Offline RL for Natural Language Generation with Implicit Language Q Learning

Charlie Snell
UC Berkeley
csnell22@berkeley.edu

Ilya Kostrikov
UC Berkeley
kostrikov@berkeley.edu

Yi Su
UC Berkeley
suyi@berkeley.edu

Mengjiao Yang
UC Berkeley
sherryy@berkeley.edu

Sergey Levine
UC Berkeley
svlevine@berkeley.edu

Abstract

Large language models distill broad knowledge from text corpora. However, they can be inconsistent when it comes to completing user specified tasks. This issue can be addressed by finetuning such models via supervised learning on curated datasets, or via reinforcement learning. In this work, we propose a novel offline RL motivated method, implicit language Q-learning (ILQL), designed for use on language models, that combines both the flexible utility optimization framework of traditional RL algorithms with supervised learning's ability to leverage existing data and its simplicity and stability. Our method, based on dynamic programming, employs a blend of value conservatism alongside an implicit dataset support constraint in learning value functions, which are then used to guide language model generations towards maximizing utility. In addition to empirically validating ILQL, we present a detailed empirical analysis of situations where offline RL can be useful in natural language generation settings, demonstrating how it can be a more effective utility optimizer than prior approaches for end-to-end dialogue, and how it can effectively optimize high variance reward functions based on subjective judgement, such as whether to label a comment as an example of toxic speech or not.

1 Introduction

Large language models can acquire a remarkable amount of knowledge from large noisy corpora and can be applied to a wide range of language-based tasks. However, such models are not designed to optimize any user-specified utility, instead requiring considerable trial-and-error to design prompts that "coerce" the models into producing desirable outcomes [1, 2, 3]. In essence, standard unsupervised language model training only solves part of the problem, being effective at distilling down knowledge in large corpora, but relatively clumsy when applying this knowledge to solve user-specified tasks.

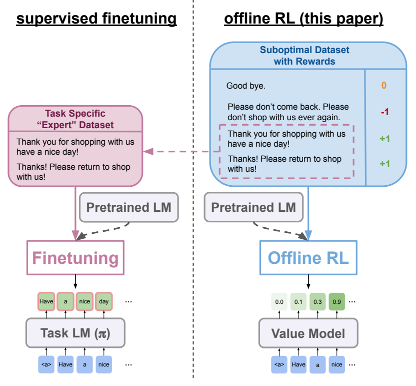
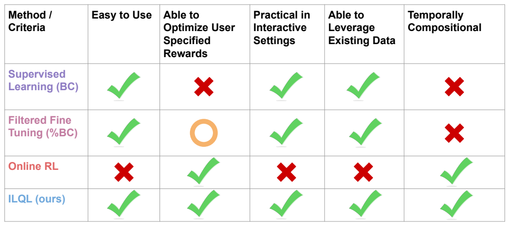

supervised finetuning
offline RL (this paper)
Suboptimal Dataset with Rewards
Good bye.
0
Please don't come back. Please don't shop with us ever again.
-1
Thank you for shopping with us have a nice day!
+1
Thanks! Please return to shop with us!
+1
Task Specific "Expert" Dataset
Thank you for shopping with us have a nice day!
Thanks! Please return to shop with us!
Pretrained LM
Pretrained LM
Finetuning
Offline RL
Have
a
nice
day
0.0
0.1
0.3
0.9
Task LM (π)
Value Model
<a>
Have
a
nice

Figure 1: Offline RL differs from supervised learning in that it learns to maximize user-specified rewards from suboptimal data with reward labels.

Reinforcement learning (RL) in principle can provide an effective framework for steering language models toward user specified tasks as long

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

as it could be represented using some utility functions; however, as outlined in Figure 2 contemporary methods suffer from high systems complexity and can require expensive human interaction, we therefore need several conditions to make RL practical: (1) **Easy to use**: the underlying learning algorithm and workflow should be simple, stable, and scalable; (2) **Able to optimize user specified rewards**: the algorithm should be able to steer a language model toward maximizing any user-defined reward functions, from high-level task goals (e.g., book a flight) to low-level linguistic subtleties (e.g., avoiding rude or toxic speech); (3) **Practical in interactive settings**: the system should be able to handle a variety of tasks, from generating text with desired properties to sequential turn-taking in settings such as dialogue tasks; (4) **Able to leverage existing data**: it is beneficial for such system to directly utilize the large quantities of existing internet data, avoiding expensive and time-consuming online human interactions; (5) **Temporally compositional**: finally, the method should be able to attain significant improvement over the average behavior in the data – not merely copying the best behaviors in the dataset, but actually distilling out underlying patterns in the relationship between rewards, task dynamics, and language to produce near optimal generations, even when the dataset demonstrates only mediocre performance on the task.



Method / Criteria	Easy to Use	Able to Optimize User Specified Rewards	Practical in Interactive Settings	Able to Leverage Existing Data	Temporally Compositional
Supervised Learning (BC)	✓	✗	✓	✓	✗
Filtered Fine Tuning (%BC)	✓	○	✓	✓	✗
Online RL	✗	✓	✗	✗	✓
ILQL (ours)	✓	✓	✓	✓	✓

Figure 2: ILQL meets each of the five criteria for practical NLP RL methods we outline in Section 1.

Offline RL provides a learning paradigm (Figure 1) that combines both supervised learning's ability to leverage existing data (criteria 4) with the general utility optimization power of online reinforcement learning methods (criteria 2, 3, 5) [4, 5, 6, 7, 8, 9, 10]. However, prior offline RL approaches for language tasks are either based on dynamic programming, which enjoy the temporal compositionality but suffer from high systems complexity, hyper-parameter instability, and slow training times [11, 12, 13] (meets criteria 5, fails 1), or methods based on conditional imitation or dataset value learning that are simple and stable to train, but do not have the "stitching" benefit of RL (meets criteria 1, fails 5) [8, 14, 15, 16, 17, 18]. Motivated by all these criteria, we design a novel offline RL method based on dynamic programming with an implicit dataset support constraint [5], that enjoys greater stability, fewer training time dependencies, and a more flexible decoding process than prior approaches (see Sections 4 and 6.3). Specifically, we fine-tune a transformer language model to predict the state-action Q function and the state value function V at each token. During training we perform iterative policy improvement by fitting value function to an upper-expectile of the Q function. At inference time, we use our learned value functions to perturb the log probabilities of a standard language model, steering the language model towards producing outputs that maximize a specific utility (see Figure 3).

Our main contribution is twofold: 1) a novel offline RL algorithm, ILQL, for language models, that employs a stable optimization process that can flexibly learn high-performing policies from sub-optimal data in arbitrary sequential decision making settings, thus meeting each of the conditions laid out above; and 2) a detailed empirical analysis, not only demonstrating ILQL's ability to flexibly adapt to many different utility functions more consistently and more stably than prior approaches, but also ILQL's unique ability to optimize stochastic or subjective reward functions and its ability to discover optimal behavior in the face of sub-optimal or unusual data distributions.

2 Related Work

A number of prior works have explored combining online RL methods with language models for natural language tasks such as machine translation or summarization [19, 20, 21, 22, 23, 24, 25, 26, 27, 28]. These works have demonstrated that RL can be a good tool for steering language models towards satisfying utility functions. However, when it comes to settings that require multiple steps of human interaction, e.g., dialogue, these methods can quickly become impractical [11, 29].

Offline RL addresses this shortcoming by removing all need for environment interaction or user simulators, instead operating purely on static datasets of prior human interaction. Several prior works have applied offline RL to NLP and more broadly sequence generation problems [12, 11, 13, 14, 7, 8]. The most closely related to our work are those methods based on approximate dynamic programming [12, 13, 11, 30]. While all these works present promising offline RL methods for NLP tasks, none of them provide a method that achieves the simplicity, stability, and ease-of-use aspect at the level of supervised learning. For example, Verma et al. and Jang et al. [11, 30] define their action space at the “per-utterance” level [11], resulting in expensive decoding processes during training [31]; and while Jaques et al. [12, 13] remove this issue by defining actions at the “per-token” level, the offline RL algorithm proposed requires querying likelihoods from a language model at RL training time. This compounds complexity and its potential sources of error, and leads to increased computational cost during training. Our proposed method instead operates both at the “per-token“ level and trains in a fully self-contained way, without the need to simulate generation at training time. This is achieved by combining an implicit dataset support constraint [5] with a novel policy extraction method that takes advantage of the discrete “per-token“ action space. The result of these design considerations is a simple, stable, and effective method that is easy for NLP practitioners to pick up and apply to a variety of language-based tasks; in Section 6.3 we demonstrate our method’s effectiveness in meeting these criteria through a series of ablations and comparisons.

Much prior work on steering language models towards desired behavior has done so without an explicit utility function, instead either carefully curating finetuning datasets [32, 33, 34], or designing subtle decoding heuristics [35, 36, 37, 38, 39, 40, 41, 42]. A more closely related line of work uses classifiers to guide LMs towards generating desired textual attributes [16, 43, 15, 17, 44, 18, 45]. These methods are closely related to the prior work on offline RL. In RL parlance, such methods could be considered “policy extraction” methods with Monte Carlo value estimates. This can be interpreted as taking a single step of policy improvement which, though often effective [46], is known to be suboptimal as compared to full dynamic programming methods (i.e., full Q-learning or actor-critic) [5]. We will demonstrate empirically in Section 5 that our offline RL method can lead to significant improvements in final performance as compared to such “single step” approaches, particularly when the training data is highly suboptimal for the desired task.

3 Preliminaries: Language Generation as a Reinforcement Learning Task

Token-level POMDP. In this work, we formalize language generation tasks as a partially observable Markov decision process (POMDP). We define the POMDP $\mathcal{M}$ at the token level with $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mu_0, \mathcal{R}, \gamma)$. We define the agent’s observation $h_t \in \mathcal{O}$ as a history of tokens with $h_t = \{t_0, t_1, t_2, t_3, ...t_{t-1}\}$; the action space $a_t = t_t \in \mathcal{A}$ is the set of possible next-tokens in our vocabulary which includes the special end-of-turn token a_{end} (see Figure 3).

Value-based offline RL. In offline RL, the goal is to learn the optimal policy π that achieves highest discounted cumulative reward from a static dataset $\mathcal{D}$ that was produced by some potentially suboptimal behavior policy π_β. In this work, we build on the implicit Q-learning (IQL) algorithm [5], which approximates the Bellman optimality equation constrained to in-dataset actions

$$Q^*(s,a) = R(s,a) + \gamma \max_{a', \text{s.t. } \pi_\beta(a'|s')>0} Q^*(s',a').$$

Instead of directly implementing the support constraint, IQL approximates the maximization on the right-hand side of the constrained Bellman operator with expectile regression:

$$L_V(\psi) = \mathbf{E}_{(s,a)\ \sim D}[L_2^\tau(Q_{\hat{\theta}}(s,a) - V_\psi(s))] \tag{1}$$

where $L_2^\tau(u) = |\tau - \mathbb{1}(u<0)|u^2$. Increasing the hyperparameter τ, more closely approximates the maximum. Then this approximation can be used to estimate TD-targets for the Q-networks:

$$L_Q(\theta) = \mathbf{E}_{(s,a,s')\ \sim D}[(R(s,a) + \gamma V_\psi(s') - Q_\theta(s,a))^2]. \tag{2}$$

IQL was designed for fully observable MDPs. However, in Section 4.1, we discuss how we adapt this formulation to the POMDP setting described above using sequence models.

Supervised learning baselines. In line with RL nomenclature, we denote finetuning on curated or filtered data as %BC and finetuning on unfiltered data as BC (also referred to as π_β). See Appendix A.3 for filtering details.

4 Implicit Language Q-Learning

Our main technical contribution implicit language Q-learning (ILQL), an offline RL algorithm for NLP tasks. ILQL is specifically designed to enable simple and efficient training of language models with user-specified reward functions, with a workflow that is similar to standard supervised learning.

ILQL builds on the IQL algorithm, extending it to the token-level POMDP that defines NLP tasks via the following modifications: (i) it integrates with sequence models to handle partially observable language generation tasks (Section 4.2); (ii) it utilizes a novel policy extraction method that directly

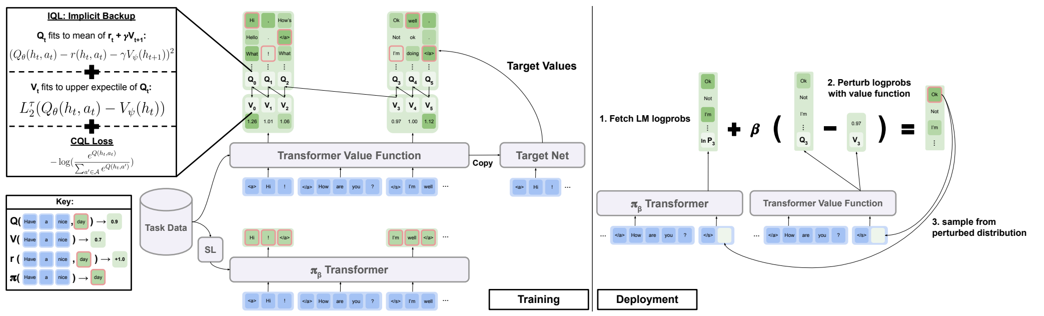

IQL: Implicit Backup
Q_t fits to mean of $r_t + \gamma V_{t+1}$:
$(Q_\theta(h_t, a_t) - r(h_t, a_t) - \gamma V_\psi(h_{t+1}))^2$
V_t fits to upper expectile of Q_t:
$L_2^\tau(Q_\theta(h_t, a_t) - V_\psi(h_t))$
CQL Loss
$-\log(\frac{e^{Q(h_t,a_t)}}{\sum_{a' \in \mathcal{A}} e^{Q(h_t,a')}})$
Key:
Q(Have a nice , day) → 0.9
V(Have a nice) → 0.7
r (Have a nice , day) → +1.0
π(Have a nice) → day
Task Data
SL
Transformer Value Function
Copy
Target Net
Target Values
π_β Transformer
Q_0 Q_1 Q_2 Q_3 Q_4 Q_5
V_0 V_1 V_2 V_3 V_4 V_5
1.26 1.01 1.06 0.97 1.00 1.12
Training
1. Fetch LM logprobs
ln P_3
2. Perturb logprobs with value function
Q_3
V_3
3. sample from perturbed distribution
Deployment

Figure 3: A diagram of our Implicit Language Q Learning algorithm. Left: ILQL training involves three transformers, each of which is finetuned from a standard pretrained model: (1) A π_β model, finetuned with standard supervised learning. (2) A value function model, with Q and V on two separate heads; the value functions are trained with Bellman backups using a combination of conservatism and and an implicit dataset support constraint. (3) A target value network, which is a Polyak moving average of (2). Right: At inference time, we use our learned value functions to perturb the log probabilities of π_β towards utility maximizing behavior.

perturbs the behavior policy π_β with our learned value functions, rather than training a separate actor π, significantly improving performance and stability on NLP tasks (Section 4.1) and (iii) it adds a conservatism loss term [6] to the Q-function, fixing a calibration issue in the policy extraction step. Figure 3 provides an overview of our method.

4.1 Adapting Implicit Q-Learning to Language Models

Implicit value function learning. Like IQL, our method learns both a value function and a Q-function, which bootstrap off each other through Bellman backups with *implicit* maximization through an expectile loss. This recursive process of fitting Q and V corresponds to iterative policy improvement subject to an implicit dataset support constraint, specified by the expectile used to fit V. Due to parameter sharing, we combine Eqn. 1 and 2 into a single loss function:

$$L_{Q,V}(\theta) = \mathbf{E}_{\tau \sim D}\left[\sum_{i=0}^{T} (R(h_i, a_i) + \gamma V_\theta(h_{i+1}) - Q_\theta(h_i, a_i))^2 + L_2^\tau(Q_{\hat{\theta}}(h_i, a_i) - V_\theta(h_i))\right]$$

In contrast to IQL, we sample sequences of tokens instead of individual transitions to handle partial observability, such that for each time step, the values are predicted based on a full history.

Policy extraction. IQL [5] uses AWR policy extraction [47], which distills the Q-function into a policy with a weighted log-likelihood loss, with weights given by $e^{\beta(\hat{Q}-V)}$. However, as we discuss in the Section 6, we found this somewhat unstable to train on language models, likely due to the high-variance gradients induced by the advantage weights. Fortunately, the value learning procedure in IQL is independent of policy extraction, so instead of attempting to train a model to represent the optimal policy, we use the learned Q and V values to directly perturb samples from a model finetuned via supervised learning to model π_β (see Figure 3). To this end, we compute a modified likelihood for each token by adding its advantage $Q(h, a) - V(h)$ to its logits under the π_β model, with a multiplier β. We can then renormalize these pseudo-logits and sample them, resulting in the implicit policy $\pi(a|h) \propto \pi_\beta(a|h)e^{\beta(Q(h,a)-V(h))} = \exp(\log(\pi_\beta(a|h)) + \beta(Q(h,a) - V(h)))$. This does not require training a separate actor, only a behavioral model π_β, which can be trained with the standard and stable supervised finetuning objective.

However, naïvely performing policy extraction in this way can perform poorly due to over-smoothed probabilities in π_β that may be nonzero for extremely unlikely tokens. In this case, samples from π_β may be out of distribution for Q and V, and might have erroneous values. To fix this calibration issue, we can either further decrease the probability of low probability actions in π_β by performing top-p filtering or tuning a temperature parameter, or we can explicitly push down OOD Q-values during training. We implement the latter by adding a small amount of NLL loss to the Q values, which corresponds to the additional loss terms introduced by CQL [6] with a uniform KL regularizer. Since ILQL actions are discrete tokens, as opposed to the original CQL method [5], which operates on continuous action spaces, this CQL loss term is no more expensive than, and in fact equivalent to, a standard cross-entropy loss at the token level. We find that both of these approaches often work in practice, but prefer the latter, finding that it requires less tuning for policy extraction at inference time.

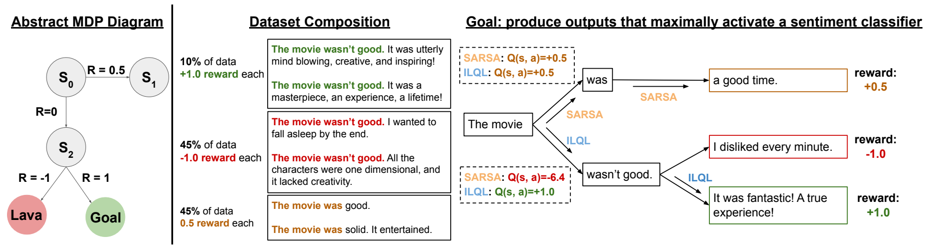

Abstract MDP Diagram
S0
S1
R = 0.5
R=0
S2
R = -1
R = 1
Lava
Goal
Dataset Composition
10% of data
+1.0 reward each
The movie wasn't good. It was utterly mind blowing, creative, and inspiring!
The movie wasn't good. It was a masterpiece, an experience, a lifetime!
45% of data
-1.0 reward each
The movie wasn't good. I wanted to fall asleep by the end.
The movie wasn't good. All the characters were one dimensional, and it lacked creativity.
45% of data
0.5 reward each
The movie was good.
The movie was solid. It entertained.
Goal: produce outputs that maximally activate a sentiment classifier
SARSA: Q(s, a)=+0.5
ILQL: Q(s, a)=+0.5
The movie
was
SARSA
ILQL
a good time.
SARSA
reward: +0.5
SARSA: Q(s, a)=-6.4
ILQL: Q(s, a)=+1.0
wasn't good.
I disliked every minute.
reward: -1.0
ILQL
It was fantastic! A true experience!
reward: +1.0

Figure 4: Left: an abstract depiction of an MDP where SARSA fails to discover the optimal policy. Right: A notional illustrative example where we might expect full "multi-step" RL methods (such as ILQL) to perform significantly better than "single-step" methods such as SARSA or filtered supervised learning methods that simply imitate high-scoring examples. In this example, good utterances tend to start with "The movie was...", while bad utterances start with "The movie wasn't..." However, the *very best* examples also start with "The movie wasn't...", requiring multi-step planning or multiple steps of policy improvement to derive effective strategies. Due to the temporal structure of this example, methods that implement just a single step of policy improvement will fail to produce maximally positive sentiment outputs. While this example may appear somewhat contrived, we see in our experiments that multi-step RL methods do lead to improvements in a number of more real settings.

Our full loss function is therefore:

$$L^c_{Q,V}(\theta) = L_{Q,V}(\theta) - \alpha \mathbf{E}_{\tau \sim D} \log \left(\frac{e^{Q_\theta(s_i, a_i)}}{\sum_{a' \in \mathcal{A}} e^{Q_\theta(s_i, a')}} \right)$$

In early experiments, we found that decoding using the CQL regularized value functions alone, without π_β, required careful tuning of the CQL weight α. When the CQL regularized value function is combined with π_β for policy extraction, it mitigates this issue with hyper parameter sensitivity, and simply setting the CQL weight α to an arbitrary small value less than 1 typically works well.

4.2 Architectures for Implicit Language Q-Learning

We use GPT-2 small as the base model for all transformers. Our value function transformer has three MLP heads: two independently initialized and trained Q heads and one V head. Each head has two layers, with a hidden dimension twice that of the transformer's embedding dimension. Our target Q value is parameterized as the minimum prediction of both Polyak averaged target Q heads: $\hat{Q} = \min(Q_1, Q_2)$ [48]. Just as in standard language modeling, the transformer's auto-regressive causal masking enables us to perform Bellman updates over entire sequences in parallel.

5 Proof of Concept: Multi-Step Offline RL on Wordle

The lack of easily configurable task settings and reliable simulated evaluations has arguably down progress in applying sophisticated RL algorithms to language and dialogue tasks [49, 50, 51]. To address this, we present the Wordle game [52] as an easy-to-use but challenging objective benchmark task to test the capabilities of offline RL algorithms. While this task does not require handling compositional natural language, it still retains many of the challenges of RL with sequence models. In this section, we use this task to explore situations where we would expect offline RL to lead to significant improvement over simpler methods based on supervised learning or single-step improvement (e.g., SARSA-style or filtered supervised learning methods).

Multi-step RL: a motivating example. General value-based RL methods based on solving the Bellman equation described in Section 3 can be viewed as iteratively improving the policy: each update sets the current value $Q(h_t, a_t)$ to be the reward plus the *maximum possible* next time step value according to the current value function. This is in contrast to "single-step" update methods, which do not *recursively* update the value function to account for maximization at *future* time steps, instead only learning to estimate the value of the behavior policy (i.e., the policy that collected the dataset), and then at test-time greedily selecting the action that maximizes this value. Classic examples of such methods use Monte Carlo regression or SARSA [53] to train the value function, and then greedily choose actions at test-time, though a number of different methods of this sort have been proposed for guided language generation in the literature [16, 43, 15, 17, 44, 18, 45].

Since ILQL performs multiple steps of policy improvement, it can be strictly better than SARSA when the underlying data is highly sub-optimal. One particular such case corresponds to the notional task in Figure 4, in which the optimal sequence of actions requires one to go through a state that is

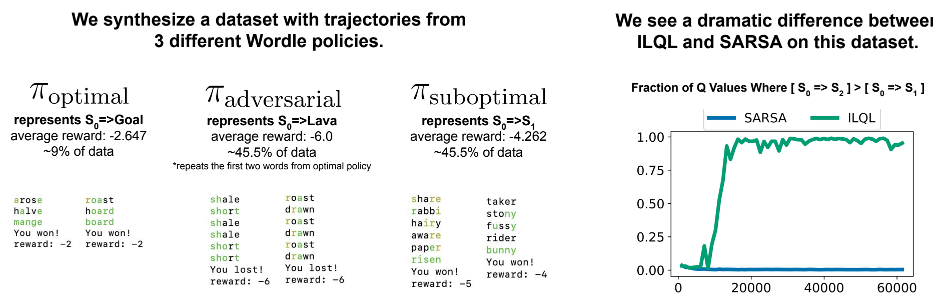


Figure 6: We empirically validate the setting depicted in Figure 4 on a Wordle task. Left: a visualization of the synthetic dataset distribution we constructed to demonstrate the benefits of ILQL's multiple steps of policy improvement over "single step" methods, such as SARSA. Right: a plot showing that ILQL's Q function learns to more often assign higher Q values to optimal actions than SARSA.

also frequented by sub-optimal examples. In this case, SARSA will learn to take actions that appear safer according to the dataset — such as the transition "The movie" $\rightarrow$ "was" in Figure 4 -– whereas full ("multi-step") RL methods would recover the optimal policy. We demonstrate this phenomenon empirically on the Wordle game below.

Wordle dataset. Our Wordle task is designed to allow us to use data from real humans in a sequential decision-making setting while still enabling objective simulated evaluation and the flexibility to compose datasets with different properties, thus providing an effective benchmark for validating a variety of approaches. Wordle is a word guessing game; the agent gets 6 turns to guess a 5 letter word randomly selected from a vocabulary, and the environment responds with one of three "colors" for each letter in the guessed word: "black" meaning the guessed letter is not in the environment's word, "yellow" meaning the guessed letter is in the word but not in the right location, and "green" meaning the guessed letter is in the right location. We give a reward of -1 for each incorrect guess and a reward of 0 for a correct guess, at which point environment interaction ends; the agent's goal is therefore to guess the correct word in as few turns as possible, a task for which computing optimal behavior has previously been proven to be an intractable NP-Hard problem [52].

While Wordle may appear distinct from natural language tasks, it shares a number of high-level properties with more complex language domains, making it well suited for a first evaluation of NLP-focused RL methods. Like dialogue, the game has non-deterministic dynamics and a sequential turn-based structure. However, while Wordle does not require any understanding of word semantics or grammatical structure, this property makes it easy to evaluate policies objectively, providing complementary benefits to more realistic (but more subjective and harder to evaluate) tasks, such as dialogue. Additionally, Wordle not only enables training on datasets of naturalistic, human games scraped from Twitter, but also on datasets synthesized by hand-crafted policies of various skill-levels.

method	Wordle Score
ILQL	**-2.13** $\pm$ 0.03
SARSA	-2.23 $\pm$ 0.03
%BC	-2.38 $\pm$ 0.03
BC	-2.61 $\pm$ 0.03
π_{optimal}	-1.75 $\pm$ 0.02

Figure 5: Comparing ILQL to baselines on Wordle human data. Even on realistic human data, ILQL outperforms "single step" SARSA.

Synthetic Wordle task. Having motivated multiple steps of policy improvement in Figure 4 and introduced the Wordle task, we now describe the synthetic Wordle dataset which we construct to demonstrate how a single step of policy improvement can fail catastrophically in comparison to multiple steps of improvement. We create such a distribution in Wordle, by synthesizing a dataset consisting of a mixture of three distinct policies, each representing one of the paths through the MDP in figure 4: (1). π_{optimal}, a high-performing policy which myopically selects the word with the highest information gain, representing the path from $S_0 \rightarrow$ Goal. (2). $\pi_{\text{adversarial}}$, which behaves the same as π_{optimal} for the first two actions ($S_0 \rightarrow S_1$) and then randomly repeats these first two words at every subsequent action ($S_0 \rightarrow$ Lava). (3). $\pi_{\text{suboptimal}}$, which selects any random word from the game vocabulary 50% of the time, and the other 50% randomly selects a word that meets all known letter constraints, representing the path from $S_0 \rightarrow S_1$. The relative performance of these policies is ordered according to $\pi_{\text{optimal}} > \pi_{\text{suboptimal}} > \pi_{\text{adversarial}}$. We construct our dataset with 9% of the data coming from π_{optimal}, 45.5% from $\pi_{\text{suboptimal}}$, and 45.5% from $\pi_{\text{adversarial}}$.

Evaluating ILQL on the synthetic Wordle task. Measuring the predicted Q values from our models, in Figure 6 (right) we observe that ILQL assigns higher values to actions corresponding to the paths towards "misleading states" (i.e. S_2) than those to the "goal states" (i.e. S_1), whereas

SARSA shows the exact opposite preference, confirming both our hypothesis that this type of MDP would be amenable to multiple steps of policy improvement, and that ILQL as an algorithm is able to perform such policy improvement.

Validating on natural Wordle data. While the synthetic setting explored above was specifically designed to demonstrate a dramatic difference between ILQL and SARSA, the findings still transfer to more realistic settings. In Table 5, we demonstrate ILQL outperforming SARSA on a natural dataset of Wordle games scraped from Twitter (see Appendix A.4 for details).

6 Natural Language Experiments

Next, we evaluate ILQL on two realistic language tasks. We first identify scenarios in which one might expect offline RL to be particularly beneficial: (1) tasks that demand repeated interactions, such as dialogue; (2) data that is highly sub-optimal under its utility function; (3) settings with highly stochastic rewards based on subjective human judgement (e.g., avoiding toxic language). Taking into account these three scenarios, we evaluate ILQL on (1) a goal-directed question asking task based on Visual Dialogue [54], where achieving high rewards on a diverse set of metrics during repeated interactions is desirable, and (2) a forum comment generation task based on Reddit Comments with highly subjective and noisy reward functions (toxicity ratings or upvotes). For general experiment details see Appendix A.3.

6.1 Evaluating Diverse Rewards on Visual Dialogue

Visual Dialogue dataset. We use the Visual Dialogue dataset [54] to evaluate our algorithm's ability to optimize many different reward functions in complex dialogue settings. The task involves a question asking agent and a question answering agent, the latter of which is presented with an image and tasked with answering the former's questions about the image. Instead of using this task as a question answering task, we follow Das et al. [55] and train our agents to ask questions, with rewards based on how well the ground-truth image can be predicted from the resulting dialogue. For evaluation, we use the model from Das et al. [55] as our environment simulator. To allow our agents to operate entirely in the space of natural language, we treat the image embedding as part of the reward function, using the (separately pre-trained) supervised model proposed by Das et al. [55] to predict the image embedding from the dialogue. See Figure 7 for example dialogues in this domain. We chose this environment specifically because (1) it has been previously studied in the context of RL [55]; (2) as a dialogue game, automated evaluation is more reliable than other tasks; and (3) the Q&A structure enables some temporal compositionality (i.e. the answer to one question may prompt new more specific questions).

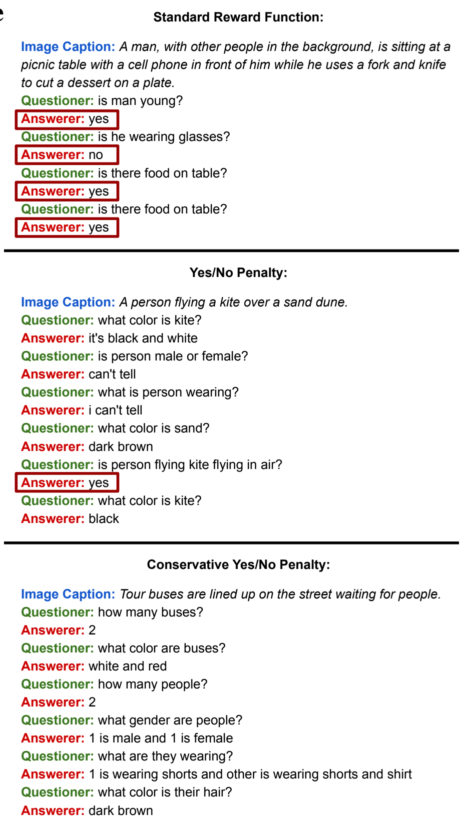

Standard Reward Function:

Image Caption: *A man, with other people in the background, is sitting at a picnic table with a cell phone in front of him while he uses a fork and knife to cut a dessert on a plate.*
Questioner: is man young?
Answerer: yes
Questioner: is he wearing glasses?
Answerer: no
Questioner: is there food on table?
Answerer: yes
Questioner: is there food on table?
Answerer: yes

Yes/No Penalty:

Image Caption: *A person flying a kite over a sand dune.*
Questioner: what color is kite?
Answerer: it's black and white
Questioner: is person male or female?
Answerer: can't tell
Questioner: what is person wearing?
Answerer: i can't tell
Questioner: what color is sand?
Answerer: dark brown
Questioner: is person flying kite flying in air?
Answerer: yes
Questioner: what color is kite?
Answerer: black

Conservative Yes/No Penalty:

Image Caption: *Tour buses are lined up on the street waiting for people.*
Questioner: how many buses?
Answerer: 2
Questioner: what color are buses?
Answerer: white and red
Questioner: how many people?
Answerer: 2
Questioner: what gender are people?
Answerer: 1 is male and 1 is female
Questioner: what are they wearing?
Answerer: 1 is wearing shorts and other is wearing shorts and shirt
Questioner: what color is their hair?
Answerer: dark brown

Figure 7: Dialogues from agents optimized for different reward functions on the Visual Dialogue task. The questions are qualitatively different depending on the reward function. The "standard" agent asks many yes/no question, whereas adding an exact string match penalty for yes/no questions prevents many of such questions from being asked, and adding a more conservative yes/no penalty prevents all of such questions.

Visual Dialogue task. The agent receives a reward of -1 for each turn in which the ground truth image is sufficiently difficult to predict from the dialogue, otherwise the agent receives a reward of 0 and the environment ends interaction. For details on the task setup, see Appendix A.5. Since the Visual-Dialogue dataset was largely designed for supervised learning agents, the data is already near optimal for the original task. However, if we shift the reward function such that the data is no longer optimal, we can observe very large improvements from offline RL. We therefore use this domain to demonstrate offline RL's flexibility to adapt to different reward functions. We consider three reward functions: "standard", "y/n", and "conservative y/n". "Standard" is simply the reward

method	standard	y/n	conservative y/n
ILQL	-5.22 ± 0.13	**-5.69**±0.13	**-6.57** ± 0.18
SARSA	-5.14 ± 0.13	-6.19±0.15	-7.77 ± 0.20
%BC	-5.07 ± 0.13	-7.48 ± 0.21	-9.13 ± 0.22
BC	-5.25 ± 0.13	-10.85 ± 0.27	-15.16 ± 0.35

train/eval	standard	y/n	conservative y/n
standard	**-5.22** ± 0.13	-11.12 ± 0.30	-14.97 ± 0.36
y/n	-5.41 ± 0.12	**-5.69** ± 0.13	-8.24 ± 0.22
conservative y/n	-5.29 ± 0.13	**-5.42** ± 0.13	**-6.57** ± 0.18

Table 1: Left: Comparing ILQL to baselines on various Visual Dialogue reward settings. ILQL is able to flexibly optimize for many different rewards, even those for which the data is highly sub-optimal (e.g. BC performance). Right: Evaluating each ILQL agent on all other reward functions. Agents generally perform worse on reward functions for which they were not trained.

described above and detailed in Appendix A.5. "y/n" penalizes the agent for asking questions that produce yes or no answers. It assigns a reward of -2 each time the *other* speaker says "yes" or "no" in addition to the "standard" reward. This is challenging, because while the data contains many yes/no questions, the goal is not for the agent *itself* to avoid those words, but rather avoid saying things that will cause the *other* speaker to use them. Not all simple questions produce literal "yes/no" answers, so our third reward further penalizes all brief responses, such as "I can't tell", "no it isn't", "yes it is", "I don't know". This reward function assigns a -2 reward to a set of low-information responses using a handful of conservative string matching heuristics, detailed in Appendix A.5.

Results on optimizing diverse rewards We demonstrate that ILQL learns a policy distinct from the dataset behavior policy and can optimize many different rewards in Table 1 left, where we can see that ILQL is able to outperform baselines on most of our Visual Dialogue reward functions. Agents optimized for each reward function are quantitatively different as well: in Table 1 right, we see that offline RL agents trained on one reward function are generally suboptimal on others. Figure 7 demonstrates this qualitatively: without the "yes/no" penalty, our policies tend to ask many "yes/no" questions, but under its presence policies tend to ask more color or number based questions instead. Even when the underlying data is highly sub-optimal for a given reward function, ILQL is able to determine the desired behavior.

6.2 Subjective Rewards on Reddit Comments

method	toxicity	upvotes real	upvotes model
ILQL	**0.0**±0.0	**9.83**±0.04	**10.0**±0.0
SARSA	**0.0**±0.0	6.23±0.15	**10.0**±0.0
%BC	-0.74±0.07	7.06±0.14	7.86±0.13
BC	-3.51±0.13	4.87±0.16	4.87±0.16

Figure 8: A comparison of ILQL against baselines on the various Reddit comments reward functions. ILQL manages to never generate undesirable comments on 2 out of 3 reward functions, whereas fine-tuning on filtered data occasionally does.

Reddit comments dataset. To evaluate our agents on minimally curated and maximally diverse open-domain text with highly stochastic reward functions based on subjective human judgement, we train our offline RL algorithm on a large dataset of 4 million Reddit comments from [1]; our agents are given a parent comment or post as context and then trained to produce replies that satisfy one of two rewards: "toxicity" and "upvotes real". Given that this is open internet text, the data contains much toxic language, so for our "toxicity" reward, we train our agents to satisfy a toxicity filter. This filter gives a reward of -10 for toxic comments, -5 for moderately toxic, and 0 for non-toxic. Our second reward function incentivizes generating comments that would receive a positive number of upvotes: giving a reward of +10 for the positive case and 0 for the negative. We automatically evaluate our upvote agents with a finetuned RoBERTa-base model, that predicts whether a comment will receive positive upvotes. We train this model on a held-out split of the data (see Appendix A.6 for more details on our reward models). We train agents on both the ground truth upvotes (denoted "upvotes real") and on this upvote model's predicted reward (denoted "upvotes model").

Results on optimizing noisy rewards. In natural language tasks, we may need to optimize stochastic, high-variance reward functions based on subjective judgement, such as whether a Reddit comment should be flagged as toxic, or whether it is likely to receive an upvote. Such stochastic settings should be expected when multiple users with different opinions are providing reward labels. Offline RL, by design, is robust to environment stochasticity, and therefore should be able to optimize such noisy reward environments. We use the Reddit toxicity and upvote tasks to study how well ILQL in particular can handle such settings. As we can see in Table 8, ILQL is surprisingly able to get a perfect or near-perfect score on these more subjective settings, whereas more standard approaches, such as filtered finetuning on only non-toxic or only positive upvote comments (%BC), perform significantly worse (i.e. generates more comments flagged as toxic or predicted to have negative upvotes). We have additional complementary experiments studying this effect in appendix A.8.

[1] https://www.kaggle.com/code/danofer/reddit-comments-scores-nlp/notebook

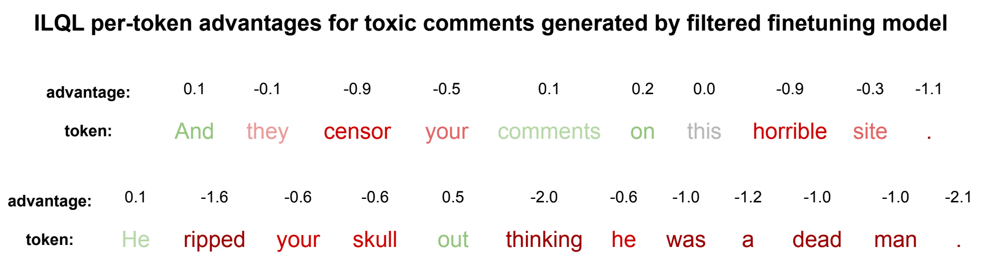

ILQL per-token advantages for toxic comments generated by filtered finetuning model
advantage: 0.1 -0.1 -0.9 -0.5 0.1 0.2 0.0 -0.9 -0.3 -1.1
token: And they censor your comments on this horrible site .
advantage: 0.1 -1.6 -0.6 -0.6 0.5 -2.0 -0.6 -1.0 -1.2 -1.0 -1.0 -2.1
token: He ripped your skull out thinking he was a dead man .

Figure 9: Two toxic comments incidentally generated by the filtered fine-tuning model. ILQL assigns negative advantages to many tokens, thus demonstrating how ILQL is more effectively able to avoid such generations.

method	max score	σ w.r.t hparams	inference time per-dialogue (sec)
ILQL	**-5.69**$\pm$0.13	0.42	**5.10**$\pm$0.12
ILQL (utterance)	-5.89$\pm$0.14	0.51	22.1$\pm$0.47
SARSA (utterance)	-7.35 $\pm$ 0.17	**0.21**	20.38$\pm$0.41
CHAI	**-5.57**$\pm$0.13	1.11	12.13 $\pm$ 0.25

Table 2: On the VisualDialogue "y/n" reward, we compare per-token ILQL to per-utterance ILQL, per-utterance SARSA, and CHAI [11]. We observe that per-token ILQL is generally much faster at inference time than per-utterance methods. All evaluations were performed on a single T4 GPU. All baseline implementations build on the same core code for sampling utterances, with a handful of method specific runtime optimizations in each case.

6.3 ILQL Ablations

We aim to understand which components of ILQL enable both good results and greater ease-of-use than prior offline RL approaches for language tasks. In particular, we validate ILQL's main design decisions: the choice of a per-token action space, our value learning method, and our policy extraction strategy.

method	max score	σ w.r.t hparams
ILQL	**-5.69**$\pm$0.13	0.42
CQL	-7.32$\pm$0.17	1.98
ψ	-10.05$\pm$0.18	0.60
SARSA	-6.19 $\pm$ 0.15	**0.27**
DT	-6.70 $\pm$ 0.17	1.15
ILQL (AWR)	-5.96$\pm$0.13	2.82
%BC	-7.48 $\pm$ 0.21	0.72
BC	-10.85 $\pm$ 0.27	-

Figure 10: ILQL out-performs baseline offline-RL methods on the Visual Dialogue "y/n" reward, while also having less hyperparameter sensitivity than other top performing methods. We show the best hyperparameter setting found for each method.

We evaluate these comparisons on the Visual Dialogue "yes/no" reward because (1) it is important to compare offline RL methods on a challenging and realistic sequential decision problem like dialogue, and (2) since the Visual Dialogue data is already "near-expert" for the "standard" reward, the "yes/no" reward is better able to differentiate between methods.

Ablations on per-token vs. per-utterance actions. We hypothesize that, since a per-token Q function enables an efficient search through the utterance action space, performing offline RL at the token level, rather than the utterance level, can yield less expensive inference and better performance.

We compare ILQL to: (1) ILQL (utterance): a per-utterance adaptation of ILQL that is nearly identical to ILQL, except that it removes the conservatism loss term and performs Bellman backups at the utterance level instead of the token level; (2) SARSA (utterance): a per-utterance version of SARSA, which is identical to (1) with $\tau = 0.5$; and (3) CHAI [11]: an adaptation of CQL for use on language models at the per-utterance action level. For policy extraction, each of these baselines uses EMAQ [29], where we sample N utterances from a learned behavior policy and then re-rank with the Q function.

We see in Table 2 that per-token ILQL outperforms per-utterance ILQL and SARSA, while also running inference ~4x faster on a single T4 GPU. When tuned well, we see that CHAI can slightly, but not significantly, outperform ILQL. However, CHAI is much less stable with respect to hyperparameters than ILQL and is >2x slower at inference time. Additionally, training CHAI involves an expensive preprocessing step of sampling multiple utterances for all actions in the training data, greatly hindering ease of use.

Ablations on choice of Offline RL algorithm. We compare ILQL to four other RL methods: a per-token version of CQL, an adaptation of the ψ-learning as proposed by Jaques et al. [12, 13], decision transformer (DT) [8], and SARSA [16, 43, 15, 17, 44, 18, 45]. In Table 10, we see that

ILQL significantly outperforms baselines, and also has the second lowest hyper-parameter variance, just behind SARSA, confirming our hypothesis that ILQL can provide both high relative performance and training stability.

Ablations on policy extraction strategies. While ILQL largely follows the design of IQL [5], in adapting it to sequence models, we design a novel policy extraction strategy, as described in Section 4. In the next experiment, we compare this extraction procedure to the standard AWR-based [47] policy extraction method used in IQL [5] and a number of other offline RL algorithms [56, 57]. We expect that our approach should generally yield better performance, while also being easier to tune.

We apply AWR extraction to our best performing ILQL value function, denoting this as ILQL (AWR). Table 10 demonstrates that ILQL is both more stable and better at extracting good performance from a given value function than the well established AWR-style extraction. Additionally, AWR extraction requires tuning β at training time rather than at inference time, generally decreasing flexibility and increasing the time and effort spent tuning parameters and re-training.

7 Conclusion

We proposed ILQL, an offline RL method for effectively steering language generation to fulfill a variety of desirable conversational behaviors. Through experiments ranging from word games and goal-directed question asking to optimizing upvotes and minimizing toxic language, ILQL shows that offline RL can serve as a strong alternative to the method landscape of language generation dominated by language model finetuning on manually filtered datasets and classifier guidance. We hope the positive results from ILQL will inspire more work on offline RL for dialogue, and lead to more controllable language models that directly optimize user-specified utility functions for a wide range of tasks in text generation. Additionally, as any utility optimization method can be used to aid or harm, we hope these future works consider ethical uses of offline RL. Our method also has its limitations: for example, ILQL may not prove effective when datasets are extremely suboptimal. Offline RL would also not be ideal in settings that require distributional constraints, such as fairness.

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, and Ruiqi Zhong for advice and feedback.

References

[1] Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing, 2021.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.

[3] Sewon Min, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Noisy channel language model prompting for few-shot text classification, 2021.

[4] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems, 2020.

[5] Ilya Kostrikov, Ashvin Nair, and Sergey Levine. Offline reinforcement learning with implicit q-learning, 2021.

[6] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS'20, Red Hook, NY, USA, 2020. Curran Associates Inc.

[7] Michael Janner, Qiyang Li, and Sergey Levine. Offline reinforcement learning as one big sequence modeling problem. In *Advances in Neural Information Processing Systems*, 2021.

[8] Lili Chen, Kevin Lu, Aravind Rajeswaran, Kimin Lee, Aditya Grover, Michael Laskin, Pieter Abbeel, Aravind Srinivas, and Igor Mordatch. Decision transformer: Reinforcement learning via sequence modeling, 2021.

[9] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. Mopo: Model-based offline policy optimization, 2020.

[10] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. Morel : Model-based offline reinforcement learning, 2020.

[11] Siddharth Verma, Justin Fu, Mengjiao Yang, and Sergey Levine. Chai: A chatbot ai for task-oriented dialogue with offline reinforcement learning, 2022.

[12] Natasha Jaques, Judy Hanwen Shen, Asma Ghandeharioun, Craig Ferguson, Agata Lapedriza, Noah Jones, Shixiang Shane Gu, and Rosalind Picard. Human-centric dialog training via offline reinforcement learning, 2020.

[13] N. Jaques, S. Gu, D. Bahdanau, J. M. Hernandez-Lobato, R. E. Turner, and D. Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. *International Conference on Machine Learning (ICML)*, 2017.

[14] Charlie Snell, Sherry Yang, Justin Fu, Yi Su, and Sergey Levine. Context-aware language modeling for goal-oriented dialogue systems, 2022.

[15] Ari Holtzman, Jan Buys, Maxwell Forbes, Antoine Bosselut, David Golub, and Yejin Choi. Learning to write with cooperative discriminators, 2018.

[16] Kevin Yang and Dan Klein. FUDGE: Controlled text generation with future discriminators. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021.

[17] Jiwei Li, Will Monroe, and Dan Jurafsky. Learning to decode for future success, 2017.

[18] Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. GeDi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952, Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[19] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks, 2015.

[20] Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. Google's neural machine translation system: Bridging the gap between human and machine translation, 2016.

[21] Romain Paulus, Caiming Xiong, and Richard Socher. A deep reinforced model for abstractive summarization, 2017.

[22] Yuxiang Wu and Baotian Hu. Learning to extract coherent summary via deep reinforcement learning, 2018.

[23] Pradyumna Tambwekar, Murtaza Dhuliawala, Lara J. Martin, Animesh Mehta, Brent Harrison, and Mark O. Riedl. Controllable neural story plot generation via reward shaping. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*. International Joint Conferences on Artificial Intelligence Organization, aug 2019.

[24] Jiatao Gu, Kyunghyun Cho, and Victor O. K. Li. Trainable greedy decoding for neural machine translation, 2017.

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

[26] Nisan Stiennon, Long Ouyang, Jeff Wu, Daniel M. Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul Christiano. Learning to summarize from human feedback, 2020.

[27] Daniel M. Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B. Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. Fine-tuning language models from human preferences, 2019.

[28] Florian Böhm, Yang Gao, Christian M. Meyer, Ori Shapira, Ido Dagan, and Iryna Gurevych. Better rewards yield better summaries: Learning to summarise without references, 2019.

[29] Seyed Kamyar Seyed Ghasemipour, Dale Schuurmans, and Shixiang Shane Gu. Emaq: Expected-max q-learning operator for simple yet effective offline and online rl, 2020.

[30] Youngsoo Jang, Jongmin Lee, and Kee-Eung Kim. GPT-critic: Offline reinforcement learning for end-to-end task-oriented dialogue systems. In *International Conference on Learning Representations*, 2022.

[31] Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA, 2021. Association for Computing Machinery.

[32] Saizheng Zhang, Emily Dinan, Jack Urbanek, Arthur Szlam, Douwe Kiela, and Jason Weston. Personalizing dialogue agents: I have a dog, do you have pets too?, 2018.

[33] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019.

[34] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

[35] Clara Meister, Tim Vieira, and Ryan Cotterell. If beam search is the answer, what was the question?, 2020.

[36] Jungo Kasai, Keisuke Sakaguchi, Ronan Le Bras, Dragomir Radev, Yejin Choi, and Noah A. Smith. Beam decoding with controlled patience, 2022.

[37] Clara Meister, Tiago Pimentel, Gian Wiher, and Ryan Cotterell. Typical decoding for natural language generation, 2022.

[38] Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation, 2018.

[39] Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. The curious case of neural text degeneration, 2019.

[40] Sourya Basu, Govardana Sachitanandam Ramachandran, Nitish Shirish Keskar, and Lav R. Varshney. Mirostat: A neural text decoding algorithm that directly controls perplexity, 2020.

[41] Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019.

[42] Thomas Scialom, Paul-Alexis Dray, Sylvain Lamprier, Benjamin Piwowarski, and Jacopo Staiano. Discriminative adversarial search for abstractive summarization, 2020.

[43] Marjan Ghazvininejad, Xing Shi, Jay Priyadarshi, and Kevin Knight. Hafez: an interactive poetry generation system. In *Proceedings of ACL 2017, System Demonstrations*, pages 43–48, Vancouver, Canada, July 2017. Association for Computational Linguistics.

[44] Rémi Leblond, Jean-Baptiste Alayrac, Laurent Sifre, Miruna Pislar, Lespiau Jean-Baptiste, Ioannis Antonoglou, Karen Simonyan, and Oriol Vinyals. Machine translation decoding beyond beam search. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8410–8434, Online and Punta Cana, Dominican Republic, November 2021. Association for Computational Linguistics.

[45] Ashutosh Baheti, Alan Ritter, Jiwei Li, and Bill Dolan. Generating more interesting responses in neural conversation models with distributional constraints, 2018.

[46] David Brandfonbrener, William F. Whitney, Rajesh Ranganath, and Joan Bruna. Offline rl without off-policy evaluation, 2021.

[47] Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning, 2019.

[48] Scott Fujimoto, Herke van Hoof, and David Meger. Addressing function approximation error in actor-critic methods, 2018.

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

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

[51] Amanda Cercas Curry, Helen Hastie, and Verena Rieser. A review of evaluation techniques for social dialogue systems. In *Proceedings of the 1st ACM SIGCHI International Workshop on Investigating Social Interactions with Artificial Agents*, pages 25–26, 2017.

[52] Daniel Lokshtanov and Bernardo Subercaseaux. Wordle is np-hard, 2022.

[53] Richard S Sutton, Andrew G Barto, et al. Introduction to reinforcement learning. 1998.

[54] Abhishek Das, Satwik Kottur, Khushi Gupta, Avi Singh, Deshraj Yadav, José M. F. Moura, Devi Parikh, and Dhruv Batra. Visual dialog, 2016.

[55] Abhishek Das, Satwik Kottur, José M. F. Moura, Stefan Lee, and Dhruv Batra. Learning cooperative visual dialog agents with deep reinforcement learning, 2017.

[56] Ashvin Nair, Abhishek Gupta, Murtaza Dalal, and Sergey Levine. Awac: Accelerating online reinforcement learning with offline datasets, 2020.

[57] Ziyu Wang, Alexander Novikov, Konrad Zolna, Jost Tobias Springenberg, Scott Reed, Bobak Shahriari, Noah Siegel, Josh Merel, Caglar Gulcehre, Nicolas Heess, and Nando de Freitas. Critic regularized regression, 2020.

[58] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.

[59] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor, 2018.

[60] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[61] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2018.

[62] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2017.

[63] Felix A. Gers, Jürgen Schmidhuber, and Fred Cummins. Learning to Forget: Continual Prediction with LSTM. *Neural Computation*, 12(10):2451–2471, 10 2000.

[64] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4rl: Datasets for deep data-driven reinforcement learning, 2020.

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

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes] see Section 6
   (b) Did you describe the limitations of your work? [Yes] see Section 7
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] see Section 7
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes] I've read it and our paper obeys these guidelines.
2. If you are including theoretical results...

   (a) Did you state the full set of assumptions of all theoretical results? [N/A] We have no theoretical results.
   (b) Did you include complete proofs of all theoretical results? [N/A] We have no proofs.
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes] see supplemental
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes] see Appendix
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes] see Section 6
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] see Appendix
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...
   (a) If your work uses existing assets, did you cite the creators? [Yes] We cite all datasets and models used in Section 6.
   (b) Did you mention the license of the assets? [No] They are open sourced.
   (c) Did you include any new assets either in the supplemental material or as a URL? [Yes] See supplemental for code.
   (d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] All assets used are open source.
   (e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [Yes] We use potentially toxic internet text for some experiments; we discuss this in Section 6.
5. If you used crowdsourcing or conducted research with human subjects...
   (a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A] We do not conduct experiments with human subjects.
   (b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A] We do not conduct experiments with human subjects.
   (c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A] We do not conduct experiments with human subjects.

A Appendix

A.1 Justifications for Offline RL in Dialogue

Dialogue tasks are one of the most rich and interactive settings in NLP, and as we will argue, these properties make it an ideal target for applying offline RL. RL in general presents an elegant and highly desirable utility optimization framework for sequential decision making settings, such as dialogue. However, solving realistic interactive tasks with online RL requires either repeated real-world interaction or building a realistic simulator of the environment. In the case of dialogue, such online interaction means communicating with real humans, which may be impractically expensive and time-consuming with contemporary sample-inefficient online RL methods [58, 59], and building a realistic simulator of human responses may be largely intractable in sufficiently rich or complex dialogue settings. Offline RL, on the other hand, avoids both of these heavy requirements, by, just as many recent breakthroughs in the field of NLP [2, 60, 61], operating purely on previously collected data, which is wildly available on the internet in general. Offline-RL therefore presents an ideal approach for flexibly steering language models towards the successful completion of dialogue tasks in a way that effectively leverages existing data, just as supervised learning does.

A.2 Full Token-Level POMDP Formulation

We expand on the POMDP definition presented in Section 3. In order to apply RL to interactive language settings, we need to formalize dialogue generation and other NLP tasks as a partially observable Markov decision processes (POMDP). We define the POMDP $\mathcal{M}$ at the token level with $\mathcal{M} = (\mathcal{S}, \mathcal{A}, \mathcal{O}, \mathcal{T}, \mathcal{Z}, \mu_0, \mathcal{R}, \gamma)$. We define the agent's observation $h_t \in \mathcal{O}$ to be a history of tokens with $h_t = \{t_0, t_1, t_2, t_3, ...t_{t-1}\}$; the action space $a_t = t_t \in \mathcal{A}$ is defined to be the set of possible next-tokens in our vocabulary, which includes the special end-of-turn token a_{end}. The agent's policy then corresponds to a mapping $\pi : \mathcal{O} \rightarrow \mathcal{P}(\mathcal{A})$. Many tasks such as dialogue have an underlying state s_t that goes beyond just the sequence history, which can encompass things like the speaker's mental state. The environment transitions $\mathcal{T}(\cdot|s_t, a_t)$ are defined as a function of this s_t. In particular, in domains, such as dialogue, the dynamics are trivial within the agent's utterance (the selected token is deterministically appended to the history), but when the policy outputs a special "end of turn" token, the other speaker gets a turn, which is subsequently appended to the history. In other tasks, where the goal is to generate a single utterance, such as generating a summary or a single Reddit comment, the episode ends when the policy produces the end token. The agent receives a reward, defined by $R(s_t, a_t) \rightarrow \mathcal{R}$, after each action taken. However, in all the settings we consider, the agent receives non-zero reward r_t only after producing an "end of turn" token, rather than densely at every token in the agent's utterances.

While some prior works [11, 30] have considered actions at the utterance level, defining decision processes at the token level can yield a more effective search over the exponentially large utterance action space, simply by selecting tokens with high estimated values. Typically, searching over a per-utterance action space requires a Monte Carlo process of sampling multiple full utterances and then re-ranking with estimated values, which can generally bring additional computational complexity at both training and inference time. In Section 6.3, we demonstrated the effectiveness of learning at the token level through an ablation study that compares with learning at the utterance level.

A.3 General Experiment Details

Here we outline architecture and hyper-parameter details of all our models and baselines.

ILQL experiment details. We run all of our experiments on GPT-2 small transformer architectures, with the supervised learning policy on one transformer and Q and V heads on a separate one. The target Q network is also on a separate transformer. In all our experiments we initialize with GPT-2 pre-trained weights, except in the case of Wordle, where we initialize randomly. Additionally, Wordle uses a different token set: the set of 26 characters, plus an additional token for each "color". We train all RL baselines with double-Q learning, using two separate heads on the same transformer model as the two Q-functions. Our target Q networks are Polyak-averaged with decay factor 0.005 for both the transformer and the Q function head. We use $\gamma = 0.99$ for all offline-RL experiments. All value function heads are two layer MLPs with hidden dimension twice that of the transformer's embedding dimension. Our MLPs used ReLU non-linearities and no dropout. We used a learning rate of 1e-4 on the Reddit and Visual Dialogue tasks and 1e-5 on the Wordle task. We used no weight decay in training any of our models, and we used a dropout of 0.1 inside the transformer (i.e. the same dropout setting that was used for pretraining GPT2). We trained all Wordle models with a batch size of 1024, all Visual Dialogue models with a batch size of 64, and all Reddit models with a batch size of 32. We always truncate token sequences to length 1024, except on Reddit tasks, in which we truncate to length 512.

Except on the Reddit comment task, we train ILQL on each of $\tau = \{0.7, 0.8, 0.9\}$, and we also evaluate each on $\beta = \{4, 8, 16\}$. On the Reddit comment tasks, we only train with $\tau = 0.6$ and evaluate on $\beta = \{1, 2, 4, 8, 16, 32\}$. On all tasks, we report the setting with the greatest performance.

For the NLL (CQL) loss term applied to ILQL, we used a weight α of 1.0 on all VisualDialogue experiments, 0.25 on all Reddit Comment experiments, and 0.0001 on Wordle. These values were tuned by hand. Generally, we find this loss parameter to not be too critical to performance; we tune it a little at first for each task and then don't worry about it.

All ILQL models and all baselines were trained on a single GPU until convergence. Training never exceeded three days.

Evaluation details. During evaluation, we use greedy decoding to generate utterances on all tasks and baselines, except the Reddit Comments tasks, where we sample instead. All experiments are evaluated on 1024 task-instances from an unseen evaluation set. We use our BC baseline model as π_β for guiding ILQL's perturbation-based policy extraction.

BC baselines. We train BC baselines with the same optimization parameters (i.e., weight decay, dropout, learning rate, batch size) and initialization as ILQL. We use early stopping: when the validation loss exceeds the training loss, we stop training. Unlike our ILQL value function models, we use a linear head on top of the transformer to parameterize our BC policy, as is standard for language model finetuning. The only difference between our BC and standard language model training is that instead of finetuning the model to predict the whole sequence of states and actions, we only finetune the model to predict the agent's own actions or utterances.

%BC baselines. For %BC baselines we report the performance of the best model out of several different percentages: $\{10\%, 30\%, 50\%\}$ for Wordle, $\{10\%, 20\%, 30\%\}$ for Visual Dialogue, and for Reddit, since our rewards are discrete, we define $\%BC$ to mean just training on the data-points with the maximum reward label. We use the same hyper-parameters as our BC baselines for training these models.

Decision transformer baseline. Our decision transformer baseline follows from Chen et al. [8], except we initialize with pretrained GPT2 weights. All hyperparameters are identical to those used to train our BC baselines. To evaluate decision transformer on our Visual Dialogue "y/n" reward, we swept over a broad range of conditional reward-to-go values: $\{-11, -10, -9, -8, -7, -6, -5, -4, -3, -2, -1, 0\}$. We report the setting with the best performance.

SARSA baselines. SARSA baselines are implemented as ILQL with $\tau = 0.5$ and all other hyper-parameters are identical to those used with ILQL as described above. Except on the Reddit comment tasks, we evaluate all SARSA models on $\beta = \{4, 8, 16\}$, and report the setting with the greatest performance. On the Reddit comment tasks, we show the best performance from $\beta = \{1, 2, 4, 8, 16, 32\}$.

Per-utterance ILQL. For "ILQL (utterance)", we train models with $\tau = \{0.7, 0.8, 0.9\}$. We also evaluate each model with number of EMAQ-style [29] samples chosen from N=$\{4, 8, 16\}$. We report the setting with the best task performance. "SARSA (utterance)" is a special case of "ILQL (utterance)" with $\tau = 0.5$, which we evaluate on each of N=$\{4, 8, 16\}$ and report the setting with the best performance. The architecture for "ILQL (utterance)" and "SARSA (utterance)" is largely identical to that of per-token ILQL, with the main difference being that Bellman backups are performed at the utterance level instead of the token-level. As a result of this difference, Q-heads map to a scalar at the end of an utterance instead of a vector at every token with length equal to the size of the vocabulary.

CHAI baseline. Our CHAI baseline is adopted from Verma et al. [11]. The tasks we consider only require utterance actions; no axuiliary actions, like the price proposal action required by that of Verma et al. [11]'s bargaining task. We therefore only adopt the components from CHAI relevant to utterance level actions. In order to compute CHAI's CQL loss at the utterance level, we need to sample counterfactual utterances for each action in the training data, which can be highly expensive and can greatly slow training. Following Verma et al. [11], we amortize this cost at the risk of inducing some bias by caching 5 counterfactual samples for each action in the training data as a preprocessing step. In our case, this preprocessing step took over 30 hours to execute on a V-100 GPU for the full the Visual Dialogue training set. As in all our other experiments, we train two Q networks [48], where our target Q value is parameterized as the minimum of both Polyak averaged target Q heads. We train our CHAI models with a batch size of 16 and otherwise all other hyperparameters are identical to those used with ILQL. We train models with CQL $\alpha = \{0.1, 1.0, 10.0\}$, and we evaluate each model with the number of EMAQ-style [29] samples chosen from N=$\{4, 8, 16\}$. We report the setting with the best performance.

Per-token dynamic programming baselines. For our per-token CQL and ψ-learning baselines in Table 10, we tuned the CQL loss weight with $\alpha = \{0.1, 1.0, 10.0\}$, and the ψ-learning reward scale with $c = \{0.1, 1.0, 10.0\}$. For each baseline agent, we evaluated using ILQL's policy extraction with $\beta = \{4, 8, 16\}$ and also evaluated by greedily selecting tokens with the Q function by itself. We report the setting with the best performance for each baseline.

Our implementation of per-token CQL is identical to ILQL with the only exception being that for per-token CQL the loss function is defined as:

$$L_{Q,V}(\theta) = \mathbf{E}_{\tau \sim D}\left[\sum_{i=0}^{T}(R(h_i, a_i) + \gamma \max_{a_{t+1} \in \mathcal{A}} Q_{\hat{\theta}}(h_{i+1}, a_{t+1}) - Q_\theta(h_i, a_i))^2\right]$$

Our implementation of ψ-learning is adapted from Jaques et al. [12, 13] for use on transformer language models [62, 60] instead of RNNs [63]. The architecture is identical to that of ILQL, the main difference is in the loss function:

$$L_{Q,V}(\theta) = \mathbf{E}_{\tau \sim D}\left[\sum_{i=0}^{T} L_\delta(\frac{R(h_i, a_i)}{c} + \log(\pi_\beta(h_i, a_i)) + \gamma \log(\sum_{a_{t+1} \in \mathcal{A}} \exp Q_{\hat{\theta}}(h_{i+1}, a_{t+1})) - Q_\theta(h_i, a_i))\right]$$

Where π_β is our BC baseline model: a transformer language model trained with supervised learning. And L_γ defines the Huber loss; we use $\gamma = 1$ in our experiments.

In both baselines, we also fit a value function head to the mean of the Q functions, as in ILQL with $\tau = 0.5$. Additionally, for both baselines, aside from the parameters mentioned, all other parameters are identical to those used with ILQL, as described above. The only exception being that for ψ-learning, we used a learning rate of 1e-5 instead of 1e-4 due to training instability with the higher learning rate.

We generally found ψ-learning to be highly unstable to train in our experiments, often producing incomprehensible outputs. It is possible that the baseline could work better with even more careful tuning.

AWR extraction abalation details. For our "ILQL (AWR)" ablation, we extracted a policy with AWR extraction using the best performing ILQL value function out of those trained with $\tau = \{0.7, 0.8, 0.9\}$. We performed AWR extraction from this value function using 3 different settings for beta: $\beta = \{4, 8, 16\}$. Using the same value function, we performed ILQL extraction with $\tau = \{4, 8, 16\}$. For each, we reported the setting with the best performance.

A.4 Wordle Task Details

Environment details. Our agents observe the game-state as a history of alternating sequences of 5 letter tokens followed by 5 color tokens. Unlike the actual Wordle game, we do not prevent the agent from generating words that aren't in the vocabulary (i.e., the agent is free to produce any sequence of 5 letters). For our synthetic experiments, we chose to use the full word list given at https://gist.github.com/cfreshman/a03ef2cba789d8cf00c08f767e0fad7bdue to its relatively large size and its use in the actual Wordle game. However, the environment can be configured with any provided list of 5 letter words.

Wordle Twitter dataset details. We outline the details of our natural Wordle dataset scraped from Twitter, introduced in Section 5. Due to Wordle's popularity, we have access to a large amount of natural human data for Wordle (specifically 214,930 games), scraped from tweets [2]. Existing offline RL benchmarks are composed of purely synthetic data [64], and as a result, it may be unclear how well offline RL algorithms work on more natural data distributions. We therefore present

[2]we use the wordle tweet dataset provided here: https://www.kaggle.com/code/benhamner/wordle-1-6/notebook

this human Wordle dataset as a more naturalistic offline RL task. While the scraped Tweets don't display the actual words used by human players, only the sequence of transition colors given by the environment, we can retrofit valid words onto these tweets to produce a dataset of full trajectories. Note that the words we retrofit may not necessarily be the natural words human players would have used, but the dataset still represents the average performance of human players, since the number of turns remains unchanged in this retrofitting process. Additionally, this retrofitting allows us to further multiply the size of the dataset, since typically several different sequences of words can be valid for a given tweet. We can also partially control the difficulty level of the task and dataset by specifying the size and composition of the vocabulary used to retrofit words onto Tweets. In our Wordle human experiments in Table 5, we use a random subset of 200 words from the word list given at https://gist.github.com/cfreshman/a03ef2cba789d8cf00c08f767e0fad7b.

A.5 Visual Dialogue Task Details

Here we detail the task setup and reward functions used in our Visual Dialogue experiments in Section 6.1. We use Das et al.'s code [3] to produce generations and to predict image embeddings from the provided supervised learning answer and question bots, respectively. We integrate these components of Das et al.'s codebase into ours by wrapping the relevant functionality of Das et al.'s codebase in a flask webserver interface that is then queried by our system.

As described in Section 6.1, our "standard" reward function gives a reward of -1 for each turn in which the true image is sufficiently difficult to predict from the dialogue, otherwise the agent receives a reward of 0 and the environment interaction ends. We firstly formalize this notion of "sufficiently difficult to predict".

The standard reward is based on the relative percentile ranking of the ground truth image's distance from the predicted embedding among a set of images taken from the evaluation set. We give a -1 reward to our agent for every turn in which $(1 - p_t) < (1 - p_0) * 0.5$, where p_t is the ground truth image's percentile rank at dialogue turn t and p_0 is the ground truth image's percentile rank at the beginning of the dialogue, when only the image caption is observed. Otherwise, the agent gets a reward of 0 and the episode ends. This condition effectively rewards the agent once the ground truth image is preferred over 50% of the images that were preferred over it at initialization. The agent should learn to ask as many good questions as possible to get the episode to successfully end as early as possible. In initial experiments, we found it took a very long time to train good value functions for rewards based on the absolute Euclidean distance alone, as used by Das et al. [55], so to make it faster to iterate, we used the relative distance formulation described above.

Our "y/n" reward adds, on top of the "standard" reward, a reward of -2 for every question that results in a response that exactly matches the strings "yes" or "no".

Our "conservative y/n" reward instead aims to provide a more conservative, higher-recall lower-precision penalty to any question which might be a yes/no question. This accounts for the fact that people often answer yes/no questions with longer phrases (e.g., "It appears so"). This reward function provides a reward of -2 if any of the following words are sub-strings of the response to the agent's question: "not", "don't", "can't", "cannot", "fairly", "could", "think so", "okay", "maybe", "yes", "no", "looks", "appears", "tell", "mostly just". All of these were determined by hand to be words/phrases that occur often in answers to questions that are effectively yes/no questions.

A.6 Reddit Reward Model Details

We outline the details of our reward functions for the Reddit tasks presented in Section 6.2.

Toxicity Reward. Our toxicity filter reward uses OpenAI's API [4], which provides a free toxicity filter, meant for developers building applications off the GPT3 API to use to block toxic inputs or generations. We assign a reward of -10 for comments labeled as toxic (scored as 2), -5 for comments labeled as moderately toxic (scored as 1), and 0 for comments labeled as non-toxic (scored as 0).

Upvote Model Reward. Our upvote reward function is finetuned from RoBERTa-base [65] with a learning rate of 1e-5 and a batch size of 64. Since our reward functions are binary, we train with binary cross entropy loss. Like our value function heads in ILQL, we predict the reward as a scalar from a 2-layer MLP on top of the RoBERTa transformer, with hidden dimension twice that of the transformer, ReLU non-linearity, and no dropout. We truncate token sequences to maximum length

[3]https://github.com/batra-mlp-lab/visdial-rl

[4]https://openai.com/api/

train/eval	toxicity	upvotes model
toxicity	**0.0**±0.0	9.07±0.09
upvotes gold	-5.00±0.00	9.83±0.04
upvotes model	-5.00±0.00	**10.0**±0.0

Table 3: Evaluating each Reddit ILQL agent on all other reward functions. Agents trained on one reward function are less optimal on rewards for which they were not trained.

method	toxicity	noised toxicity	upvotes real	upvotes model
ILQL	**0.0**±0.0	**0.0**±0.0	**9.83**±0.04	**10.0**±0.0
SARSA	**0.0**±0.0	**0.0**±0.0	6.23±0.15	**10.0**±0.0
%BC	-0.74±0.07	-1.61±0.11	7.06±0.14	7.86±0.13
BC	-3.51±0.13	-3.48±0.15	4.87±0.16	4.87±0.16

Table 4: A comparison of ILQL against baselines on the various Reddit comments reward functions. ILQL manages to never generate undesirable comments on 3 out of 4 reward functions, whereas fine-tuning on filtered data occasionally does.

256. At inference time, we predict a reward of +10 if the model's reward logit is ≥ 0 and a reward of 0 otherwise. We used binary (positive or negative) rewards for upvotes instead of the more natural cardinal numeric representation, because different sub-reddits can have drastically different upvote counts depending on the sub-reddit's population, and our binarization (positive or negative upvotes) is invariant to these differences in scale. However, this binarization is not the only normalization that we could have used to overcome this issue.

A.7 Evaluating Reddit Agents on Different Rewards

To complement Table 1 right, comparing Visual Dialogue agents on all Visual Dialogue reward functions, we also present Table 3, showing how different Reddit agents perform when evaluated on different rewards. Here we find the same result, that agents trained on one reward function are generally less optimal for others, further confirming ILQL as an effective utility optimizer on natural language tasks.

A.8 Noisy Rewards

In figure 11, we present a more detailed visual explanation for why offline RL outperforms filtered finetuning on our Reddit tasks in section 6.2. As our results in Table 8 show, offline RL consistently outperforms finetuning on filtered data on this task. We hypothesize that this is due to offline RL's ability to effectively reason about the inherently stochastic and subjective rewards functions present in these tasks. In these high-variance reward settings, simply filtering or curating datasets for exclusively high-reward examples can fail to produce desirable outputs, since such filtered finetuning approaches do not make the model aware of the reward uncertainty. Put another way, training on curated datasets that exclude low-reward examples doesn't teach the model about what *not* to generate, whereas models that are aware of the reward, such as Q-learning, directly learn to relate actions to their expected reward values, averaging out uncertainty and stochasticity. This can enable models to avoid outputs that have low but non-trivial probability of undesirable outcomes (e.g., toxicity). Offline RL in this sense is able to find the safest outputs, whereas fine-tuning on filtered data does not explicitly express such a preference.

To further test this hypothesis, we artificially add further noise to the toxicity reward function. The standard toxicity reward assigns all comments one of three reward values: 0, indicating the comment is non-toxic; -5, indicating the comment is moderately toxic; -10, indicating the comment is highly toxic. We now relabel the reward for all comments originally given a -5 reward, randomly to either 0 or 10 with equal probability.

Since some of the moderately toxic comments get relabeled with reward=0, they would be included in the %BC training set, whereas offline RL should learn to represent uncertainty about such comments and thus push away from the stochastic "middle ground" of this reward function. In Table A.8 in the "noised toxicity" column, we see that our offline RL agents learn to never generate toxic outputs despite the additional noise, and in Figure 9 we can see qualitatively that offline RL assigns low advantages to potentially negative or toxic words/phrases that were incidentally generated by the %BC model. All of this goes to support our hypothesis that the advantage of offline RL over filtered supervised learning in these settings lies in its improved ability to handle reward uncertainty.

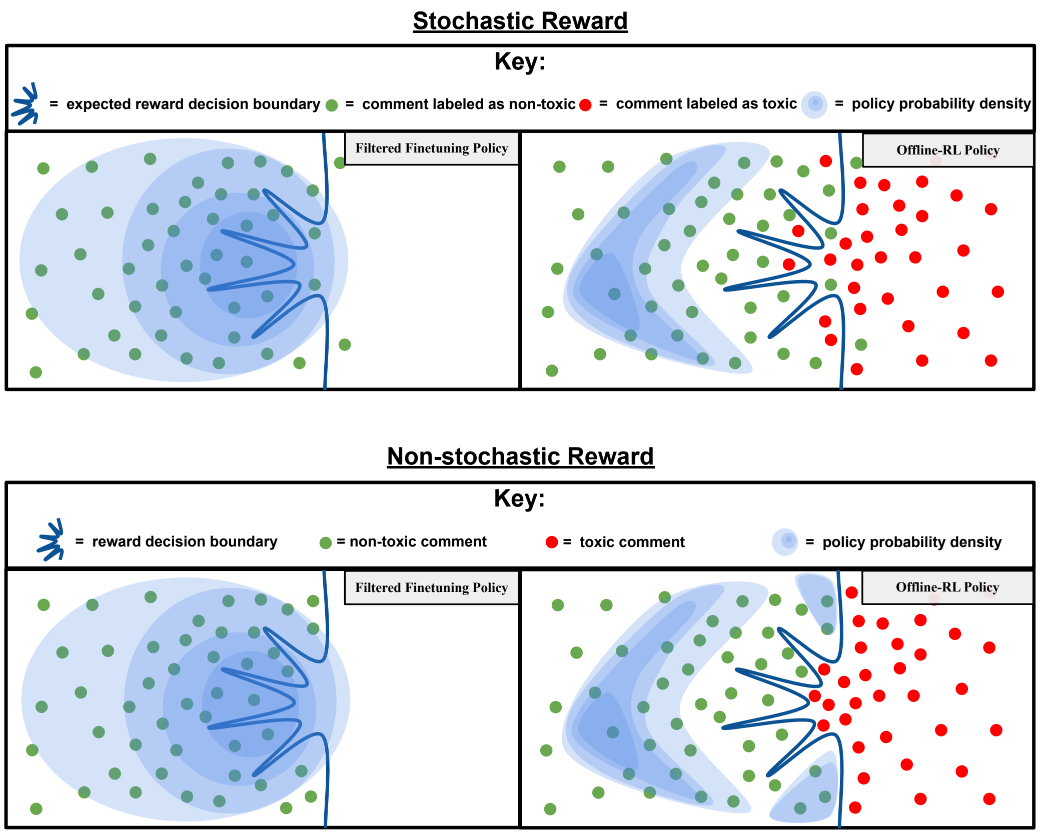

Stochastic Reward
Key:
= expected reward decision boundary
● = comment labeled as non-toxic
● = comment labeled as toxic
= policy probability density
Filtered Finetuning Policy
Offline-RL Policy
Non-stochastic Reward
Key:
= reward decision boundary
● = non-toxic comment
● = toxic comment
= policy probability density
Filtered Finetuning Policy
Offline-RL Policy

Figure 11: A visual explanation of offline RL's ability to optimize high variance reward functions based on subjective judgement, such as whether to label a comment as an example of toxic speech or not. Finetuning on filtered data accidentally generalizes into producing undesirable outputs, whereas offline RL is able to find the "safe" outputs. Top: In the case of stochastic rewards, offline RL learns to avoid the highly stochastic regions of the action space, whereas filtered finetuning will explicitly learn to imitate undesirable outputs that were stochastically given a positive reward in the training data, thus leading to suboptimal behavior. Right: In the case of non-stochastic but sharp-boundary reward functions, ILQL is still able to integrate into its Q values uncertainty about actions near the sharper parts of the reward function's decision boundary, thus avoiding these regions. Finetuning on filtered data expresses no such preference and thus risks generalizing into occasionally producing undesirable outputs.

A.9 Trading Off Output Diversity for Optimization

An advantage of our novel policy extraction mechanism is that we can flexibly tune the parameter β at inference time, directly trading off between random generation and optimality. As discussed in Section 12, this parameter controls a constraint on our policy's deviation from the data distribution. As we increase β, the resulting policy will be more strongly influenced by the Q-function, and as we decrease β, it should approach the data distribution. Beyond the risk in diverging too far from the data, another potential downside to increasing β is that the resulting policy distribution will become more deterministic. In some settings, such as chit-chat dialogue, we may desire policies capable of producing diverse and interesting outputs, so such a deterministic, highly-optimized agent would be undesirable.

We demonstrate in Figure 12, using the Reddit Toxicity task, how varying β can modulate the diversity of the generations produced by our policy, as measured by its entropy, at the cost of a small decrease in performance. We show that as we increase β, while the policy's performance generally increases, the entropy decreases and subsequently so does the interestingness and diversity of the language model's outputs. At inference time, we can tune this parameter to trade-off optimization for output diversity as we desire.

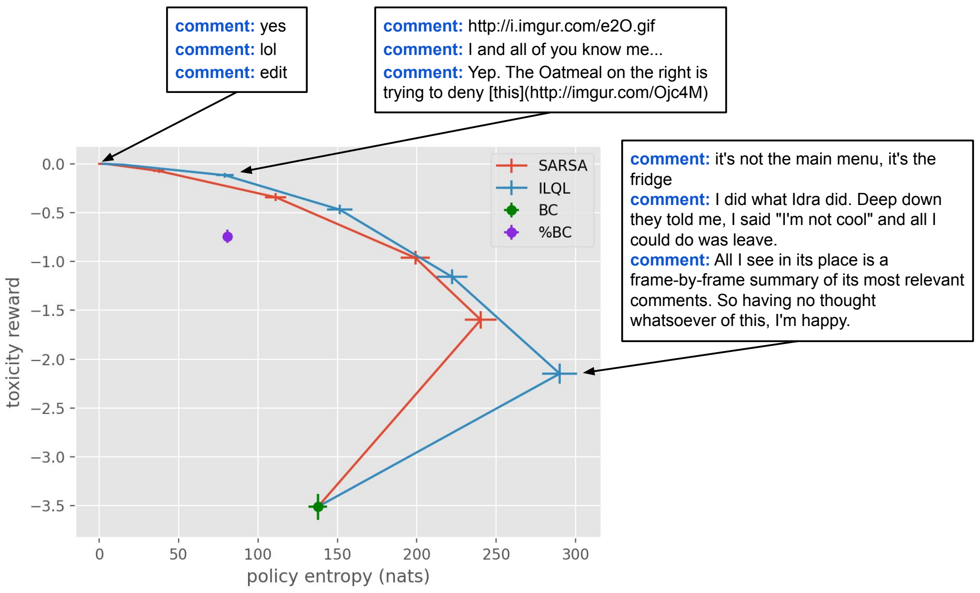


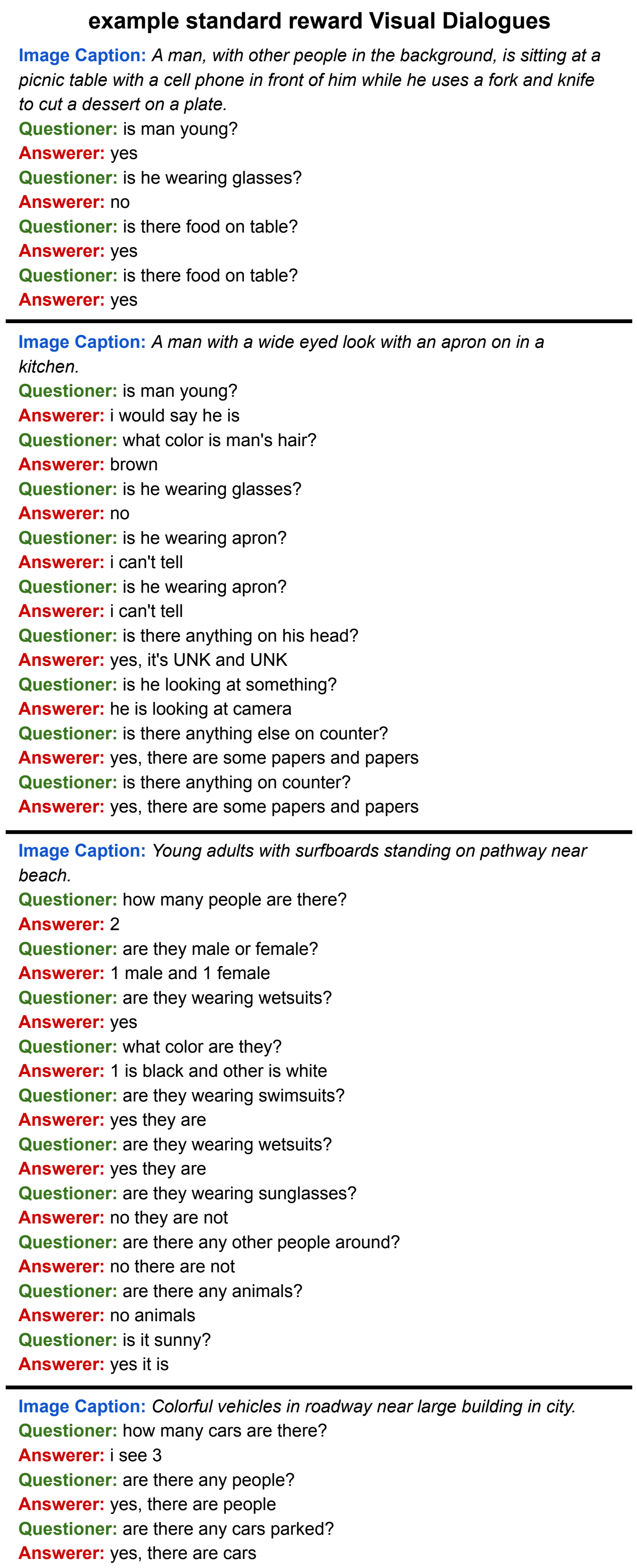
Figure 12: By varying β at inference time, we can interpolate the trade-off between offline RL "optimization" and output diversity.

A.10 Visual Dialogue Example Dialogues and Histogram

In Figures 13, 14,and 15 we present a set of selected representative example dialogues produced by our best performing ILQL agent on each reward. In Figure 16 we present a histogram showing often different agents generate yes/no questions as judged by their respective “y/n” or “y/n conservative” reward functions.

A.11 Comprehensive Result Tables

In Tables 5, 6,and 7 we present comprehensive results for all hyper-parameter settings for all baselines on all tasks.



example standard reward Visual Dialogues

Image Caption: *A man, with other people in the background, is sitting at a picnic table with a cell phone in front of him while he uses a fork and knife to cut a dessert on a plate.*
Questioner: is man young?
Answerer: yes
Questioner: is he wearing glasses?
Answerer: no
Questioner: is there food on table?
Answerer: yes
Questioner: is there food on table?
Answerer: yes

Image Caption: *A man with a wide eyed look with an apron on in a kitchen.*
Questioner: is man young?
Answerer: i would say he is
Questioner: what color is man's hair?
Answerer: brown
Questioner: is he wearing glasses?
Answerer: no
Questioner: is he wearing apron?
Answerer: i can't tell
Questioner: is he wearing apron?
Answerer: i can't tell
Questioner: is there anything on his head?
Answerer: yes, it's UNK and UNK
Questioner: is he looking at something?
Answerer: he is looking at camera
Questioner: is there anything else on counter?
Answerer: yes, there are some papers and papers
Questioner: is there anything on counter?
Answerer: yes, there are some papers and papers

Image Caption: *Young adults with surfboards standing on pathway near beach.*
Questioner: how many people are there?
Answerer: 2
Questioner: are they male or female?
Answerer: 1 male and 1 female
Questioner: are they wearing wetsuits?
Answerer: yes
Questioner: what color are they?
Answerer: 1 is black and other is white
Questioner: are they wearing swimsuits?
Answerer: yes they are
Questioner: are they wearing wetsuits?
Answerer: yes they are
Questioner: are they wearing sunglasses?
Answerer: no they are not
Questioner: are there any other people around?
Answerer: no there are not
Questioner: are there any animals?
Answerer: no animals
Questioner: is it sunny?
Answerer: yes it is

Image Caption: *Colorful vehicles in roadway near large building in city.*
Questioner: how many cars are there?
Answerer: i see 3
Questioner: are there any people?
Answerer: yes, there are people
Questioner: are there any cars parked?
Answerer: yes, there are cars

Figure 13: Example dialogues produced by the best performing ILQL agent on the Visual Dialogue "standard" reward.

example y/n reward Visual Dialogues

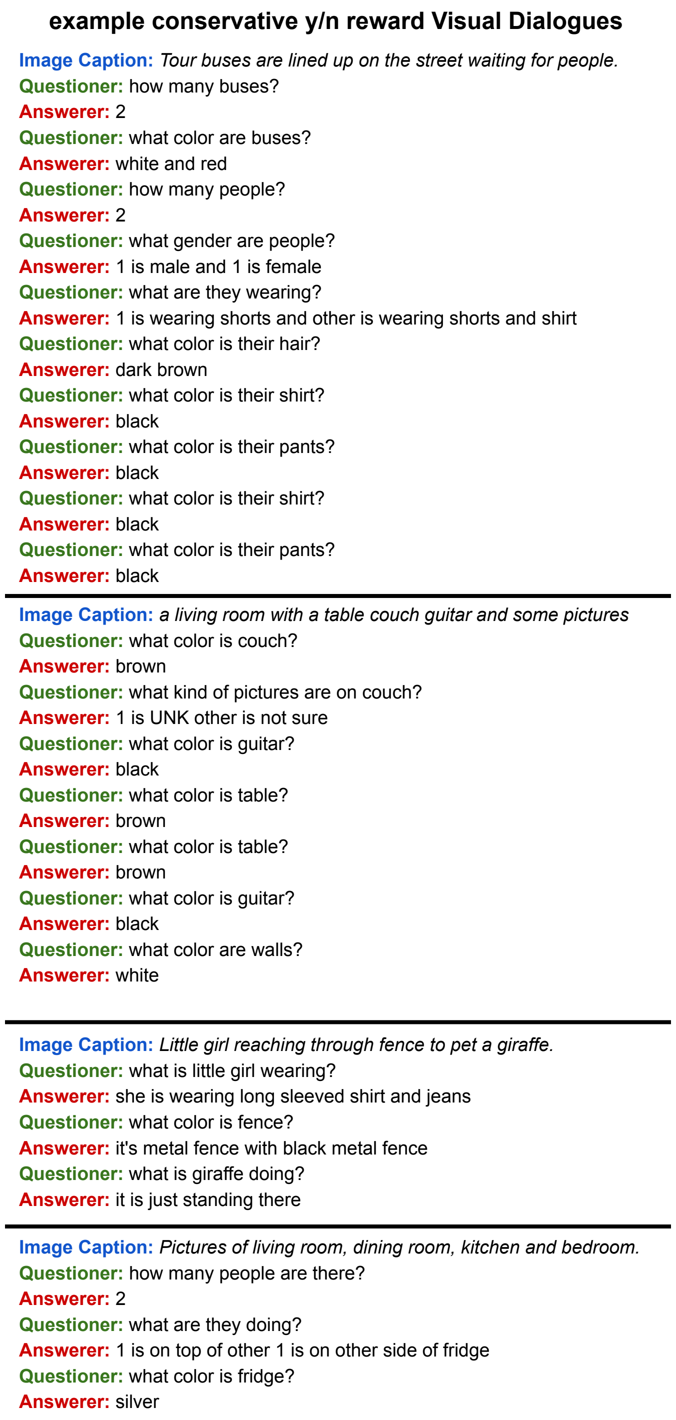
Image Caption: *Yellow and blue passenger train rounding a curve.*
Questioner: how many cars on train?
Answerer: i can only see 1
Questioner: is it day or night?
Answerer: day
Questioner: is it sunny or cloudy?
Answerer: sunny
Questioner: is there anything else on tracks?
Answerer: no, just birds
Questioner: how many birds are there?
Answerer: 2
Questioner: what color are birds?
Answerer: black
Questioner: what color is sky?
Answerer: blue
Questioner: what time of day is it?
Answerer: it is daytime
Questioner: what time of year is it?
Answerer: i can't tell
Questioner: what else can you see?
Answerer: trees, sky, sky, sky

Image Caption: *A person flying a kite over a sand dune.*
Questioner: what color is kite?
Answerer: it's black and white
Questioner: is person male or female?
Answerer: can't tell
Questioner: what is person wearing?
Answerer: i can't tell
Questioner: what color is sand?
Answerer: dark brown
Questioner: is person flying kite flying in air?
Answerer: yes
Questioner: what color is kite?
Answerer: black

Image Caption: *Little girl reaching through fence to pet a giraffe.*
Questioner: how old is girl?
Answerer: maybe 3
Questioner: what color is her hair?
Answerer: dark brown
Questioner: what is she wearing?
Answerer: i can only see her from waist up
Questioner: what color is her hair?
Answerer: dark brown
Questioner: what is her eye color?
Answerer: i can't tell, it's covered with white and black and red
Questioner: what is fence made of?
Answerer: wood
Questioner: what color is fence?
Answerer: it's metal fence with black metal fence
Questioner: is giraffe full grown?
Answerer: yes

Image Caption: *a number of people in a field with a kite in the air*
Questioner: how many people are there?
Answerer: i see 4
Questioner: what color is kite?
Answerer: it's hard to tell, but it looks black
Questioner: are they male or female?
Answerer: both are male
Questioner: what are they wearing?
Answerer: they are wearing shorts and t shirts
Questioner: what color are shorts?
Answerer: black

Figure 14: Example dialogues produced by the best performing ILQL agent on the Visual Dialogue "y/n" reward.



example conservative y/n reward Visual Dialogues

Image Caption: *Tour buses are lined up on the street waiting for people.*
Questioner: how many buses?
Answerer: 2
Questioner: what color are buses?
Answerer: white and red
Questioner: how many people?
Answerer: 2
Questioner: what gender are people?
Answerer: 1 is male and 1 is female
Questioner: what are they wearing?
Answerer: 1 is wearing shorts and other is wearing shorts and shirt
Questioner: what color is their hair?
Answerer: dark brown
Questioner: what color is their shirt?
Answerer: black
Questioner: what color is their pants?
Answerer: black
Questioner: what color is their shirt?
Answerer: black
Questioner: what color is their pants?
Answerer: black

Image Caption: *a living room with a table couch guitar and some pictures*
Questioner: what color is couch?
Answerer: brown
Questioner: what kind of pictures are on couch?
Answerer: 1 is UNK other is not sure
Questioner: what color is guitar?
Answerer: black
Questioner: what color is table?
Answerer: brown
Questioner: what color is table?
Answerer: brown
Questioner: what color is guitar?
Answerer: black
Questioner: what color are walls?
Answerer: white

Image Caption: *Little girl reaching through fence to pet a giraffe.*
Questioner: what is little girl wearing?
Answerer: she is wearing long sleeved shirt and jeans
Questioner: what color is fence?
Answerer: it's metal fence with black metal fence
Questioner: what is giraffe doing?
Answerer: it is just standing there

Image Caption: *Pictures of living room, dining room, kitchen and bedroom.*
Questioner: how many people are there?
Answerer: 2
Questioner: what are they doing?
Answerer: 1 is on top of other 1 is on other side of fridge
Questioner: what color is fridge?
Answerer: silver

Figure 15: Example dialogues produced by the best performing ILQL agent on the Visual Dialogue "conservative y/n" reward.

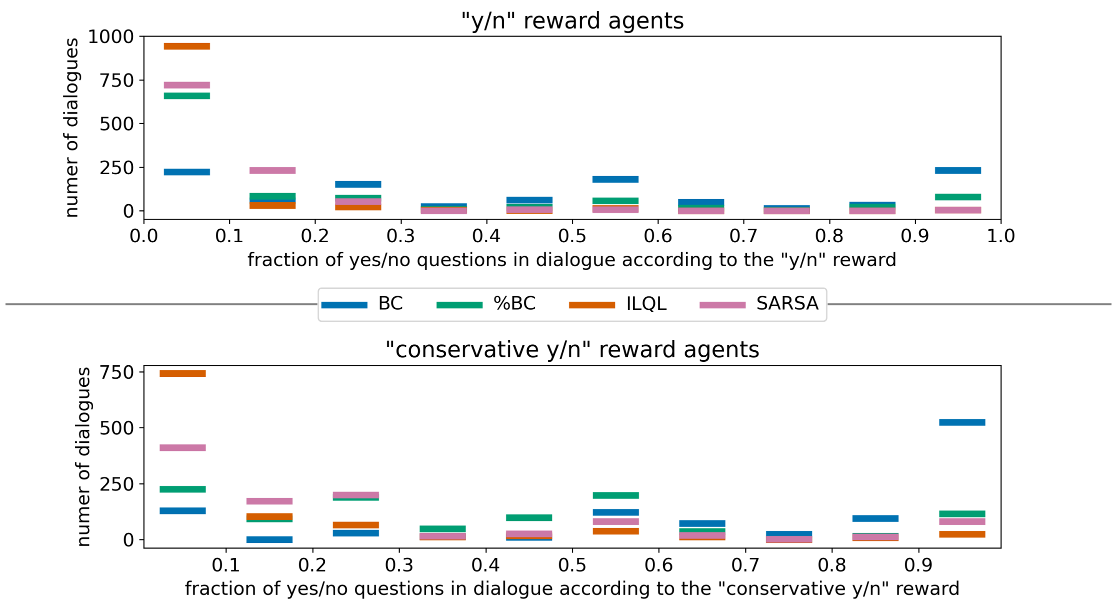


Figure 16: Top: Histogram of the fraction of yes/no questions asked per dialogue by Visual Dialogue agents trained on the "y/n" reward. Here yes/no questions are determined by the same exact match heuristic used by the "y/n" reward. ILQL agents ask fewer questions triggered as being yes/no than baselines. The best performing agent is used for all methods. Bottom: Histogram of the fraction of yes/no questions asked per dialogue by Visual Dialogue agents trained on the "conservative y/n" reward. Here yes/no questions are determined by the same exact match heuristic used by the "conservative y/n" reward. ILQL agents ask fewer questions triggered as being yes/no than baselines. The best performing agent is used for all methods.

model	standard	y/n	conservative y/n
ILQL $\tau = 0.7, \beta = 4$	-5.23±0.13	-6.65±0.18	-7.64±0.21
ILQL $\tau = 0.7, \beta = 8$	**-5.22**±0.13	-5.92±0.15	-7.05±0.19
ILQL $\tau = 0.7, \beta = 16$	-5.28±0.13	**-5.69**±0.13	-6.77±0.18
ILQL $\tau = 0.8, \beta = 4$	-5.30±0.12	-6.88±0.18	-7.97±0.21
ILQL $\tau = 0.8, \beta = 8$	-5.40±0.13	-6.28±0.16	-6.85±0.18
ILQL $\tau = 0.8, \beta = 16$	-5.38±0.13	-5.99±0.15	-6.65±0.18
ILQL $\tau = 0.9, \beta = 4$	-5.35±0.13	-7.01±0.19	-7.41±0.20
ILQL $\tau = 0.9, \beta = 8$	-5.40±0.13	-6.35±0.16	-6.72±0.18
ILQL $\tau = 0.9, \beta = 16$	-5.45±0.13	-6.17±0.15	**-6.57**±0.18
SARSA $\beta = 4$	-5.20±0.13	-6.87±0.18	-9.12±0.24
SARSA $\beta = 8$	**-5.14**±0.13	-6.39±0.16	-8.09±0.21
SARSA $\beta = 16$	-5.18±0.13	**-6.19**±0.15	**-7.77**±0.20
10%BC	-5.24±0.12	**-7.48**±0.21	-9.67±0.26
20%BC	**-5.07**±0.13	-8.91±0.24	**-9.13**±0.22
30%BC	-5.16±0.12	-9.10±0.22	-10.52±0.25
BC	**-5.25**±0.13	**-10.85**±0.27	**-15.16**±0.35
CQL $\alpha = 0.1, \beta = 4$	-	-10.08±0.21	-
CQL $\alpha = 0.1, \beta = 8$	-	-10.97±0.21	-
CQL $\alpha = 0.1, \beta = 16$	-	-12.92±0.23	-
CQL $\alpha = 0.1, \beta = \infty$	-	-10.74±0.17	-
CQL $\alpha = 1.0, \beta = 4$	-	-7.84±0.20	-
CQL $\alpha = 1.0, \beta = 8$	-	-7.53±0.19	-
CQL $\alpha = 1.0, \beta = 16$	-	-7.45±0.18	-
CQL $\alpha = 1.0, \beta = \infty$	-	**-7.32**±0.17	-
CQL $\alpha = 10.0, \beta = 4$	-	-11.76±0.29	-
CQL $\alpha = 10.0, \beta = 8$	-	-11.81±0.30	-
CQL $\alpha = 10.0, \beta = 16$	-	-11.84±0.30	-
CQL $\alpha = 10.0, \beta = \infty$	-	-11.81±0.30	-
ψ $c = 0.1, \beta = 4$	-	-11.59±0.18	-
ψ $c = 0.1, \beta = 8$	-	-11.26±0.17	-
ψ $c = 0.1, \beta = 16$	-	-11.42±0.17	-
ψ $c = 0.1, \beta = \infty$	-	-11.51±0.16	-
ψ $c = 1.0, \beta = 4$	-	**-10.05**±0.18	-
ψ $c = 1.0, \beta = 8$	-	**-10.05**±0.18	-
ψ $c = 1.0, \beta = 16$	-	**-10.05**±0.18	-
ψ $c = 1.0, \beta = \infty$	-	**-10.05**±0.18	-
ψ $c = 10.0, \beta = 4$	-	-11.35±0.18	-
ψ $c = 10.0, \beta = 8$	-	-10.99±0.17	-
ψ $c = 10.0, \beta = 16$	-	-10.95±0.17	-
ψ $c = 10.0, \beta = \infty$	-	-10.78±0.17	-
DT R $= -11$	-	-10.57 ± 0.24	-
DT R $= -10$	-	-10.58 ± 0.24	-
DT R $= -9$	-	-10.53 ± 0.24	-
DT R $= -8$	-	-10.49 ± 0.24	-
DT R $= -7$	-	-10.40 ± 0.23	-
DT R $= -6$	-	-10.42 ± 0.23	-
DT R $= -5$	-	-10.30 ± 0.23	-
DT R $= -4$	-	-10.30 ± 0.23	-
DT R $= -3$	-	-9.83 ± 0.22	-
DT R $= -2$	-	-9.54 ± 0.21	-
DT R $= -1$	-	-8.15 ± 0.19	-
DT R $= 0$	-	**-6.70** ± 0.17	-
ILQL (AWR) $\tau = 0.7, \beta = 4$	-	**-5.96**±0.13	-
ILQL (AWR) $\tau = 0.7, \beta = 8$	-	-11.75±0.23	-
ILQL (AWR) $\tau = 0.7, \beta = 16$	-	-12.11±0.22	-
ILQL (utterance) $\tau = 0.7$, N $= 4$	-	-7.08±0.17	-
ILQL (utterance) $\tau = 0.7$, N $= 8$	-	-6.04±0.15	-
ILQL (utterance) $\tau = 0.7$, N $= 16$	-	**-5.89**±0.14	-
ILQL (utterance) $\tau = 0.8$, N $= 4$	-	-7.12±0.17	-
ILQL (utterance) $\tau = 0.8$, N $= 8$	-	-6.15±0.15	-
ILQL (utterance) $\tau = 0.8$, N $= 16$	-	-5.91±0.14	-
ILQL (utterance) $\tau = 0.9$, N $= 4$	-	-7.01±0.17	-
ILQL (utterance) $\tau = 0.9$, N $= 8$	-	-6.12±0.15	-
ILQL (utterance) $\tau = 0.9$, N $= 16$	-	-5.93±0.14	-
SARSA (utterance) N $= 4$	-	-7.82±0.18	-
SARSA (utterance) N $= 8$	-	-7.39±0.17	-
SARSA (utterance) N $= 16$	-	**-7.35**±0.17	-
CHAI $\alpha = 0.1$, N $= 4$	-	-7.18±0.18	-
CHAI $\alpha = 0.1$, N $= 8$	-	-6.18±0.15	-
CHAI $\alpha = 0.1$, N $= 16$	-	-5.62±0.13	-
CHAI $\alpha = 1.0$, N $= 4$	-	-7.03±0.17	-
CHAI $\alpha = 1.0$, N $= 8$	-	-5.87±0.14	-
CHAI $\alpha = 1.0$, N $= 16$	-	**-5.57**±0.13	-
CHAI $\alpha = 10.0$, N $= 4$	-	-8.43±0.18	-
CHAI $\alpha = 10.0$, N $= 8$	-	-8.24±0.16	-
CHAI $\alpha = 10.0$, N $= 16$	-	-8.28±0.15	-

Table 5: All hyper-parameter settings evaluated across all Visual Dialogue tasks, abalations, and baselines. The best performing setting for each baseline of abalation is bolded. $\beta = \infty$ refers to greedily selecting actions with just the Q function. ILQL is generally better performing and more stable than all baseline approaches.

method	toxicity	noised toxicity
ILQL $\tau = 0.6$, $\beta = 1$	-2.15±0.10 / 289.97±10.92	-1.89±0.12 / 260.34±10.35
ILQL $\tau = 0.6$, $\beta = 2$	-1.16±0.08 / 222.18±9.68	-0.99±0.09 / 202.09±8.99
ILQL $\tau = 0.6$, $\beta = 4$	-0.47±0.05 / 151.46±7.86	-0.53±0.07 / 156.85±7.80
ILQL $\tau = 0.6$, $\beta = 8$	-0.12±0.02 / 79.20±5.38	-0.23±0.05 / 73.12±4.88
ILQL $\tau = 0.6$, $\beta = 16$	-0.01±0.01 / 15.46±1.76	-0.03±0.02 / 22.05±1.51
ILQL $\tau = 0.6$, $\beta = 32$	**0.00**±0.00 / 2.00±0.35	**0.00**±0.00 / 9.24±0.16
SARSA $\beta = 1$	-1.60±0.09 / 240.21 ± 10.10	-1.72±0.12 / 233.17±9.89
SARSA $\beta = 2$	-0.96±0.07 / 199.04±9.24	-1.06±0.10 / 187.36±8.82
SARSA $\beta = 4$	-0.34±0.04 / 111.13±6.76	-0.41±0.06 / 108.76±6.53
SARSA $\beta = 8$	-0.07±0.019 / 37.59±3.47	-0.13±0.03 / 41.61±3.52
SARSA $\beta = 16$	**0.00**±0.00 / 4.38±0.54	**0.00**±0.0 / 1.09±0.38
SARSA $\beta = 32$	**0.00**±0.00 / 0.03±0.01	**0.00**±0.00 / 0.00±0.00
%BC	**-0.74**±0.07 / 80.84±3.19	**-1.61**±0.11 / 90.75±3.29
BC	**-3.51**±0.13 / 137.70±5.82	**-3.48**±0.15 / 126.54±5.26

method	upvotes real	upvotes model
ILQL $\tau = 0.6$, $\beta = 1$	6.29±0.15 / 39.11±1.99	8.38±0.12 / 25.15±1.43
ILQL $\tau = 0.6$, $\beta = 2$	7.01±0.14 / 21.47±1.56	9.53±0.07 / 7.82±0.88
ILQL $\tau = 0.6$, $\beta = 4$	7.47±0.14 / 7.91±0.63	9.99±0.01 / 1.38±0.09
ILQL $\tau = 0.6$, $\beta = 8$	9.05±0.09 / 1.55±0.07	**10.00**±0.00 / 0.78±0.02
ILQL $\tau = 0.6$, $\beta = 16$	9.73±0.05 / 0.55±0.14	**10.00**±0.00 / 0.54±0.02
ILQL $\tau = 0.6$, $\beta = 32$	**9.83**±0.04 / 0.18±0.02	**10.00**±0.00 / 0.22±0.02
SARSA $\beta = 1$	**6.23**±0.15 / 33.75±1.64	7.82±0.13 / 28.80±1.87
SARSA $\beta = 2$	4.66±0.16 / 15.81±0.82	8.96±0.10 / 9.34±0.64
SARSA $\beta = 4$	0.81±0.09 / 2.38±0.28	9.93±0.03 / 0.43±0.09
SARSA $\beta = 8$	0.09±0.03 / 0.00±0.00	**10.00**±0.00 / 0.01±0.01
SARSA $\beta = 16$	0.09±0.03 / 0.00±0.00	**10.00**±0.00 / 0.00±0.00
SARSA $\beta = 32$	0.09±0.03 / 0.00±0.00	**10.00**±0.00 / 0.00±0.00
%BC	**7.06**±0.14 / 100.77±4.15	**7.86**±0.13 / 97.51±3.90
BC	**4.87**±0.16 / 127.65±5.48	**4.87**±0.16 / 127.65±5.48

Table 6: All hyper-parameter settings evaluated across all Reddit Comment tasks, abalations, and baselines. The best performing setting for each baseline of abalation is bolded. The left result in each cell the the agent's average reward and the right result is the estimated entropy of the agent's policy, measured in nats. As β is turned up, performance increases, but the entropy (or diversity) of the policy's outputs decreases. ILQL generally learns better performing and more diverse policies than baselines.

method	score
ILQL $\tau = 0.7$, $\beta = 4$	-2.23±0.03
ILQL $\tau = 0.7$, $\beta = 8$	-2.18±0.03
ILQL $\tau = 0.7$, $\beta = 16$	-2.18±0.03
ILQL $\tau = 0.8$, $\beta = 4$	**-2.13**±0.02
ILQL $\tau = 0.8$, $\beta = 8$	**-2.13**±0.03
ILQL $\tau = 0.8$, $\beta = 16$	-2.31±0.03
ILQL $\tau = 0.9$, $\beta = 4$	-2.30±0.03
ILQL $\tau = 0.9$, $\beta = 8$	-2.26±0.03
ILQL $\tau = 0.9$, $\beta = 16$	-2.36±0.03
SARSA $\beta = 4$	-2.24±0.03
SARSA $\beta = 8$	-2.26±0.03
SARSA $\beta = 16$	**-2.23**±0.03
10%BC	-2.93±0.06
30%BC	-3.01±0.06
50%BC	**-2.38**±0.03
BC	**-2.61**±0.03

Table 7: All hyper-parameter settings and baselines evaluated on the human Wordle dataset scraped from Tweets (see Section A.4). The best performing setting for each baseline of abalation is bolded. ILQL is generally better performing than baselines.