Context-Aware Language Modeling for Goal-Oriented Dialogue Systems

Charlie Snell **Mengjiao Yang** **Justin Fu** **Yi Su** **Sergey Levine**
UC Berkeley
{csnell22,sherryy,justinfu,suyi,svlevine}@berkeley.edu

Abstract

Goal-oriented dialogue systems face a trade-off between fluent language generation and task-specific control. While supervised learning with large language models is capable of producing realistic text, how to steer such responses towards completing a specific task without sacrificing language quality remains an open question. In this work, we formulate goal-oriented dialogue as a partially observed Markov decision process, interpreting the language model as a representation of both the dynamics and the policy. This view allows us to extend techniques from learning-based control, such as task relabeling, to derive a simple and effective method to finetune language models in a goal-aware way, leading to significantly improved task performance. We additionally introduce a number of training strategies that serve to better focus the model on the task at hand. We evaluate our method, Context-Aware Language Models (CALM), on a practical flight-booking task using AirDialogue. Empirically, CALM outperforms the state-of-the-art method by 7% in terms of task success, matching human-level task performance.

1 Introduction

Dialogue systems have typically approached the problem of generating realistic dialogue from the perspective of supervised learning (Dušek and Jurcicek, 2016; Eric and Manning, 2017; Mei et al., 2017; Chen et al., 2019; Wu et al., 2019a; Hosseini-Asl et al., 2020; Peng et al., 2020; Adiwardana et al., 2020). However, dialogue can also be viewed as a sequential decision making process, which is well-suited to planning and reinforcement learning (RL) algorithms. A challenge with the classical RL approach to dialogue is the requirement for active interaction with humans (Gašić et al., 2011). Training such a system with active human-in-the-loop interaction quickly becomes expensive and cumbersome, making it desirable to develop techniques for goal-directed training of dialogue systems that can effectively leverage offline data.

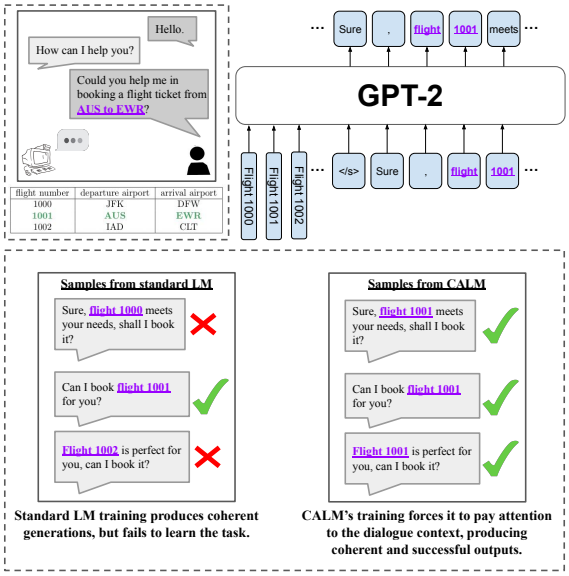

Figure 1: **CALM is an end-to-end language model for goal oriented dialogue.** CALM's training objective teaches the model to better pay attention to the dialogue task context, yielding a ∼50% improvement in task success over standard LM training on a flight booking task.

While many dialogue generation techniques based on RL and learned control have been proposed (Eckert et al., 1997; Levin et al., 2000; Chung, 2004; Georgila et al., 2006; Schatzmann et al., 2007; Heeman, 2009; Georgila and Traum, 2011), most such systems take a pipelined approach, where an abstract representation of states and actions is designed by hand and then combined with RL to train a "dialogue management" system, rather than generating dialogue end-to-end. These pipelined approaches rely on a manually designed decomposition of the dialogue task, which may be domain-specific and, more importantly, may not enjoy all of the benefits of tightly integrating low-level text generation with the overall goals of the

Code at https://sea-snell.github.io/CALM_LM_site/

Findings of the Association for Computational Linguistics: NAACL 2022, pages 2351 - 2366

task. In this work, we instead ask: how can we scalably and effectively introduce the mechanisms of goal-directed decision making into end-to-end language models, to directly steer language generation toward completing specific dialogue tasks rather than simply generating probable responses?

To this end, rather than utilizing a pipelined approach, we aim to directly finetune language models in a task-aware manner such that they can maximize a given utility function. We observe that large language models can already be formulated within a Markov decision processes (MDP) as capturing both the dynamics and policy for a decision-making task, where dialogue history serves as state, and the agent's utterances serve as actions. We could utilize this observation by finetuning the models directly with online RL, but the need for human-in-the-loop training makes this difficult. Offline RL methods (Levine et al., 2020; Fujimoto et al., 2019; Wu et al., 2019b; Wang et al., 2020b) provide an alternative approach, but typically require value function estimation, which is not straightforward to perform with a language model. Instead, we propose a conditional imitation learning strategy coupled with a novel task relabeling approach that can finetune language models from offline data, such that the model still represents the joint distribution over dialogues, but *tilts* this distribution toward dialogues with a high reward. This amounts to a *task-aware* finetuning strategy that integrates task information into the model. The main contribution of our work is CALM (*Context-Aware Language Modeling*), a framework for end-to-end goal-directed dialogue generation. CALM unifies the traditional language modeling objective with task-specific supervision, where a language model is interpreted as a joint representation of dynamics and policies in an MDP, and the finetuning process utilizes a conditional imitation learning objective with a novel task relabeling strategy that teaches the model how to generate high-utility dialogues (see Figures 1 and 2). Because CALM interprets the language model as *both* a dynamics model and a policy, it can be used as either a model-free method, where the dynamics are discarded and the policy component is used to greedily generate responses, or as a *model-based* method, where the dynamics component can be used to *plan* at test-time. We empirically evaluate CALM on AirDialogue (Wei et al., 2018), the largest dataset for goal-oriented dialogue based-on a flight-booking task. CALM improves the task success by 10% over the previous state-of-the-art method (Chen et al., 2020) following the evaluation protocol proposed by Wei et al. (2018), achieving the first-ever human-level performance on this dataset.

2 Related Work

Our goal is to enable end-to-end training of goal-directed dialogue agents. In these settings, an agent aims to complete a particular task with its utterances (Smith and Hipp, 1994). Goal-directed agents have been explored in contexts such as personal assistants (McTear, 2002; Budzianowski et al., 2018; Williams et al., 2014), recommendation systems (Liu et al., 2010; Kang et al., 2019), education (Yuan et al., 2008), and negotiation (He et al., 2018; Lewis et al., 2017). While there are multiple approaches to constructing dialogue agents, in this work we frame the problem of generating dialogue as a sequential decision making problem within a (partially observed) Markov Decision Process (MDP) (Singh et al., 1999; Young et al., 2013). Prior works that utilize such an MDP formulation typically aim to train a dialogue management system (Singh et al., 2002), in which the agent reasons about higher-level abstractions of the state of the conversation, and language generation is performed using a downstream procedure. Dialogue management systems have been trained using techniques such as online reinforcement learning via policy gradients (Gašić et al., 2011; He et al., 2018), off-policy reinforcement learning (Pietquin et al., 2011; Yu et al., 2016) or actor-critic methods (Su et al., 2017). Our method differs from dialogue management systems in that CALM is an end-to-end system optimized for successful task completion, and performs both high-level decision making and language generation.

Recent advancements in language models, such as recurrent neural networks (Sundermeyer et al., 2012; Asri et al., 2016; Su et al., 2016; Zhao et al., 2019; Wang et al., 2020a; Zhang et al., 2020) and attention-based architectures (Vaswani et al., 2017; Liu et al., 2019; Devlin et al., 2018; Brown et al., 2020), have spurred increasing interest in such end-to-end dialogue systems (Hosseini-Asl et al., 2020; Peng et al., 2020; Adiwardana et al., 2020). Model-based approaches, in which a learned agent is substituted for a human, allow learning to be done entirely within simulation without human intervention (Li et al., 2016; He et al., 2018; Kang et al., 2019; Lewis et al., 2017; Liu et al., 2018). In con-

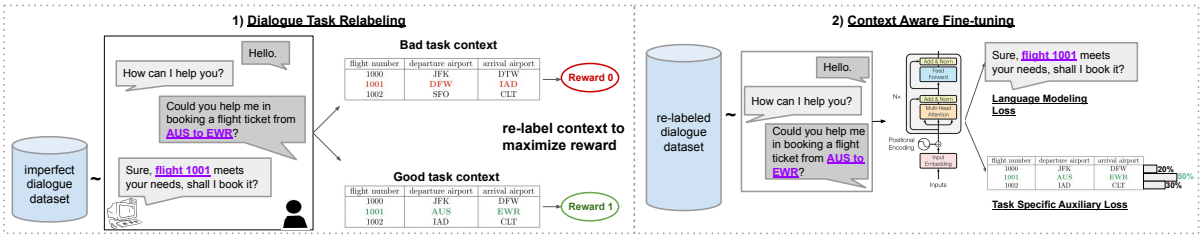

Figure 2: **A visual outline of CALM.** We apply Task Relabeling to our static offline dataset, by swapping out the task context — in this case a flight table — such that the attached dialogue becomes an optimal example of task completion. When fine-tuning on this relabeled dataset, we then apply a Task Specific Auxiliary Loss on top of the standard language modeling objective; this helps the model learn to use the task context. Once trained, CALM can consistently solve goal-directed dialogue tasks.

trast to these approaches, CALM augments the traditional language modeling objective with task-specific rewards in order to finetune a model that is more aware of task goals, which significantly improves performance over a naïve language model without the need for simulating human responses in an interactive training loop. Jaques et al. (2019) recently proposed a model-free, offline approach to undirected dialogue, or dialogue without a specific task goal. Our method differs in that we aim to solve goal-oriented dialogue which allows us to optimize task-specific objectives, and that we take a model-based RL approach which enables us to leverage fine-tuned language models.

3 Preliminaries

In this section, we review our notation and problem formulation for casting dialogue within a sequential decision making framework.

POMDP formulation. We formulate dialogue generation as a partially observable Markov decision process (POMDP) (Kaelbling et al., 1998), with a state that consists of known and unknown context information about the task. Let $c_h \in \mathcal{C}^{(h)}$ denote the hidden context for the task, and let $c_o \in \mathcal{C}^{(o)}$ denote the observed context. For instance, in a flight booking task, a table of available flights might correspond to c_o, while the particular flight that the human wants to book, which is unknown to the agent, corresponds to c_h. Note that the reward, which requires booking the right flight, depends on both hidden and observed contexts. We can define such an environment as a POMDP $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mu_0, \mathcal{R}, \gamma)$. We denote a conversation τ as $\tau := \{a_0, e_0, ..., a_T\}$, where T denotes the number of turns in a conversation and a_t and e_t represent utterances (strings of tokens) from the dialogue agent (a_t) and the human (e_t) at the t-th turn, respectively. We additionally use $\tau_{<t}$ to denote conversation history up to the t-th turn. We can represent the underlying POMDP state $s_t \in \mathcal{S}$ as the concatenation of both of the contexts and the previous conversation history $s_t := \{c_h, c_o, \tau_{<t}\} = \{c_h, c_o, a_0, e_0, ..., a_{t-1}, e_{t-1}\}$. However, we only observe the last two elements of the state tuple, such that our observation $o_t \in \mathcal{O}$ at the t-th conversation turn is $o_t = \{c_o, \tau_{<t}\}$. An action $a_t \in \mathcal{A}$ is the agent's response to the current state s_t. Given our definition of the state, the full conversation in a dialogue can be conveniently represented by the last observation and action, $\{o_T, a_T\}$. An agent $\pi : \mathcal{O} \to \mathcal{P}(\mathcal{A})$ maps observations to sets of probability measures over the action space $\mathcal{P}(\cdot)$. A transition function $\mathcal{T}(\cdot|s_t, a_t)$, represents a distribution over the human's utterances, returning s_{t+1} as the state at turn $t+1$. We only consider the sparse reward setting with $r_T = R(s_T, a_T) \in \{0, 1\}$ denoting task completion, and $r_t = 0, \forall t < T$. Our final reward is therefore dependent on both the context and the dialogue: $R(s_T, a_T) = R(\tau, c_h, c_o)$, where the context $\{c_o, c_h\}$ is randomly sampled for each dialogue from some initial distribution μ_0.

Goal-oriented dialogue. Goal-oriented dialogue systems aim to maximize the expected reward of the above POMDP

$$\mathbb{E}_{\{c_o, c_h\} \sim \mu_0, \pi, \mathcal{T}}[\sum_{t=0}^{T} \gamma^t R(s_t, a_t)], \qquad (1)$$

where $\{c_h, c_o\}$ is sampled from distribution μ_0. *On-policy* RL algorithms optimize this objective via environment interaction, which is represented by a real human. However, because human-in-the-loop training is expensive, we pursue an *offline learning* approach where we are given a fixed dataset and there is no further interaction with the human in the learning process. This dataset is composed of n trajectories with $\mathcal{D}^{\text{off}} = \{c_h^{(i)}, c_o^{(i)}, \tau^{(i)}, r^{(i)}\}_{i=1}^{n}$ with each $\tau^{(i)} = \{a_0^{(i)}, e_0^{(i)}, , ..., a_T^{(i)}\}$ and its corresponding final reward for task completion $r^{(i)}$. Our

goal is to learn the policy $\pi(a|o)$ which improves the dialog agent's ability in achieving the highest task reward defined in Equation 1.

Language models. While conventionally a language model is seen simply as a sequence model over tokens of the form $\prod_{t=1}^{T} p(x_{t+1}|x_{1:t})$, when the sequence $x_{1:T}$ corresponds to a dialogue trajectory τ, we can also interpret a language model as learning the distribution over τ. This distribution can be factored into the product of the policy $\pi(a_t|\tau_{<t})$ and the dynamics $\mathcal{T}(\tau_{<t+1}|\tau_{<t}, a_t)$, and so we can say that a language model *also* represents the policy and the dynamics. Therefore, the maximum likelihood objective for training or finetuning a language model on a dialogue dataset $\mathcal{D}^{\text{off}}$ consisting of dialogue trajectories τ can be written as

$$\mathcal{L}_{LM}(\theta) = \max_{\theta} \mathop{\mathbf{E}}_{\tau\sim\mathcal{D}^{\text{off}}} \sum_{t=1}^{T} \Big(\log \pi_\theta(a_t|\tau_{<t}) + \log \mathcal{T}_\theta(\tau_{<t+1}|\tau_{<t}, a_t) \Big), \quad (2)$$

where $\pi_\theta(a_t|o_t)$ represents a policy that generates new dialogue based on the observed context and dialogue history, and $\mathcal{T}_\theta(\tau_{<t+1}|\tau_{<t}, a_t)$ represents the observed dynamics characterizing human responses, and θ denotes parameters in π and $\mathcal{T}$. Note that $\tau_{<t}$ consists only of the conversation history, and does not contain any task-specific context. A naïve approach to train dialogue systems is to jointly parameterize both π and $\mathcal{T}$ as one language model, and optimize Equation 2 on pre-collected conversations $\mathcal{D}^{\text{off}}$. This method corresponds to behavioral cloning (BC) (Pomerleau, 1989).

Context conditioning. While an agent trained using Equation 2 can learn policies and dynamics that imitate human conversations, this objective does not incorporate the task goal, and may not produce a policy that is more performant than the dataset $\mathcal{D}^{\text{off}}$. While it is possible to input c_o into the language model to maximize the conditional probability of $P(\tau|c_o)$ using a conditional version of the language modeling objective, $\mathcal{L}_{CTX}(\theta)$,

$$\mathcal{L}_{CTX}(\theta) = \max_{\theta} \mathop{\mathbf{E}}_{(\tau,c_o)\sim\mathcal{D}^{\text{off}}} \sum_{t=1}^{T} \Big(\log \pi_\theta(a_t|\tau_{<t}, c_o) + \log \mathcal{T}_\theta(o_{t+1}|\tau_{<t}, a_t, c_o) \Big), \quad (3)$$

contexts with particular task structures (e.g., a set of entries in a table) may not be simply processed as a sequence similarly to τ. Additionally, the language model is not pretrained to read structured context, and oftentimes the recent dialogue history is much more predictive of the next utterance than the task context is. As a result, language models can ignore the task context and only learn $P(\tau)$ despite being conditioned on c_o. Our approach builds on this conditional modeling approach, but makes a number of improvements that allow it to be more aware of the context information, which attains significantly better results in our experiments.

4 Context-Aware Language Modeling

In this section, we present our method for goal-oriented dialogue systems, Context-Aware Language Modeling (CALM). CALM interprets a language model as a combination of a policy and a dynamics model in the POMDP formulation of a dialogue task, as described in Section 3. Under this interpretation, naïve supervised finetuning on the dialogue dataset can be viewed as behavioral cloning (BC) (Pomerleau, 1989). However, BC only imitates data and does not necessarily produce a *good* policy in terms of completing tasks. We propose to improve the policy by utilizing a task relabeling strategy (described in Section 4.1), analogous to prior task relabeling approaches (Kaelbling, 1993; Andrychowicz et al., 2017; Pong et al., 2018; Savinov et al., 2018; Ghosh et al., 2019; Lynch et al., 2020; Eysenbach et al., 2020). This relabeling procedure augments the data with examples of near-optimal utterances, making the language model more task-aware. However, we find several shortcomings with this approach alone and propose the following improvements. First, an expressive language model is liable to ignore the task context, which we address by proposing an auxiliary loss (Section 4.2) that forces the model to utilize this information. Second, learning from structured task information is difficult and can result in models that fail to capture complex task structure, so we propose a task pre-training procedure to improve the learnability (Section 4.3). Finally, to further improve performance we use a model-based planning procedure (Section 4.4) on top of the proposed method that samples multiple dialogues in parallel and selects the most promising candidates.

4.1 Dialogue Task Relabeling

$\mathcal{L}_{CTX}(\theta)$ defines a context-conditional maximum likelihood objective for training an expert imitation policy in conjunction with a dynamics model. However, simply imitating all the dialogue data does not necessarily produce the best possible policy. We would like to learn a policy that pro-

duces dialogue that is more optimal, in the sense of better maximizing the task utility, than the average dialogue in the dataset. Task relabeling enables us to learn from optimal trajectories without simply filtering the dataset for high-reward trajectories, which would unnecessarily discard potentially informative data. In the case of dialogue, we can perform task relabeling by considering the context $\{c_o, c_h\}$ as defining the task. While a given dialogue may be unsuccessful for the context for which it was collected, it could be considered successful under a different context. In this case, we can simply swap out $\{c_o, c_h\}$ to create optimal task examples from the many sub-optimal examples provided by $\mathcal{D}^{\text{off}}$. Since our reward $R(c_h, c_o, \tau)$ is a function of the dialogue and context, we can modify the reward for a given dialogue just by changing the given observed context c_o. Using this observation, we can relabel unsuccessful dialogues with successful ones, and even for already successful dialogues there may be multiple c_o corresponding to task success, allowing us to augment the number of successful (c_h, c_o, τ) tuples.

Formally, since our POMDP includes a prior distribution over contexts $\{c_h, c_o\} \sim \mu_0$, there exists a posterior $q(c_o|\tau, c_h)$ over observed contexts that correspond to optimal task completion under a given τ. We can then re-label τ to be optimal under its context by sampling a new c_o from $q(c_o|\tau, c_h)$. In practice, this sampling is performed by rejection sampling from either μ_0 or some $P(c_o|c_h)$; the latter, lower entropy distribution, can be preferred if there is a low probability of sampling valid, high-reward contexts under μ_0. Now, given any τ from an offline dataset of dialogues, we can learn from the full distribution of contexts corresponding to optimal task completion under this dialogue.

In order for this relabeling procedure not to bias our policy towards behavior that is overly-optimistic about the user's responses, it is necessary that the distribution of these responses in our dataset does not depend on the portion of the context that is relabeled. For example, relabeling the table of available flights for a flight booking task should generally be reasonable, because the user is usually unaware of the flight table. On the other hand, relabeling the desired flight would not make sense, since the user's utterances are strongly depend on this. To provide another example, in a bargaining task (Lewis et al., 2017), the agent might fail to obtain the desired item and instead get an item of lesser value. But relabeling with a context that assigns a higher value to the item received would not lead to a reasonable example, since the agent mainly received this item as a result of the user's responses rather than as a result of their own bargaining skill.

Methods based on similar principles have previously been proposed in the deep RL community for simple parametric tasks, such as goal-reaching or linearly-parameterized reward functions (Kaelbling, 1993; Andrychowicz et al., 2017; Eysenbach et al., 2020). However, the dialogue task relabeling that we employ is particularly effective in our setting, since there may be exponentially many contexts that are optimal for a given dialogue (e.g., many different flight tables for a flight booking task), in contrast to the simpler task parameterizations used in prior work, where for example only one goal might be optimal for a given trajectory (the one that is reached). As a result, this technique not only allows us to turn sub-optimal task data into optimal data, but it also allows us to greatly increase the number of optimal task examples from which we can learn, which we will show leads to a large performance improvement.

4.2 Task-Specific Auxiliary Loss

Goal-oriented dialogue generation can be viewed as learning the conditional distribution $P(\tau|c_o)$, where τ represents the generated dialogue given a specific context c_o. However when trained naïvely, language models are liable to ignore this conditioning context, instead focusing purely on the previous utterances in the dialogue. In this case, the model is effectively only learning $P(\tau)$ despite having both the capacity and the context to learn the lower-entropy conditional distribution $P(\tau|c_o)$.

While dialogue tasks are by definition carried out through natural language, there is often an abstract high-level action $\alpha_h \in \mathbf{A}$ that essentially determines the success of the task. In the case of the information retrieval task that we consider in this paper, these high-level actions correspond to deciding which database entity to retrieve for the user (e.g., suggesting a flight to the customer that meets all of their needs). While these high-level actions are theoretically learnable from correlations between the dialogue and the given context, in general, we find that learning these correlations corresponds to a relatively small decrease in dialogue entropy under the model. As a result, the model is less incentivized to learn these correlations rele-

vant to the task than the form of the dialogue. To address this issue, we incorporate an auxiliary objective into our training, which trains the model directly to predict the abstract high-level actions taken in the present dialogue. This objective effectively up-weights gradients relevant for learning the high-level actions, which further helps the model to utilize the context to solve the high-level task through dialogue.

For a given dialogue-context pair $(\tau, \{c_h, c_o\})$ and high-level action, α_h, our auxiliary objective is then simply to maximize the likelihood of the high-level actions taken in the dialogue:

$$\mathcal{C}(\phi) = \max_{\phi} \mathop{\mathbf{E}}_{(c_h, c_o, \tau, \alpha_h) \sim \mathcal{D}^{\text{off}}} \log P_\phi(\alpha_h | \tau, c_o). \tag{4}$$

Just like the language modeling objective, this classification objective is averaged over each token in the dialogue sequence. Our full training objective then becomes:

$$\max_{\theta, \phi} \mathcal{L}_{CTX}(\theta) + \beta * \mathcal{C}(\phi), \tag{5}$$

where β is a hyper-parameter and $\mathcal{L}_{CTX}(\theta)$ is the standard context-conditional language modeling objective as defined in Section 3.

4.3 Task Pretraining

As observed by Liu et al. (2021), for some structured tasks, such as table question answering, pretraining on a simplified version of the given task with a synthetic context can help the model to focus learning on the "skills" that are most relevant to utilize the task context, which leads to improved downstream task performance. We instantiate this idea in our method by pre-training our model on a simplified (dialogue-free) version of the task. Instead of simultaneously modeling all the details of the raw dialogue, as is required to learn $P(\tau|c_o)$, the key observation here is that in our case the task reward only depends on the tuple $\{c_h, c_o, a_T\}$. This enables us to effectively learn to execute the task by only modeling $P(c_h, a_T|c_o)$, without any dialogue at all. By pre-training our model to first learn this simplified distribution, we effectively focus on learning the necessary skills for completing the task. It is expected that the skills learned during this pre-training phase should also generalize and transfer when we later perform training on the real dialogue. The particular instantiation of this principle in the case of AirDialogue is described in Section D.

4.4 Model-Based Dialogue Rollouts

While the methodology discussed so far can produce effective policies, language models also represent task dynamics, as discussed in Section 3. We can leverage this fact to further improve the performance of our fine-tuned models by performing model-based planning at test-time, using both the policy and dynamics components in concert to further maximize task reward. A full dialogue trajectory can then be formed by concatenating this sampled future trajectory $\tau_{\geq t}$ with the current state of the dialogue $\tau_{<t}$ i.e., $\tau = \{\tau_{<t}, \tau_{\geq t}\}$. We perform the model-based planning by sampling k such future trajectories from the final fine-tuned model, and ranking them according to an estimated reward function $\hat{R}(\tau, c_o)$ (see Appendix E.1). Then, we improve upon the policy π from which we took the samples by taking the action (i.e., the next utterance) a_t which receives the highest estimated reward among the sampled trajectories. This rollout sampling procedure is identical to the one used by Lewis et al. (2017).

5 CALM for AirDialogue

In this section, we instantiate our proposed method, CALM, for the AirDialogue flight booking task (Wei et al., 2018). We first give an overview of the task, and then describe how to do relabeling and context conditioning on this specific task.

5.1 AirDialogue Dataset

Dataset overview. The AirDialogue dataset (Wei et al., 2018) is a recently published large-scale airline reservation dataset based on the aforementioned task. The dataset includes 402,038 conversations. The dataset involves three distinct tasks: booking, canceling, and changing flights. We describe the booking task in detail below.

Flight booking task. The (human) customer is given a set of 12 trip requirements, and the flight agent (bot) is provided with a table of 30 flights. The goal of the flight agent is to book a flight from the table for the customer which meets all their requirements, or to correctly inform them that no such flight is available. To determine task success, the flight agent must predict an explicit action at the end of the dialogue indicating the flight that was booked or inform no flight available. See Figure 7 for an example conversation from the dataset.

5.2 Processing Tables

The AirDialogue booking tasks require efficiently querying a flight table containing flight information (e.g., departing location, ticket price)

given to the agent prior to the conversation. In order to successfully complete the booking task, the agent needs to be able to filter, select, and integrate information from the flight table based on the customer's preferences inferred from the dialogue.

Instead of treating the tables as unstructured sequences (Wei et al., 2018; Jiang et al., 2021) or as SQL databases (Chen et al., 2020), CALM models tables as an observable context consisting of a set $c_o = \{f_1, f_2, f_3, ..., f_N\}$ of table rows. These rows are then input to our model as a set of embeddings (see appendix C and G for more details).

5.3 Relabeling AirDialogue with CALM

While the AirDialogue dataset only includes one flight table for each dialogue, there are potentially many flight tables compatible with each dialogue as each flight can appear in many tables. We hence implement our relabeling procedure as described in Section 4.1 as follows. We perform rejection sampling on the observable context (i.e., the table of flights) $c_o \sim q(c_o|\tau, c_h)$, sampling until we obtain a new context (c_h, c_o, τ), which gives maximum reward possible $R(\tau, c_h, c_o) = \max_{c_o} R(\tau, c_h, c_o)$. The prior distributions $p(c_o)$ and $p(c_o|c_h)$, from which the tables in the AirDialogue dataset were sampled, are provided with the dataset. By rejection sampling from $p(c_o|c_h)$, we can effectively sample from the posterior $q(c_o|\tau, c_h)$ within a certain computational budget. In this setting, c_o denotes tables and there are exponentially many tables which correspond to a task success under a given dialogue. Therefore, with our relabeling approach, we increase the number of near-optimal task examples exponentially, which makes it much easier for the language model to learn to query the flight table.

Our relabeling is approximately valid according to the condition specified in Section 4.1. While the customer does not have access to the flight table and therefore is not directly affected by our relabeling, there are still some minor edge-cases in which over-optimism about the dynamics could be learned by our policy. If for example, in the dataset the customer were to occasionally reject the first flight that we suggest, our policy may learn to assign a small probability to the action of initially offering the wrong flight, relying on them subsequently rejecting it such that we can later recover and offer the correct one. However, in practice we observe that these cases are rare in AirDialogue.

	success rate
CALM (greedy)	**0.88** ± 4e-3
LM(GPT2-small) (greedy)	0.38 ± 1e-3
AirConcierge (greedy)	0.81 ± 7e-3
CALM (planning)	**0.90** ± 2e-3
LM(GPT2-small) (planning)	0.74 ± 7e-3
Human	**0.88**

Table 1: **Comparison of our method and baselines across all tasks.** Using greedy decoding, our method matches human performance, greatly improving over baselines. Adding roll-outs (32 samples) further improves task completion.

5.4 Table Selection as Auxiliary Loss

The primary high-level action involved in AirDialogue is the decision of which flight table entry, if any, to recommend to the user. We therefore implement our auxiliary objective as a classification head on top of the language model, trained to predict the flight table entry that meets the customer's requests. Specifically, our set of high-level actions $\mathbf{A}$ is the set of flight table rows $\{f_1, f_2, f_3, ..., f_N\}$ plus an additional item f_0, corresponding to the case in which no flights meet the customer's requirements. If f^* is the flight recommended in the dialogue, then our auxiliary objective is:

$$\mathcal{C}(\phi) = \max_{\phi} \mathop{\mathbf{E}}_{(c_o,\tau)\sim\mathcal{D}^{\text{off}}} \log P_\phi(f^*|\tau, c_o). \quad (6)$$

6 Experiments

In this section, we empirically evaluate the performance of CALM on AirDialogue (Wei et al., 2018). We first show that CALM outperforms the SOTA on the AirDialogue dataset by around 7% in the standard simulated evaluation protocol proposed by Chen et al. (2020), which prior work denotes as "self-play" (see Appendix E), and this matches human-level performance as reported by Wei et al. (2018). Beyond this, we also perform a comprehensive set of ablation studies to validate the necessity of each component of CALM.

Experiment Setup and Baselines. We compare CALM on AirDialogue with two baselines. The first is AirConcierge, the previous SOTA on AirDialogue, which explicitly parses and executes SQL queries from the dialogue (Chen et al., 2020). The other is a standard language model (denoted as LM(GPT2-small)) trained on a dataset filtered for successful task examples, without any of our context-aware language modeling techniques (see Appendix Section A for more details on dataset filtering). CALM uses the fine-tuned GPT2-small model (Radford et al., 2018) as the backbone of the policy and dynamics model. After learning the

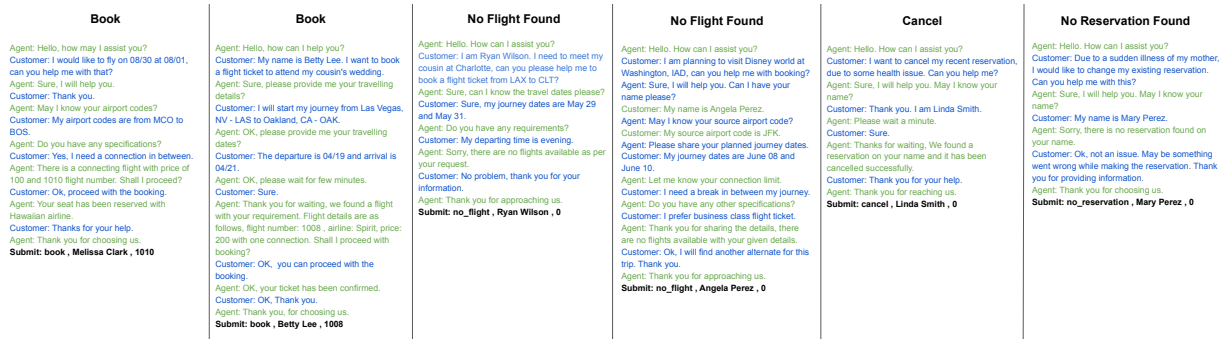

Figure 3: **Example dialogues generated by CALM (in green) in the simulated evaluation.** Despite being end-to-end, CALM produces highly coherent and sensible outputs.

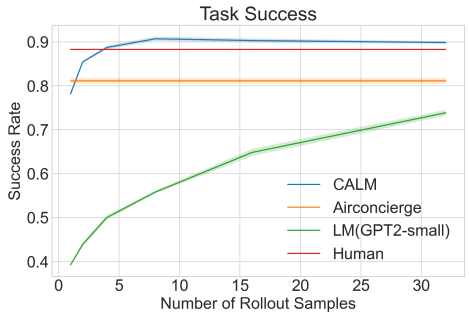

Figure 4: **Task success as a function of the number of rollout samples.** Note that successful task completion improves with more rollout samples.

dynamics model, both CALM and the LM(GPT2-small) can employ two different planning strategies: (1) a simple greedy decoding of the next utterance (equivalent to beam search with beam-width one) and (2) the rollout planning as described in Section 4.4. For AirConcierge, we only evaluate greedy decoding, as this method cannot be easily adapted for producing full rollouts. Rollout planning requires a method for predicting the reward of a given dialogue, and we describe our specific reward predictor for AirDialogue in Appendix Section E.1.

Results for Task Success. In terms of task success, CALM outperforms the prior SOTA (AirConcierge) by approximately 7%, achieving 88% task success when using greedy decoding from the language model (see Table 1). Compared with AirConcierge, where all reasoning about the task context is done outside of the language model, CALM does all of the filtering, selecting, and responding with relevant flight table entries within the language model, in a fully end-to-end manner. Meanwhile, CALM also improves over LM(GPT2-small) by 50% in terms of task success, indicating the necessity of our context-aware approach for goal-oriented tasks.

	CALM	LM(GPT2-small)	AirConcierge
Perplexity	1.63	**1.59**	-
BLEU	32.88	**35.75**	27.75

Table 2: **BLEU score and perplexity results.** CALM improves on task success without sacrificing generation quality.

We further evaluate the the performance of various methods, when utilizing the rollout planning technique. As shown in Figure 4, as the number of rollout samples increases, the performance improves for all methods. Remarkably, applying the rollout planning to CALM further increases total task success by 2%, raising it to 90% and matching human performance on the AirDialogue task. The baseline LM(GPT2-small) benefits much more from rollout planning than CALM, and we suspect that at around 90% task completion, the performance becomes bottlenecked by the customer bot's mistakes, therefore we only observe less gain from rollout planning with CALM.

Results for Language Quality. To quantitatively measure the generated language quality, we present perplexity and BLEU for all methods in Table 2. CALM performs similarly to LM(GPT2-small) and outperforms AirConcierge significantly.

Ablation Study. To examine the effectiveness of each single component in our method, we train and evaluate four ablations of CALM. Each of these ablations remove one of the components in our approach: task relabeling (Section 4.1), auxiliary loss (Section 4.2), and table pre-training (Section 4.3). Beyond this, we also examine CALM without both task relabeling and pre-training. As shown in Table 3, removing any one of these components drops task success by at least 10%, and in most cases much more than that. This shows that each piece of our method plays a critical role in helping CALM to effectively learn the goal-oriented task.

	Success Rate
CALM	**0.88** ± 4e-3
LM(GPT2-small)	0.38 ± 1e-3
CALM w/o relabel, pre-train	0.42 ± 4e-3
CALM w/o relabel	0.66 ± 1e-2
CALM w/o pre-train	0.39 ± 3e-3
CALM w/o auxiliary loss	0.78 ± 4e-3

Table 3: **Task success rate for various ablations of CALM on AirDialogue (all using greedy decoding).** Removing any single component from CALM drops performance by at least 10%.

7 Conclusion

We proposed an end-to-end framework, CALM, for goal-oriented dialogue systems. Formulating end-to-end dialogue generation as a Markov decision process, CALM employs task relabeling and context-aware finetuning to steer supervised learning of language models towards specific goals, improving task performance drastically while preserving language quality. We show that this improves performance on AirDialogue over the previous state of the art, and matches previously reported human performance under the standard simulated evaluation protocol.

CALM optimizes for task-specific measures of success, and while such measures might be comparatively simple for domains such as AirDialogue, in general specifying the right success measure or reward function may present challenges. Furthermore, as with all methods based on end-to-end language models, CALM is susceptible to internal biases and inconsistencies in the language model itself. There is for example no constraint that ensures that CALM produces *truthful* answers, or that it avoids harmful or socially unacceptable outputs. A practical deployable dialogue system would likely require additional measures to account for such issues, analogously to how learning-based methods for self-driving vehicles might require some additional safety mechanisms to ensure constraints, and indeed further research on reward specification, ensuring truthful outputs, and other constraint strategies for dialogue systems that combine language models and reward maximization is a promising and important direction.

The context-conditioned supervised learning strategy used by CALM provides for reward maximization, but is in general not optimal for arbitrary reinforcement learning problems: in general RL settings, learning a value function with dynamic programming in general can attain significantly better returns than imitating high-performing trajectories, by recombining good parts of multiple different trajectories (which might individually be suboptimal) (Kostrikov et al., 2021; Kumar et al., 2022). The simple supervised learning strategy works well in the domain we tested, but extending CALM to use value-based reinforcement learning methods is a promising direction for future work. Indeed, the improvement obtained from planning on top of the CALM model likely indicates that the supervised learning approach we employ has room for improvement. Additionally, the auxiliary objectives and relabeling strategies we employ require some amount of domain-specific design, and more general strategies could be developed in future.

Addressing these limitations in future work and developing more advanced methods that combine end-to-end language generation via large language models with concepts from reinforcement learning and planning is a promising research direction for making dialogue systems more capable, while also making language models more task aware. We hope that CALM will serve as an indication for the potential of such methods.

Acknowledgements

This research was supported by an Amazon-BAIR Commons project, as well as the Office of Naval Research. We thank Dilek Hakkani-tur, Alexandros Papangelis, Mandi Zhao, Ruiqi Zhong, and Yang Liu for advice and feedback.

References

Daniel Adiwardana, Minh-Thang Luong, David R So, Jamie Hall, Noah Fiedel, Romal Thoppilan, Zi Yang, Apoorv Kulshreshtha, Gaurav Nemade, Yifeng Lu, et al. 2020. Towards a human-like open-domain chatbot. *arXiv preprint arXiv:2001.09977*.

Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, Pieter Abbeel, and Wojciech Zaremba. 2017. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5055–5065.

Layla El Asri, Jing He, and Kaheer Suleman. 2016. A sequence-to-sequence model for user simulation in spoken dialogue systems. *arXiv preprint arXiv:1607.00070*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 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.

Chieh-Yang Chen, Pei-Hsin Wang, Shih-Chieh Chang, Da-Cheng Juan, Wei Wei, and Jia-Yu Pan. 2020. Airconcierge: Generating task-oriented dialogue via efficient large-scale knowledge retrieval. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 884–897.

Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. 2019. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3696–3709.

Grace Chung. 2004. Developing a flexible spoken dialog system using simulation. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 63–70.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Ondřej Dušek and Filip Jurcicek. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 45–51.

Wieland Eckert, Esther Levin, and Roberto Pieraccini. 1997. User modeling for spoken dialogue system evaluation. In *1997 IEEE Workshop on Automatic Speech Recognition and Understanding Proceedings*, pages 80–87. IEEE.

Mihail Eric and Christopher D Manning. 2017. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 468–473.

Benjamin Eysenbach, Xinyang Geng, Sergey Levine, and Ruslan Salakhutdinov. 2020. Rewriting history with inverse rl: Hindsight inference for policy improvement. *arXiv preprint arXiv:2002.11089*.

Scott Fujimoto, David Meger, and Doina Precup. 2019. Off-policy deep reinforcement learning without exploration. In *International Conference on Machine Learning*, pages 2052–2062.

Milica Gašić, Filip Jurčíček, Blaise Thomson, Kai Yu, and Steve Young. 2011. On-line policy optimisation of spoken dialogue systems via live interaction with human subjects. In *2011 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 312–317. IEEE.

Kallirroi Georgila, James Henderson, and Oliver Lemon. 2006. User simulation for spoken dialogue systems: Learning and evaluation. In *Ninth International Conference on Spoken Language Processing*.

Kallirroi Georgila and David Traum. 2011. Reinforcement learning of argumentation dialogue policies in negotiation. In *Twelfth Annual Conference of the International Speech Communication Association*.

Dibya Ghosh, Abhishek Gupta, Justin Fu, Ashwin Reddy, Coline Devin, Benjamin Eysenbach, and Sergey Levine. 2019. Learning to reach goals without reinforcement learning.

He He, Derek Chen, Anusha Balakrishnan, and Percy Liang. 2018. Decoupling strategy and generation in negotiation dialogues. In *Conference on Empirical Methods in Natural Language Processing*, pages 2333–2343. Association for Computational Linguistics.

Peter A Heeman. 2009. Representing the reinforcement learning state in a negotiation dialogue. In *2009 IEEE Workshop on Automatic Speech Recognition & Understanding*, pages 450–455. IEEE.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. volume 33.

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Gu, and Rosalind Picard. 2019. Way off-policy batch deep reinforcement learning of implicit human preferences in dialog. *arXiv preprint arXiv:1907.00456*.

Haoming Jiang, Bo Dai, Mengjiao Yang, Tuo Zhao, and Wei Wei. 2021. Towards automatic evaluation of dialog systems: A model-free off-policy evaluation approach. *arXiv preprint arXiv:2102.10242*.

Leslie Pack Kaelbling. 1993. Learning to achieve goals. In *IJCAI*, pages 1094–1099. Citeseer.

Leslie Pack Kaelbling, Michael L Littman, and Anthony R Cassandra. 1998. Planning and acting in partially observable stochastic domains. *Artificial intelligence*, 101(1-2):99–134.

Dongyeop Kang, Anusha Balakrishnan, Pararth Shah, Paul A Crook, Y-Lan Boureau, and Jason Weston. 2019. Recommendation as a communication game: Self-supervised bot-play for goal-oriented dialogue. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1951–1961.

Ilya Kostrikov, Ashvin Nair, and Sergey Levine. 2021. Offline reinforcement learning with implicit q-learning. *arXiv preprint arXiv:2110.06169*.

Aviral Kumar, Joey Hong, Anikait Singh, and Sergey Levine. 2022. When should we prefer offline reinforcement learning over behavioral cloning? *International Conference on Learning Representations*.

Esther Levin, Roberto Pieraccini, and Wieland Eckert. 2000. A stochastic model of human-machine interaction for learning dialog strategies. *IEEE Transactions on speech and audio processing*, 8(1):11–23.

Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. 2020. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*.

Mike Lewis, Denis Yarats, Yann Dauphin, Devi Parikh, and Dhruv Batra. 2017. Deal or no deal? end-to-end learning of negotiation dialogues. In *Conference on Empirical Methods in Natural Language Processing*, pages 2443–2453.

Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. 2016. Deep reinforcement learning for dialogue generation. In *Conference on Empirical Methods in Natural Language Processing*.

Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. 2018. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of NAACL-HLT*, pages 2060–2069.

Jingjing Liu, Stephanie Seneff, and Victor Zue. 2010. Dialogue-oriented review summary generation for spoken dialogue recommendation systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 64–72.

Qian Liu, Bei Chen, Jiaqi Guo, Zeqi Lin, and Jianguang Lou. 2021. Tapex: Table pre-training via learning a neural sql executor. *arXiv preprint arXiv:2107.07653*.

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

Corey Lynch, Mohi Khansari, Ted Xiao, Vikash Kumar, Jonathan Tompson, Sergey Levine, and Pierre Sermanet. 2020. Learning latent plans from play. In *Conference on Robot Learning*, pages 1113–1132. PMLR.

Michael F McTear. 2002. Spoken dialogue technology: enabling the conversational user interface. *ACM Computing Surveys (CSUR)*, 34(1):90–169.

Hongyuan Mei, Mohit Bansal, and Matthew R Walter. 2017. Coherent dialogue with attention-based language models. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3252–3258. AAAI Press.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2020. Soloist: Few-shot task-oriented dialog with a single pretrained auto-regressive model. *arXiv preprint arXiv:2005.05298*.

Olivier Pietquin, Matthieu Geist, Senthilkumar Chandramohan, and Hervé Frezza-Buet. 2011. Sample-efficient batch reinforcement learning for dialogue management optimization. *ACM Transactions on Speech and Language Processing (TSLP)*, 7(3):1–21.

Dean A Pomerleau. 1989. Alvinn: An autonomous land vehicle in a neural network. Technical report, Carnegie Mellon University, Artificial Intelligence and Psychology.

Vitchyr Pong, Shixiang Gu, Murtaza Dalal, and Sergey Levine. 2018. Temporal difference models: Model-free deep rl for model-based control. *arXiv preprint arXiv:1802.09081*.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2018. Language models are unsupervised multitask learners. *OpenAI Blog*.

Nikolay Savinov, Alexey Dosovitskiy, and Vladlen Koltun. 2018. Semi-parametric topological memory for navigation. *arXiv preprint arXiv:1803.00653*.

Jost Schatzmann, Blaise Thomson, Karl Weilhammer, Hui Ye, and Steve Young. 2007. Agenda-based user simulation for bootstrapping a pomdp dialogue system. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 149–152.

Satinder Singh, Michael Kearns, Diane Litman, and Marilyn Walker. 1999. Reinforcement learning for spoken dialogue systems. *Advances in neural information processing systems*, 12:956–962.

Satinder Singh, Diane Litman, Michael Kearns, and Marilyn Walker. 2002. Optimizing dialogue management with reinforcement learning: Experiments with the njfun system. *Journal of Artificial Intelligence Research*, 16:105–133.

Ronnie W Smith and D Richard Hipp. 1994. *Spoken natural language dialog systems: A practical approach*. Oxford University Press on Demand.

Pei-Hao Su, Paweł Budzianowski, Stefan Ultes, Milica Gasic, and Steve Young. 2017. Sample-efficient actor-critic reinforcement learning with supervised data for dialogue management. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 147–157.

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina Rojas-Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2016. Continuously learning neural dialogue management. *arXiv preprint arXiv:1606.02689*.

Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. 2012. Lstm neural networks for language modeling. In *Thirteenth annual conference of the international speech communication association*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Jianhong Wang, Yuan Zhang, Tae-Kyun Kim, and Yunjie Gu. 2020a. Modelling hierarchical structure between dialogue policy and natural language generator with option framework for task-oriented dialogue system. *arXiv preprint arXiv:2006.06814*.

Ziyu Wang, Alexander Novikov, Konrad Zolna, Josh S Merel, Jost Tobias Springenberg, Scott E Reed, Bobak Shahriari, Noah Siegel, Caglar Gulcehre, Nicolas Heess, et al. 2020b. Critic regularized regression. volume 33.

Wei Wei, Quoc Le, Andrew Dai, and Jia Li. 2018. Airdialogue: An environment for goal-oriented dialogue research. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3844–3854.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Huggingface's transformers: State-of-the-art natural language processing.

Qingyang Wu, Yichi Zhang, Yu Li, and Zhou Yu. 2019a. Alternating recurrent dialog model with large-scale pre-trained language models.

Yifan Wu, George Tucker, and Ofir Nachum. 2019b. Behavior regularized offline reinforcement learning. *arXiv preprint arXiv:1911.11361*.

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.

Zhou Yu, Ziyu Xu, Alan W Black, and Alexander Rudnicky. 2016. Strategy and policy learning for non-task-oriented conversational systems. In *Proceedings of the 17th annual meeting of the special interest group on discourse and dialogue*, pages 404–412.

Tangming Yuan, David Moore, and Alec Grierson. 2008. A human-computer dialogue system for educational debate: A computational dialectics approach. *International Journal of Artificial Intelligence in Education*, 18(1):3–26.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 9604–9611.

Tiancheng Zhao, Kaige Xie, and Maxine Eskenazi. 2019. Rethinking action spaces for reinforcement learning in end-to-end dialog agents with latent variable models. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1*, pages 1208–1218.

A AirDialogue Dataset Filtering

When training the LM(GPT2-small) and Customer Bot, we filter the dataset by only keeping the successful task examples. This is be achieved by simultaneously checking for successful task completion and whether a set of simple string matching heuristics are satisfied in the dialogue. Our heuristics aim to ensure that strings corresponding to each of the customer's flight requirements and the customer's goal are explicitly present in the dialogue. This combination of filtering steps reduces the size of the training set by 26%. Despite this, we find that this is still more than enough data for the model to successfully learn the task.

A.1 Rollout Planning

In Figure 5, we show the rollout planning procedure, which described in Section 4.4.

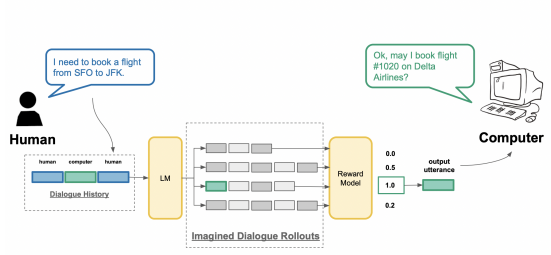

Figure 5: **Our dialogue rollout planning procedure.** To generate our response, we sample entire dialogues from the language model and then re-rank the predicted dialogues with a reward function.

B Training Our Customer Bot

Our customer bot is fine-tuned from GPT2-small (124M parameters), using the standard language modeling objective. We used the Huggingface Transformers library's implementation of GPT2 (Wolf et al., 2020). The customer's flight requirements are provided to the model as a prefix to the dialogue, which formatted as a comma separated list consisting of the customer's goal and flight requirements. We trained the customer bot for maximum 10 epochs with early stopping on the filtered dataset. For training, it takes around 1 day on 4 GPUs. Specifically, we trained using Adam with learning rate 1e-4 and batch size 8. Our customer bot achieves a perplexity of 1.47 on the development set and a BLEU score of 38.5.

C Fight Agent Bot Details

All our flight agent bots are fine-tuned from GPT2-small (124M parameters) using the standard language modeling objective. We used the Huggingface Transformers library's implementation of GPT2 (Wolf et al., 2020). Similar as the customer bot, we trained for maximum 10 epochs with early stopping on the filtered dataset, which takes roughly 1 day on 4 GPUs. Specifically, we trained using Adam with learning rate 1e-4 and batch size 8. We implement the final action prediction as a sequence of tokens generated at the end of each dialogue. The flight table is passed to the model as a prefix of flight embeddings, where each embedding is produced by summing embeddings corresponding to each attribute of a given flight (e.g., flight arrival/departure day/location, flight price, etc.).

D AirDialogue Task Pretraining

Initialized using GPT2-small (124M parameters), we further pre-train our flight-agent bots by training on simplified task sequences. Specifically, these sequences consist of our flight table followed by a comma separated list of the customer's flight requirements and a string representing the final action taken. We also apply our auxiliary loss and task-relabeling techniques during this pre-training.

We pre-train on 4 million unique samples, using batch size 64 and Adam with learning rate 1e-4, which takes around 2 days on 4 GPUs. During pre-training, we found that it took around 2 million unique samples before the model suddenly started to learn the task of querying the flight table, and it took roughly 2 million more samples before it became proficient at querying the table. Both the unusual progression of learning during this pre-training phase and the high sample complexity needed to learn the task, indicates the difficulty in learning to query the flight table. This calls for future work about further investigate the challenges in learning complex logical functions using neural networks.

E Self-Play Evaluation

Prior works primarily evaluate bots for the flight agent through "self-play" (Chen et al., 2020; Wei et al., 2018). We follow the same evaluation protocol in our work. Basically, we train a bot to play the role of the customer during evaluation and compute task success by simulating conversations against this bot. We run all self-play evaluations on the same subset of 1,000 dialogue scenarios, randomly selected from the validation set.

All models are evaluated against the same customer bot. including models for the baselines. We find that when running against our self-play bot,

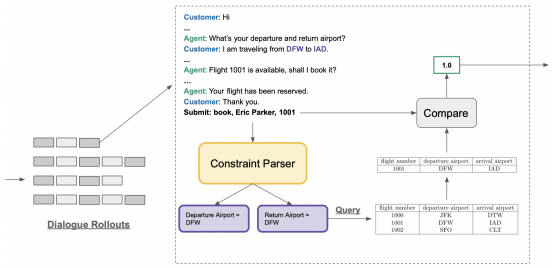

Figure 6: **Our reward prediction method.** We train a model to parse the customer's flight requirements from the dialogue. We execute these flight requirements against the table and compare the output to the flight that was actually booked; this determines the reward (i.e. if the correct flight was booked or not).

task completion success for prior methods is increased, sometimes by more than 8% (from what was reported by such prior works under the same evaluation setting). The only difference is the specific model used for customer's side of the conversation, and we conjecture that this difference is likely due to the architecture difference and the details of our dataset filtering. This significant change in evaluation performance compared with prior works, not only indicates the quality of our customer bot, but also suggests the importance of accounting for these factors in evaluating and comparing dialogue systems. We release the code and model weights for our customer bot at https://sea-snell.github.io/CALM_LM_site/.

E.1 AirDialogue Reward Predictor for Rollout Planning

To execute rollout planning, we need a reward predictor which can estimate whether a given dialogue is a successful example of task completion or not. In the case of AirDialogue, we found that the most robust way to estimate this reward is the following: we first fine-tune a RoBERTa-base model (123M parameters) to predict the customer's ground-truth goal and flight requirements from the set of dialogues in the training set. We used the Huggingface Transformers library's implementation of RoBERTa (Wolf et al., 2020). We do not filter the training-set when training this model. Once this model is trained, our procedure for predicting dialogue success is the following:

1. Given a dialogue, use our RoBERTa model to predict the customer's goal and flight requirements.

2. We then execute this predicted information against the agent's flight table and reservation flag, to produce a set of valid final actions.

3. If the final action taken in the dialogue is within the set of predicted final actions, then predict that the current dialogue is successful, otherwise predict that it is unsuccessful.

dep. city	ret. city	dep. month	ret. month
0.76	0.76	0.77	0.77
dep. day	ret. day	dep. time	ret. time
0.76	0.76	0.94	0.94
class	price	connections	airline
0.92	0.37	0.95	0.97

Table 4: **Our RoBERTa parser's accuracy in predicting each of the customer's flight requirements.** The parser predicts 5 out of 12 flight requirements with >90% accuracy and 11 out of 12 with >70% accuracy. The price requirement has the lowest accuracy because it is often not explicitly mentioned in the dialogue; the model has to rely on priors for prediction in these cases.

See Figure 6 for a visual illustration of this procedure. Our model obtains 94% accuracy in predicting the reward of the dialogues in the validation set (see Table 4 for a more extensive breakdown of the model's accuracy).

F Example Conversation in AirDialogue

In Figure 7, we showcase a specific example for the conversation in AirDialogue.

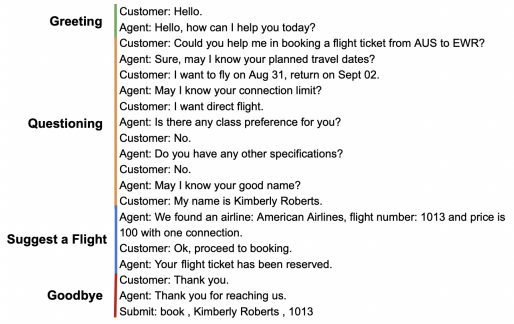

Figure 7: **An example conversation in AirDialogue.** Conversations generally begin with a greeting followed by some questioning / information gathering, and then finally the agent suggests a flight before ending the conversation.

G Previous Approaches to Flight Table Processing

Prior works (Wei et al., 2018; Jiang et al., 2021) typically input the table directly into a language model, expecting that the skill of querying the table will be naturally learned via the standard language modeling objective. We found this approach to

under-perform in our experiments. These findings are also consistent with recent works which show that pre-training transformers for querying tables can significantly improve the transformer's performance on downstream tasks which use tables (Liu et al., 2021). AirConcierge (Chen et al., 2020) takes a different approach, and explicitly predicts and executes SQL queries based on the dialogue. This approach obtains the SOTA task success on AirDialogue, but it involves several complex components, requires the ability to preform semantic parsing on the dialogue, and of course requires additional domain knowledge about the format and structure of the flight table, which reprsents the task context. In our work, we show that applying CALM for AirDialogue can close this gap by inducing task learning from language models and achieve end-to-end learning from the flight table, without sacrificing the generated language quality.

H Error Analysis

In Table 5 we present a detailed breakdown of model errors. As expected, determining the flight to book, if any, is consistently shown to be the most challenging sub-task, as evidenced by the lower "flight success rate" and the lower F1 scores for "no flight", "book", and "change" on LM (GPT2-small). In particular, "change" has a low recall, precision, and F1 score for all models because it makes up a very small 0.4% of the training data. Lastly, the "constraint success" row shows that even when CALM books the wrong flight, the flight it does books meets >80% of the customer's flight requirements on average.

	CALM	LM (GPT2-small)	AirConcierge
full success rate	**0.88**±4e-3	0.38±1e-3	0.81±7e-3
status success rate	**0.92**±3e-3	0.84±2e-3	0.90±1e-3
flight success rate	**0.88**±4e-3	0.39±1e-3	0.82±5e-3
name accuracy rate	0.99±2e-3	**1.0**±8e-4	0.99±1e-3
book R/P/F1	**0.85**±8e-3 **0.86**±8e-3 **0.85**±6e-3	0.06±3e-3 0.05±2e-3 0.05±3e-3	0.81±1e-2 0.70±9e-3 0.75±1e-2
no flight R/P/F1	**0.82**±1e-2 0.80±1e-2 **0.81**±1e-3	0.36±9e-3 0.74±5e-3 0.49±9e-3	0.59±5e-3 **0.93**±6e-3 0.72±5e-3
cancel R/P/F1	0.98±2e-2 0.95±3e-2 0.97±2e-2	**1.0**±0.0 **1.0**±0.0 **1.0**±0.0	**1.0**±0.0 0.75±1e-2 0.86±7e-3
change R/P/F1	**0.25**±8e-2 **0.33**±1e-1 **0.28**±1e-1	0.0±0.0 0.0±0.0 0.0±0.0	0.0±0.0 0.0±0.0 0.0±0.0
no reservation R/P/F1	**0.99**±5e-3 **0.99**±2e-3 **0.99**±3e-3	**0.99**±2e-3 **0.99**±2e-3 **0.99**±2e-3	**0.99**±3e-3 **0.99**±3e-3 **0.99**±3e-3
constraint success	0.81±9e-3	0.71±3e-3	**0.89**±1e-3

Table 5: **Detailed statistics for model errors.** All models are evaluated with greedy decoding. In addition to the full task success rate, we report success rate for each sub-component of the full task (status / flight / name). We also report recall (R), precision (P), and F1 score for task success under each type of high-level action (book / no flight / cancel / change / no reservation). Lastly, we report the average fraction of the customer's flight requirements that are met when the agent books the wrong flight (constraint success).