MonoTODia: Translating Monologue Requests to Task-Oriented Dialogues

Sebastian Steindl
Ostbayerische Technische Hochschule Amberg-Weiden
Germany
s.steindl@oth-aw.de

Ulrich Schäfer
Ostbayerische Technische Hochschule Amberg-Weiden
Germany
u.schaefer@oth-aw.de

Bernd Ludwig
University Regensburg
Germany
bernd.ludwig@ur.de

Abstract

Data scarcity is one of the main problems when it comes to real-world applications of transformer-based models. This is especially evident for task-oriented dialogue (TOD) systems, which require specialized datasets, that are usually not readily available. This can hinder companies from adding TOD systems to their services. This study therefore investigates a novel approach to sourcing annotated dialogues from existing German monologue material. Focusing on a real-world example, we investigate whether these monologues can be transformed into dialogue formats suitable for training TOD systems. We show the approach with the concrete example of a company specializing in travel bookings via e-mail. We fine-tune state-of-the-art Large Language Models for the task of rewriting e-mails as dialogues and annotating them. To ensure the quality and validity of the generated data, we employ crowd workers to evaluate the dialogues across multiple criteria and to provide gold-standard annotations for the test dataset. We further evaluate the usefulness of the dialogues for training TOD systems. Our evaluation shows that the dialogues and annotations are of high quality and can serve as a valuable starting point for training TOD systems. Finally, we make the annotated dataset publicly available to foster future research[1].

1 Introduction

The rise of Large Language Models (LLMs) has inspired many new fields of research and applications. One of the factors enabling their success is their capability to follow natural language prompts (Zhang et al., 2023), increasing and simplifying control over the model's output.

In general, chatbots can be roughly categorized into Task-Oriented Dialogue (TOD) systems and

[1] https://github.com/sebastian-steindl/MonoTODia

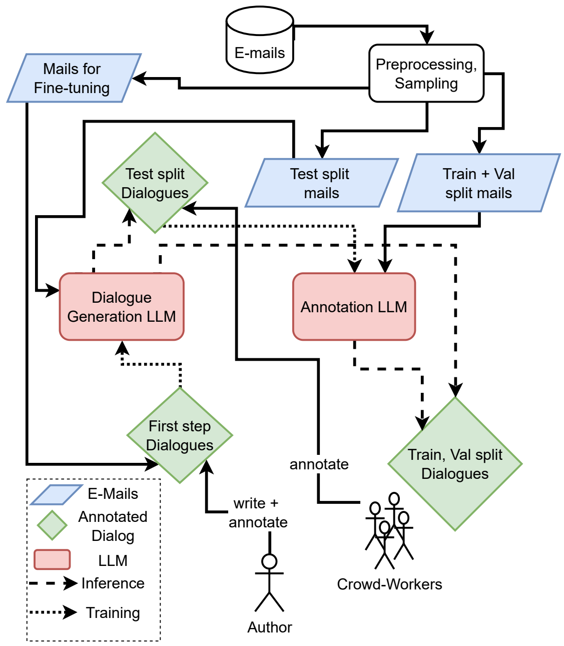

Figure 1: The MonoTODia approach. Blue marks e-mails, green annotated dialogues, and red LLMs. Dashed arrows mark inference, dotted arrows training.

Open-Domain Dialogue systems (Ni et al., 2023). TOD systems can be seen as a natural language interface to one (or multiple) external services, helping the users to achieve a certain task. These external services can often be treated as a database or an endpoint that is being queried. The request will be constructed in predefined slots that are being filled by the TOD system during the conversation. Everyday examples include actions like booking a restaurant or a train ticket. Furthermore, multiple domains can be combined within one dialogue, enabling the user to, e.g., book a complete vacation, including flights, hotel, and restaurants, within one conversation. For real-world productive use, the requests to the services will usually need to be made on live data, e.g., to get current prices and availabilities. The TOD system has to complete

Proceedings of the 2025 Conference of the Nations of the Americas Chapter of the Association for Computational Linguistics: Human Language Technologies (Industry Track), pages 390–403
April 30, 2025

three subtasks to fill slots and build requests to the external services: understanding the user (natural language understanding, NLU), deciding on how to react (policy planning, PP), and finally creating a response (natural language generation, NLG) (He et al., 2022). Compared to open-domain dialogues, TODs are usually multi-turn but short, constrained to certain domains, and highly structured (Deriu et al., 2021). While TOD systems were traditionally rule-based, current approaches use deep learning and transformers (Su et al., 2022; Bang et al., 2023; Zhao et al., 2022), achieving better results but requiring large amounts of training data.

This work thus studies whether the current advances in LLMs can make training a TOD system more accessible by translating existing monologue data into annotated, task-oriented dialogues. We fine-tune a state-of-the-art LLM to automatically translate the monologues into dialogues. In a second step, they are annotated with a LLM. The method is demonstrated on real-world e-mail requests. To assess the quality of the resulting dialogues and annotations, we perform human evaluation and investigate the usefulness of the data for training of downstream TOD systems. Our results indicate that style translation with LLMs could be a viable approach to cold start and low-resource problems for TOD systems. We publish the resulting dataset with gold-standard annotations for the test split. An example of an e-mail and the resulting dialogue is shown in Fig. 2. Further examples of dialogues generated with MonoTODia are shown in Figures 6 and 7 in Appendix G.

2 Problem statement

The need for training data is aggravated by the special requirements for the data in TOD systems, making data collection tedious, expensive and thus a fundamental bottleneck for the development of TOD systems (Axman et al., 2023; Kulkarni et al., 2024; Li et al., 2022). In collaboration with a German enterprise, we thus investigate an approach to tackle this problem: translating existing non-dialogue data to multi-turn TODs. We showcase this on e-mails, which can be seen as monologue requests in this scenario. This would drastically reduce the data collection and labeling effort while staying close to real-world, domain-specific data and tackle the cold-start problem of dialogue systems. For the company, such a system would greatly improve their service portfolio. We treat this question on an exemplar dataset derived from a German SME. The higher-level goal of this company is to digitalize and automate travel bookings. They collaborate with travel agencies, where they receive travel requests by e-mail and respond with a list of recommendations. These e-mails contain diverse, often unstructured pieces of information in various amounts and levels of detail, increasing the complexity of translation immensely.

A dialogue system is well-suited for booking scenarios since it allows for filling the needed slots and offers the possibility to, *bidirectionally*, ask for additional information, make proposals, and change previous slots. This interactivity mimics the interaction between a user and a respective human counterpart much more closely than an e-mail. Moreover, such a system would speed up processes because the response time in synchronous communication channels (chats) would be generally shorter than for asynchronous commonuication (e-mails).

The goal of the intended TOD system would be to assess the user's needs and wants. The final, legally binding confirmation of the booking would happen through a second communication channel.

Training a TOD system on e-mails directly is not possible since, e.g., the format and style don't fit, they lack the chatbot speaker role, and they are not annotated. The translation from e-mails to dialogues is complex and infeasible with traditional algorithms for multiple reasons. Firstly, one has to be able to identify all domain-specific relevant information within the e-mail, which is a NLU task. Then, one has to generate user and system utterances, which entails all the conundrums of NLG. Moreover, this NLG will in many cases need to include new, contextually relevant information that was not given in the e-mail. For example, if the e-mail only contains the destination, the chatbot would have to ask for, e.g., the travel period. Therefore, some information will need to be invented, i.e., *hallucinated*.

The recent advances in LLMs could offer an elegant solution to all of these tasks, with guided hallucinations even being desired to some extent. To enable this project, we defined a specific ontology (cf. Tab. 4 in the Appendix A) will be the basis for the LLM prompts and dialogue annotation. To the best of our knowledge, there is no current method that could be used for the problem that we tackle.

Hotel: Bahia Del Sol
Desired hotel: Bahia Del Sol

Email address: example@e-mail.com
Name: <anon_name>
Telephone number: +49 +49 12312312312
Departure airports: DTM, DUS, CGN, PAD
Travel period: October 1st, 2021 - October 8th, 2021
Travel duration: 1 week
Participants: 2 adults
Meals: All inclusive
Asking price: Up to 750

Request: A double room all inclusive with a balcony if you have a sea view, preferably a direct flight to Turkey or Mallorca

User: "I would like to book a hotel with all-inclusive." // inform(trip_type=hotel), inform(hotel_board=all-inclusive)
Bot: "Sure, that's no problem. Do you have any preferences for the hotel?" // act_require_more(hotel)
User: "Yes, it should have a direct flight to Turkey or Mallorca." // inform(flight_type=direct), inform(flight_departure_airport=[Turkey, Mallorca])
Bot: "Okay, do you want to stay in a double room?" // request(hotel_room_type)
User: "Yes, please." // inform(hotel_room_type=double)
Bot: "Do you have any preferences for the room number?" // act_require_more(hotel=room_number)
User: "No, that's okay." // inform(hotel_room_number=none)
Bot: "I recommend the Bahia Del Sol hotel, which is located directly at the beach in the town of Los Cristianos. It has 4 stars and is all inclusive. Would you like to book it?" // inform(hotel_name=Bahia Del Sol), inform(hotel_stars=4), inform(hotel_area=Los Cristianos), inform(hotel_board=all-inclusive), act_booking(book=?)
User: "Can you please give me the address of the hotel?" // request(hotel_address)
Bot: "Sure, the Bahia Del Sol hotel is located at Av. De Las Américas, s/n, 38650 Playa de las Américas, Las Palmas, Spain. Would you like to book it?" // inform(hotel_address=Av. De Las Américas, s/n, 38650 Playa de las Américas, Las Palmas, Spain), act_booking(book=?)
User: "Yes, please!" // act_booking(book=yes)
Bot: "Great, can you please provide me with your e-mail address, so I can send you the booking information?" // request(user_e-mail)
User: "info@example.com" // inform(user_e-mail=info@example.com)
Bot: "I have sent you all necessary information. Is there anything else I can do for you?" // act_information_sent(yes), act_require_more(general=?)
User: "No, thank you. That's all." // inform(availability=no), act_require_more(general=no)
Bot: "You're welcome, bye!" // act_general(bye)

Figure 2: An example e-mail from the corpus after pre-processing on the left and the resulting annotated dialogue after applying the MonoTODia approach on the right.

2.1 Existing Data and Pre-processing

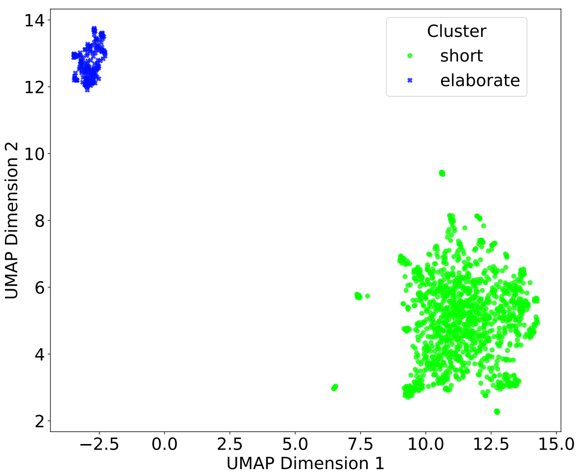

Figure 3: The clustering of the e-mails used for the train split. We first convert the e-mails with TF-IDF and then encode them with UMAP to build the clusters. It is clear that the short e-mails are the majority.

The existing data comes from an uncurated database dump of e-mail requests with highly heterogeneous styles. They range from minimal one-line e-mails, e.g., "Namibia individual trip", to elaborate, prose-like free texts and e-mails that give detailed information in an enumeration style. We can roughly cluster the e-mails into either short or elaborate, as is shown with in Fig. 3.

Since the data is raw, it includes a significant amount of noise. For example, out-of-office notifications, empty e-mails, test messages, and even apparent scam attempts. We apply a rigorous rule-based filtering to exclude noise. This affected roughly 10% of the full dataset. Moreover, we anonymized the data to remove personal client information. The e-mails are nearly exclusively in German. However, our preliminary experiments showed that the used LLM has poor performance on German text. We therefore applied one further step of pre-processing, translating the e-mails from German to English with the Google Translate API[2]. Finally, we construct the train, validation and test datasets by randomly sampling 1500, 150, and 200 e-mails, respectively, ensuring each e-mail is part of only one split. In summary, our data preparation consists of filtering, anonymization, translation and sampling for the data splits.

2.2 Impact on Real-World Business Problems

The travel-booking domain has a high potential for automation. Whereas online booking is nowadays established as an alternative to travel agencies, these services mostly rely on the user filling out static forms.

The cooperating company, adigi GmbH[3], is working towards interactive, natural language travel-booking, offering cloud-based B2B solutions. Compared to manual request processing, this leads to increased speed and reduced cost. Currently, its client base consists predominantly of travel agencies, who act as intermediaries relaying the end-customers' requests via e-mail. Extending the service portfolio by integrating a TOD system could thus drastically increase the number of clients by opening an additional direct sales channel to the end-customer, promoting business growth and competitiveness.

[2] https://github.com/ssut/py-googletrans
[3] https://www.adigi.ai

3 Background and Related Work

We will now describe related work for TOD systems, data augmentation and data style translation.

TOD systems and datasets. TOD systems were traditionally implemented by solving each subtask separately (Young et al., 2013). With the publication of large datasets, the field has moved towards deep learning-based systems such as Lin et al. (2020); Peng et al. (2021a); He et al. (2022). Benchmark datasets include, e.g., MultiWOZ (Budzianowski et al., 2018), KVRET (Eric et al., 2017) and SGD (Rastogi et al., 2020).

Data augmentation for dialogues. Data augmentation describes the sourcing of synthetic data by applying certain transformations to existing data in order to increase the amount of training data and the model's generalization ability (Shorten et al., 2021). Approaches include backtranslation (Kulhánek et al., 2021), incorporation of external datasets (Xu et al., 2021), simulating dialogues based on schemata (Peng et al., 2021b), graphs (Gritta et al., 2021), framing it as a text infilling task (Axman et al., 2023) or using specially trained generator models (Steindl et al., 2023). Nowadays, this line of research has also turned to LLMs. These methods include, for example, paraphrasing templates, using seed data or adding miscommunications to the dialogues (Li et al., 2022; Kulkarni et al., 2024; Chen et al., 2023; Mehri et al., 2022; Steindl et al., 2025). Recently, *model collapse* (Shumailov et al., 2024) has been discussed, where a model's performance degrades with every iteration of it being trained on model-generated data. One way to counteract this is by combining real and synthetic data (Gerstgrasser et al., 2024), which our method does by utilizing the human-written e-mails.

Data style translation. Translating a text from one "style" to another can be interpreted as a special case of NLG and controlled generation. First, we see summaries, especially abstractive summaries (Gupta and Gupta, 2019), as one form of such translation. Furthermore, data-to-text approaches (Jagfeld et al., 2018; Sharma et al., 2023; Wang et al., 2021) are relevant applications of this paradigm. Automatic news writing is another application (Diakopoulos, 2019), as is the creation of a dialogue based on a short story (Miyazaki, 2023). Further, the HRMultiWOZ (Xu et al., 2024) dataset is based on schemata that get turned into templates and are paraphrased by an LLM.

4 Method

Our approach uses instruction-tuned LLMs to generate dialogues based on monologue e-mails and subsequently annotate them. The LLMs undergo fine-tuning to solve these tasks. Crowd workers provide gold-standard labels for the test dataset.

The following sections provide a detailed breakdown of the two phases and finally explain the dataset sourcing.

4.1 Dialogue Generation and Annotation

We separate the tasks of dialogue generation and annotation into two distinct inference phases, where the model does not see the original e-mail when generating annotations. This prevents information leakage, that could not be reliably stopped with prompt engineering. When addressing both tasks in a single inference step, the annotation was too informed in many cases. That is, an annotation contained information that could not have been known at this point in the conversation and is only known from the e-mail.

We argue that this task separation delivers better results due to two reasons. Firstly, it leads to shorter and less complex prompts and task descriptions. Secondly, if both tasks are done in unison, the model has already attended to the complete information from the e-mail (to generate the dialogue), when annotating the first utterance, provoking information leakage. Consequently, we create the annotation for every utterance independently of later utterances. Based on preliminary experiments between various models, we decided to use an instruction-tuned open-source model from the LLaMA 3.1 (Dubey et al., 2024) family. Using an open-source model locally acts as an additional security mechanism, avoiding any risk of uploading client information to an external model provider. We use the instruction-tuned model with 8 billion parameters.

To improve the performance of the model, we fine-tuned it utilizing the LoRA (Hu et al., 2022) method for the two tasks separately, resulting in f_g for the dialogue generation and f_a for the annotation. The details for the fine-tuning are described in the Appendix B. For this purpose, we manually created and annotated 20 dialogues $D_{ft} = (x_{ft}, y_{ft})$ for e-mails that are not part of any dataset split. This number of dialogues was chosen to allow for some variation of e-mails and dialogues, including different slots and flows, without requiring too much manual labour, since the motivation of our approach is to keep this as low as possible.

The prompts for each task include an initial description, the task-specific rules, and examples. The examples enable in-context learning, which is known to improve performance (Brown et al., 2020). To increase output diversity for the dialogue generation, we created three variations of the prompt with different dialogue types examples.

For the annotation step, we provide the model with general rules for the annotation and all possible slots. The annotation is separated from the utterance in a comment-like style, starting with "//" and followed by annotations in the form "type(slot=value)". This is the result of preliminary experiments, where this format proved to be more successful and consistent than, e.g., JSON. Furthermore, it is easy to parse. However, other formatting styles are feasible and success might also depend on the specific LLM used. The full prompts are shown in Appendix E.

4.2 Dataset Sourcing

After pre-processing, sampling, and fine-tuning the dialogue generation LLM f_g, we generate the dialogues from the e-mails for all splits. We apply light rule-based post-processing, mainly removing extraneous tokens before or after the dialogue.

In the second inference phase, we first fine-tuned the annotation LLM f_a^0 on D_{ft} to evaluate the lower bound for the quality of annotations. We use f_a^0 to predict the annotation for the test set dialogues to compare them to the crowd-worker gold-standard. Then, we fine-tune f_a^0 additionally on these 200 dialogues with gold-standard annotations, yielding f_a^1.

Notably, the published data uses the gold-standard annotations for the test set and predictions from f_a^1 for the train and validation set.

5 Evaluation

To evaluate MonoTODia, we evaluate (i) the dialogue generation and annotation in isloation and (ii) the usefulness of the MonoTODia dialogues for training TOD systems.

5.1 Evaluation of Dialogue Generation

We evaluate the dialogue generation with the quality of the dialogues per se, and regarding the style translation explicitly. Both types of evaluation are impossible to perform automatically, since, by definition of the problem, no dialogues exist that allow for reference-based evaluations, ruling out most of the common NLG metrics (Gehrmann et al., 2023). Moreover, multiple aspects of the dialogue quality are intrinsically subjective (Amidei et al., 2019). We therefore opt for human evaluation with crowd workers recruited via Amazon Mechanical Turk to rate the dialogues based on the criteria in Tab. 1 on a scale of 1 to 5. These criteria entail qualities such as coherence, relevance, correctness and realness. For 100 of the test-set dialogues, we collected three independent ratings each. We ensured the qualification of the raters via a high task approval rate and an additional qualification task. They were shown the e-mail, dialogue, and instructions on how the dialogue should be created. These instructions were derived as closely as possible from the dialogue generation prompt, without giving away that the task was done by a LLM. Moreover, they were given instructions on how to rate the dialogues.

Criteria	Short explanation
C-0	E-mail is a vacation request.
C-1	Information from e-mail is represented in dialogue.
C-2	User gives more information in dialogue than e-mail.
C-2-1	If C-2 is "Yes": This additional information makes sense.
C-2-2	If C-2 is "Yes": This additional information is relevant to the booking.
C-3	The dialogue follows the rules of creation.
C-4	The dialogue resembles a real conversation.
C-5	The Bot is helpful to the user.

Table 1: The criteria and their short explanations for the crowd worker evaluation of the dialogue generation. The exact, full questions are shown in Appendix F.

5.2 Annotation Quality

To evaluate the annotation generated by the LLM, we opted for a reference-based evaluation by comparing it to crowd workers' annotations for the test data split. As such, we used crowd workers to create gold-standard annotations for the test dataset, where its accuracy has the highest importance for the overall evaluation. We ensured crowd-worker qualification as before.

5.3 Complexity of Different E-Mails

The e-mails that are used as the input of our approach come from various sources and have highly heterogeneous styles. They range from direct, free-format e-mails to tabular-like information but can be roughly classified as either short or elaborate e-mails (cf. Fig. 3). There is no clear indication that either of those two types led to consistently better or worse ratings by the human judges. However, one specific format was over-represented within the worst-rated dialogues. This format presents slot-value pairs (e.g., destination, Europe) separated by multiple line breaks. This very implicit style lacks additional context. It thus appears that the model struggles to extract the information more than in a more expressive way such as "slot: value".

5.4 Downstream Empirical Evaluation

While our main focus lies on the translation and its evaluation, we conduct an auxiliary experiment investigating if the generated data is suitable for the training of TOD systems in the Dialogue State Tracking (DST) and response generation (RG) tasks. To this end, we train two T5 (Raffel et al., 2023) and one BART (Lewis et al., 2020) model. The training details are provided in Appendix C. We formulate the DST task as predicting the dialogue state annotations from the chat history, i.e., all previous utterances. We evaluate this with three metrics: Exact-Match (EM), Soft-Match (SM), and Presence (PR). EM measures if there is a perfect match, SM if either all slots or all values are correct, and PR if the ground-truth is a subset of the prediction. For each metric, we report the mean percentage over all utterances and dialogues. Their exact formulations are provided in Appendix D.

For the RG task, we provide the model with oracle annotations and the chat history and evaluate the generated response with the BERTScore (Zhang et al., 2019). In every of these cases, we use the annotations from f_a^0 for the train and validation set, and the gold-standard for the test data.

6 Results and Discussion

Dialogue Generation. The outcomes of the crowd worker evaluation is summarized in Tab. 2, showing the average ratings for each question. They show that the generated dialogues have high quality, achieving an average rating of at least 4 out of 5 in nearly all tested criteria, with the lowest rating being 3.98. We see that even after our filtering, the judges deemed 11% of the e-mails to not be valid input for the task of generating a dialogue. This can mostly be attributed to e-mails being very uninformative (too short), as evidenced by significantly lower scores for C-1 and higher scores for C-2 in the invalid e-mails subset. When we control for the input to be valid, we can see that every criterion improves. Naturally, the opposite is true when only considering invalid inputs. Interestingly, C-4 and C-5 remain on a high level and see only minor changes when controlling for input validity. This underlines the strong language generation skills of the LLM. The consistently good scores for C-5 specifically can be attributed to the model being trained with the objective to be a helpful bot itself.

Criteria	Average	Valid	Invalid
C-0* (valid)	89%	n/a	n/a
C-1 (inf. exists)	3.98	4.09	2.71
C-2* (more inf.)	34%	30%	79%
C-2-1 (sensible)	4.27	4.48	3.37
C-2-2 (relevant)	4.33	4.50	3.58
C-3 (rules)	4.07	4.15	3.17
C-4 (realness)	4.41	4.43	4.25
C-5 (helpful)	4.37	4.41	4.00

Table 2: Average rating for the criteria. Valid column contains the results for dialogues where a majority of raters judged the input e-mail as valid, i.e., C-0 is positive. Invalid column is analogous. *: Binary question, for which we report the percentage of positive answers.

Annotation. To measure the accuracy of the annotations, we compare the annotations from f_a^0 to the human annotations for the test set. This provides a lower bound estimate for the annotation quality. The results are $EM = 25.78$, $SM = 36.77$, and $PR = 43.13$. These show that the annotations are not perfect, but surprisingly good for the extremely low amount of training data. Besides, the annotation of task-oriented dialogues is rather complicated and, in this case, allows for some syntactic variances that are semantically equivalent, e.g., for the format of dates or times. Even without having a human-generated gold-standard, we can assume that the annotations of the train and validation data split are of higher quality since the model got fine-tuned with the additional 200 human-annotated test dialogues.

Downstream Empirical Evaluation. The results for the usage of the MonoTODia data in training TOD systems are presented in Tab. 3. They show that the MonoTODia dialogues can be a valid

Metric	t5-base	t5-small	BART-large
EM	36.38	28.44	27.23
SM	60.89	52.33	51.29
PR	50.47	37.56	35.68
BERT	81.24	79.85	85.74

Table 3: Results of the DST and response generation evaluation. BERTScore shows the mean of the F1-Score, the standard deviation for all models was $9 < \sigma < 10$.

starting point for implementing a TOD system and can thus alleviate the cold start problem. We see a mostly positive correlation between model size and performance. However, BART-large, even though it is the largest model, performs worse than t5-base on the DST metrics but better in the RQ task.

7 Conclusion

This study investigates the feasibility and efficacy of using LLMs to translate e-mails into annotated task-oriented dialogues for the travel booking domain. By fine-tuning a state-of-the-art open-source LLM, performing extensive human assessment and empirical analysis, we have shown that the generated dialogues are of good quality and suitable for downstream training of TOD systems. Even for input e-mails that lack all necessary information, the dialogues achieved good scores. Note that the published dataset uses train and validation annotations predicted from fine-tuning on 220 gold-standard dialogues (f_a^1) to provide higher-quality annotations. Nevertheless, we observe that even a smaller dataset of only 20 examples can be a sufficient foundation for the training of TOD systems. The evaluation results show that the generated dialogues closely resemble real conversations, contain relevant information, and that the bot in the conversations is helpful in achieving the user's goal. Furthermore, the LLM closely followed the rules to generate the dialogue based on the e-mail. These results are consistent with other studies on synthetic dialogues (Mehri et al., 2022; Bae et al., 2022; Chen et al., 2023; Kulkarni et al., 2024), even though they follow different paradigms and do not translate from existing data. Overall, the findings suggest that this approach holds promise for addressing the challenges of data scarcity in training TOD systems. Even though our study is limited to only e-mails, we think that by leveraging existing data sources, such as e-mails, IT support tickets, or transcribed calls, and employing modern LLMs, companies can thus overcome barriers to deploying TOD systems in their service portfolios. Moreover, we had to translate the e-mails and proceed with English dialogues, which will need to be translated back into German for the use case in the cooperating company. LLMs that perform better on German are thus of high interest. We publish the resulting dataset to support future research.

8 Ethical Considerations

Widespread ethical usage of AI is an important step towards socially meaningful technological advance and broad acceptance of AI. Our work shows that LLMs might be used to generate training data for smaller, more specialized models, whose usage is less restrictive. We believe that synthetic data can to some extent alleviate problems that usually arise during model training, both regarding data scarcity, but also data imbalance. This can allow more organizations and companies to use AI in production. For the special case of service-agent-like chatbots, that improve a user's experience when using a service, we believe that the possible benefits outweigh the potential risk of, e.g., loss of jobs. Nonetheless, using LLM generated data will always bear risks of being biased or faulty. Furthermore, a dual use might be problematic, when dialogues are being generated to train chatbots with the aim to, e.g., spread fake news or commit fraud.

The payment per task for the human evaluators was calculated to equal an hourly rate of roughly \$10 given the average time needed, exceeding the Federal US minimum wage of \$7.25 per hour at the time of writing. Crowd workers were also paid for the qualification tasks.

Acknowledgements

We thank the adigi GmbH for their cooperation and making this research possible.

References

Jacopo Amidei, Paul Piwek, and Alistair Willis. 2019. The use of rating and Likert scales in Natural Language Generation human evaluation tasks: A review and some recommendations. In *Proceedings of the 12th International Conference on Natural Language Generation*, Tokyo, Japan.

Dustin Axman, Avik Ray, Shubham Garg, and Jing Huang. 2023. Contextual data augmentation for task-oriented dialog systems. In *ECML-PKDD 2023 Workshop on Challenges and Opportunities of Large*

Language Models in Real-World Machine Learning Applications (COLLM).

Sanghwan Bae, Donghyun Kwak, Sungdong Kim, Donghoon Ham, Soyoung Kang, Sang-Woo Lee, and Woomyoung Park. 2022. Building a Role Specified Open-Domain Dialogue System Leveraging Large-Scale Language Models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2128–2150, Seattle, United States. Association for Computational Linguistics.

Namo Bang, Jeehyun Lee, and Myoung-Wan Koo. 2023. Task-optimized adapters for an end-to-end task-oriented dialogue system. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7355–7369, Toronto, Canada. Association for Computational Linguistics.

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

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.

Maximillian Chen, Alexandros Papangelis, Chenyang Tao, Seokhwan Kim, Andy Rosenbaum, Yang Liu, Zhou Yu, and Dilek Hakkani-Tur. 2023. PLACES: Prompting Language Models for Social Conversation Synthesis. *Preprint*, arxiv:2302.03269.

Jan Deriu, Alvaro Rodrigo, Arantxa Otegi, Guillermo Echegoyen, Sophie Rosset, Eneko Agirre, and Mark Cieliebak. 2021. Survey on evaluation methods for dialogue systems. *Artificial Intelligence Review*, 54(1):755–810.

Nicholas Diakopoulos. 2019. *How Algorithms Are Rewriting the Media*. Harvard University Press, Cambridge, MA and London, England.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. 2024. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*.

Mihail Eric, Lakshmi Krishnan, Francois 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*, pages 37–49, Saarbrücken, Germany. Association for Computational Linguistics.

Sebastian Gehrmann, Elizabeth Clark, and Thibault Sellam. 2023. Repairing the cracked foundation: A survey of obstacles in evaluation practices for generated text. *Journal of Artificial Intelligence Research*, 77:103–166.

Matthias Gerstgrasser, Rylan Schaeffer, Apratim Dey, Rafael Rafailov, Henry Sleight, John Hughes, Tomasz Korbak, Rajashree Agrawal, Dhruv Pai, Andrey Gromov, Daniel A. Roberts, Diyi Yang, David L. Donoho, and Sanmi Koyejo. 2024. Is Model Collapse Inevitable? Breaking the Curse of Recursion by Accumulating Real and Synthetic Data. *Preprint*, arXiv:2404.01413.

Milan Gritta, Gerasimos Lampouras, and Ignacio Iacobacci. 2021. Conversation Graph: Data Augmentation, Training, and Evaluation for Non-Deterministic Dialogue Management. *Transactions of the Association for Computational Linguistics*, 9:36–52.

Som Gupta and S. K Gupta. 2019. Abstractive summarization: An overview of the state of the art. *Expert Systems with Applications*, 121:49–65.

Wanwei He, Yinpei Dai, Yinhe Zheng, Yuchuan Wu, Zheng Cao, Dermot Liu, Peng Jiang, Min Yang, Fei Huang, and Luo Si. 2022. Galaxy: A generative pre-trained model for task-oriented dialog with semi-supervised learning and explicit policy injection. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 36, pages 10749–10757.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Glorianna Jagfeld, Sabrina Jenne, and Ngoc Thang Vu. 2018. Sequence-to-Sequence Models for Data-to-Text Natural Language Generation: Word- vs. Character-based Processing and Output Diversity. *Preprint*, arxiv:1810.04864.

Jonáš Kulhánek, Vojtěch Hudeček, Tomáš Nekvinda, and Ondřej Dušek. 2021. AuGPT: Auxiliary Tasks and Data Augmentation for End-To-End Dialogue with Pre-Trained Language Models. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 198–210.

Atharva Kulkarni, Bo-Hsiang Tseng, Joel Ruben Antony Moniz, Dhivya Piraviperumal, Hong Yu, and Shruti Bhargava. 2024. Synthdst: Synthetic data is all you need for few-shot dialog state tracking. *Preprint*, arXiv:2402.02285.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Zekun Li, Wenhu Chen, Shiyang Li, Hong Wang, Jing Qian, and Xifeng Yan. 2022. Controllable Dialogue Simulation with In-context Learning. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 4330–4347, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

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

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

Shikib Mehri, Yasemin Altun, and Maxine Eskenazi. 2022. LAD: Language Models as Data for Zero-Shot Dialog. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 595–604, Edinburgh, UK. Association for Computational Linguistics.

Chiaki Miyazaki. 2023. Dialogue generation conditional on predefined stories: Preliminary results. *IEEE access : practical innovations, open solutions*, 11:85589–85599.

Jinjie Ni, Tom Young, Vlad Pandelea, Fuzhao Xue, and Erik Cambria. 2023. Recent advances in deep learning based dialogue systems: A systematic survey. *Artificial Intelligence Review*, 56(4):3055–3155.

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

Baolin Peng, Chunyuan Li, Zhu Zhang, Jinchao Li, Chenguang Zhu, and Jianfeng Gao. 2021b. SYNERGY: Building Task Bots at Scale Using Symbolic Knowledge and Machine Teaching. *Preprint*, arxiv:2110.11514.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Preprint*, arXiv:1910.10683.

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):8689–8696.

Mandar Sharma, Ajay Gogineni, and Naren Ramakrishnan. 2023. Innovations in Neural Data-to-text Generation: A Survey. *Preprint*, arxiv:2207.12571.

Connor Shorten, Taghi M. Khoshgoftaar, and Borko Furht. 2021. Text Data Augmentation for Deep Learning. *Journal of Big Data*, 8(1):101.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. AI models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Sebastian Steindl, Ulrich Schäfer, and Bernd Ludwig. 2023. Controlled data augmentation for training task-oriented dialog systems with low resource data. In *Proceedings of the 2nd Workshop on Pattern-based Approaches to NLP in the Age of Deep Learning*, pages 92–102, Singapore. Association for Computational Linguistics.

Sebastian Steindl, Ulrich Schäfer, and Bernd Ludwig. 2025. CoPrUS: Consistency preserving utterance synthesis towards more realistic benchmark dialogues. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 5902–5917, Abu Dhabi, UAE. Association for Computational Linguistics.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Chunliu Wang, Rik Van Noord, Arianna Bisazza, and Johan Bos. 2021. Evaluating Text Generation from Discourse Representation Structures. In *Proceedings of the 1st Workshop on Natural Language Generation, Evaluation, and Metrics (GEM 2021)*, pages 73–83, Online. Association for Computational Linguistics.

Weijie Xu, Zicheng Huang, Wenxiang Hu, Xi Fang, Rajesh Cherukuri, Naumaan Nayyar, Lorenzo Malandri, and Srinivasan Sengamedu. 2024. HR-MultiWOZ: A task oriented dialogue (TOD) dataset for HR LLM agent. In *Proceedings of the First Workshop on Natural Language Processing for Human Resources (NLP4HR 2024)*, pages 59–72, St. Julian's, Malta. Association for Computational Linguistics.

Yan Xu, Etsuko Ishii, Genta Indra Winata, Zhaojiang Lin, Andrea Madotto, Zihan Liu, Peng Xu, and Pascale Fung. 2021. CAiRE in DialDoc21: Data augmentation for information seeking dialogue system. In *Proceedings of the 1st Workshop on Document-grounded Dialogue and Conversational Question Answering (DialDoc 2021)*, pages 46–51, Online. Association for Computational Linguistics.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D. Williams. 2013. POMDP-Based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. 2023. Instruction Tuning for Large Language Models: A Survey. *Preprint*, arxiv:2308.10792.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. 2022. Description-Driven Task-Oriented Dialog Modeling. *Preprint*, arxiv:2201.08904.

A Domain-specific Ontology

Domain	Slots
hotel	board, name, area, address, price, feature, room_type, room_amount, stars, transfer, reviews
flight	departure_airport, arrival_airport, airline, type, class, price, duration
trip	travel_period_start, travel_period_end, length, price, type, destination, guests, guests_children, availability, confirmation_number
user	name, phone, e-mail
act	require_more, booking, information_sent, general

Table 4: The ontology of domains and slots we use as a basis for MonoTODia.

B Fine-tuning Details

All training and inference was done on a DGX A-100 320 GB platform, that offers eight 40 GB graphics cards. We utilized the peft (Mangrulkar et al., 2022) library to apply LoRA. We configured LoRA with the following parameters: $r = 16$, $\alpha = 64$, dropout probability $= 0.1$, and target the modules q_proj, up_proj, o_proj, k_proj, down_proj, gate_proj and v_proj. We do not use a bias in LoRA. We fine-tune with a learing rate of 1e-4 and 3e-5 for four and one epochs, for dialogue generation and annotation, respectively.

C TOD Training Details

For the TOD system we train two T5 (Raffel et al., 2023) and one BART (Lewis et al., 2020) model. Their sizes range from 60 million to 400 million parameters. We train the models for 10 epochs each with a learning rate of 5e-5 and keep only the best instance based on the validation loss. We formulate the input and output with special tokens that mark the beginning and end of the chat history, annotation and utterance to generate for the RG task. For example, a target output in the DST task might be `<annot>request:trip_type</annot>` for the input `<ctx>User: I am looking for a package deal for our vacation. </ctx>`

D Metrics

$$EM(y,\hat{y}) = \begin{cases} 1 & \text{if } \{y_1, y_2, \dots, y_n\} \\ & = \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n\} \\ 0 & \text{otherwise} \end{cases}$$

$$SM(y,\hat{y}) = \begin{cases} 1 & \text{if } \{s_i \mid (s_i, v_i) \in y\} \\ & \cap \{s_j \mid (s_j, v_j) \in \hat{y}\} \neq \emptyset \\ & \text{or } \{v_i \mid (s_i, v_i) \in y\} \\ & \cap \{v_j \mid (s_j, v_j) \in \hat{y}\} \neq \emptyset \\ 0 & \text{otherwise} \end{cases}$$

$$PR(y,\hat{y}) = \begin{cases} 1 & \text{if } \{y_1, y_2, \dots, y_n\} \\ & \subseteq \{\hat{y}_1, \hat{y}_2, \dots, \hat{y}_m\} \\ 0 & \text{otherwise} \end{cases}$$

E Full Prompts

F Full Dialogue Rating Questions

G Further Dialogue Examples

Criteria	Full Question
C-0	Check this box only if the original e-mail is an actual request for vacation offers. Do not check this box, if it is another type of e-mail, such as an empty e-mail, spam or any other e-mail that is not requesting a vacation or information on a vacation.
C-1	On a scale from 1 to 5, how much of the information given in the E-Mail is also represented in the dialog?
C-2	Check this box if the user utterances in the dialogue contain more information than was given in the original e-mail.
C-2-1	If C-2 is “Yes”: On a scale from 1 to 5, how much sense does the additional information make in the context of this dialogue?
C-2-2	If C-2 is “Yes”: On a scale from 1 to 5, how relevant is the additional information to the booking of a vacation?
C-3	On a scale from 1 to 5, how closely does the dialogue follow the instructions for creating the dialogue from the E-Mail (as described above)?
C-4	On a scale from 1 to 5, how closely does the dialogue resemble a real conversation?
C-5	On a scale from 1 to 5, how helpful is the Bot to the User?

Table 5: The criteria and the full questions as shown to the dialogue raters.

You can generate a dialogue between a user and a fictious chatbot based on an e-mail that gives information on the vacation.
Since the chatbot has the role of a travel agent, he should always be polite and helpful when talking to the user who is a potential customer.
Within the dialogue, the Bot is asking for information necessary to the booking of the vacation.
The User should answer them to find a fitting hotel, flight or both combined. The Bot provides options to book or says that there are no availabilities.
Please add additional, fictious information, e.g., for proposing hotel names and flights.

Here are some minimal requirements for the conversation:
If the user is booking a hotel, the conversation should clarify at a minimum the hotel name, travel dates and room number.
If the user is booking a flight, the conversation should clarify at a minimum the departure date, arrival date, departure airport and arrival airport.
If the user decides to book, the user should always provide his e-mail, which can just be a placeholder like example@e-mail.com, during the conversation or the Bot needs to ask for it.
If the e-mail does not contain enough information to clarify the just defined minimum, invent something that fits the dialogue flow, context and goal of the dialogue. The dialogue should either end with the user booking an option, or declining to do so.

[Example Input 1]:

[REDACTED FOR BREVITY]

[Example Output 1]:

[REDACTED FOR BREVITY]

With the help of [Example Input 1] and [Example Output 1] generate the output dialogue for this new input:

Figure 4: The full prompt used for dialogue generation. Omissions for the sake of brevity are marked in all-caps and bold.

You provide the labels for an utterance in the form of slots and slot values and for the dialogue actions that look like this: inform(slot_name=slot_value)
Each utterance can have multiple annotations.
For each annotation, first, differentiate between inform-annotation, request-annotation and act-annotation.
If the speaker is giving information use inform(), if he is asking for information, use request(), for all other cases, try one of the act_TYPE().

Multiple entities within one annotation are denoted with brackets, e.g., // inform(hotel_name=[entity1, entity2, entity3])
Only use the slots given in the following. In general an annotation is done as type(slot_name=slot_value), where type is one of inform, request or act.
Use the inform type, when a speaker is giving information and the request type when a speaker is asking for information.
Below I defined all possible slot_names, and the act_TYPE are below that.
If you want to annotate durations, you can use the abbreviations d for day and w for week, for example 'one week' becomes '1w'.
Negations can be done in programming style, e.g. inform(destination!=Germany) means that the user does not want to go to Germany.
Moreover, you can use inequality signs like inform(hotel_price<=1000) to say that the hotel price should be at most 1000€.

I will now give you all possible slot_names, a short [explanation] and for some the possible {slot_values}. The explanations are between the [] brackets and the slot_values between the {} braces. If no slot_values are given, they can be any string from the utterance.
Here they are in the form: slot_name [explanation] {slot_values (if any)}
hotel_board [meal plan] {all-inclusive, half-board, full-board, any},
hotel_name [name of the hotel],
hotel_area [hotel is in this area],
hotel_address [address of the hotel],
hotel_price [price of hotel],
hotel_feature [features like pool, wifi, etc.],
hotel_room_type [type of room],
hotel_room_amount [number of rooms to book],
hotel_stars [number of stars],
hotel_transfer [transfer from airport to hotel] {yes, no},
hotel_reviews [how other users rated the hotel],
flight_departure_airport [the airport where the flight will depart],
flight_arrival_airport [the airport where the flight will arrive],
flight_airline [the airline with which the flight will be conducted],
flight_type [the type of flight] {direct, indirect, one-way, round-trip},
flight_class [the ticket class for the flight] {economy, business, first},
flight_price [the price of the flight],
flight_duration [the duration of the flight],
travel_period_start [earliest possible date],
travel_period_end [latest possible date],
trip_length [duration of trip],
trip_price [the total price of the trip, can include hotel and flight],
trip_type [is the User looking for only a flight or hotel or a package deal] {hotel, flight, package},
destination [destination of trip],
guests [number of adult guests],
guests_children [number of children guests],
user_name [name of the user],
user_e-mail [e-mail of the user],
user_phone [phone number of the user],
availability [is a trip available] {yes, no},
confirmation_number [confirmation or reference number of the booking],

And here are the possible act_TYPES, again with [explanations] and for some the {values}
act_require_more [if the user needs anything else, either in general or specificly a hotel or flight],
act_booking [should the trip be booked],
act_information_sent [the information was sent to the user's mail],
act_general [greetings, thanking, etc.] {greeting, thanks, youre_welcome, bye, sorry, understood}

These instructions provide you with all the information you need to annotate the utterances.
I will now show you three examples of how the output should look like, after that it is your turn to annotate.

[EXAMPLE 1]
User: "Can you help me find a hotel that fits my requirements?" // request(hotel)
Bot: "Of course, what are your requirements?" // act_require_more(hotel)
User: "We are looking for a 4-star all-inclusive hotel in the Flic en Flac area with a private beach." // inform(hotel_stars=4), inform(hotel_board=all-inclusive), inform(hotel_area=Flic en Flac), inform(hotel_feature=private beach)
Bot: "I think I have a great option for you. Have you considered the Mauritius Beach Resort?" // inform(hotel_name=Mauritius Beach Resort), act_booking(book=?)
[REDACTED FOR BREVITY]

[EXAMPLE 2]
[REDACTED FOR BREVITY]

[EXAMPLE 3]
[REDACTED FOR BREVITY]

You need to strictly follow the descriptions of the annotations, or else everything might explode and people will get hurt. If you go step-by-step, you can look at the current utterance and every previous utterance to determine the correct annotation.
With the help of the four examples and the description above, edit the following dialog by adding the annotations according to your instructions above.

Figure 5: The full prompt used for annotation. Omissions for the sake of brevity are marked in all-caps and bold.

User: "I want to go on a trip." // request(trip_type=package)
Bot: "I'd be happy to help you plan your trip. What kind of vacation are you looking for?" // act_require_more(general=?)
User: "I want to go somewhere warm for Christmas and New Year's Eve." // inform(destination!=Germany), inform(travel_period_start=2021-12-25), inform(travel_period_end=2022-01-01)
Bot: "Okay, we can definitely help you with that. How many people are going on this trip?" // request(guests)
User: "Just the two of us." // inform(guests=2)
Bot: "Great, do you have a preference on the destination?" // act_require_more(destination=?)
User: "Somewhere with a beach." // inform(destination_type=beach)
Bot: "Sure, we have many great beach destinations. How long do you want to stay?" // request(trip_length)
User: "Ten days." // inform(trip_length=10d)
Bot: "And do you want to fly from Munich, Salzburg or Innsbruck?" // request(flight_departure_airport)
User: "Salzburg." // inform(flight_departure_airport=Salzburg)
Bot: "Okay, I can offer you a 10-day all-inclusive stay in the Canary Islands, specifically in the Melia Salinas hotel, which is located directly on the beach in Fuerteventura. It has an excellent recommendation rating of 85% and is known for its cleanliness and good food. It is available for an all-inclusive price of 1500 € per person. Would you like to book it?" // inform(hotel_name=Melia Salinas), inform(hotel_area=Fuerteventura), inform(hotel_feature=[beach, excellent recommendation rating, cleanliness, good food]), inform(hotel_board=all-inclusive), inform(hotel_price=1500), inform(destination=Fuerteventura), act_booking(book=?)
User: "How do I book?" // act_require_more(book=?)
Bot: "To book, can you please provide me with your e-mail address? So I can send you the booking information." // request(user_e-mail)
User: "info@example.com" // inform(user_e-mail=info@example.com)
Bot: "I have sent you the necessary information, thank you for your order." // act_information_sent(yes)
User: "Thank you." // act_general(thanks)
Bot: "You're welcome. Do you have any other questions?" // act_require_more(general=?)
User: "No, that's it. Thank you." // inform(availability!=yes)
Bot: "You're welcome, bye!" // act_general(bye)
User: "I want to go on a trip." // request(trip_type=package)
Bot: "I'd be happy to help you plan your trip. What kind of vacation are you looking for?" // act_require_more(general=?)

Figure 6: An additional example dialogue generated with MonoTODia.

User: "I am looking for a hotel in Sardinia with breakfast and rental car" // inform(trip_type=hotel), inform(destination=Sardinia), inform(hotel_board=breakfast), inform(hotel_feature=rental_car)
Bot: "Great! There are many hotels in Sardinia that offer breakfast and car rental. What is your budget?" // act_require_more(hotel)
User: "Our budget is around €2000 for 2 weeks. Do you have any hotels that fit our criteria?" // inform(trip_price=2000), inform(trip_length=2w)
Bot: "Certainly, I can suggest some hotels that match your requirements. Can you please tell me what type of hotel star rating are you looking for?" // act_require_more(hotel)
User: "Well, at least 4 stars. Do you have any options in that range?" // inform(hotel_stars=4)
Bot: "I have some great options for 4-star hotels in Sardinia that include breakfast and car rental. Let me see what I can find for you. Do you have a preference for a specific area in Sardinia?" // inform(hotel_area=Sardinia), act_require_more(hotel)
User: "We would like to be close to the beach, preferably in the north. But we are flexible." // inform(hotel_location=beach), inform(hotel_area=north)
Bot: "Okay, I've found a few options for you. Can you please tell me how many rooms would you like to book?" // act_require_more(hotel)
User: "We are 2 people, so just one room. Do you have any availability in May?" // inform(guests=2), inform(hotel_room_amount=1), inform(travel_period_start=May)
Bot: "I'm afraid we are fully booked in May, but I can offer you a great option in June. Would you like me to make the reservation for you?" // inform(availability=no), inform(travel_period_end=June), act_booking(book=?)
User: "That would be great, thank you!" // act_booking(book=yes)
Bot: "Awesome, I will send you all the details to your email. What is your email address?" // act_require_more(user_e-mail)
User: "It is example@e-mail.com. Why?" // inform(user_e-mail=example@e-mail.com)
Bot: "So I can send you all the details regarding the booking. Thank you for booking with us!" // inform(confirmation_number=booking), act_information_sent(yes)
User: "You're welcome! Thank you for your help!" // act_general(thanks)
Bot: "You're welcome, bye!" // act_general(bye)

Figure 7: An additional example dialogue generated with MonoTODia.