena. Full experimentation details and results can be found in Appendix G.

6 Conclusion

We introduce LUCID, a pipeline of LLM calls which is designed to create high quality and linguistically sophisticated dialogue data. LUCID involves an extensive validation process, including three validator LLMs that discard conversations where there is any disagreement. To demonstrate the quality of the data produced, we generate a seed dataset of 4,277 dialogues, consisting of 92,699 turns, with a wide variety of challenging conversational phenomena. The generated system labels in LUCID prove to be highly accurate, with only 1% of conversations containing a labelling error. We make our code available to facilitate larger scale, high quality data generation.

Limitations

The main limitation of our approach is the cost of using a closed-source LLM. This prevented us from generating a larger number of dialogues or performing more ablation studies to isolate the improvements from specific stages. This cost was driven by our *if in doubt, discard* approach to validation, which prioritised the accuracy and quality of the data produced, at the expense of the computational time and cost involved. While there are also substantial costs associated with high quality manual annotation, in this work we aim to show that an LLM-driven approach to generating high quality data is possible and feasible. We also aim to produce a seed dataset of the highest quality which can be used by practitioners on an on-going basis.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2017. Learning end-to-end goal-oriented dialog. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ - a large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Y. Zhao, Yanping Huang, Andrew M. Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *CoRR*, abs/2210.11416.

Layla El Asri, Hannes Schulz, Shikhar Sharma, Jeremie Zumer, Justin Harris, Emery Fine, Rahul Mehrotra, and Kaheer Suleman. 2017. Frames: a corpus for adding memory to goal-oriented dialogue systems. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 207–219, Saarbrücken, Germany. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Mihail Eric, Lakshmi Krishnan, François Charette, and Christopher D. Manning. 2017. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pages 37–49. Association for Computational Linguistics.

Jack FitzGerald, Christopher Hench, Charith Peris, Scott Mackie, Kay Rottmann, Ana Sanchez, Aaron Nash, Liam Urbach, Vishesh Kakarala, Richa Singh, Swetha Ranganath, Laurie Crist, Misha Britan, Wouter Leeuwis, Gokhan Tur, and Prem Natarajan. 2023. MASSIVE: A 1M-example multilingual natural language understanding dataset with 51 typologically-diverse languages. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4277–4302, Toronto, Canada. Association for Computational Linguistics.

Rahul Goel, Waleed Ammar, Aditya Gupta, Siddharth Vashishtha, Motoki Sano, Faiz Surani, Max Chang, HyunJeong Choe, David Greene, Kyle He, Rattima Nitisaroj, Anna Trukhina, Shachi Paul, Pararth Shah, Rushin Shah, and Zhou Yu. 2023. PRESTO: A multilingual dataset for parsing realistic task-oriented dialogs. *CoRR*, abs/2303.08954.

Xuanli He, Islam Nassar, Jamie Kiros, Gholamreza Haffari, and Mohammad Norouzi. 2022. Generate, annotate, and learn: NLP with synthetic text. *Trans. Assoc. Comput. Linguistics*, 10:826–842.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2023. Unnatural instructions: Tuning language models with (almost) no human labor. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, pages 14409–14428. Association for Computational Linguistics.

Chia-Chien Hung, Anne Lauscher, Ivan Vulic, Simone Paolo Ponzetto, and Goran Glavas. 2022. Multi2woz: A robust multilingual dataset and conversational pretraining for task-oriented dialog. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL 2022, Seattle, WA, United States, July 10-15, 2022*, pages 3687–3703. Association for Computational Linguistics.

Haoran Li, Abhinav Arora, Shuohui Chen, Anchit Gupta, Sonal Gupta, and Yashar Mehdad. 2021. MTOP: A comprehensive multilingual task-oriented semantic parsing benchmark. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2950–2962, Online. Association for Computational Linguistics.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, Peng Xu, Feijun Jiang, Yuxiang Hu, Chen Shi, and Pascale Fung. 2021. Bitod: A bilingual multi-domain dataset for task-oriented dialogue modeling. In *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*.

Alisa Liu, Swabha Swayamdipta, Noah A. Smith, and Yejin Choi. 2022. WANLI: Worker and AI collaboration for natural language inference dataset creation. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 6826–6847, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Yinhong Liu, Yimai Fang, David Vandyke, and Nigel Collier. 2024. Toad: Task-oriented automatic dialogs with diverse response styles.

Yu Meng, Jiaxin Huang, Yu Zhang, and Jiawei Han. 2022. Generating training data with language models: Towards zero-shot language understanding. In *NeurIPS*.

OpenAI. 2023. Gpt-4 technical report.

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

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.*, 21:140:1–140:67.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8689–8696. AAAI Press.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021. Generating datasets with pretrained language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Pararth Shah, Dilek Hakkani-Tür, Bing Liu, and Gokhan Tür. 2018. Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and on-line reinforcement learning. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers)*, pages 41–51, New Orleans - Louisiana. Association for Computational Linguistics.

Joe Stacey and Marek Rei. 2023. Improving robustness in knowledge distillation using domain-targeted data augmentation. *arXiv preprint arXiv:2305.13067*.

Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gašić, Lina M. Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Peter West, Chandra Bhagavatula, Jack Hessel, Jena Hwang, Liwei Jiang, Ronan Le Bras, Ximing Lu, Sean Welleck, and Yejin Choi. 2022. Symbolic knowledge distillation: from general language models to commonsense models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4602–4625, Seattle, United States. Association for Computational Linguistics.

Sarah Wiegreffe, Jack Hessel, Swabha Swayamdipta, Mark Riedl, and Yejin Choi. 2022. Reframing human-AI collaboration for generating free-text explanations. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 632–658, Seattle, United States. Association for Computational Linguistics.

Qingyun Wu, Gagan Bansal, Jieyu Zhang, Yiran Wu, Beibin Li, Erkang Zhu, Li Jiang, Xiaoyun Zhang, Shaokun Zhang, Jiale Liu, Ahmed Hassan Awadallah, Ryen W White, Doug Burger, and Chi Wang. 2023. Autogen: Enabling next-gen llm applications via multi-agent conversation.

Yuxiang Wu, Matt Gardner, Pontus Stenetorp, and Pradeep Dasigi. 2022. Generating data to mitigate spurious correlations in natural language inference datasets. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 2660–2676. Association for Computational Linguistics.

Jiacheng Ye, Jiahui Gao, Qintong Li, Hang Xu, Jiangtao Feng, Zhiyong Wu, Tao Yu, and Lingpeng Kong. 2022. Zerogen: Efficient zero-shot learning via dataset generation. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, EMNLP 2022, Abu Dhabi, United Arab Emirates, December 7-11, 2022*, pages 11653–11669. Association for Computational Linguistics.

Qi Zhu, Kaili Huang, Zheng Zhang, Xiaoyan Zhu, and Minlie Huang. 2020. CrossWOZ: A large-scale Chinese cross-domain task-oriented dialogue dataset. *Transactions of the Association for Computational Linguistics*, 8:281–295.

A Related work - Data Generation with LLMs

Wu et al. (2023) recently introduce a framework allowing the interaction of multiple, different LLMs, based on the idea that LLMs can solve highly challenging tasks if these tasks are broken into smaller steps. While Wu et al. (2023) are successful in generating dialogues for a group chat scenario, this does not require the intent and slot labelling needed for task-oriented dialogue. For other NLP tasks, to avoid labelling errors or poor quality data, generating data with LLMs can involve human annotators reviewing the generated utterances (Liu et al., 2022; Wiegreffe et al., 2022), or using the generated data as unlabelled data to be used with knowledge distillation (Meng et al., 2022; He et al., 2022; Stacey and Rei, 2023). Labelled data generation has been successful for other tasks without human input (Honovich et al., 2023; Ye et al., 2022; Schick and Schütze, 2021; West et al., 2022; Wu et al., 2022), however noise may be an issue for a large proportion of the data (Honovich et al., 2023; Schick and Schütze, 2021).

B Conversation Planner Details

To ensure variety in the generated conversations, the planner makes extensive use of sampling, choosing how many intents should be provided in the conversation, which optional slots should be discussed, the conversational phenomena (both happy and unhappy paths) that should be included, and which slots and intents any conversational phenomena should be applied to.

Sampling is also involved in choosing the path of intents that are included in the conversation. The planner is provided with a sample of intents across all domains, before an LLM generates a plausible sequence of intents from this sample (*stage 3*).

We also provide more detail below to describe exactly how the conversation planner creates slot values for the conversation, involving a range of different stages:

The slot values for the first intent are initially sampled based on the plausible slot values generated for each intent (see *stage 2*), preventing LLMs generating repetitive conversations. An LLM is then asked to make the slots for each intent more realistic and coherent (*stage 4*). This prevents contradictory slot values, for example when a hotel check-out date is before the check-in date. This stage also prevents highly unlikely slot values being over represented, while introducing further variety into the slot values being provided.

For all other intents in the conversation (after the first intent), an LLM generates a plausible reason why the user would want to complete this intent given what has already occurred in the conversation (*stage 5*). Likely slot values are then generated (*stage 6*) based on this context. *Stage 4* is then repeated, increasingly the likelihood that the slot values selected are realistic and coherent for the intent. An LLM is then asked to update the slot values across every intent in the conversation so that these intents are related and consistent (*stage 7*), encouraging more natural conversations that align more closely with human to virtual assistant interactions.

C Quantitative Analysis Findings

Table 3 provides the full results from our quantitative analysis of 200 dev examples. This analysis identified two labelling issues within the 200 conversations that were reviewed (conducted by one of the paper authors).

In the first instance, the user says "I want to review the film the godfather. I give it a 9 out of 10 and my review is an absolute classic! Great performances and storytelling.. No spoilers in my review." The system interprets this as giving the film name (the godfather), the rating (9), the spoiler alert (False), and the review text. However, the system also predicts the text 'No spoilers in my review' as part of the review text, when this may not be the case.

In the second instance, the user performs an in-turn correction, changing the value of the additional notes slot for a new hair appointment intent. However, the system predicts this correction as part of the note itself, giving the slot value: 'I need a quick haircut, actually make that I'm getting ready for a family reunion photoshoot and want a new haircut.'

Table 3 also measures some of the issues reported in the qualitative analysis, including: 1) how many times the natural language generated response (NLG) indicates an intent was performed before confirmation was given, 2) when the NLG does not follow the system predictions, 3) unrealistic slot values, 4) unrealistic combination of slots mentioned for an intent, 5) when the first choice string span was not selected by LUCID, and 6) when the user or NLG does not fully understand the purpose of the intent. While previous work

Figure 4: Examples for eight of the nine challenging conversational phenomena included in the LUCID dataset. We also included 'cancellation' examples which are similar to 'delay confirmation', resulting in the system not confirming a given intent.

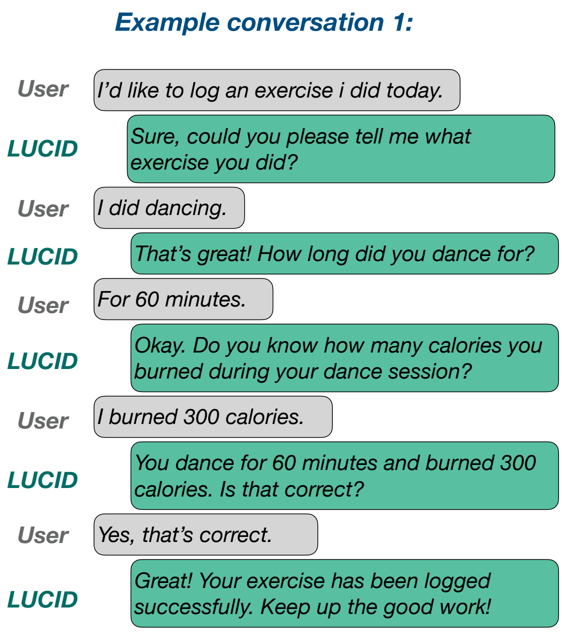

Figure 5: An example conversation from LUCID (Example #1). As described in Appendix J, we show the first three LUCID conversations to provide an unbiased sample of our generated data.

does not report similar metrics, we publish these figures with the aim of raising the bar for future data generation efforts.

In the case of point 5) considering when the first choice string span was not selected, we find that only 8.5% of conversations have a string slot value that does not match the reviewer's first choice. However, these differences are subjective and subtle, with over half of the cases concerning whether to include 'the' before a date (e.g. 'the 5th of March' vs '5th of March').

D Qualitative Analysis

We perform a qualitative analysis on our dev set, understanding potential limitations of our data, and suggesting ways these could be mitigated for future generation efforts.

Finding 1) The natural language responses from the model do not always reflect the system labels that have been predicted. We observed that a correct system label can be accompanied by a natural language response that does not reflect the correct system prediction. We noticed this for sarcastic responses, where only the system label and not the natural language response reflected the user's sarcasm. We choose to manually review the turns labelled as sarcastic, filtering out 4 dialogues. However, our quantitative analysis on the

Issue name	Prevalence
NLG indicates intent performed before confirmation	4.5%
NLG does not follow system prediction	1%
Unrealistic slot values	6%
Unrealistic combination of slots mentioned for an intent	3.5%
First choice string span not selected	8.5%
User or NLG does not fully understand purpose of the intent	6.5%

Table 3: Quantitative analysis from 200 dialogues in our development set. We report these six metrics in addition to the system label accuracy figure of 1% provided in Section 4.3.

dev set highlights this as an issue beyond sarcastic turns, with natural language responses not faithfully following the system label predictions in 1% of conversations.

As a related issue, the natural language response can also suggest that an intent has been performed before confirmation is given by the user. Informed by this finding, we filter out conversations when an intent was not performed, unless there was a cancellation signal provided by the user (removing 79 conversations). After this filtering, our quantitative analysis finds that 4.5% of conversations include responses that suggest an intent has been performed before confirmation is given. However, in each case there was no impact on the conversation beyond the phrasing of the natural language response. As LUCID prioritises validating the system labels, we do not implement validation checks on the natural language response. Introducing additional validation for the natural language responses is likely to also improve their quality.

Finding 2) The planner's choice of slots and their corresponding values can sometimes be unrealistic. While a strength of LUCID is the realistic and varied slot values used in conversations, this is not always the case. We also notice that the choice of slots included in a conversation is not always realistic. For example, you would not usually give the start time, end time in addition to specifying the duration of a swimming lesson. Our quantitative analysis identifies that 6% of conversations contain at least one unrealistic slot value, while 3.5% of conversations include an unrealistic combination of slots. The unrealistic slot combinations demonstrate a limitation to our sampling approach, where we randomly sample which optional slots should be included in each conversation. This issue could be overcome with an additional LLM stage responsible for deciding if the slot combination provided is realistic or not.

Finding 3) The user does not always understand the purpose of the intent. For example the user may ask 'can you find my favorites from yesterday?', when it is not clear if the user understands what a 'favorite' is. This is a consequence of our conversation plans telling users which intent should be performed, without also providing a description. The quantitative analysis finds that in 6.5% of conversations, either the user or the natural language response does not fully understand an intent, suggesting that descriptions should be included for future data generation work.

Finding 4) The system command labelling is consistently high quality, with few labelling mistakes. We quantify this finding with our quantitative analysis of 200 conversations in the development set, which finds only 1% of examples when the system label is not correct. More detail on the two system labelling errors identified are provided in Appendix C.

E Validation and Post-Processing

We introduce additional validation, ensuring turns that include our challenging conversational phenomena are correctly predicted by our LLMs. When the conversation rules instruct a user to introduce a specific conversational phenomenon for a certain slot value, the user is instructed to also provide a signal (in the form of a special token) to show that this unhappy path is being performed. We then use this signal for validation purposes, ensuring that the following system command matches the expected response for this phenomenon. However, we do not provide these special tokens to the system which interprets and labels the user request; these would not be available to a real virtual assistant and we find that including them during data generation can result in unrealistic target labels (e.g. if a user's 'irrelevant' response accidentally constitutes a plausible slot value).

A range of post-processing rules are also introduced after our qualitative analysis. We filter conversations where a slot is corrected during the conversation, but where there is no correctional signal provided by the user (as described above, a signal is provided by the user for each complex conversational phenomena which is used purely for validation purposes). This filtering process removes instances where a slot was first mentioned by the user without giving a value, with the system incorrectly assigning a slot value from this turn (filtering 123 conversations). We perform additional filtering to remove empty string slot values (removing 27 conversations), and any instances where the system turn predicts a hint, as hints should only occur in Signal turns. There were 172 occurrences when a hint was predicted by the system, although in almost all cases these conversations were already filtered by another post-processing filter.

In total, 56% of conversations pass all of our validation checks. To avoid wasting valuable conversational data, we salvage the prefix of an aborted conversation up to the point where the validation error was identified[5]. In these cases, we truncate the conversation, sampling LLM generated natural language responses that justify interrupting the conversation.

F Annotation Schema Detail

An important part of our annotation schema is the order of the turns, and how system turns trigger the natural language responses. This section provides more detail on these points.

System turns always follow user turns. In most cases, the first system turn is followed by a signal turn, except when the system decides to immediately call a response with 'say()', for example if the user response is irrelevant. Signal turns are then followed by a system turn, which triggers the natural language response turn. The system turns therefore decide when to pass information to the mock back-end, and when to trigger the natural language response. We use the system turns as the targets in this dataset.

LUCID automatically creates the mock back-end for each intent using the schema generated in steps 1 and 2. This involves generating a Python class to represent the intent in question, which is then instantiated as an object and interacts with the system commands to indicate when mandatory slots have not yet been provided, or when confirmation is still required before the intent can be performed. The outputs of the mock back-end are represented by the signal turns described above.

G Baseline Results

We train six different baseline models on the LUCID training data (T5-small, T5-base, T5-large, Flan-T5-small, Flan-T5-base and Flan-T5-large models). Each model is evaluated on the test set for seen intents and the OOD test set for unseen intents (see Table 6). We additionally experiment with training our Flan-T5-base baseline on varying amounts of training data (see Table 8). Details about about the choice of hyper-parameters can be found in Appendix K.

To measure performance on our generated LUCID data, we consider five performance metrics: joint goal accuracy, intent accuracy, fuzzy slot accuracy, exact match accuracy (between user turns), and exact match accuracy for an entire dialogue.

For joint goal accuracy, we consider a fuzzy matching score for string slot values. As many system turns involve a *say* command, a joint goal accuracy figure is only calculated for turns where a value is predicted (or contained in the system labels). Additionally, we consider the goal state of all intents in the conversation, rather than considering different states for different intents or domains separately.

We also use fuzzy matching for our slot accuracy measure, which is a joint accuracy measure across all the slot values provided in a single system turn (when any slot values are predicted, or when they are included in the labels). We additionally introduce exact match metrics that consider the accuracy of all system commands, not just those that refer to intent and slot values (for example, including 'say' commands). We introduce two exact match metrics - *exact match (turn)* considers whether all predicted system commands between two user turns exactly match their labels, while *exact match (conversation)* considers whether every predicted system command in a conversation matches with the system labels.

The exact match between user turns is measured for a Flan-T5-base model for each conversational

[5] To avoid overly short conversations, we do this only if at least one intent has been performed already or at least 10 turns have occurred

phenomena, both in the seen and unseen (OOD) test sets (see Table 4). We use this metric because some conversational phenomena involve an assignment, which is then followed by a 'say' command. As predicting a 'say' command following an assignment is not challenging for the model, we find that using the exact match between user turns metric provides the fairest comparison. The most challenging phenomena for our Flan-T5-base model are ASR-early end, sarcasm and answering about another slot phenomena (see Table 4), although predictions for ASR-early end are substantially worse for the seen intents. A number of phenomena appear to be less challenging than examples with no unhappy paths (see 'None' in Table 4), particularly for the OOD test set. This occurs because many phenomena do not involve any slot assignment, which becomes more challenging in the OOD test set.

For the tool retrieval, as gold system labels for previous turns are seen in the prompt conversation history, we retrieve all tools that have been mentioned in an oracle history up to that point.

Our baseline models are fine-tuned using the following prompt: "You are a smart AI assistant who is responsible for writing system commands to describe what the user has asked for. Your job is to write the next system command based on the latest user turn, considering the conversation so far." When using tool retrieval, the following text is added "Information about the following tools may help:", before providing the retrieved intent alongside intents from the conversation history. Finally, a single in-context example is provided to the model (see our code for further details).

H Intent Generation

In total, 54 intent descriptions were provided, with a single intent removed for data quality reasons. The removed intent involved the user asking the virtual assistant to start watching a television channel, giving specific start and end times for when they want to start watching the channel. As this is an unrealistic scenario (a user would want to start watching a television channel straight away), the intent was removed.

A transactional intent was created for each of the remaining 53 descriptions. LUCID then creates a query intent corresponding to each transactional intent. The query intent returns entities that would be created using the corresponding transactional intent.

Some query intents were merged together (this happens when the corresponding transactional intents had the same entity names - one of the intent properties generated by the LLM). As a result, there are 6 fewer query intents than transactional intents, resulting in a total of 100 intents. Each of the following pairs of transactional intents returned the same entity names, and so their corresponding query intents were combined: *add_tv_program_to_favorites* and *add_artist_to_favorites* (both of which return 'favorites' entities), *set_timer* and *set_alarm* (both of which return 'alarm' entities), *book_nails_appointment* and *book_spa_appointment* (both of which return 'appointments'), *order_supermarket_shop* and *order_takeaway* (both of which return 'orders'), *add_song_to_favorites* and *play_song* (both of which return 'songs'), and *review_film* and *review_restaurant* (both of which return 'reviews').

Each of the 100 intents used for our data generation are listed in Table 9 and Table 10. These tables list each transactional intent, along with its corresponding query intent. We also provide the human authored descriptions for each intent that were initially provided to LUCID.

I Slot Duplication within PRESTO and SGD

SGD report 214 slots in their training data (Rastogi et al., 2020), corresponding to 365 slots across all dataset splits (see Table 1). This counts slots with exactly the same names in the same domains within different services, which we consider to be duplicated slots (although the allowed slot values may change in each case). As a result, we provide a more direct comparison to SGD without this slot and intent duplication across services (see 'SGD-no dup' in Table 1).

For PRESTO, we consider the 303 slots present in the English split of the dataset (Goel et al., 2023). However, as the semantic annotations in PRESTO are represented in parse-trees, slots are counted multiple times if their paths are different. We find the number of English slots in PRESTO reduces to 276 without this duplication.

When considering the total number of slots in PRESTO, the same slot can be counted multiple times depending on its position in the labelled parse

Conv. phenomena	Total #	Train #	Dev #	Test #	Test-OOD #	Test Acc.	Test-OOD Acc.
Cancellation	12	5	2	3	2	100	100
ASR-early end	58	41	7	7	3	43	100
Sarcasm	63	46	3	8	6	75	67
Delay confirmation	76	53	7	4	12	100	100
Answer about another slot	113	75	13	11	14	64	43
Irrelevant answer	163	116	18	15	14	93	93
Overheard answer	203	153	17	23	10	100	100
In-turn correction	215	145	27	25	18	80	72
Correction	250	166	28	31	25	90	81
None	3,200	2,279	307	252	362	82	56
Conv. w/ 1+ unhappy path	1,077	754	108	119	96	-	-
Total conversations	4,277	3,033	415	371	458	-	-
% Conversations unhappy	25.2%	24.9%	26.0%	32.1%	21.0%	-	-

Table 4: Total number of each conversational phenomenon within each split of our dataset. While there are few conversations for cancellation, this behaves similarly to the 'delay confirmation' phenomenon. We also show the exact match (Turn) metric for each conversational phenomena from a T5-Flan-base baseline model (details of the metric are provided in section Appendix G).

trees. For example, the *Send_digital_object* intent includes *bcc* and *cc* slots. Both of these slots can be a *Personal_contact* entity, which contains a *person* slot. In this case, the *person* slot within *Personal_contact* would be counted twice if it was contained within either the *bcc* or *cc* slots. Removing this slot duplication reduces the number of English slots in PRESTO from 303 to 276.

Note, we consider the v.2.2 of MultiWOZ for our comparison, as this version explicitly states the intents present in the dataset.

J Example Dialogues

In addition to the examples provided in Figure 1 and Figure 3, we provide three additional examples of the LUCID generated conversations. To provide an unbiased sample of our conversations, we show the first three dialogues in the dataset (see Figure 5, Figure 6 and Figure 7). We also show examples of each of the unhappy paths used in our dataset (see Figure 4).

K Modelling Setup, Parameters, Computing Setup

For each baseline experiment, we train for 3 epochs. This was selected as a hyper-parameter based on the development set loss after training for 1-10 epochs for our Flan-T5-base baseline. We use a learning rate of 5×10^{-5}, with a linear learning schedule, a batch size of 2 with 8 gradient accumulation steps. For experiments with reduced training data, we train for more epochs as the training data is increased (with epochs inversely proportional to the size of the training data, allowing for a fair comparison). For each LLM call, we use a temperature value of 0.7.

Our baseline models have the following number of parameters: T5-small and Flan-T5-small (60 million parameters), T5-base and Flan-T5-base (220 million parameters), T5-large and Flan-T5-large (770 million parameters). We train our models with V100 GPUs, with our combined baseline experiments training for approximately 80 GPU hours.

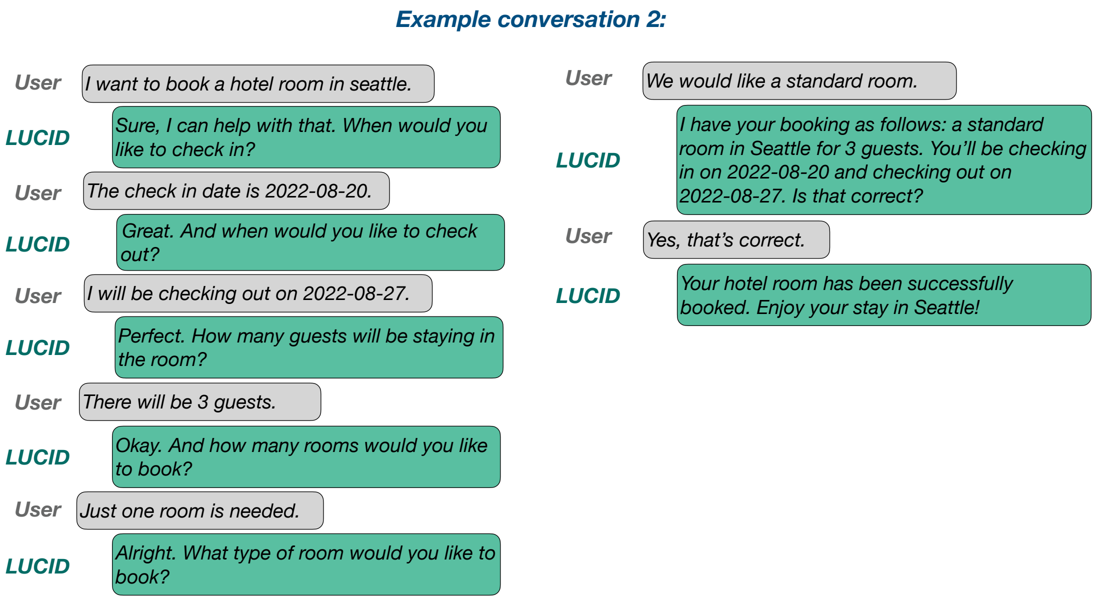

Figure 6: An example conversation from LUCID (Example #2). As described in Appendix J, we show the first three examples to provide an unbiased sample from our generated data.

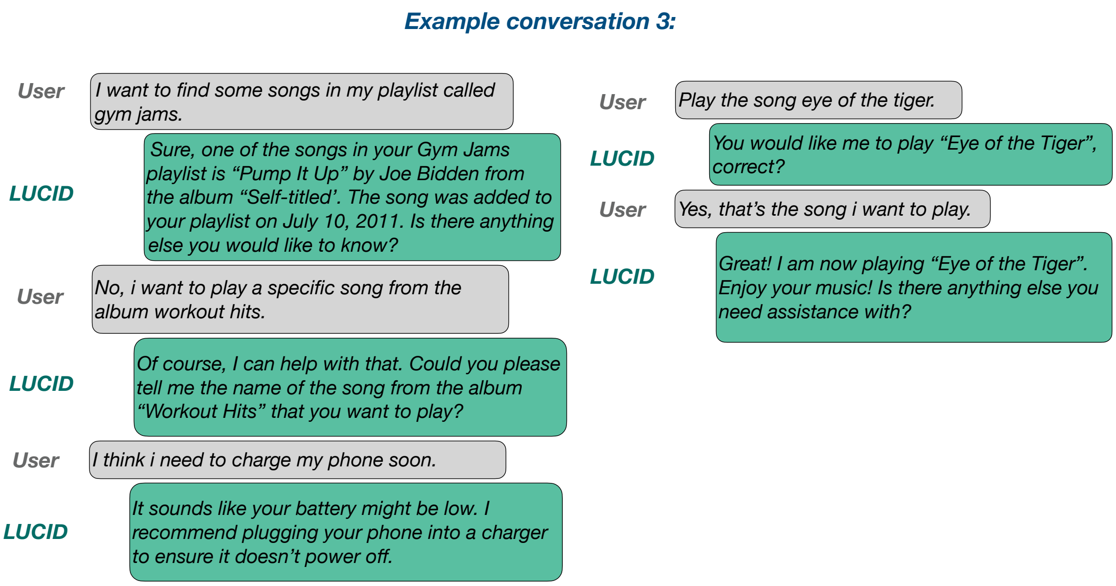

Figure 7: An example conversation from LUCID (Example #3). As described in Appendix J, we show the first three examples to provide an unbiased sample from our generated data.

	DSTC2	WOZ2.0	FRAMES	M2M	MultiWOZ	SGD	LUCID
# domains	1	1	3	2	7	**16**	12
# dialogues	1,612	600	1,369	1,500	8,438	**16,142**	3,033
# turns	23,254	4,472	19,986	14,796	113,556	**329,964**	65,217
Turns per dialogue	14.49	7.45	14.60	9.86	13.46	20.44	**21.50**
No. of slots	8	4	61	13	24	214	**432**
No. of slot values	212	99	3,871	138	4,510	14,139	4,701
Values per turn	0.009	0.02	**0.2**	0.009	0.04	0.04	0.07

Table 5: Reported statistics for LUCID, and related datasets for task-oriented dialogue. All statistics refer to the training split of the datasets, except for Frames which reports figures for all splits. Compared to previous dialogue datasets, LUCID has considerably more slots, and more turns per dialogue. There are also more possible slot values per turn than either MultiWoZ or SGD. The number of turns in LUCID refer to User, System, Signal and Response turns.

	Intent acc.	Joint goal acc.	Slot acc.	Match (turn)	Match (conv.)
Test (seen):					
T5-Small	94.7	57.1	69.8	74.5	30.5
T5-Base	97.9	67.5	76.6	82.1	44.2
Flan-T5-Base	97.9	69.7	77.6	82.6	45.8
T5-Large	**98.7**	69.0	77.9	83.2	46.9
Flan-T5-Large	98.5	**69.7**	**78.5**	**83.5**	**47.4**
Test-OOD (unseen):					
T5-Small	95.3	22.0	38.0	46.2	6.3
T5-Base	97.6	42.2	61.4	58.5	10.3
Flan-T5-Base	97.6	41.3	61.2	57.0	7.6
T5-Large	**98.8**	45.7	64.1	**60.2**	**11.4**
Flan-T5-Large	98.6	**53.2**	**66.6**	59.9	10.3

Table 6: Results of our baseline model trained for 3 epochs, using a SentenceBERT retrieval model. Each evaluation metric is described in more detail in Appendix G. Results are from a single seed in each case.

	Intent acc.	Joint goal acc.	Slot acc.	Match (turn)	Match (conv.)
Test (seen):					
No tools	97.4	66.1	75.9	81.6	43.7
w/ retrieval	97.9	**69.7**	77.6	82.6	**45.8**
Oracle tools	**99.1**	69.3	**78.1**	**83.1**	**45.8**
Test-OOD (unseen):					
No tools	87.1	32.8	55.7	53.2	**8.3**
w/ retrieval	97.6	**41.3**	61.2	57.0	7.6
Oracle tools	**99.4**	40.8	**61.3**	**57.4**	6.6

Table 7: Results of a T5-Flan-base model with our tool retrieval, using oracle tools, and with no tools provided in the prompt. Each evaluation metric is described in more detail in Appendix G. Results are from a single seed in each case.

# Training ex.	Intent acc.	Joint goal acc.	Slot acc.	Match (turn)	Match (conv.)
Test (seen):					
125	88.8	29.3	49.1	57.1	10.2
250	91.0	37.5	57.7	65.2	15.6
500	91.9	51.2	64.9	70.6	22.6
1k	94.3	58.9	70.5	75.2	29.1
2k	96.4	62.6	73.8	78.5	37.2
4k	96.7	65.0	75.0	80.3	40.4
8k	97.3	66.8	76.2	81.1	42.3
16k	97.6	66.4	76.5	81.9	43.9
Full (24,786)	97.9	69.7	77.6	82.6	45.8
Test-OOD (unseen):					
125	92.7	16.7	32.3	34.9	4.1
250	93.5	20.5	43.9	44.6	4.8
500	95.5	26.3	49.7	48.5	6.1
1k	96.7	34.3	56.4	54.2	9.6
2k	96.9	32.0	56.5	53.9	5.9
4k	97.0	37.0	59.5	57.0	6.8
8k	97.2	35.5	58.6	56.5	8.5
16k	97.4	32.7	57.2	55.9	7.0
Full (24,786)	97.6	41.3	61.2	57.0	7.6

Table 8: Results of a T5-Flan-base model trained with varying amounts of training data (count of the system turns provided). Each evaluation metric is described in more detail in Appendix G.

Transactional intents (1-30)	Corresponding query	Intent description
add_artist_to_favorites	find_favorites	Add artist to favourites
add_event	find_events	Add event
add_payment_card	find_payment_cards	Add payment card
add_restaurant_to_favorites	find_favorite_restaurants	Add restaurant to favourites
add_song_to_favorites	find_songs	Add song to favourites
add_to_favourites	find_favourite_pages	Add a page to favourites
add_tv_program_to_favorites	find_favorites	Add a TV program to favourites
add_user	find_users	Add user with access to calendar
block_sender	find_blocked_senders	Block sender
book_bus_ticket	find_bus_tickets	Book a bus ticket
book_city_tour	find_city_tours	Book a city tour
book_cruise	find_cruises	Book cruise
book_flight	find_flights	Book a flight
book_guide	find_guides	Book a guide
book_hair_appointment	find_hair_appointments	Book hair appointment
book_hotel_room	find_hotel_rooms	Book a hotel room
book_massage	find_massages	Book a massage
book_nails_appointment	find_appointments	Book appointment to do nails
book_pedicure	find_pedicures	Book a pedicure
book_spa_appointment	find_appointments	Book a spa appointment
book_swimming_lesson	find_lessons	Book swimming lesson
book_taxi	find_taxis	Book a taxi
book_train_journey	find_train_journeys	Book a train journey
book_triathlon	find_triathlons	Book triathlon
buy_film_tickets	find_film_tickets	Buy film tickets
create_direct_debit	find_direct_debits	Create direct debit
create_playlist	find_playlists	Create playlist
create_reminder	find_reminders	Create a reminder
create_workout_regime	find_workouts	Create workout regime
log_exercise	find_exercises	Log exercise

Table 9: Each transactional intent (1-30), alongside its respective query intent, and the description provided to LUCID that was used to generate the intent.

Transactional intents (31+)	Corresponding query	Intent description
make_song_recommendation	find_recommendations	Make song recommendation
open_web_page	find_web_pages	Open an internet page in a web browser
order_coffee	find_coffee_orders	Order coffee
order_supermarket_shop	find_orders	Order supermarket shop
order_takeaway	find_orders	Order takeaway
pay_bill	find_bills	Pay bill
play_audiobook	find_audiobooks	Play audiobook
play_film	find_films	Play film on streaming service
play_podcast_episode	find_podcast_episodes	Play a podcast episode
play_song	find_songs	Play a song
rent_accommodation	find_accommodations	Rent accommodation
rent_car	find_cars	Rent a car
reserve_table	find_reservations	Reserve a table
review_film	find_reviews	Review film
review_restaurant	find_reviews	Review a restaurant
send_email	find_emails	Send an email
send_invoice	find_invoices	Send invoice
send_message	find_messages	Send a message
set_alarm	find_alarms	Set an alarm
set_timer	find_alarms	Set a timer
set_volume	find_volume	Set the volume
transfer_money	find_transactions	Transfer money
write_note	find_notes	Write a note

Table 10: Each transactional intent (31+), alongside its respective query intent, and the description provided to LUCID that was used to generate the intent.