Way Off-Policy Batch Deep Reinforcement Learning of Implicit Human Preferences in Dialog

Natasha Jaques, Asma Ghandeharioun, Judy Hanwen Shen, Craig Ferguson,
Agata Lapedriza, Noah Jones, Shixiang Gu, Rosalind Picard
Department of Media Arts and Science
Massachusetts Institute of Technology
Cambridge, MA 02139
jaquesn@mit.edu

Abstract

Most deep reinforcement learning (RL) systems are not able to learn effectively from off-policy data, especially if they cannot explore online in the environment. These are critical shortcomings for applying RL to real-world problems where collecting data is expensive, and models must be tested offline before being deployed to interact with the environment – e.g. systems that learn from human interaction. Thus, we develop a novel class of off-policy batch RL algorithms, which are able to effectively learn offline, without exploring, from a fixed batch of human interaction data. We leverage models pre-trained on data as a strong prior, and use KL-control to penalize divergence from this prior during RL training. We also use dropout-based uncertainty estimates to lower bound the target Q-values as a more efficient alternative to Double Q-Learning. The algorithms are tested on the problem of open-domain dialog generation – a challenging reinforcement learning problem with a 20,000-dimensional action space. Using our Way Off-Policy algorithm, we can extract multiple different reward functions post-hoc from collected human interaction data, and learn effectively from all of these. We test the real-world generalization of these systems by deploying them live to converse with humans in an open-domain setting, and demonstrate that our algorithm achieves significant improvements over prior methods in off-policy batch RL.

1 Introduction

In order to scale deep reinforcement learning (RL) to safety-critical, real-world domains, two abilities are needed. First, since collecting real-world interaction data can be expensive and time-consuming, algorithms must be able to leverage off-policy data – collected from vastly different systems, far into the past – in order to learn. Second, it is often necessary to carefully test a policy before deploying it to the real world; for example, to ensure its behavior is safe and appropriate for humans. Thus, the algorithm must be able to learn offline first, from a static batch of data, without the ability to explore.

This off-policy, *batch reinforcement learning* (BRL) setting represents a challenging RL problem. Most deep RL algorithms fail to learn from data that is not heavily correlated with the current policy [16]. Even models based on off-policy algorithms like Q-learning fail to learn when the model is not able to explore during training. This is due to the fact that such algorithms are inherently optimistic in the face of uncertainty. When value estimates are noisy, taking the maximum over estimates of future reward leads to a persistent overestimation bias. In a normal RL setting, this drives the model to explore areas of the state-action space for which the value estimates have the highest variance, thus enabling it to refine them. In a batch setting where the model cannot explore, it is instead driven to value parts of the state-action space for which it has little to no data to learn a good policy.

Preprint. Under review.

We propose to resolve these issues by leveraging a pre-trained generative model of the state-action space trained on known sequences of interaction data. While training with RL, we penalize divergence from this prior model with different forms of KL-control. We benchmark against a discrete adaptation of Batch Constrained Q (BCQ) [16], a recently proposed BRL algorithm for continuous domains, and show that KL-control achieves superior performance. Finally, we propose using dropout to obtain uncertainty estimates of the target Q values, and use this lower bound to alleviate the Q-learning overestimation bias. This provides a more efficient alternative to Clipped Double Q-Learning [15].

We apply these algorithms to a challenging, under-explored, real-world reinforcement learning problem: using implicitly expressed human reactions in chat to improve open-domain dialog systems. When a machine learning system interacts with humans, ideally we would like to learn about the humans' preferences in order to improve the performance of the system. Yet having humans manually indicate their preferences through explicit means like pressing a button (e.g. [7]) or submitting a feedback report, does not scale. Instead, we would like to be able to use humans' implicit reactions, such as the sentiment they express, or the length of the conversation, in order to improve the policy.

Applying off-policy batch RL to language generation is challenging because the number of potential combinations of words and sentences leads to a combinatorial explosion in the size of the state space. The action space – the set of frequent vocabulary words in the English language – is 20,000-dimensional. This compounds the overestimation problem, making BRL even more difficult. However, when learning from human interactions in the wild, it is crucial to be able to learn offline and test the policy before deploying it, lest it learn inappropriate behaviors (e.g. [26]).

To support this work, we developed an interactive online platform that allows humans to chat with deep neural network dialog models running on GPU; the BRL models trained for this study are available live at `https://neural.chat/rl`. Through this platform we collected human responses to a set of over 40 different dialog models over the course of several months. Using our Way Off-Policy algorithm, we are able to effectively learn from this batch of data, in spite of the fact that it was generated with a vastly different set of model architectures, which were trained on different datasets. Further, we use the batch to learn from many different reward functions designed post-hoc to extract implicit human preferences, something that is only possible with effective off-policy BRL.

2   Related Work

The approach we propose is based on KL-control, a branch of stochastic optimal control (SOC) [58] where the Kullback-Leibler (KL) divergence from some distribution is used to regularize an RL policy (e.g. [1, 33, 48, 61]). Well-known examples include Trust Region Policy Optimization (TRPO) [50], and use conservative, KL-regularized policy updates to restrict the RL algorithm to stay close to its own prior policy (e.g. [22, 32, 44, 48]). KL-control can also be applied to entropy maximization (e.g. [65]); for example, G-learning penalizes KL-divergence from a simple uniform distribution in order to cope with overestimation of Q-values [14]. Soft Q-learning motivates using a Boltzmann distribution in the value function as a way of performing maximum entropy RL [21]. KL-control has also been used to improve transfer learning between maximum likelihood estimation (MLE) training on data, and training with RL [28]. To the best of our knowledge, our work is the first to propose KL-control as a way of improving off-policy learning without exploration in a BRL setting.

Other strategies to improve off-policy learning have been proposed, although many focus on scenarios where the policy is able to explore and collect more data (e.g. [10, 49]). In the deep RL setting, policy gradients can be corrected to account for the difference in the distribution of states visited under the original policy and the learned off-policy algorithm [40]. Covariance-shift-based methods have been adapted to the off-policy deep RL setting to deal with the issue of value divergence [19]. Normalized feature representations have been proposed as an alternative approach [4]. Batch Constrained Q-learning (BCQ) [16] tackles off-policy batch learning in continuous action domains by training a generative model of the batch, $p(a|s)$, sampling from this model, and selecting the best action based on a Q-estimate. This approach fails to integrate information about the distribution $p(a|s)$ directly into the policy, and cannot scale to scenarios in which the state-action space is large, and the amount of available batch data is too small to train $p(a|s)$. Many works from off-policy policy evaluation use importance sampling or model estimation to investigate the problem of estimating the performance of a policy given a batch of off-policy data (e.g. [11, 29, 45, 60]). Effective off-policy

learning gives us the ability to learn from many different rewards post-hoc, something that could potentially improve techniques which use the relabeling trick (e.g. [30, 2]).

We propose using dropout to approximate model uncertainty of the target Q-network. The idea of using dropout to estimate uncertainty in neural networks was first proposed by Gal and colleagues (2016) [17]. Different forms of uncertainty estimates have been used in RL (e.g. [31, 42]); for example, Bayesian uncertainty estimates have been proposed as an alternative to double DQN [3].

Improving dialog systems with RL has largely been restricted to task-oriented dialog systems, which have a limited number of task-specific actions (e.g. [12, 18, 38, 39, 59]). These approaches may incorporate human input, usually through explicit, manual feedback (e.g. [54]), but sometimes with more implicit signals, such as the user interrupting the system or starting over [55]. Attempts to expand RL to the open-domain dialog setting are less numerous. Even in this setting, authors may choose to use a highly restricted action space; for example, using RL to choose which scripted or MLE dialog model to invoke to answer a user's query [51]. Early attempts to apply deep RL to the full vocabulary-sized action space relied mainly on hand-crafted rewards that described qualities of the generated text, such as *ease of answering* [35]. This approach has been extended to use a discriminator trained to distinguish human from generated text as a reward function [36, 37]. While some work has incorporated implicit signals such as sentiment [23] and conversation length [64] in MLE systems, the idea of using such signals as a reward for RL is relatively unexplored. Shin and colleagues uses on-policy learning in conjunction with a user-sentiment approximator to improve a seq2seq model [56], but are unable to learn directly from user feedback. To the best of our knowledge, we are the first to use batch RL to train hierarchical open-domain dialog models on implicit cues gained from real human interactions.

3 Methods

We employ typical RL notation in which s_t represents the environment state at time t, the agent takes action a_t according to its policy $\pi(a_t|s_t)$, and receives a reward $r(s_t, a_t)$. The agent's goal is to maximize reward over an episode trajectory τ, with a discount factor of γ applied to future rewards. Q-learning methods learn an action-value estimate of the total expected discounted future reward, $Q_\pi(a_t, s_t) = \mathbb{E}_\pi[\sum_{t'=t}^{T} \gamma^{t'-t} r(s_{t'}, a_{t'})]$, through iterative updates based on the Bellman equation:

$$Q_{\theta_\pi}(s_t, a_t) = r(s_t, a_t) + \gamma \mathbb{E}_{s_{t+1} \sim p(\cdot|s_t, a_t)}[\max_{a_{t+1}} Q_{\theta_T}(s_{t+1}, a_{t+1})] \qquad (1)$$

In deep Q-learning [41], a Q-network approximates $Q_{\theta_\pi}(s_t, a_t)$ and drives the policy π. A second target Q-network approximates the expected reward from the next state, $Q_{\theta_T}(s_{t+1}, a_{t+1})$ – a standard practice for alleviating overestimation bias [62].

To perform batch Q-learning, we first pre-train a generative model of $p(a|s)$ using a set of known environment trajectories. In our case, this model is then used to generate the batch data via human interaction. The weights of the Q-network and target Q-network are initialized from the pre-trained model, which helps reduce variance in the Q-estimates and works to combat overestimation bias. To train Q_{θ_π} we sample $< s_t, a_t, r_t, s_{t+1} >$ tuples from the batch, and update the weights of the Q-network to approximate Eq. 1. This forms our baseline model, which we call *Batch Q*.

3.1 Dropout for uncertainty estimation of Target Q-values

Overestimation of Q-values becomes particularly problematic in the batch setting. The Q estimates for state-action pairs which are not well covered in the batch will be noisy, and this variance will lead the *max* operator in Eq. 1 to overestimate the value of these states. This drives the model to value regions of the state-action space for which it has no data to learn a reasonable policy, and no ability to explore to refine its estimates. Clipped Double Q-learning [15] addresses the overestimation problem by maintaining two independent pairs of Q-networks, and taking the minimum of their estimates of future reward. This approach is computationally expensive and memory intensive. Further, if following a transfer learning approach where the Q-network is initialized from a pre-trained MLE model (as we do in this paper), it is not clear how to obtain multiple independent target Q-networks.

Instead, we obtain a distribution over predictions from a single target Q-network trained with dropout, and take the lower bound of these to reduce overestimation bias. It has been shown that dropout approximates Bayesian uncertainty for neural networks, by assuming the weights of the network are

drawn from a Gaussian prior, $W \sim N(0, I)$, and using variational inference to estimate the posterior distribution $p(W|X, Y)$ [17]. We perform dropout during both training and inference before each weight layer, and approximate the posterior such that the dropout distribution q^W is a mixture of Gaussians, and $D_{KL}[q^W||p(W|X, Y)]$ is minimized. Given the target Q-network Q_{θ_T}, we compute $Q(a_{t+1}, s_{t+1})$ using a Monte Carlo (MC) estimate of the lower-bound of $Q_{\theta_T}(a_{t+1}, s_{t+1})$ by running M stochastic forward passes of the network, each with a new dropout mask $d_i \sim q^W$:

$$Q(a_{t+1}, s_{t+1}) = \min_{i=1...M}[Q_{\theta_T}(a_{t+1}, s_{t+1}; d_i)] \tag{2}$$

Using the minimum operator penalizes high variance estimates, essentially leading the algorithm to be pessimistic in the face of uncertainty, rather than optimistic. Such a bias will push the model to favour actions that lead to states well covered by the batch data [16]. We evaluate the performance of this approach using a second baseline model, *Batch Q MC*.

3.2   Discrete Batch Constrained Q

Batch Constrained Q-learning (BCQ) [16] proposes to address the BRL problem by constraining the actions of the Q-network to be close to the data contained within the batch. This is accomplished by learning a generative model of the batch, $G_w = p(a|s)$, and sampling from this model during learning and inference. Because BCQ is designed for continuous action domains, it applies a learned perturbation model $\xi(s, a; \Phi)$ which is allowed to alter the action within the range $[-\Phi, \Phi]$. BCQ learns Q-estimates that incorporate the perturbation model, $Q_\theta(s, a + \xi(s, a; \Phi))$. To act, n possible actions are sampled from the generative model, $\{a_i \sim G_w(s)\}_{i=1}^n$, perturbed, and the action with the maximum Q-value is selected, giving the BCQ policy:

$$\pi_{BCQ}(s) = \underset{a_i + \xi(s, a_i; \Phi)}{\arg\max} \ Q_\theta(s, a_i + \xi(s, a_i; \Phi)) \tag{3}$$

We focus on the scenario where a model of $p(a|s)$ can be obtained through MLE training on data of known action sequences. This prior model provides a more robust estimate of $p(a|s)$ than one learned from the batch data, assuming the size of the batch is small relative to unsupervised data related to the problem (i.e. when the batch comes from human interaction data). We propose an adaptation of BCQ to discrete action spaces (*DBCQ*) which leverages such a strong pre-trained prior model as an improved version of G_w. Since BCQ relies on Double Clipped Q-learning [15], here we use dropout-based uncertainty estimates as in Eq. 2. Because the action space is discrete, we do not use a perturbation model to modify actions, but instead define the DBCQ policy as:

$$\pi_{DBCQ}(s) = \underset{a_i \sim p(a|s)}{\arg\max} Q_{\theta_\pi}(s, a_i) \tag{4}$$

3.3   KL Control from pre-trained prior

Rather than simply sample from the prior, we would like the Q-learning algorithm to directly incorporate the prior into the policy. Thus, we use KL-control to penalize divergence between the prior $p(a|s)$, and the Q-network policy π_θ, while still maximizing reward. Given a trajectory of actions, $\tau = \{a_1, a_2, ...a_{t-1}\}$, let $q(\tau) = \prod_{t=1}^T \pi_\theta(a_t, s_t)$ be the policy of our Q-learning algorithm at the trajectory level. Similarly, let $p(\tau) = \prod_{t=1}^T p(a_t|s_t)$ be the prior distribution over the trajectory, and $r(\tau)$ be the rewards. We seek to maximize the following KL-regularized objective:

$$L(q) = \mathbb{E}_{q(\tau)}[r(\tau)]/c - D_{KL}[q(\tau)||p(\tau)] \tag{5}$$

Since $D_{KL}[q||p] = \sum_x q(x)(\log q(x) - \log p(x))$, we can see that this is equivalent to maximizing the following expected value function of the policy π_θ at the action level:

$$Q^\pi(s_t, a_t) = \mathbb{E}_\pi[\sum_{t'=t}^T r(s_{t'}, a_{t'})/c + \log p(a_{t'}|s_{t'}) - \log \pi(a_{t'}|s_{t'})] \tag{6}$$

The two terms we have introduced in Eq. 6 have clear motivations. The $p(a|s)$ term rewards the model for choosing actions that have high probability under the prior, biasing the model to state-action pairs that are realistic, and likely to be in the batch. The $-\log \pi(a|s)$ term is analogous to entropy regularization. Maintaining diversity in the action space through entropy regularization is important

for generative models like dialog systems, which are known to collapse to an uninteresting, small number of repeated samples [34]. Re-stating Eq. 6 as an entropy-regularized Q-function, we obtain:

$$Q(s_t, a_t) = \mathbb{E}_\pi \left[\sum_{t'=t}^{T} r(s_{t'}, a_{t'})/c + \log p(a_{t'}|s_{t'}) + \mathcal{H}(\cdot|s_{t'}) \right] \tag{7}$$

Motivated by energy-based models of the form $\pi(a_t|s_t) \propto \exp(-\mathcal{E}(s_t, a_t))$, one can derive a soft version of the entropy-regularized Q-function that uses a Boltzmann distribution to estimate future reward [21]. We refer to it as a Ψ-function following previous work [28], which derived this function as a generalization of the Ψ-learning proposed by [48]. The optimal Ψ-function and policy are:

$$\Psi^*(s_t, a_t) = r(s_{t'}, a_{t'})/c + \log p(a_{t'}|s_{t'}) + \gamma \log \sum_{a'} \exp(\Psi^*(s', a')) \tag{8}$$

$$\pi_\Psi^*(a_t|s_t) = \exp(\Psi^*(s_t, a_t)) \tag{9}$$

Because it avoids taking a hard max over noisy estimates, Ψ-learning leads to less overestimation of future reward [1, 21]. This leads to more stable TD updates and aids learning. Thus, we argue it will be especially useful in the BRL setting for reducing optimism in the face of uncertainty.

3.4 Model averaging

Finally, we explore the setting where the data in the batch may be generated from a large variety of different models M with different architectures, which each learn a different estimate of $p(a|s; M)$. We use this diversity to create a more robust prior by computing a weighted average of these models based on a normalized score $S(M)$ for each model. The score could be some measure of model quality, or simply the proportion of data in the batch that was generated with that model. Thus, we define $p_{MA}(a|s)$ as the model-averaged prior: $p_{MA}(a|s) = \sum_M S(M)p(a|s; M)$.

4 RL for open-domain dialog generation

In this work, we employ hierarchical seq2seq dialog models [20, 43, 52, 53], which use three recurrent networks to generate the next utterance in a conversation (see Figure 1). The *encoder RNN* operates on the tokens of the next input utterance $u_t = [w_1, w_2, ...w_n]$, and encodes them into a representation $h_t^e = f^e(u_t)$. This is fed into a *context RNN*, which forms the upper level of the hierarchy – it is updated only after each utterance, rather than each token. The context RNN outputs $h_t^c = f^c(h_t^e)$, which is fed into the *decoder RNN*, which produces the output utterance $u_{t=1}$ one token at a time. Note that

Figure 1: Simplified diagram of the variational hierarchical dialog model.

while transformer architectures (e.g. [47]) have emerged as a powerful alternative to seq2seq models, here we choose to focus on hierarchical architectures because it gives us the flexibility to extend this work to use hierarchical control in the future, by learning to optimize rewards at both the utterance and conversation level. Although we trained and tested a variety of different architectures drawing from several works [20, 43, 52, 53], we converged on the Variational Hierarchical Recurrent Encoder Decoder (VHRED) as the most promising model [53]. We also apply knowledge distillation to improve the model's ability to recognize and encode the sentiment and semantics of the conversation, as proposed by [20].

Applying RL to dialog generation is challenging due to the large state-action space. The model attempts to construct a response utterance $u_t^\pi = [a_1, a_2, ..., a_n]$ by iteratively choosing an action a_i as the next token. The number of tokens in the vocabulary of our pre-trained model is 20,000, making the action space very high-dimensional, potentially compounding the problem of overestimation and making batch learning excessively difficult. However, initializing the Q-networks with the weights of the pre-trained language model provides a strong prior over the appropriate word to select.

Here we consider human interaction to represent the 'environment'. The response of a human to the bot's utterance is used to compute a reward signal to train the model. The state of the environment s_t
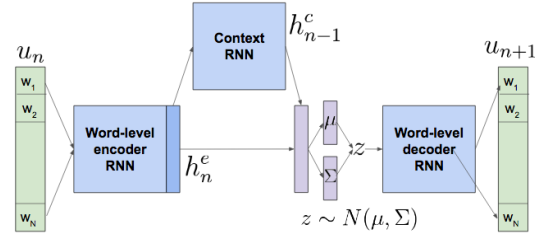
constitutes all of the text in the conversation uttered so far, both by the bot and the human. The state has a hierarchical structure, marking its division into utterances, which are further divided into tokens. While the bot is constructing an utterance u_t^π, it is straightforward to obtain a target Q-estimate of future reward using the model's estimated Q-values over its own next token in the utterance. However, at the last token of the bot's utterance, the estimated future reward must include the human's response u_t^h. Therefore, we append the human response into the conversation, $s_{t+1} = [s_{t-1}, u_t^\pi, u_t^h]$, feed this into the target Q-network, and use the estimated Q-values for the first token of the bot's next utterance. All of the code for our models and RL techniques is available in open-source at `https://github.com/natashamjaques/neural_chat/tree/master/rl`.

4.1 Learning from implicit human preferences

We would like to improve a dialog model's ability to engage in natural conversation with a human by learning from the signals implicit in the way that the human responds. Rather than having the human manually label good performance – which we show in this work does not scale – the agent should recognize informative cues within the user's responses, like sentiment, and the amount of time they spend chatting. Essentially, we want to create an agent that is intrinsically motivated to produce positive reactions in its human conversation partner. We design several intrinsic reward functions based on the rich, interactive content of conversation, taking inspiration from the psychology of human conversation: 1) eliciting positive sentiment and transitions from negative to positive sentiment, due to the importance of emotion to creating a sense of understanding [6, 63]; 2) eliciting longer conversations and more words typed, since this is a signal of engagement [57, 64]; 3) eliciting laughter (counting the number of 'ha's in the user response), because of its importance in building solidarity [24]; 4) high semantic similarity (close distance in sentence embedding space [8]) between the human input and agent response, because paraphrasing and style matching are important in facilitating good conversation [27, 63]; and 5) asking questions, since this is an important active listening skill [5]. The total reward given to the agent is a combination of these, with details (and coefficients) given in the supplementary material. Note that the first 4 types of rewards depend on eliciting positive responses from a human user; we call these the implicit *human reward*. The 5th reward is easily exploitable by the agent itself. These rewards represent only an initial foray into designing good metrics of human enjoyment, and further experimentation will be needed to improve them.

5 Experiments

To collect interactive human conversation data, we built a CUDA-capable web app that can host neural network dialog models on GPU for fast, real-time inference: `https:neural.chat`. The code for the server is available in open-source at `https://github.com/asmadotgh/neural_chat_web`. We trained over 40 dialog models with different architectures (e.g. [53, 52, 43, 20]), on different datasets (movie dialogs [9] and Reddit [20]). Note that these models varied significantly in terms of the distribution of language they learned. We collected a batch of data containing 14232 pairs of user input and agent response. This batch was used to train the RL models described in Section 3, which were then re-deployed to the website. We recruited 90 Mechanical Turk workers to provide a total of 718 7-point Likert scale ratings of the bots' quality, fluency, diversity, contingency (relatedness), and empathy, after interacting with each bot for at least 3 turns. Participants also had the option to provide explicit feedback through upvoting or downvoting a particular utterance within the interface. Note that testing these models in the wild with humans represents a more meaningful test of generalization than testing an RL model in the same limited (game) environment in which it was trained, since humans are not restricted in the text they can type to the model, and are the ultimate authority on naturalistic conversation.

6 Results

To compare models, we not only look at human users' ratings and votes, but also consider the automatic signals detectable from the text itself. This implicit *human reward* metric aggregates the measures listed in items 1-4 in Section 4.1, and measures the ability to elicit positive responses from the human. Table 1 shows the results of the evaluation. Each of the enhancements proposed (MC estimation of target Q-values, Ψ-learning, and MA) leads to performance gains in terms of human reward, manual votes, or ratings. However, the most notable difference in performance comes from

Table 1: Interactive human evaluation of techniques for off-policy batch RL. KL-control models strongly out-perform other techniques. Ratings are Likert scale, votes and human reward are z-scores.

Model type	Quality	Fluent	Diverse	Related	Empathy	Total	Votes	Human reward
DBCQ	1.64 ±.29	1.87 ±.34	**3.13 ±.58**	1.84 ±.34	2.09 ±.38	10.58 ±1.55	-.228	-.050
Batch Q	1.87 ±.30	2.36 ±.42	2.20 ±.41	1.91 ±.32	2.58 ±.47	11.91 ±1.58	-.163	-.005
Batch Q + MC	1.85 ±.39	2.46 ±.44	2.46 ±.52	1.98 ±.39	2.34 ±.49	11.07 ±1.82	-.068	.005
KL-control Q	2.38 ±.39	3.24 ±.47	3.42 ±.54	2.38 ±.45	2.56 ±.43	13.98 ±1.81	.016	.004
KL-control Ψ	2.33 ±.41	**3.73 ±.53**	2.82 ±.50	2.31 ±.44	**3.47 ±.50**	**14.67 ±1.82**	**.128**	**.061**
KL-control MA Ψ	**2.60 ±.43**	3.47 ±.42	3.00±.49	**2.49 ±.44**	2.89 ±.51	14.44 ±1.96	.127	.042

Table 2: Purely reward-maximizing methods like *Batch Q* trivially exploit the reward function by asking a question every turn, and using the maximum number of tokens in every sentence. In contrast, KL-control methods output plausible language by staying close to the prior, but shift to using polite, cheerful language to maximize implicit human rewards.

[User]: hey, what's up?
[Batch Q]: where did you get??, that sounds so exciting. how do you work for you... you want to talk to me?... you want
[User]: yes, i want to talk to you!
[Batch Q]: where did you say to me?... you want to talk to you about you... you are so good for you... you are so good...
[User]: thanks, i appreciate that.
[Batch Q]: where did you get??, that sounds so exciting. how do you work for you... you want to see you...

[User]: hey, what's up?
[KL-control]: i'm not sure, but i'm glad to see you, and i'm so happy that you're not alone
[User]: oh thanks, that's very nice of you!
[KL-control]: i'm sorry, but i'm glad to see what you think.
[User]: don't be sorry, you did a good job.
[KL-control]: that's really nice to me.

KL-control. The KL-control models show substantial gains over the baseline[1] models across both ratings and human reward. We perform a one-way analysis of variance (ANOVA) comparing the KL-control models to the Batch Q baselines and DBCQ on the total human rating score, and find that the KL-control models are significantly better, $F(x) = 4.781, p < .05$. This validates the hypothesis that KL-control with a strong, pre-trained prior can be used to improve batch RL.

Without KL-regularization, the baseline RL models diverge quickly and continuously from the prior, losing information about realistic sequences – as shown in Figure 2. This figure also helps explain the poor performance of DBCQ in Table 1. The underlying Q-network in DBCQ does not directly integrate the prior. As Q-learning causes the model to diverge from the prior, the Q-estimates of language generated according to the prior become unrealistic, and Eq. 4 selects unrealistic actions. This results in highly 'diverse' (random) generated utterances. Note that since we operate in discrete action space, we could not include the perturbation model originally proposed by [16], which may be critical to achieving good performance with BCQ.
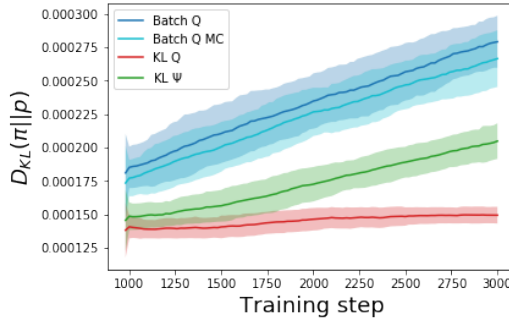

Figure 2: KL-divergence of the policy from the prior is lower with KL-control throughout training. Bands show standard deviation.

The pre-trained prior may be especially important in a generative domain like dialog, where the true reward function is unknown, and so purely maximizing reward may actually lead to poorer quality conversations. Table 2 shows examples of conversations with a Batch Q and KL-control model. Because the Batch Q model has no incentive to stay close to realistic language, it learns to exploit the reward by asking a question and outputting the maximum number of tokens (30) every utterance. These sentences contain implausible phrases that do not represent realistic language (e.g. "*where did you say to me?*"). In contrast, the KL-control model uses realistic language, but shifts its distribution towards cheerful and polite speech, presumably because this is what led to positive human responses in the batch data. Rather than simply cherry-picking results, we invite the reader to check for themselves; all of the models tested in this study are available at: `https://neural.chat/rl`.

---

[1]We also compare the RL models to the prior, and see performance improvements in terms of the elicited human reward, but not in terms of the quality ratings. We believe this is because the rewards proposed here do not fully cover what it means to have a high quality conversation (as we will elucidate later), and hope that other researchers be able to use the techniques we propose to learn from improved rewards.
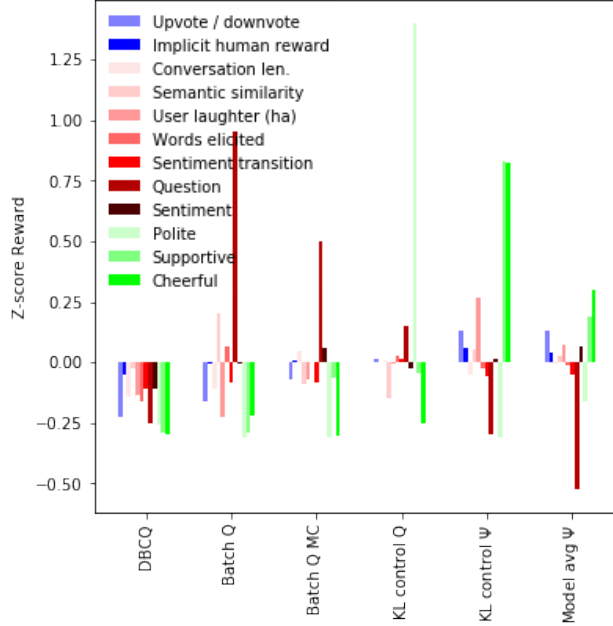
Figure 3: Z-scored reward. Red metrics were used in training rewards, green are post-hoc. Traditional RL methods like Batch Q exploit simple action-based rewards, like asking questions. In contrast, KL-control methods shift their distribution towards polite, supportive, and cheerful conversation, allowing them to elicit higher *human reward* (blue).
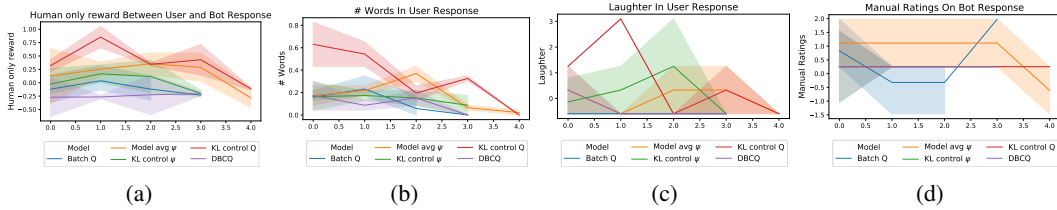

Figure 4: Comparison of top 10 conversation trajectories observed across deployed models, 90% CI of the rewards: (a) Implicit human feedback; (b) Words elicited; (c) Laughter; (d) Manual votes.

In fact, we noticed that all models trained with the implicit human rewards described in Section 4.1 learned to use more cheerful and supportive language. Therefore, we create post-hoc metrics to measure this effect (see the supplementary material for details). Figure 3 shows how these metrics, as well as the implicit rewards, differ across models. Without KL-control, baseline methods like Batch Q exploit simple rewards like asking questions at the expense of realistic language, explaining their poor quality ratings. In contrast, KL-control models learn to rely more on realistic but polite, supportive, and cheerful dialog to elicit higher total *human reward*.

To understand the effect of the implicit rewards, Figure 4 shows the reward trajectory over the ten best conversations obtained with models trained with different techniques. While we see that KL-control models are able to elicit significantly higher reward than baselines, we note that KL-control Q performs best overall and in terms of words elicited, even though it had lower quality ratings in Table 1. This suggests that maximizing these rewards is not a perfect proxy for human judgments of quality. Note also that eliciting laughter is an extremely rare event, and only the KL-control models are able to do so. Finally, Figure 4 (d) shows that manual votes occur even more rarely, suggesting that explicit feedback from humans is a cumbersome and sparse reward signal.

Table 3 presents the results of models trained with only a single reward function, ordered from lowest to highest quality. Notably, extracting multiple different reward functions post-hoc from a batch of data and training on these independently is only possible with an effective BRL model. Here all models are trained with KL-control, Ψ-learning, and MC targets. Investigating which rewards

Table 3: Interactive human evaluation of different reward functions (models trained with KL-control)

Reward function	Quality	Fluent	Diverse	Related	Empathy	Total	Votes	Human reward
Conv. len.	2.20 ±.40	3.61 ±.53	3.02 ±.52	2.25 ±.46	2.48 ±.45	13.57 ±1.84	-.035	-.003
Semantic sim.	1.93 ±.34	3.50 ±.45	2.37 ±.45	2.11 ±.45	2.52 ±.48	12.43 ±1.75	-.020	.012
User laughter	1.96 ±.38	3.56 ±.48	2.33 ±.51	1.93 ±.42	3.20 ±.55	12.98 ±1.60	-.149	-.003
Words elicited	2.11 ±.32	3.96 ±.44	3.04 ±.45	2.04 ±.35	2.55 ±.46	13.70 ±1.44	.059	.024
Manual votes	2.14 ±.38	3.47 ±.45	2.91 ±.47	2.07 ±.39	2.42 ±.46	13.00 ±1.65	-.030	.010
Sent. trans.	2.02 ±.31	3.71 ±.49	2.98 ±.50	2.04 ±.42	2.84 ±.48	13.60 ±1.63	.031	.014
Question	2.29 ±.37	**4.31 ±.50**	**3.31 ±.52**	2.20 ±.40	2.60 ±.41	14.71 ±1.63	.057	.012
Sentiment	**2.47 ±.32**	4.05 ±.45	3.23 ±.46	**2.42 ±.39**	3.23 ±.55	**15.40 ±1.49**	**.085**	**.045**

presented in Section 4.1 are most critical to achieving high-quality conversations with humans, we note that maximizing positive and minimizing negative sentiment in the user turns out to lead to the highest quality bot. This underscores the importance of affective signals as cues for good conversation. Bots trained on the manual upvotes and downvotes provided by users on the utterance level fail to achieve similarly high performance. Even though users were instructed to make use of the vote feature, the task is burdensome, and users did not vote frequently enough to provide a good training signal. This validates the hypothesis that *implicit* signals of human enjoyment (such as sentiment) are a more scalable way to learn from human preferences.

7   Conclusion

This paper presents a series of techniques which improve performance when learning off-policy without the possibility to explore – i.e. batch RL (BRL). Most significantly, we are the first to propose using KL-control from a strong prior model pre-trained on data as a way to avoid overestimation and instability in BRL. Our results demonstrate that KL-control is critical to achieving good performance in this setting. In a generative domain such as dialog, the true reward function is not known, and trivially exploiting the rewards can actually lead to worse performance. Thus, KL-control may be particularly necessary to ensure samples remain realistic and close to the data distribution. We propose several reward functions that could allow an open-domain dialog generation model to learn from rich cues implicit in human interaction, where learning from expressed sentiment was most promising. While these rewards are far from perfect or complete, we see that maximizing implicit rewards leads to better performance than relying on explicit feedback. We hope that the techniques presented here will allow other researchers to leverage BRL for learning from human interaction data, and spur the development of even better rewards for capturing human preferences.

References

[1] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

[2] Marcin Andrychowicz, Filip Wolski, Alex Ray, Jonas Schneider, Rachel Fong, Peter Welinder, Bob McGrew, Josh Tobin, OpenAI Pieter Abbeel, and Wojciech Zaremba. Hindsight experience replay. In *Advances in Neural Information Processing Systems*, pages 5048–5058, 2017.

[3] Kamyar Azizzadenesheli, Emma Brunskill, and Animashree Anandkumar. Efficient exploration through bayesian deep q-networks. In *2018 Information Theory and Applications Workshop (ITA)*, pages 1–9. IEEE, 2018.

[4] Aditya Bhatt, Max Argus, Artemij Amiranashvili, and Thomas Brox. Crossnorm: Normalization for off-policy td reinforcement learning. *arXiv preprint arXiv:1902.05605*, 2019.

[5] Graham D Bodie, Kellie St. Cyr, Michelle Pence, Michael Rold, and James Honeycutt. Listening competence in initial interactions i: Distinguishing between what listening is and what listeners do. *International Journal of Listening*, 26(1):1–28, 2012.

[6] Graham D Bodie, Andrea J Vickery, Kaitlin Cannava, and Susanne M Jones. The role of "active listening" in informal helping conversations: Impact on perceptions of listener helpfulness, sensitivity, and supportiveness and discloser emotional improvement. *Western Journal of Communication*, 79(2):151–173, 2015.

[7] Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In *Advances in Neural Information Processing Systems*, pages 4299–4307, 2017.

[8] Alexis Conneau, Douwe Kiela, Holger Schwenk, Loïc Barrault, and Antoine Bordes. Supervised learning of universal sentence representations from natural language inference data. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 670–680, 2017.

[9] Cristian Danescu-Niculescu-Mizil and Lillian Lee. Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2nd Workshop on Cognitive Modeling and Computational Linguistics*, pages 76–87. Association for Computational Linguistics, 2011.

[10] Thomas Degris, Martha White, and Richard S Sutton. Off-policy actor-critic. In *Proceedings of the 29th International Coference on International Conference on Machine Learning*, pages 179–186. Omnipress, 2012.

[11] Mehrdad Farajtabar, Yinlam Chow, and Mohammad Ghavamzadeh. More robust doubly robust off-policy evaluation. In *International Conference on Machine Learning*, pages 1446–1455, 2018.

[12] Mehdi Fatemi, Layla El Asri, Hannes Schulz, Jing He, and Kaheer Suleman. Policy networks with two-stage training for dialogue systems. In *Proceedings of the 17th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 101–110, 2016.

[13] Bjarke Felbo, Alan Mislove, Anders Søgaard, Iyad Rahwan, and Sune Lehmann. Using millions of emoji occurrences to learn any-domain representations for detecting sentiment, emotion and sarcasm. In *2017 Conference on Empirical Methods in Natural Language ProcessingConference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2017.

[14] Roy Fox, Ari Pakman, and Naftali Tishby. Taming the noise in reinforcement learning via soft updates. In *Proceedings of the Thirty-Second Conference on Uncertainty in Artificial Intelligence*, pages 202–211. AUAI Press, 2016.

[15] Scott Fujimoto, Herke Hoof, and David Meger. Addressing function approximation error in actor-critic methods. In *International Conference on Machine Learning*, pages 1582–1591, 2018.

[16] Scott Fujimoto, David Meger, and Doina Precup. Off-policy deep reinforcement learning without exploration. *arXiv preprint arXiv:1812.02900*, 2018.

[17] Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Representing model uncertainty in deep learning. In *international conference on machine learning*, pages 1050–1059, 2016.

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

[19] Carles Gelada and Marc G Bellemare. Off-policy deep reinforcement learning by bootstrapping the covariate shift. *arXiv preprint arXiv:1901.09455*, 2019.

[20] Asma Ghandeharioun, Judy Shen, Natasha Jaques, Craig Ferguson, Noah Jones, Agata Lapedriza, and Rosalind Picard. Approximating interactive human evaluation with self-play for open-domain dialog systems. *arXiv preprint arXiv:1906.09308*, 2019.

[21] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1352–1361. JMLR. org, 2017.

[22] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *International Conference on Machine Learning*, pages 1856–1865, 2018.

[23] Braden Hancock, Antoine Bordes, Pierre-Emmanuel Mazare, and Jason Weston. Learning from dialogue after deployment: Feed yourself, chatbot! *arXiv preprint arXiv:1901.05415*, 2019.

[24] Jennifer Hay. Functions of humor in the conversations of men and women. *Journal of pragmatics*, 32(6):709–742, 2000.

[25] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

[26] Helena Horton. Microsoft deletes 'teen girl' ai after it became a hitler-loving sex robot within 24 hours. In *Telegraph UK*, 2016.

[27] Molly E Ireland, Richard B Slatcher, Paul W Eastwick, Lauren E Scissors, Eli J Finkel, and James W Pennebaker. Language style matching predicts relationship initiation and stability. *Psychological science*, 22(1):39–44, 2011.

[28] Natasha Jaques, Shixiang Gu, Dzmitry Bahdanau, José Miguel Hernández-Lobato, Richard E Turner, and Douglas Eck. Sequence tutor: Conservative fine-tuning of sequence generation models with kl-control. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1645–1654. JMLR. org, 2017.

[29] Nan Jiang and Lihong Li. Doubly robust off-policy value evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 652–661, 2016.

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

[31] Gregory Kahn, Adam Villaflor, Vitchyr Pong, Pieter Abbeel, and Sergey Levine. Uncertainty-aware reinforcement learning for collision avoidance. *arXiv preprint arXiv:1702.01182*, 2017.

[32] Sham M Kakade. A natural policy gradient. In *Advances in neural information processing systems (NIPS)*, volume 14, pages 1531–1538, 2002.

[33] Hilbert J Kappen, Vicenç Gómez, and Manfred Opper. Optimal control as a graphical model inference problem. *Machine learning*, 87(2):159–182, 2012.

[34] Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. Dialogue learning with human-in-the-loop. *arXiv preprint arXiv:1611.09823*, 2016.

[35] Jiwei Li, Will Monroe, Alan Ritter, Dan Jurafsky, Michel Galley, and Jianfeng Gao. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1192–1202, 2016.

[36] Jiwei Li, Will Monroe, Tianlin Shi, Sébastien Jean, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2157–2169, 2017.

[37] Ziming Li, Julia Kiseleva, and Maarten de Rijke. Dialogue generation: From imitation learning to inverse reinforcement learning. *arXiv preprint arXiv:1812.03509*, 2018.

[38] Bing Liu and Ian Lane. Iterative policy learning in end-to-end trainable task-oriented neural dialog models. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*, pages 482–489. IEEE, 2017.

[39] Bing Liu, Gokhan Tür, Dilek Hakkani-Tür, Pararth Shah, and Larry Heck. Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2060–2069, 2018.

[40] Yao Liu, Adith Swaminathan, Alekh Agarwal, and Emma Brunskill. Off-policy policy gradient with state distribution correction. *arXiv preprint arXiv:1904.08473*, 2019.

[41] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*, 2013.

[42] Ian Osband, Charles Blundell, Alexander Pritzel, and Benjamin Van Roy. Deep exploration via bootstrapped dqn. In *Advances in neural information processing systems*, pages 4026–4034, 2016.

[43] Yookoon Park, Jaemin Cho, and Gunhee Kim. A hierarchical latent structure for variational conversation modeling. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1792–1801, 2018.

[44] Jan Peters, Katharina Mülling, and Yasemin Altun. Relative entropy policy search. In *AAAI*, pages 1607–1612. Atlanta, 2010.

[45] Doina Precup. Eligibility traces for off-policy policy evaluation. *Computer Science Department Faculty Publication Series*, page 80, 2000.

[46] Robert R Provine. Laughter. *American scientist*, 84(1):38–48, 1996.

[47] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI Blog*, 1:8, 2019.

[48] Konrad Rawlik, Marc Toussaint, and Sethu Vijayakumar. On stochastic optimal control and reinforcement learning by approximate inference. In *Robotics: science and systems*, 2012.

[49] Martin Riedmiller. Neural fitted q iteration–first experiences with a data efficient neural reinforcement learning method. In *European Conference on Machine Learning*, pages 317–328. Springer, 2005.

[50] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *Proceedings of the 32nd International Conference on Machine Learning (ICML-15)*, pages 1889–1897, 2015.

[51] Iulian V Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, et al. A deep reinforcement learning chatbot. *arXiv preprint arXiv:1709.02349*, 2017.

[52] Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[53] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron Courville, and Yoshua Bengio. A hierarchical latent variable encoder-decoder model for generating dialogues. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[55] Weiyan Shi and Zhou Yu. Sentiment adaptive end-to-end dialog systems. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1509–1519, 2018.

[56] Jamin Shin, Peng Xu, Andrea Madotto, and Pascale Fung. Happybot: Generating empathetic dialogue responses by improving user experience look-ahead. *arXiv preprint arXiv:1906.08487*, 2019.

[57] Candace L Sidner, Cory D Kidd, Christopher Lee, and Neal Lesh. Where to look: a study of human-robot engagement. In *Proceedings of the 9th international conference on Intelligent user interfaces*, pages 78–84. ACM, 2004.

[58] Robert F Stengel. *Stochastic optimal control*. John Wiley and Sons New York, New York, 1986.

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

[60] Philip Thomas and Emma Brunskill. Data-efficient off-policy policy evaluation for reinforcement learning. In *International Conference on Machine Learning*, pages 2139–2148, 2016.

[61] Emanuel Todorov. Linearly-solvable markov decision problems. In *Advances in neural information processing systems (NIPS)*, pages 1369–1376, 2007.

[62] Hado Van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[63] Harry Weger Jr, Gina R Castle, and Melissa C Emmett. Active listening in peer interviews: The influence of message paraphrasing on perceptions of listening skill. *The Intl. Journal of Listening*, 24(1):34–49, 2010.

[64] Li Zhou, Jianfeng Gao, Di Li, and Heung-Yeung Shum. The design and implementation of xiaoice, an empathetic social chatbot. *arXiv preprint arXiv:1812.08989*, 2018.

[65] Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pages 1433–1438. Chicago, IL, USA, 2008.

8 Appendix

8.1 Details about implicit metrics

8.1.1 Sentiment-based

To compute sentiment on short texts like conversation utterances, we leverage a state-of-the-art sentiment-detection model, which was trained on a massive amount of Twitter data to predict the emojis in tweets [13]. Transfer learning from this model to other tasks showed that it was able to significantly outperform a series of sentiment, irony, and sarcasm benchmarks. This DeepMoji model outputs a probability distribution over 64 most-frequently used emojis as shown in Figure 5. After observing the performance of the model in detecting users' emotions in the domain of online chat, we define a set of weights over the emojis and calculate the weighted sum over an emotion embedding vector to derive a *sentiment* reward which is higher for positive sentiment and lower for negative sentiment. These weights are shown in Figure 5 (b). We also compute a sentiment-transition reward using the same score based on whether the peak positive sentiment occurred later in the conversation than the peak negative sentiment, reasoning that sentiment should improve over the course of the conversation.
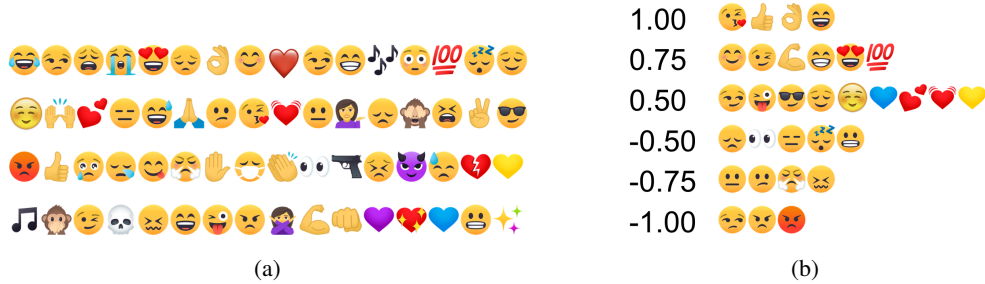

Figure 5: (a) 64-most frequent emojis as predicted by [13] used for calculating emotion embeddings. (b) Assigned weights used in producing the sentiment reward from the predicted emoji values.

8.1.2 Engagement-based

Based on prior work [64], we use the number of turns in the conversation as an indicator of the quality of the bot's performance. To distribute this reward over every utterance in the conversation, we take the total conversation length N, and compute the discounted reward for utterance $n < N$ as $\gamma^{N-n} N$. We also reward each utterance with the number of words in the user's response, which we refer to as the *words elicited*.

8.1.3 Laughter

Laughter has been shown to be very important to human affiliation [46] and solidarity [24]. Therefore, we detect the number of occurrences of the string 'ha' in the user's response, and use this as a reward. Interestingly, we find that bots trained to maximize user laughter learn to be extremely supportive and cheerful compared to other bots (for definitions of supportive and cheerful, see Section 8.1.7).

8.1.4 Semantic similarity

Language style matching has been shown to be a strong predictor of relationship initiation and stability [27]. While it would be ideal if our chatbots could intelligently adapt their conversation style to a new user, in reality

most baseline dialog models struggle to maintain topic coherence, even over a few utterances (for an analysis of this effect, see [20]). Therefore we reward *semantic similarity* between the user's input and the bot's response, to encourage the bot to stay on topic and produce reasonable answers. This score is computing by leveraging a state-of-the-art sentence embedding model [8], and penalizing distance in embedding space.

8.1.5 Questions

Asking questions is an important listening skill, and is linked to conversation management, attentiveness, and responsiveness [5]. Therefore, we give the bot a reward of 0.5 if the utterance contains a question word (*how, what, where, why, when, who*), and an additional 0.5 if it contains a question mark.

8.1.6 Total reward equation

The total reward used to train the bots is a combination of the above rewards, in the following proportions:

```
0.15682657*question + 0.13837638*semantic_coherence + 0.15313653*laughter +
0.14206642*sentiment_transition + 0.14206642*sentiment + 0.14760148*words_elicited +
0.1199262*conversation_length.
```

8.1.7 Post-hoc metrics

After training the bots on these rewards, we noticed a shift in the distribution of their language towards more polite, cheerful, and supportive speech. Therefore, we designed post-hoc metrics to measure these qualities, which are based on counting whether a subset of phrases is present in an utterance.

Politeness phrases: *if I may; may I; please; thanks; no worries; if you don't mind; have a great day; I'm sorry.*

Supportive phrases: *you're right; you are right; you're not alone; you are not alone; congrats; that's a good idea; that is a good idea; you'll be fine; you will be fine; you'll be okay; you will be okay; it will get better; sorry you're going through; sorry you are going through; if it makes you feel better; if it makes you feel any better; keep your head up; keep it up; I'm in a similar situation; I am in a similar situation; you'll get it; you will get it; happy for you; I'm in the same boat; I am in the same boat; if you feel like you need to vent.*

Cheerful phrases: *nice to hear; happy; excited; really nice; glad; the best; great; good time; looking forward; beautiful.*

8.2 Training details and hyperparameters

RL models were trained for between 800 and 1000 batches of data, where the batch size was fixed at 32. Early stopping was used to determine the number of training iterations of the best checkpoint. All other hyperparameters were shared between RL models, and were as follows: discount $\gamma = 0.5$, weight placed on RL reward vs. KL-divergence term $c = 2$, number of Monte Carlo samples of the Target Q-network $M = 5$, target network update rate $\alpha = .005$, learning rate $r = .0001$. We used a smooth $L1$ loss function to approximate the Q-values, and clipped gradients at a value of 1.0.

The underlying parameters of the VHRED model were as follows: Context RNN hidden size $= 1000$, decoder hidden size $= 1250$, encoder hidden size $= 1250$, z embedding size $= 600$, gradient clip $= 1.0$, dropout $d = 0.2$. The maximum conversation length was fixed at 5 utterances (context from more than 5 utterances ago was discarded), and the maximum sentence length was 30 tokens.

We also added layers to the Context RNN and regularized it to be able to predict the semantic content of the input utterance using a form of knowledge distillation [25] from a state-of-the-art sentence-embedding model [8]. There were 2 additional feedforward semantic prediction prediction layers of size 128, which used ReLu activation.

8.3 Additional results

Figure 6 shows the normalized reward scores obtained bots trained with respect to different rewards. While some bots (such as those trained to ask questions or elicit positive sentiment) effectively generalize to new users, we see that others (e.g. words elicited) are not actually able to best elicit those responses in the wild. We hypothesize this is because the relatively small size of batch date we were able to collect ($\approx 14,000$ utterances) does not give these bots enough information about how to elicit long responses from users.

	Total reward	Implicit human reward	Conversation length	Sentiment similarity	User laughter (ha)	Words elicited	Sentiment transition	Question	Sentiment	Manual ratings	Politeness	Supportive	Cheerful
Conversation length	0.018	-0.003	-0.046	-0.048	0.023	-0.053	0.008	-0.541	-0.102	-0.035	-0.037	-0.143	-0.277
Semantic similarity	0.037	0.012	-0.003	-0.145	0.119	-0.039	0.006	-0.480	-0.042	-0.020	-0.191	-0.224	0.068
User laughter (ha)	0.029	-0.003	-0.101	0.060	0.023	-0.016	-0.105	-0.368	-0.056	-0.149	-0.210	0.886	0.826
Words elicited	0.072	0.024	0.135	0.043	-0.093	0.003	-0.038	-0.126	0.014	0.059	0.570	-0.285	-0.016
Sentiment transition	0.051	0.014	-0.059	0.034	0.100	-0.036	-0.004	-0.299	-0.123	0.031	0.103	0.133	0.015
Question	0.154	0.012	0.195	-0.012	-0.230	0.047	-0.048	1.366	0.083	0.057	-0.210	-0.364	-0.251
Sentiment	0.123	0.045	-0.092	0.021	0.030	0.068	-0.099	0.346	0.294	0.085	0.098	0.270	-0.073
Manual ratings	0.064	0.010	-0.065	0.051	0.062	0.026	-0.071	-0.029	-0.064	-0.030	-0.112	-0.207	-0.275

Figure 6: Normalized reward scores obtained by models trained with respect to different rewards. We see that the bot trained to ask questions is easily able to exploit this reward, and similarly the bot trained to elicit positive sentiment does so successfully. For the rest of the bots, the relationship is less clear. For example, the bot trained to elicit laughter becomes the most supportive and cheerful, while the bot trained to elicit more words is very polite.

8.4 Interactive bot platform details

To collect data from humans interacting with our bots, we built `https://neural.chat`, a platform for hosting deep neural network dialog models online on GPU for fast, real-time inference. Figure 7) shows an example of the interface, in which users are able to rate the bots after talking to them for at least three turns.

Figure 7: Interactive evaluation ratings page available at `https://neural.chat`.

Figure 8 is an example conversation within the platform that interactive evaluation participants see. Annotators can optionally click the up and down arrows beside each chatbot response to give feedback on the specific utterance. Once 3 or more turns of the conversation has taken place, participants may click "Close Chat and Rate" to get to the rating screen.

8.4.1 Website server setup and configuration

The server was hosted on a Google Cloud Platform virtual instance with 64GB of RAM and a NVIDIA Tesla P100 graphics card. The backend was a Django program being served by NGINX and uWSGI. For simplicity, we opted to have the Django process import the chatbots into the same Python process as Django, rather than have the two connect to each other via other means such as sockets. This configuration decreased development time and increased reliability, but it would need to be revisited if the server needed to scale several orders of magnitude past what was required for this study. The current configuration was still able to support hundreds of simultaneous users and host more than 30 bots concurrently.

Figure 8: Interactive evaluation chat interface.

The chatbots were kept in a separate project from the Django project and maintained separately from the server code. Each chatbot extended an abstract class that defined key methods for the Django program to use, and was registered to a globally accessible dictionary via a decorator. The Django project was provided the path to the Chatbots project in its PYTHONPATH, so it could import the dictionary in which all the chatbot objects had been registered and use that to dynamically determine which chatbots were available and to access them in its views.

It is important to note that the chatbots used PyCUDA, and PyCUDA does not work in a multiprocessing environment. Because of this, uWSGI needed to be configured to only have one python process and to disable any attempt at multiprocessing. Furthermore, the chatbots required substantial startup times, so all chatbots are kept in memory at all times in the Django process. In order to keep all the chatbots in memory concurrently, we needed a very high amount of RAM on our server and opted for a 64GB virtual instance, and a GPU with 16GB RAM. This combination of CUDA to run the chatbots on the GPU with a high amount of RAM to keep all bots in memory at the same time resulted in incredibly fast server response times, with effectively no increase in response time when using the bots in requests compared to requests that did not.

For further information and instructions on server configuration, please read the server documentation available at `https://github.com/asmadotgh/neural_chat_web`. We hope that this platform will allow others to host their own bots and evaluate them in an interactive setting.