Computer Science
&
Data Science

Capstone Report - Fall 2023

Dialogue State Tracking using Large Language Models

Bingsen Chen,
Peiyang Wu,
Xiaocheng Yang

supervised by
Professor Yik-Cheung Tam & Professor Hongyi Wen

Preface

Bingsen Chen is pursuing a Bachelor's degree in Data Science concentrating on Artificial Intelligence at New York University Shanghai. Xiaocheng Yang is pursuing a Bachelor's degree in Computer Science from the same institution. Peiyang Wu is pursuing a Bachelor's degree in Computer Science and Data Science concentrating on Marketing at New York University Shanghai. As task-oriented dialogue systems, such as restaurant virtual receptionists, gain ground nowadays, it is important to improve the dialogue understanding abilities of such systems. This project means to test the feasibility of Large Language Models on the Dialog State Tracking task and further understand the performance of LLMs across different training settings.

Acknowledgements

We are profoundly grateful to Professor Yik-Cheung Tam, whose invaluable guidance has played a pivotal role throughout this Capstone research. His expertise and insights have been instrumental in our academic journey. Our sincere appreciation also goes to Professor Hongyi Wen for his unwavering support in achieving the success of this capstone project.
Furthermore, we extend thanks to our parents and friends whose support has been a great source of encouragement.

Abstract

This project explores the task of Dialogue State Tracking (DST) in task-oriented dialogue systems. DST involves understanding and tracking user intentions during multi-turn dialogues, typically represented as slot-value pairs. The study reviews various DST methods, highlighting limitations in classification-based and span extraction-based approaches. The paper adopts a generative approach, leveraging large language models (LLMs) to improve performance. Experimenting with different formulations, model scaling, and data scaling, the study demonstrates the feasibility of using LLMs for DST, achieving a new state-of-the-art on MultiWOZ 2.2 and 2.4 benchmarks. The slot-level question-answering formulation is shown to enhance LLM performance, reducing hallucination. Additionally, the paper provides insights into the influence of model scaling and data scaling on DST performance, contributing to a comprehensive understanding of the subject.

Keywords

Natural Language Processing, Large Language Models, Dialogue State Tracking

Contents

1. Introduction

Task-oriented dialogue systems, such as ticket-booking chatbots or call-answering virtual receptionists, have increasingly abundant real-world applications nowadays. To complete a user-specified task like making a restaurant reservation, such systems need to first understand and keep track of users' intentions throughout multi-turn dialogues, which is the task of Dialogue State Tracking (DST). In particular, the dialogue state in DST is represented as slot-value pairs (e.g. {"restaurant-price": "expensive", "restaurant-area": "center"}), and modern-day dialogue systems typically employ a neural network-based model to track the dialogue state by filling slots based on a given schema.

Previous methods vary in the specific formulation to tackle this task. Classification-based methods [1] predict the value for a given slot within a fixed set of possible answers. Span extraction-based methods [2, 3, 4] predict the value for each slot by extracting the most likely text span within the dialogue context. Both methods have inherent limitations. Classification-based DST cannot predict non-categorical slots, such as "the name of a restaurant", where there are infinite possible answers; it also suffers from domain shift when being applied to an unseen domain of slots. Span extraction falls short of handling implicit meanings. For instance, "Can you book a hotel room for my wife and me?" infers booking a room for 2 people, but there is no number "2" to be extracted for the slot "number of booking people". Generation-based methods thus become the new paradigm for the DST task, especially with the rise of pretrained language models. Such methods directly generate the slot values token-by-token, which makes it possible to generalize to unseen domains and values. Given the impressive performance of pretrained transformer models on various generation tasks [5, 6, 7], we also adopt a generative approach to DST, but we further break the task down into slot-level question-answering format, which reduces hallucination during decoding and will be expounded in Section 3.2.

Meanwhile, we have witnessed the power of scaling law [8] and the emergent abilities [9] of large language models (LLMs) to solve natural language processing tasks [10, 11]. LLMs have shown the ability to understand and generate text of complex meaning, demonstrate improved reasoning skills, and exhibit a better grasp of nuanced contexts, beyond what was explicitly programmed or trained for. These observations are highly relevant to our task since in DST, the model is expected to understand and obtain the key information within a long-form spoken dialogue. As previous DST methods reached a bottleneck on the joint goal accuracy metric (See Section 3.3

for details) with relatively small language models, we probed into finetuning LLMs to tackle this long-standing challenge.

In particular, this study experimented with different task formulations, model scaling, and data scaling to shed light on how we can finetune LLMs for DST. Our main contributions are:

- **We proved the feasibility of using LLMs for DST and achieved the new state of the art on the MultiWOZ 2.2 and 2.4 benchmarks.** Specifically, our best-performing model based on Llama-2-chat(13b) surpassed the previous leading method on both versions of MultiWOZ in terms of joint goal accuracy.

- **We demonstrated that the slot-level question-answering formulation enhanced LLM's performance on DST** In our experiments, generating one slot value at a time effectively reduces hallucination as opposed to generating a JSON string that contains all dialogue states.

- **We clarified the influence of model scaling and data scaling on DST performance.** We provided an extensive analysis of base model scaling, LoRA rank scaling, and training data scaling in our experiments.

2. Related Work

DST task has been tackled in various formulations in terms of how DST models predict the slot values. In a broad sense, prediction can be made from a fixed ontology or open vocabulary [12]. Under fixed ontology, we assume there exists a predefined set of values for each slot. Open vocabulary is a more practical yet challenging approach. Many models of different architectures have been proposed to tackle the DST task. In the big picture, we observe a transition from RNN-based architectures to transformer-based architectures. Also, as pretrained transformer models demonstrate their impressive generation capabilities, we observed a pivot from classification or span-extraction methods to end-to-end generation for flexibility and simplicity.

Initially dominant in DST, RNN-based architectures, such as the Flat Joint State Tracker (FJST) and the Hierarchical Joint State Tracker (HJST), employed bidirectional Long Short-Term Memory structures for encoding dialogue history. FJST utilized a feedforward neural network for slot value decoding, while HJST adopted a hierarchical recurrent encoder-decoder structure [13, 14]. The Transferable Dialogue State Generator (TRADE) implemented Gated Recurrent

Units for dialogue encoding and slot value decoding [15]. Before the rise of transformers, attention mechanisms gained prominence, as seen in the Dialogue State Tracking via Question Answering model, incorporating bidirectional GRU and bidirectional attention flow for enhanced interaction between dialogue and queries [16, 17]. The Schema-guided Multi-domain Dialogue State Tracker with Graph Attention Networks utilized a graph attention matching network and recurrent graph attention network for schema extraction and leveraging previous conversation states.[18].

Recognizing the Transformers' potential, researchers integrated Transformers into DST models. Early examples include the Non-Autoregressive Dialogue State Tracking model, using attention layers for encoding and two transformers for decoding dialogues and slots [19]. The Slot-Utterance Matching Belief Tracker applied BERT for encoding utterances and slot types, combining them through multi-head attention layers and RNN for final encoding [20]. Another paradigm shift involved relying entirely on a single pre-trained transformer, as exemplified by the TOD-BERT model, which utilized BERT for comprehension and slot projections for each slot, determining prediction values based on similarity with BERT hidden states [1]. Departing further, Simple-TOD, a GPT-2-based model, conceived dialogue state tracking as a generative process, directly generating slot values [21]. Our ongoing research project aligns with this trend, exploring the potential benefits of larger pre-trained models for improved dialogue state tracking outcomes.

Traditionally, researchers formulate DST as a classification or span-extraction task. A DST classifier is used in formulations where the goal is to predict dialogue states in predefined categories or labels (i.e. fixed ontology). Instead of generating text, a classifier decoder assigns a class label or category to the input text. For instance, the obtained TOD-BERT model provides an improvement over the standard BERT model on DST tasks, by proposing a contrastive objective function to simulate the classification task, which is based on BERT architecture but using two loss functions: masked language modeling loss and response contrastive loss [1]. This approach is analogous to text classification tasks such as sentiment analysis and topic classification. However, such fixed ontology approach in reality is impractical. The set of possible answers might be either too large or simply not accessible. The Span Extraction decoder selects and extracts existing text spans from the dialogue, combining them to generate the current dialogue state. The Triple Copy Strategy (Tripy)[22] utilizes diverse copy mechanisms for slot value assignment, eliminating the need for predefined lists. Conversational Semantic Parsing (CSP) [3] builds upon Tripy, converting natural language queries into formal language with inductive biases in a Pre-trained Language Model. CSP is evaluated for Dialogue State Tracking (DST),

employing controllable counterfactuals for robust evaluation in diverse scenarios. Additionally, Multi-Perspective Dialogue Collaborative Selection [4] improves the efficiency of state tracking by dynamically selecting relevant dialogue contents for each updated slot. However, slot values can be implicit or have different wordings (e.g.“expensive” can be synonymous with “high-end”), which can cause problems if solely on the dialogue context to extract value spans.

As pretrained transformer decoder and encoder-decoder models show great capability in generation tasks [23, 24, 5], the DST task can thus be formulated as simple as an end-to-end autoregressive generation problem. SimpleTOD [21] and Soloist [25] both use GPT-2 to generate state, action, and response generation, unifying all the task-oriented dialogue tasks. Yet, SimpleTOD adopts transfer learning to finetune the GPT-2 pretrained on a large general-purpose text corpus, while Soloist proposes a multi-task pretraining step to ground the causal language model in end-user goals. Similar to Soloist, PPTOD [26] also proposes a task-specific pretraining objective, but it uses prepended task instructions to allow the model to adapt to downstream tasks.

As instruction finetuning shows great adaptability across downstream tasks, prompt-style DST has also gained ground in recent years. Prompting the language model with natural or semi-natural language instruction is highly flexible in terms of cross-domain generalizability. Lee et al. used schema-driven prompts that break down a prediction for a single turn into multiple prompts, where each prompt asks the T5 model to generate the value for a single slot given the natural language description of the slot [20]. As opposed to generating one slot at a turn, D3ST adopts a large prompt that encloses all possible slots for a single dialogue turn [27]. The current state of the art is the BREAK model that builds on top of purely prompting the language model [28]. It also trained a re-ranker model that selects the best slot-value candidate from beam search decoding, which helps them surpass the previous best JGA reported by a margin of around 24%. Our method also falls into this subcategory of prompt-style DST via instruction finetuning.

3. Solution

In this section, we present our methods and experiment setup to investigate how we can leverage the generative abilities of LLMs to tackle the DST task.

3.1. Dataset

In our experiments, we used the training data from Multi-Domain Wizard-of-Oz (MultiWOZ) 2.2 [29], and evaluated the models on both MultiWOZ 2.2 and 2.4 [30]. MultiWOZ 2.2 is the most extensively used benchmark for DST, containing a large-scale multi-domain dialogue dataset that contains about 10k multi-turn dialogues spanning over 8 domains. However, although it is already a refined version of the original MultiWOZ dataset [31], it still contains many annotation errors. MultiWOZ 2.4 rectified the annotations in the validation set and test set in MultiWOZ 2.1, leaving the training set intact. We treat the MultiWOZ 2.4 validation set and test set as clean data to evaluate our model upon.

Therefore, training on MultiWOZ 2.2 that contains noisy data better simulates the real-world setting where data is never error-free. Evaluating on MultiWOZ 2.4 reveals the absolute performance of our models. We still evaluated our models on MultiWOZ 2.2 to compare our results with the existing literature since most previous methods are evaluated on MultiWOZ 2.2.

It should be noted that the dataset is fully labeled by human researchers, thus noise and mistakes are unavoidable. The noises could have unexpected impacts on the fine-tuning process. We would express our appreciation to the researchers who keep working on this dataset.

3.2. Task formulation

In MultiWOZ, a dialogue is composed of multiple turns where each turn is an utterance from either the user or the AI assistant. We predict the dialogue state right after each user's utterance. In particular, the raw dataset can be defined as $\mathcal{D} = \{C_i, S_i, V_i\}_{i=1}^{N}$, where C_i, S_i, and V_i stand for the dialogue context, all possible slots, and ground truth values of a dialogue turn i respectively. To clarify, we consider each dialogue turn as a sample, independent of the full dialogue it belongs to. $S_i = (s_{i,1}, s_{i,2}, s_{i,3}, \dots)$ where each $s_{i,j}$ represents a possible slot to be filled in turn i. $V_i = (v_{i,1}, v_{i,2}, v_{i,3}, \dots)$ are the corresponding true values for all slots in S_i.

In our slot-level question answering (QA) formulation, we further break down each sample where each slot in a dialogue turn becomes a sample. Formally, each sample is a triplet of $(C_i, s_{i,j}, v_{i,j})$. Inspired by schema-driven prompting [20] and instruction finetuning [10], we format an input prompt as an instruction that asks the model for the predicted value for a certain slot. We also augment the prompt with the textual description of the slot name (e.g. the description of "restaurant-bookday" is "day of the restaurant booking"). The prompt can be expressed as $p_{i,j} = (\tau_{\text{QA}}, C_i, s_{i,j}, d_{i,j})$, where τ_{QA} is the instruction template for slot-level QA format (See

Appendix A.4 for an example) and $d_{i,j}$ is the description for slot $s_{i,j}$. Thus, given an LLM π_θ, the predicted slot value is produced by $\hat{v}_{i,j} = \pi_\theta(p_{i,j})$.

We also compared this formulation with generating all slot values for a dialogue turn as a JSON string simultaneously in Section 4.3. In this case, each instruction prompt is $p_i = (\tau_{\text{JSON}}, C_i, S_i, D_i)$, where τ_{JSON} is the instruction prompt template for JSON output formulation (See Appendix A.3 for an example). $D_i = (d_{i,1}, d_{i,2}, \dots)$ contains text descriptions for each slot in S_i. At inference time, the model would predict all slot values $\widehat{V}_i$ at the same time.

3.3. Evaluation metrics

Given the instructions from the authors of MultiWOZ 2.2[1], we wrote the evaluation scripts following TRADE's implementation[2]. We report the following metrics for each experiment run:

Joint Goal Accuracy (JGA) JGA calculates the accuracy of DST models on a dialogue turn level. In other words, it measures the fraction of dialogue turns where all slot values are predicted correctly over the total number of dialogue turns, which can be mathematically expressed as

$$\text{JGA} = \frac{\sum_{i=1}^{N} \mathbb{1}_{\{\hat{v}_{i,j} = v_{i,j} \ \forall j\}}}{\text{N}}$$

JGA is generally considered a more challenging metric as it requires the DST model to not make any errors in tracking the dialogue state for a single turn. JGA is also more similar to the real-world setting where task-oriented dialogue systems should always accurately capture users' demands for reliability. Most methods on the leaderboard[3] can predict over 90% of the slots correctly, but their JGA is generally within 50% to 60%.

Slot F1 Slot F1 is calculated at dialogue turn level as well and averaged across all the turns. It is the harmonic mean of precision and recall of the predictions for a dialogue turn. We define the "None" values (i.e. the dialogue has not mentioned the slot) as negative samples. Thus, predicting the correct value for a slot that has been mentioned is regarded as one true positive count, while predicting "None" when the ground truth is "None" is counted as a true negative. Therefore, aggregated precision and recall for all N turns are computed by

$$\text{precision}_{\text{agg}} = \frac{1}{N}\sum_{i=1}^{N} \text{precision}_i = \frac{1}{N}\sum_{i=1}^{N} \frac{\text{TP}_i}{\text{TP}_i + \text{FP}_i},$$

[1] https://github.com/budzianowski/multiwoz
[2] https://github.com/jasonwu0731/trade-dst/blob/master/models/TRADE.py
[3] https://github.com/budzianowski/multiwoz

$$\text{recall}_{\text{agg}} = \frac{1}{N}\sum_{i=1}^{N}\text{recall}_i = \frac{1}{N}\sum_{i=1}^{N}\frac{\text{TP}_i}{\text{TP}_i + \text{FN}_i}$$

TP_i, FP_i, and FN_i respectively represent the true positive count, false positive count, and false negative count for turn i. Precision_i and recall_i then stand for the precision and recall for turn i. We calculate the slot F1 score for a single turn i with the harmonic mean of precision and recall. Lastly, slot F1 score for all turns is computed by averaging the per-turn slot F1.

$$\text{F1}_{\text{agg}} = \frac{1}{N}\sum_{i=1}^{N}\text{F1}_i = \frac{1}{N}\sum_{i=1}^{N}\frac{2 \cdot \text{precision}_i \cdot \text{recall}_i}{\text{precision}_i + \text{recall}_i}$$

3.4. Model and Training

In our experiments, we used three different LLMs – Llama 2, Llama 2 chat, and Falcon – to tackle the DST task. To train such large-scale models, full-parameter finetuning is not feasible given the resource and time constraints.

Llama 2, an LLM developed by MetaAI, is based on the transformer decoder architecture and has been trained on a newly crafted corpus consisting of data that is publicly available [32]. Compared with its previous generation, Llama, the size of the training corpus increased by 40% [32]. As a base model, it can already perform well in the field of commonsense reasoning, world, knowledge, and reading comprehension. Its performance in those fields surpasses other base models including Falcon, MPT, and Llama [32]. Therefore, we choose to fine-tune this base model and hope to obtain a state-of-the-art result from it.

On top of Llama 2, MetaAI employed the technique of Reinforcement Learning from Human Feedback (RLHF) when finetuning the Llama 2 chat version. RLHF aligns a pretrained LLM with human preference for chatting purposes [33]. They first trained a reward model, of which the objective is to make the score of a chosen response higher than its counterpart. Having the reward model ready, they employed Proximal Policy Optimization, which is a standard method for RLHF, and Rejection sampling, where they sampled K outputs from the model, selected the best candidate with our reward, and used the rejection sampled data from the larger model to distill a large 70B model into smaller models. The human evaluation shows that Llama 2-chat is competitive with ChatGPT and outperforms the PaLM-bison chat model by a large percentage on our prompt set [32]. Although the architecture of the model has not changed, we test whether RLHF can further impact the model's capacity to understand dialogue data, as RLHF training is based on human-AI dialogues.

Falcon is a large language model developed by the Technology Innovation Institute. It is also based on the transformer architecture and features some innovative techniques. In the aspect of architecture, it introduces the attention and MLP block parallelization [34]. In a standard transformer, one MLP block stacks upon one attention layer. The input of the MLP block is the output of the attention layer. In Falcon, they calculate the MLP result and the attention result simultaneously. The MLP block and the attention layer share the same input and their outputs are aggregated using summation. This technique in principle also accelerates the parallel training by reducing the communication costs associated with tensor parallelism. In this project, we finetuned this model in order to provide a comparison with Llama 2 and Llama 2-chat.

In order to finetune the three mentioned base models, we employ the parameter-efficient fine-tuning (PEFT) technique so that we can achieve reasonably satisfying results using fewer hardware resources. It uses QLoRA, which is a quantized version of Lower Rank Adaption(LoRA). The central key of LoRA is that the weights of a model are matrices and each matrix can be decomposed into a base matrix plus a product of two small matrices [35]. Therefore, the task of fine-tuning a matrix can be rephrased as training the two matrices in the decomposition and the initial matrix is a base. In this manner, it is possible to use less number of parameters to mimic the capacity of a large model. QLoRA goes one step further. It introduces a brand new data type, NF4, which empirically yields better results, double quantization, which further saves space for the weights, and a paged optimizer, which utilizes the NVIDIA unified memory to avoid gradient checkpointing memory spikes [36].

4. Results and Discussion

4.1. Experiments

During the training process, we first formulate the raw training data from MultiWOZ into slot-level QA and JSON format. Subsequently, we employ this refined data to fine-tune Llama-2-7b, Llama-2-7b-chat, Llama-2-13b-chat, and Falcon-7b. We specified hyper-parameters, especially: model type, data size, LoRA r, and Data format. Throughout the fine-tuning phase, we diligently save checkpoints at regular intervals, capturing the evolving state of the model.

When achieving the convergence of training loss, we pivot to evaluations centered around the checkpoints corresponding to the point of convergence. This evaluation encompasses generating results using the fine-tuned model, subsequent post-processing (see Section 5.4 for details) of

these results, and then computing relevant evaluation metrics.

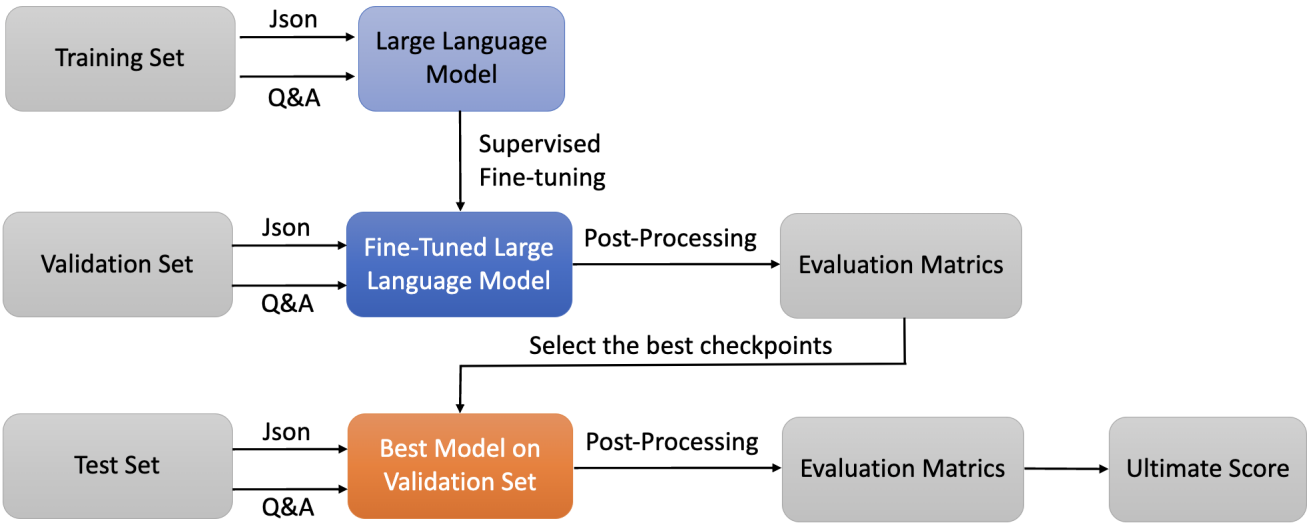

Figure 1: General Experiment Pipeline

4.2. Main results

Descriptive table comparing our result with related works

	Model Size	MultiWOZ 2.2		MultiWOZ 2.4	
		JGA	Slot F1	JGA	Slot F1
BREAK-greedy (GPT2) [28]	117M	53.7	-	63.1	-
BREAK-greedy (T5) [28]	60M	54.8	-	68.0	-
SDP-DST [20]	770M	57.6	-	-	-
TOATOD [37]	220M	63.8*	94.0	-	-
D3ST [27]	11B	58.7	-	75.9*	-
ChatGPT	Unknown	61.52	-	**83.16**	-
Falcon (QA)	7B	67.1	90.9	71.0	95.0
Llama2-chat (QA)†	13B	**73.4**	**94.9**	**82.4**	**97.1**

Table 1: The results of finetuning different base models

Table 1 demonstrates the performance of the two best models finetuned in this project compared with the previous models as strong baselines. The baselines include BREAK-greedy [28] of model size 117M, BREAK-greedy [28] of size 60M, SDP-DST [20] of size 770M, TOATOD [37] of size 220M, and D3ST [27] of size 11B. The baselines are generally leveraging models that have no more than 1 billion parameters except for D3ST. Here we use two corpora, MultiWOZ 2.2 and MultiWOZ 2.4, to examine the performance of the models. For MultiWOZ 2.2, the best performance is given by TOATOD, whose JGA reaches 63.8. For MultiWOZ 2.4, the best performance is given by D3ST, whose JGA reaches 75.9. In our project, we leverage Falcon of size

7B and Llama2-chat of size 13B. The finetuned 7B Falcon, whose JGA reaches 67.1 can already outperform all the baselines substantially on MultiWOZ 2.2. The finetuned 13B Llama2-chat further pushes the JGA to a level as high as 73.4 on MultiWOZ 2.2 and 82.4 on MultiWOZ 2.4, both of which outperform all the counterparts, suggesting the validity of finetuning large language models to fulfill the DST task.

On the other hand, the greater capacity of fine-tuned Llama2-chat could be due to the following aspects. First, the size of the Llama base model is larger than the Falcon base model. Empirically speaking, a larger model can achieve better results. Second, the Falcon-7b base model was released in June 2023 while the Llama2-7b base model was released in July 2023. As a newer product, it is expected that Llama2 would outperform Falcon in some sense. Third, Llama2-chat was finetuned by means of Reinforcement Learning from Human Feedback (RLHF), which might have some influence over the capacity of downstream tasks.

4.3. Comparison of JSON and Slot-level QA

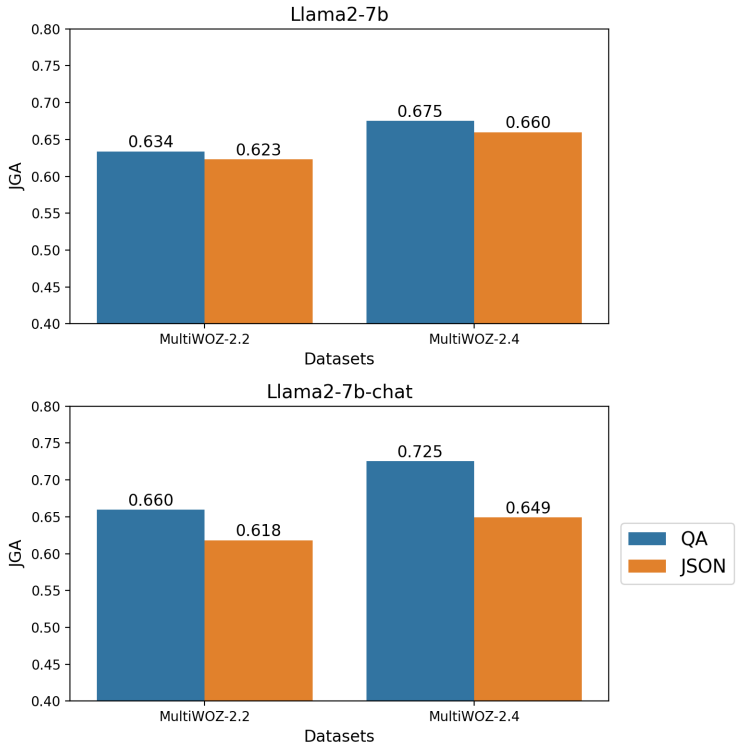

Figure 2: A comparison between QA formulation and JSON formulation

Figure 2 demonstrates how different data formulations can impact the model performance in the DST task. The upper plot contains the performance of fine-tuned Llama2-7b and the lower

plot contains the performance of fine-tuned Llama2-7b-chat. It is clear that the QA format outperforms the JSON format in all cases. It is expected because in principle, compared with JSON, where the model has to generate all slot values in a dialogue, QA decomposes the task into several one-intent retrieval steps. In each step of generation, we only ask the model to generate the value for one slot, greatly reducing the difficulty of the task. Furthermore, since the output length is reduced significantly, the output turns out to be more stabilized. While generating the JSON format, the finetuned model would exhibit hallucinations from time to time, which will be discussed in detail in Section 5.1

4.4. Model Scaling

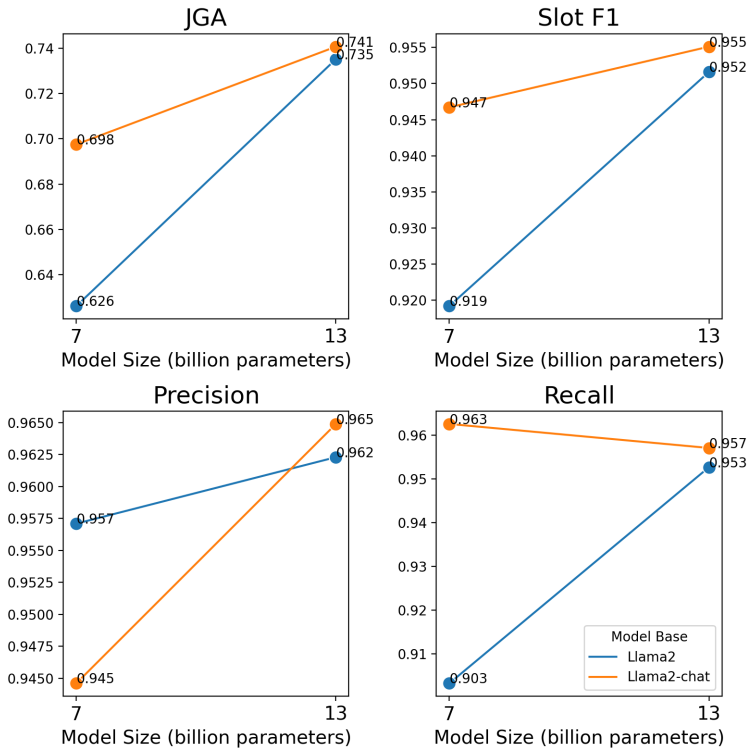

Figure 3: A comparison between 7B models and 13 models

Figure 3 refers to how the model scaling will impact the tine-tuned output. We employed the Llama-7b and Llama-13b models for this experiment. Llama-7b has 7 billion parameters, while Llama-13b has 13 billion parameters. This will elicit Llama-13b more powerful and cumbersome than Llama-7b. We undertake the fine-tuning under Lora = 16 and 30% of training data. It turns out that for both Llama-2 and Llama-2-chat, with the increment of model size, the performance of the model can be significantly increased. Furthermore, when the model size is increased from

7 billion to 13 billion, the performance gap between the aligned chat version and the non-chat version is mitigated from around 5 percent to less than 1 percent, suggesting that the size of the model can actually mitigate the difference in the pertaining process while being fine-tuned for the downstream task of DST. However, one anomaly observed here was the decrement of recall of the finetuned chat version when the model size was increased from 7 billion to 13 billion, for which we could not fully understand the reason.

4.5. LoRA Scaling

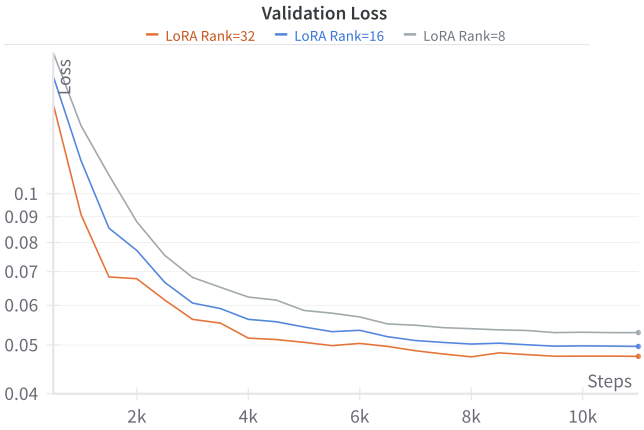

Figure 4: A comparison between evaluation loss using different LoRA ranks for finetuning Llama2-7b

Figure 4 shows the dynamics of validation loss during finetuning using different LoRA ranks. We experimented with rank=8, rank=16, and rank=32. It was observed that a larger rank of LoRA led to a smaller validation loss during all phases of finetuning. One explanation could be that essentially LoRA decomposed one weight matrix into a product of two trainable ones. The larger the rank is, the more trainable parameters there will be. Since a larger LoRA rank offers more trainable parameters, more parameters will actually be updated during gradient descent.

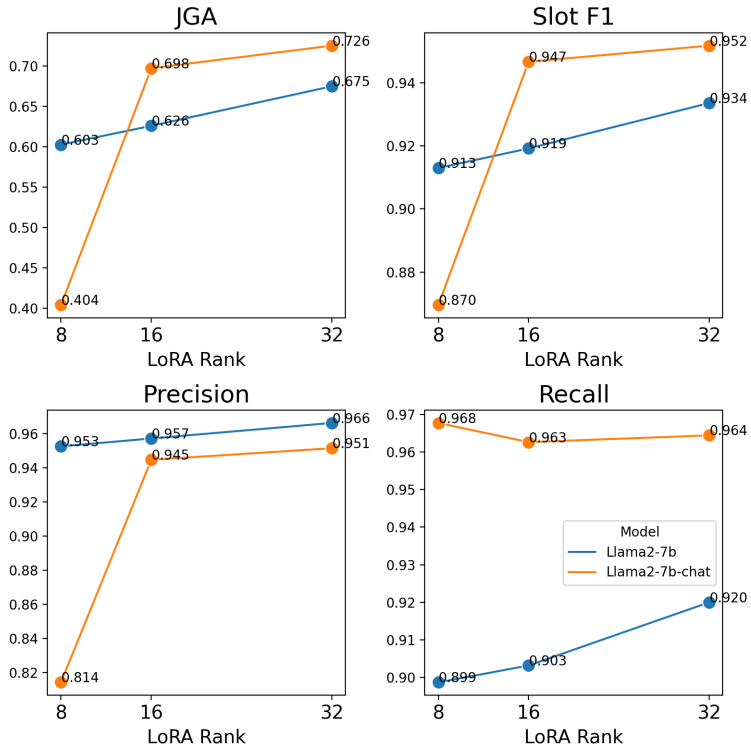

Figure 5: A comparison between finetuned models using different LoRA ranks

Figure 5 shows how different ranks could lead to different model performances in the DST task. Here we tested it using finetuned Llama2-7b and Llama2-7b-chat. Since a larger rank leads to lower validation loss, it is not surprising that the larger the rank is, the better the performance will be. However, there are two anomalies observed here. First, although the performances of both models grew, the performance of the non-chat version almost grew linearly as the rank increased while the performance of the chat version experienced a great leap when the rank increased from 8 to 16. Second, the precision of the chat version actually decreased when the rank was increased from 8 to 16, which was extremely counter-intuitive. The only difference in their settings was that the chat version used Reinforcement Learning from Human Feedback (RLHF) while the non-chat version did not. It is possible that it was due to the RLHF that caused the different sensitivities to rank value.

4.6. Data Scaling

	MultiWOZ-2.2				MultiWOZ-2.4			
	JGA	**Slot F1**	**Precision**	**Recall**	**JGA**	**Slot F1**	**Precision**	**Recall**
30%	67.37%	93.50%	94.20%	94.62%	74.07%	95.51%	96.49%	95.71%
60%	73.35%	94.88%	95.81%	95.40%	82.35%	97.10%	98.26%	96.62%

Table 2: Base Model: Llama2-13b-chat

	MultiWOZ-2.2				MultiWOZ-2.4			
	JGA	**Slot F1**	**Precision**	**Recall**	**JGA**	**Slot F1**	**Precision**	**Recall**
30%	63.29%	90.23%	90.27%	92.16%	66.37%	91.02%	92.21%	91.52%
100%	67.14%	90.93%	93.13%	90.58%	70.95%	95.03%	95.64%	94.26%

Table 3: Base Model: Falcon-7b

Table 2 and table 3 demonstrate how data scaling can impact the finetuned model's performance for DST task. For Llama2-13B-chat, we adopted 30% and 60% of the full dataset to experiment with. For Falcon-7B, we used 30% and 100% to experiment with. It turns out that for both models, by raising the amount of leveraged data, the performance of a model can be improved significantly. Finetuned Llama-13B-chat's JGA on MultiWOZ 2.2 is enhanced by 6 percent and 8 percent on MultiWOZ 2.4. Finetuned Falcon-7B's JGA is enhanced by 4 percent on MultiWOZ 2.2 and also 4 percent on MultiWOZ 2.4. This is a substantial improvement because in order to enhance JGA, the model needs to generate correct slot values for all slots in a dialogue. It is possible that although the accuracy of the model is satisfying, errors are distributed in different dialogues, making the JGA score less desirable. On the other hand, it is difficult to increase the precision and recall and to finally increase the slot F1. We observe only around 1 to 2 percent of increment in F1 in all case except for Falcon-7b finetuned on MultiWOZ 2.4. Falcon-7b finetuned on MultiWOZ 2.4 can obtain an increment of 4 percent probably due to the cleanness of the corpus so that the model can better leverage the large size of the corpus.

5. Discussion

5.1. Hallucination Puzzle on JSON Format Generation

Hallucination indicates the behavior in which the model speaks false knowledge as if it is accurate. In other words, the model may generate incorrect, nonsensical, and unreal output. Empirically, this phenomenon is more likely to occur when generating long text.

In our experiments, we noticed very rare hallucination occurs when generating JSON format. It embodies and repeatedly generates some slot-value pairs. One example can be shown in the figure below:

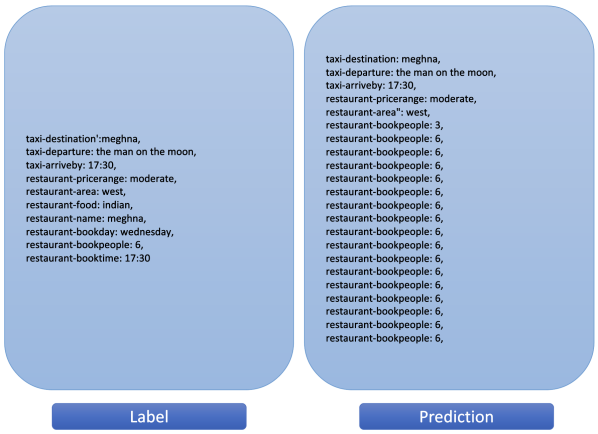

Figure 6: Hallucination may occur as one slot-value-pair repeatedly being generated

5.2. FP16 Training Collapse

Initially, Llama2 was pretrained using BF16 precision. However, the computation resources on the Greene HPC platform, including RTX8000 and V100, do not support training in such precision. One solution was to use the FP16 precision as a substitute. However, the BF16 precision is not completely compatible with the FP16 precision, causing overflow from time to time. It was observed that overflow may happen on the 30th and 31st decoder layers of llama-2. The 30th layer, which was an MLP, output infinite values. Then in the next layer, which was a layer normalization, while calculating the standard deviation, those infinite values were taken and 0's were consequently produced. Those 0 values took part in the multiplication with hidden states, producing tons of nan values. Therefore, we observed that during finetuning on Llama2 using the FP16 precision, the evaluation loss would become nan and the finetuning process could not move forward properly. One possible solution was to use clamp in the forward function in order to avoid overflow. However, that significantly slowed down the finetuning. Another option was to give up the 16-bit precision and use the 32-bit precision at a small cost of training speed.

5.3. Negative Downsampling

In the MultiWOZ corpus, one dialogue turn sample will be assigned to one specific domain and there will be a certain amount of potential slots to be filled in. When using the QA format, one

dialogue turn sample could be split into several QA samples depending on how many slots the domain had. In the initial corpus, there were around 40k turns while the corpus in the QA format would expand in size to as large as 600k samples, which was a great drain on the computation time. In fact, the QA took a lot of negative samples, of which the slot values were just none, into account. Therefore, we hope to downsample the negative samples in order to, first, keep a reasonable balance between positive samples and negative samples and, second, downscale the corpus to make the finetuning more efficient. However, it turned out that a model, no matter whether the base model was Llama2 or Falcon, as long as it was finetuned on a downsampled corpus, would produce a high recall and a low precision, meaning it was overconfident about the slots and overestimated the number of positive samples in the development set. Subsequently, we chose to give up the downsampling technique in the end, which turned out to be the right direction because once we gave up negative downsampling, we immediately obtained well-balanced precisions and recalls.

5.4. Post-processing Correction

We conduct the post-process step to normalize the model-generated content and introduce flexibility to the final estimations. Some representative generations differ from the ground truth can be corrected through normalization. For instance, the timestamp generation "3 pm" is equal to "15:00", abbreviations "St Catherines college" should be equal to "Saint Catherines college". Normalization also corrects typos in the dataset, for example, "Shanghi family restaurant" in the label should be "Shanghai family restaurant" instead. We manually identified normalization cases in the test and validation dataset.

5.5. Resource Limitation

The problem of resource scarcity notably barricades our training process and plan, especially when it comes to the context of training Large Language Models. One significant issue is the extended queue time for accessing computational resources, such as large memory allocation and graphics processing units (GPUs). Our models' find-tuning calls for a high demand for GPU resources, and even with the NYU Greene HPC cluster, the best supercomputer in the top university, we still need to queue for a significantly long time to run a GPU job.

Personal Contribution

Bingsen Chen mainly contributed to the slot-level QA formulation and experiments with Llama-2 and Llama-2-chat. He initially processed the MultiWOZ 2.2 benchmark datasets into instruction finetuning format for everyone and wrote evaluation scripts for both MultiWOZ 2.2 and 2.4. He ran all the experiments with Llama2 and Llama-2-chat in slot-QA format. Then, he proposed the unlikelihood training method as described in Appendix A.5. However, its performance is not as competitive so it is not included in the main text of the report. Also, he investigated and reported the FP16 training collapse as described in Section 5.2. Bingsen Chen also managed the GitHub repository for this Capstone project.

6. Conclusion

In conclusion, this project proposes the possibility of leveraging the understanding power of large language models to tackle the DST problem, improving the Joint Goal Accuracy (JGA) to a higher level, which has long been an unsolved problem. By finetuning three current large language models, including Llama2, Llama2-chat, and Falcon, using Parameter-Efficient Fine-Tuning (PEFT) within the computation capacity of resources that we can have access to, we ultimately obtained a finetuned Falcon-7b which outperformed the baselines on MultiWOZ 2.2 and a finetuned Llama2-13b-chat, which outperformed the baselines both on MultiWOZ 2.2 and on MultiWOZ 2.4. We investigated the validity of QA and JSON output formulation and decided to use the QA formulation since the model proved to perform better using this formulation. We also studied how model scale, LoRA scaling and data scaling affected the model performance in the DST task and concluded that increasing the model size, enlarging the LoRA rank, and enlarging the corpus are all effective methods to enhance model performance for the DST task.

On the other hand, we document the difficulties that we have confronted during the research, including the Hallucination Puzzle on JSON Format Generation, the FP16 Training Collapse, the negative effect of negative downsampling, problematic labels in the MultiWZO ground truth. We also document the anomaly of the decrement of finetuned Llama-2-chat, which leaves space for future research.

This project serves as a starting point for further research leveraging language models to handle the task of DST, Having witnessed the effectiveness of this method, further work can follow this trajectory to try other large language models and hack better data formulations and training

configurations in order to further push the state-of-the-art in this field.

References

[1] C.-S. Wu, S. Hoi, R. Socher, and C. Xiong, "Tod-bert: Pre-trained natural language understanding for task-oriented dialogue," 2020.

[2] T. Yu, R. Zhang, O. Polozov, C. Meek, and A. H. Awadallah, "Score: Pre-training for context representation in conversational semantic parsing," in *International Conference on Learning Representations*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:235613495

[3] S. Li, S. Yavuz, K. Hashimoto, J. Li, T. Niu, N. Rajani, X. Yan, Y. Zhou, and C. Xiong, "Coco: Controllable counterfactuals for evaluating dialogue state trackers," *arXiv preprint arXiv:2010.12850*, 2020.

[4] J. Guo, K. Shuang, J. Li, Z. Wang, and Y. Liu, "Beyond the granularity: Multi-perspective dialogue collaborative selection for dialogue state tracking," in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland: Association for Computational Linguistics, May 2022, pp. 2320–2332. [Online]. Available: https://aclanthology.org/2022.acl-long.165

[5] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *J. Mach. Learn. Res.*, vol. 21, pp. 140:1–140:67, 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:204838007

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[7] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[8] J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu, and D. Amodei, "Scaling laws for neural language models," 2020.

[9] J. Wei, Y. Tay, R. Bommasani, C. Raffel, B. Zoph, S. Borgeaud, D. Yogatama, M. Bosma, D. Zhou, D. Metzler, E. H. Chi, T. Hashimoto, O. Vinyals, P. Liang, J. Dean, and W. Fedus, "Emergent abilities of large language models," 2022.

[10] J. Wei, M. Bosma, V. Y. Zhao, K. Guu, A. W. Yu, B. Lester, N. Du, A. M. Dai, and Q. V. Le, "Finetuned language models are zero-shot learners," 2022.

[11] A. Chowdhery, S. Narang, J. Devlin, M. Bosma, G. Mishra, A. Roberts, P. Barham, H. W. Chung, C. Sutton, S. Gehrmann, P. Schuh, K. Shi, S. Tsvyashchenko, J. Maynez, A. Rao, P. Barnes, Y. Tay, N. Shazeer, V. Prabhakaran, E. Reif, N. Du, B. Hutchinson, R. Pope, J. Bradbury, J. Austin, M. Isard, G. Gur-Ari, P. Yin, T. Duke, A. Levskaya, S. Ghemawat, S. Dev, H. Michalewski, X. Garcia, V. Misra, K. Robinson, L. Fedus, D. Zhou, D. Ippolito, D. Luan, H. Lim, B. Zoph, A. Spiridonov, R. Sepassi, D. Dohan, S. Agrawal, M. Omernick, A. M. Dai, T. S. Pillai, M. Pellat, A. Lewkowycz, E. Moreira, R. Child, O. Polozov, K. Lee, Z. Zhou, X. Wang, B. Saeta, M. Diaz, O. Firat, M. Catasta, J. Wei, K. Meier-Hellstern, D. Eck, J. Dean, S. Petrov, and N. Fiedel, "Palm: Scaling language modeling with pathways," 2022.

[12] L. Jacqmin, L. M. Rojas-Barahona, and B. Favre, ""do you follow me?": A survey of recent approaches in dialogue state tracking," in *SIGDIAL Conferences*, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:251196750

[13] M. Eric, R. Goel, S. Paul, A. Kumar, A. Sethi, P. Ku, A. K. Goyal, S. Agarwal, S. Gao, and D. Hakkani-Tur, "Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines," 2019.

[14] I. V. Serban, A. Sordoni, Y. Bengio, A. Courville, and J. Pineau, "Building end-to-end dialogue systems using generative hierarchical neural network models," 2016.

[15] C.-S. Wu, A. Madotto, E. Hosseini-Asl, C. Xiong, R. Socher, and P. Fung, "Transferable multi-domain state generator for task-oriented dialogue systems," 2019.

[16] L. Zhou and K. Small, "Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering," 2020.

[17] M. Seo, A. Kembhavi, A. Farhadi, and H. Hajishirzi, "Bidirectional attention flow for machine comprehension," 2018.

[18] L. Chen, B. Lv, C. Wang, S. Zhu, B. Tan, and K. Yu, "Schema-guided multi-domain dialogue state tracking with graph attention neural networks," *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 05, p. 7521–7528, 2020.

[19] H. Le, R. Socher, and S. C. H. Hoi, "Non-autoregressive dialog state tracking," 2020.

[20] C.-H. Lee, H. Cheng, and M. Ostendorf, "Dialogue state tracking with a language model using schema-driven prompting," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, Nov. 2021, pp. 4937–4949. [Online]. Available: https://aclanthology.org/2021.emnlp-main.404

[21] E. Hosseini-Asl, B. McCann, C.-S. Wu, S. Yavuz, and R. Socher, "A simple language model for task-oriented dialogue," 2022.

[22] M. Heck, C. van Niekerk, N. Lubis, C. Geishauser, H.-C. Lin, M. Moresi, and M. Gašić, "Trippy: A triple copy strategy for value independent neural dialog state tracking," 2020.

[23] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019. [Online]. Available: https://api.semanticscholar.org/CorpusID:160025533

[24] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. J. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," *ArXiv*, vol. abs/2005.14165, 2020. [Online]. Available: https://api.semanticscholar.org/CorpusID:218971783

[25] B. Peng, C. Li, J. Li, S. Shayandeh, L. Lidén, and J. Gao, "Soloist: Building task bots at scale with transfer learning and machine teaching," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 807–824, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:236937204

[26] Y. Su, L. Shu, E. Mansimov, A. Gupta, D. Cai, Y.-A. Lai, and Y. Zhang, "Multi-task pre-training for plug-and-play task-oriented dialogue system," in *Annual Meeting of the Association for Computational Linguistics*, 2021. [Online]. Available: https://api.semanticscholar.org/CorpusID:238226978

[27] J. Zhao, R. Gupta, Y. Cao, D. Yu, M. Wang, H. Lee, A. Rastogi, I. Shafran, and Y. Wu, "Description-driven task-oriented dialog modeling," *ArXiv*, vol. abs/2201.08904, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246240888

[28] S. Won, H. Kwak, J. Shin, J. Han, and K. Jung, "BREAK: Breaking the dialogue state tracking barrier with beam search and re-ranking," in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Toronto, Canada: Association for Computational Linguistics, Jul. 2023, pp. 2832–2846. [Online]. Available: https://aclanthology.org/2023.acl-long.159

[29] X. Zang, A. Rastogi, S. Sunkara, R. Gupta, J. Zhang, and J. Chen, "Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines," in *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, ACL 2020*, 2020, pp. 109–117.

[30] F. Ye, J. Manotumruksa, and E. Yilmaz, "Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation," 2022.

[31] P. Budzianowski, T.-H. Wen, B.-H. Tseng, I. Casanueva, S. Ultes, O. Ramadan, and M. Gašić, "Multiwoz – a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling," 2020.

[32] H. Touvron, L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, S. Batra, P. Bhargava, S. Bhosale, D. Bikel, L. Blecher, C. C. Ferrer, M. Chen, G. Cucurull, D. Esiobu, J. Fernandes, J. Fu, W. Fu, B. Fuller, C. Gao, V. Goswami, N. Goyal, A. Hartshorn, S. Hosseini, R. Hou, H. Inan, M. Kardas, V. Kerkez, M. Khabsa, I. Kloumann, A. Korenev, P. S. Koura, M.-A. Lachaux, T. Lavril, J. Lee, D. Liskovich, Y. Lu, Y. Mao, X. Martinet, T. Mihaylov, P. Mishra, I. Molybog, Y. Nie, A. Poulton, J. Reizenstein, R. Rungta, K. Saladi, A. Schelten, R. Silva, E. M. Smith, R. Subramanian, X. E. Tan, B. Tang, R. Taylor, A. Williams, J. X. Kuan, P. Xu, Z. Yan, I. Zarov, Y. Zhang, A. Fan, M. Kambadur, S. Narang, A. Rodriguez, R. Stojnic, S. Edunov, and T. Scialom, "Llama 2: Open foundation and fine-tuned chat models," 2023.

[33] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin, C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. E. Miller, M. Simens, A. Askell, P. Welinder, P. F. Christiano, J. Leike, and R. J. Lowe, "Training language models to follow instructions with human feedback," *ArXiv*, vol. abs/2203.02155, 2022. [Online]. Available: https://api.semanticscholar.org/CorpusID:246426909

[34] E. Almazrouei, H. Alobeidli, A. Alshamsi, A. Cappelli, R. Cojocaru, M. Debbah, Étienne Goffinet, D. Hesslow, J. Launay, Q. Malartic, D. Mazzotta, B. Noune, B. Pannier, and G. Penedo, "The falcon series of open language models," 2023.

[35] E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang, and W. Chen, "Lora: Low-rank adaptation of large language models," 2021.

[36] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," 2023.

[37] N. Bang, J. Lee, and M.-W. Koo, "Task-optimized adapters for an end-to-end task-oriented dialogue system," 2023.

A. Appendix

A.1. Tables of Comparison of JSON and Slot-level QA

	MultiWOZ-2.2				MultiWOZ-2.4			
	JGA	Slot-f1	Precision	Recall	JGA	Slot-f1	Precision	Recall
QA	63.38%	91.93%	94.81%	91.33%	67.51%	93.36%	96.62%	92.00%
JSON	62.33%	88.27%	91.68%	87.84%	65.99%	89.84%	93.65%	88.53%

Table 4: Base Model: Llama2-7b

	MultiWOZ-2.2				MultiWOZ-2.4			
	JGA	Slot-f1	Precision	Recall	JGA	Slot-f1	Precision	Recall
QA	65.96%	93.15%	92.83%	95.31%	72.55%	95.18%	95.14%	96.44%
JSON	61.81%	87.81%	91.53%	87.24%	64.90%	89.12%	93.24%	87.71%

Table 5: Base Model: Llama2-7b-chat

A.2. Table of Model Scaling Experiments

	MultiWOZ-2.2				**MultiWOZ-2.4**			
Base Model	JGA	Slot F1	Precision	Recall	JGA	Slot F1	Precision	Recall
llama-7b	58.65%	90.26%	93.66%	89.47%	62.61%	91.92%	95.71%	90.32%
llama-13b	67.00%	93.20%	93.97%	94.23%	73.52%	95.16%	96.23%	95.27%
llama-7b-chat	63.37%	92.63%	92.14%	95.12%	69.75%	94.67%	94.46%	96.26%
llama-13b-chat	67.37%	93.50%	94.20%	94.62%	74.07%	95.51%	96.49%	95.71%

Table 6: lora_r=16, training on 30% data

A.3. JSON Prompt

Given an input dialogue between the user and the assistant, look for information related to specified slots, **{placeholder for slot description and possible values}**. Respond in JSON format, providing key-value pairs for each mentioned slot. If the slot is not mentioned in the dialogue, exclude it from the response. Ensure the correct input format for the dialogue and specify the expected JSON response format. Note that the order of key-value pairs can be ignored.

A.4. Q&A

Non-Categorical Prompt: Based on the input dialogue between the user and the assistant, answer **{placeholder for slot description}**. If it's not mentioned in the dialogue, please answer NONE.

Categorical Prompt: Based on the input dialogue between the user and the assistant, choose the correct answer for **{placeholder for slot description}** from **{placeholder for possible values}**. If it's not mentioned in the dialogue, please choose NONE.

A.5. Unlikelihood Training

For a casual language model π_θ, we use it to generate the value $\hat{y} = (\hat{y}_1, \hat{y}_2, \ldots$
$, \hat{y}_T)$ for a given slot, where each $\hat{y}_i$ is a random variable that corresponds to a token. The ground

truth value is $y = (y_1, y_2, \ldots, y_T)$. We pre-computed B beam search candidates $\{\tilde{y}_i\}_{i=1}^{B}$, where each $\tilde{y}_i = (\tilde{y}_{i,1}, \tilde{y}_{i,2}, \ldots, \tilde{y}_{i,\tilde{T}_i})$. We define our loss function as

$$\mathcal{L}(\theta) = -\sum_{t=1}^{T} \ln \mathbb{P}(\hat{y}_t = y_t \mid p, y_{<t}) + \frac{\gamma}{B} \sum_{i=1}^{B} \sum_{t=1}^{\tilde{T}_i} w_{i,t} \cdot \ln \mathbb{P}(\hat{y}_t = \tilde{y}_{i,t} \mid p, \tilde{y}_{i,<t})$$

In this formula, γ is a coefficient to control the unlikelihood penalty. p is the instruction prompt. $w_{i,t}$ is a binary mask defined as

$$w_{i,t} = \begin{cases} 0, & \tilde{y}_{i,<t} = y_{<t} \text{ and } \tilde{y}_{i,t} = y_t \\ 1, & \tilde{y}_{i,<t} \neq y_{<t} \text{ or } \tilde{y}_{i,t} \neq y_t \end{cases}$$

In other words, when tokens in the beam candidate start to deviate from the ground truth answer, we should penalize the model for generating all the subsequent tokens.