End-to-End Offline Goal-Oriented Dialog Policy Learning via Policy Gradient

Li Zhou
Amazon
Seattle, WA
lizhouml@amazon.com

Kevin Small
Amazon
Seattle, WA
smakevin@amazon.com

Oleg Rokhlenko
Amazon
Seattle, WA
olegro@amazon.com

Charles Elkan
Amazon
Seattle, WA
elkanc@amazon.com

Abstract

Learning a goal-oriented dialog policy is generally performed offline with supervised learning algorithms or online with reinforcement learning (RL). Additionally, as companies accumulate massive quantities of dialog transcripts between customers and trained human agents, encoder-decoder methods have gained popularity as agent utterances can be directly treated as supervision without the need for utterance-level annotations. However, one potential drawback of such approaches is that they myopically generate the next agent utterance without regard for *dialog-level* considerations. To resolve this concern, this paper describes an *offline* RL method for learning from *unannotated* corpora that can optimize a goal-oriented policy at both the utterance and dialog level. We introduce a novel reward function and use both on-policy and off-policy policy gradient to learn a policy offline without requiring online user interaction or an explicit state space definition.

1 Introduction

Companies are increasingly interested in building goal-oriented dialog systems for domains such as customer service and reservation systems. One difference between building chatbots in industry and academia is that companies usually possess a vast amount of dialogs where conversation transcripts are between customers and *trained* human agents. For example, customer service live chat systems are usually operated by human agents, and these agents have to take weeks of training courses before being deemed qualified to handle customer issues. Thus, agent utterances in these dialogs can be treated as actions produced by a near-optimal policy for machine learning algorithms. To build industry-quality goal-oriented chatbots, one important direction of research is effectively utilizing these corpora of *trained agent-customer transcripts (TACTs)* to learn dialog policies offline while minimizing the dependence on domain knowledge and corresponding human effort.

Recently, several supervised learning algorithms have been proposed to learn goal-oriented dialog policies [3, 28]. In these algorithms, the agent utterances are treated as labels, and the models are trained to maximize the likelihood of agent utterances. While this is intuitive, supervised learning models are trained to only optimize myopic rewards (i.e., likelihood of the next utterances instead of the goal of the whole dialog). Simply imitating the agent utterances in the training data is problematic and can suffer from compounding errors in multi-turn dialogs [18].

Workshop on Conversational AI, NIPS 2017, Long Beach, CA, USA.

Reinforcement learning (RL), on the other hand, is a natural choice to learn goal-oriented dialog policies as it optimizes long-term cumulative reward and selects the best action to achieve the goal given the current state. Recently, many RL-based algorithms have been proposed and achieved good results [15, 5, 30, 22]. However, there are two main drawbacks when directly applying reinforcement learning to dialog policy learning: (1) RL algorithms for learning dialog managers are generally trained online, requiring interaction with real humans or user simulators during training and are not naturally suited to fully utilize large TACT corpora, (2) Most proposed algorithms require pre-defining a set of discrete actions and slots for the state and action representation, which requires significant human effort and domain knowledge – thus, not readily generalizing to new domains.

To overcome the first problem, one can pre-train RL models offline with supervised learning methods and then fine-tune policies online with real humans (or a user simulator) in the loop [21, 28, 23]. In addition to still requiring human interaction, pre-training also requires human effort to annotate each agent utterance in the training data with a pre-defined action and annotate entities with pre-defined slots (i.e., they require annotated data). Kandasamy et al. [8] proposed a batch policy gradient algorithm, which weighs trajectories in the dataset by importance sampling. Thus, it can learn dialog policy offline, but may suffer from large gradient variance due to importance sampling. Secondly, this algorithm also requires that reward signals exist in the dataset, which is not necessarily the case in practice.

To overcome the problem of requiring a pre-defined action space, several works [19, 6, 10] proposed using an encoder-decoder (i.e., sequence-to-sequence) neural architecture to learn a model for directly generating dialog responses. These approaches enable end-to-end learning and do not need domain knowledge to define the action or state space such that they can easily generalize to different domains. However, similar to the previously mentioned supervised learning algorithms, these algorithms myopically optimize the likelihood of next utterance and neglect the overarching goal of the dialog. They also tends to generate very generic responses [8, 10].

Inspired by recent RL-based algorithms for sequence prediction [2, 17], we propose a RL-based end-to-end dialog policy learning algorithm that can overcome these two drawbacks. Specifically, we model agent response generation as a Markov decision process (MDP) in which each episode (from an initial state to a terminal state) corresponds to a sequence of words in an agent utterance. Note that this is different from the MDP defined in related literature [5, 23], where each episode corresponds to a sequence of dialog acts. This MDP has a known transition function and defined reward function, so we can learn the optimal policy offline with an on-policy algorithm without interacting with real users. We adopt an neural encoder-decoder architecture to parameterize the policy such that our algorithm can be trained on unannotated data without defining dialog acts or slots, and still reap the benefits of the latest encoder-decoder architectures. The main contributions of this paper are as follows:

1. We propose a novel reward function that takes into account both utterance-level and dialog-level rewards. This reward function guides the algorithm to optimize not only the next agent utterances, but also the overall dialog goals.
2. When learning the optimal policy of the defined MDP, we propose to use off-policy policy gradient to accelerate the convergence of on-policy policy gradient.
3. Our algorithm achieves better results than state-of-the-art sequence-to-sequence models on bAbI task 6,[1] a widely used goal-oriented dialog dataset.
4. The proposed algorithm learns dialog policy from unannotated dataset offline without interacting with real users. This enables us to utilize the vast amount of existing dialogs in TACTs, and makes building industry-quality chatbots possible.

2 Related Work

There are two main approaches to learning a dialog policy in the literature, action prediction (i.e., utterance-level prediction) and sequence prediction (i.e., token-level prediction).

Dialog policy learning as action prediction. In this line of work, the algorithms first predict the agent's next action, and then generate agent utterances based on the action. Some supervised learning algorithms treat each action as one class, and then perform multi-class classification. For

[1] https://github.com/facebook/bAbI-tasks

example, Bordes and Weston [3] and Williams et al. [28] proposed to use Memory Network and recurrent neural network (RNN) as the underlying multi-class classification model respectively. Usually, these algorithms are followed by training with reinforcement learning to further refine the model [21, 28, 23, 27]. Williams et al. [28] achieved the state-of-the-art on bAbI dialog dataset with a neural architecture, by relying on injecting domain-specific knowledge and constraints, such as action masks. Meanwhile, in this paper, we focus on learning dialog policy without such domain knowledge. Many RL algorithms have also been proposed [15, 5, 30, 22]. Usually, these algorithms need to interact with real users or user simulators, which makes them difficult to scale to industry-quality applications. Some batch reinforcement learning algorithms have been proposed [16, 11]. However, these algorithms requires annotated agent actions and rewards, which are often not available.

Dialog Policy Learning as sequence prediction. The algorithms in this line of work generate agent utterances token by token given the dialog context [19, 6, 20]. Usually these algorithms adopt an encoder-decoder architecture, some having an additional belief tracker [26] or latent intent variable [25]. Outside of dialog management, RL-based sequence prediction algorithms have been proposed. Specifically, Ranzato et al. [17] and Bahdanau et al. [2] use policy gradient and actor critic to learn to generate sequences for machine translation and document summarization. However, when directly applied to dialog generation, these methods would only optimize the reward of next utterance instead of the overarching goal of the dialog. Li et al. [10] proposed a RL-based dialog generation algorithm which optimizes a set of rewards such as information flow and semantic coherence, focusing on open-domain dialogs. Kandasamy et al. [8] proposed a batch policy gradient method that can train dialog policies offline from existing dataset. Our proposed algorithm is a sequence prediction method, but differs from their work in that we do not assume the rewards are available in the dataset. Instead, we make the more realistic assumption that the trained agent utterances in the dataset are of high quality and can be treated as targets to learn, which is the case in TACT datasets.

3 Method

3.1 Problem Formulation

We represent each dialog in the training dataset as a sequence of pairs $D = \{(x^k, y^k)\}_{k=1}^K$, where K is the number of turns, x^k is the user utterance at turn k, and y^k is the agent utterance at turn k. We define the context of the dialog at turn k as $c^k = (x^1, y^1, ..., y^{k-1}, x^k)$, that is, the concatenation of all utterances before y^k. Usually, in a goal-oriented dialog dataset, there are API calls which are issued by agents to external systems. We treat an API call as an agent utterance starts with a special `api_call` token followed by the parameters of that API call.

Given the context c^k of the dialog, the algorithm generates an agent utterance, token by token. We formulate this process as a Markov decision process (MDP). The action space $\mathcal{A}$ of this MDP is the agent's vocabulary. At each time t, an action is performed, that is, a token from the vocabulary is generated. Let $z_{t-1}^k = (a_1^k, a_2^k, ..., a_{t-1}^k)$ be the sequence of tokens generated until time $t-1$. Then, the state of the MDP is defined as $s_t^k = (c^k, z_{t-1}^k)$, that is, the concatenation of the dialog context and the tokens that have already generated. The stochastic policy $\pi_\theta(\cdot|s_t^k)$ produces a distribution over the tokens in the vocabulary based on the current state, where θ is the parameters of the policy we try to learn. One token $a_t^k \sim \pi(\cdot|s_t^k)$ is sampled from the distribution, and the state is deterministically transitioning to $s_{t+1}^k = (c^k, z_t^k)$. The terminal states are the states in which the last generated token is a special EOS (end-of-sentence) token. We use z^k to denote the generated agent utterance, which has a EOS token at the end. The length of z^k is denoted by T. We will shortly describe how we define the reward function of this MDP. The goal of the learning algorithm is to maximize the cumulative rewards of the generated sequences, which is defined in section 3.2.1.

3.2 Algorithm

We parameterize our stochastic policy $\pi_\theta(\cdot|s_t^k)$ as an attention-based sequence-to-sequence (SEQ2SEQ) model [1]. Given a state, s_t^k, the encoder reads the dialog context, c^k, and the decoder outputs a probability distribution over the vocabulary conditioned on the last generated token, a_{t-1}^k, and the RNN hidden states of both c^k and z_{t-1}^k. Note our algorithm is agnostic to the choice of the underlying model used to parameterize the policy; thus, state-of-the-art encoder-decoder techniques, such as different attention mechanisms, can be easily applied here to improve the model.

One natural choice to optimize our model parameters is to use *on-policy* policy gradient [29]. In this setting, the algorithm explores the action space $\mathcal{A}$ and generates an agent utterance z^k, then z^k is scored by a pre-defined reward function $r(z^k, D)$. One problem with this is that the action space $\mathcal{A}$ is the agent token vocabulary, which is quite large and extremely difficult to effectively explore in order to learn a good policy. To solve this problem, Ranzato et al. [17] and Bahdanau et al. [2] proposed to start from optimizing cross-entropy loss, and then slowly deviate from it to let the model explore.

In this paper, we propose to combine *on-policy* policy gradient with *off-policy* policy gradient [4, 8]. We show that if the actions in the dataset have high returns (future cumulative rewards), such as the actions in TACT datasets, then off-policy policy gradient can further speed up the convergence of the overall algorithm. Moreover, off-policy learning enables us to utilize any existing rewards in the dataset.

3.2.1 Learning with On-policy Policy Gradient

Since the transition function of the defined MDP is known (i.e., an action that adds a word to an utterance will always transition to the state where the word is concatenated to the said utterance), once we define a reward function $r(z^k, D)$ based on the generated utterance z^k and the dialog D in the dataset, we can use on-policy policy gradient to learn the optimal policy.

We define two types of rewards: *utterance-level* rewards (i.e., rewards distributed within a single utterance) and *dialog-level* rewards (i.e., rewards that cross single utterance boundaries). Utterance-level rewards capture the quality of the generated agent utterances z^k compared with the existing agent utterances y^k in the training data. Example reward functions can include the semantic distance between the two utterances or simply the cosine similarity. In our experiments, we use BLEU scores [14] to derive utterance-level rewards, which is one of the most popular metrics in machine translation and has been shown to correlate well with human judgement.

The dialog-level rewards, on the other hand, captures the contribution of the generated utterances to achieving the dialog goals. We observe that goal-oriented dialogs are usually driven by issuing API calls to a database or knowledge base. For example, in a scenario in which a customer wants to return a product bought from a shopping website, a customer service agent has to first issue an API call to pull out the customer's profile. Then, the agent needs to check whether the customer is eligible for a refund. Finally, the agent issues another API call to start the refund process, and the goal of the dialog is achieved. API calls usually have parameters. In the above example, the parameters for the first API call could be the customer's email address, and the parameters for the second one could be the order id and produce id. API calls are usually logged and available for training.

It is critical for a policy to predict API calls and the parameters of the API calls correctly. When a API call is issued later than the one in the training data, it increases the number of turns of the dialog and may decrease the user experience. A worse case is when a API call is issued earlier than the one in the training data, since it has a risk to guess the parameters of the API call. Worst of all is the case when the API call is never issued and the goal of the dialog is missed. As a results, the dialog-level reward function we define gives different negative rewards for API calls that are made too late or too early, and a positive reward for each correct parameter of a API call that is made on time.

We define the reward of a generated agent utterance z^k as the sum of utterance-level and dialog-level rewards:

$$r(z^k, D) = \text{BLEU}(z^k, y^k) + \begin{cases} 0 & \text{if } z^k \text{ and } y^k \text{ are not API calls} \\ -\lambda_a & \text{if } z^k \text{ is not an API call but } y^k \text{ is} \\ -\lambda_b & \text{if } z^k \text{ is an API call and } k < k' \\ -\lambda_c & \text{if } z^k \text{ is an API call and } k > k' \\ \lambda_d * \text{\#correct\_parameters} & \text{if } z^k \text{ is an API call and } k = k' \end{cases} \quad (1)$$

where λ_a, λ_b, λ_c, and λ_d are hyper-parameters of the algorithm and are between 0 and 1, and k' is the actual turn in which the corresponding API call was issued in the training data. BLEU scores are between 0 and 1.

The reward $r(z^k, D)$ is assigned to the last action in z^k (an EOS token) and all the intermediate actions get a reward of 0. As a result, the reward signals are sparse and the learning process can be slow. Similar to Bahdanau et al. [2], we use reward shaping [13] to deal with this problem. We define the potential function $\Phi(z_t^k) = \text{BLEU}(z_t^k, y^k)$ for incomplete utterances and $\Phi(z^k) = 0$ for

complete ones. Then the reward for action a_t^k is $r_t^k = \Phi(z_t^k) - \Phi(z_{t-1}^k)$ for all $t < T$, and the reward for the last action a_T^k is $r_T^k = r(z^k, D)$. The optimal policy doesn't change under reward shaping.

The objective function to maximize is

$$J(\theta) = \mathbb{E}\left\{\sum_{t=1}^{T} \gamma^{t-1} r_t^k \mid \pi_\theta\right\}$$

where γ is the discount factor. The policy gradient is estimated by [24]:

$$\nabla J_{\text{on-policy}}(\theta) = \sum_{t=1}^{T} \left[\nabla_\theta \log \pi_\theta(a_t^k|s_t^k) \left(Q^\pi(s_t^k, a_t^k) - b(s_t^k)\right)\right] \tag{2}$$

where $b(s_t^k)$ is called the baseline [29], which is the average return of the current policy at s_t^k. We can use a function approximator such as another neural network to estimate $Q^\pi(s_t^k, a_t^k)$. Then the algorithm falls into the actor-critic framework [2]. However, for simplicity, in this paper we use Monte Carlo estimate of $Q^\pi(s_t^k, a_t^k)$, which is similar to the REINFORCE [29] algorithm:

$$Q^\pi(s_t^k, a_t^k) = \sum_{t'=t}^{T} \gamma^{t'-t} r_{t'}^k$$

3.2.2 Acceleration with Off-Policy Policy Gradient

When the action and state space are large, on-policy RL algorithms can converge slowly. One reason is that on-policy algorithms have to explore and experience actions with high returns before it can learn to pick these good actions. However, in our case, the number of all possible sequences to generate is $|\mathcal{A}|^T$, where $|\mathcal{A}|$ is the size of the agent token vocabulary and on-policy exploration is difficult. Meanwhile, in TACT datasets, actions are generated by trained agents and are expected to have high returns. As the dialogs in these datasets are demonstrations of good trajectories, it is sensible to reduce exploration efforts by using this information.

In this paper, we propose to use off-policy policy gradient [4] to exploit such information. Off-policy policy gradient essentially maximize the probability of actions in the dataset, weighted by importance sampling ratios and the returns of these actions. Denoting the i-th token in y^k as o_i^k and $\tilde{T}$ as the length of y^k, the state of the MDP is represented by $\tilde{s}_t^k = (c_t^k, o_1^k, ..., o_{t-1}^k)$, similar to the on-policy setting. Off-policy policy gradient is estimated by [8]:

$$\nabla J_{\text{off-policy}}(\theta) = \sum_{t=1}^{\tilde{T}} \left[\frac{\pi_\theta(o_t^k|\tilde{s}_t^k)}{q(o_t^k|\tilde{s}_t^k)} \nabla_\theta \log \pi_\theta(o_t^k|\tilde{s}_t^k) \left(Q^\pi(\tilde{s}_t^k, o_t^k) - b(\tilde{s}_t^k)\right)\right] \tag{3}$$

where

$$Q^\pi(\tilde{s}_t^k, o_t^k) = \left[\prod_{t'=t}^{\tilde{T}} \frac{\pi_\theta(o_t^k|\tilde{s}_t^k)}{q(o_t^k|\tilde{s}_t^k)}\right] \left[\sum_{t'=t}^{\tilde{T}} \gamma^{t'-t} \tilde{r}_t^k\right] \tag{4}$$

and $q(\cdot|\tilde{s}_t^k)$ is the behavior policy, that is, the policy used to generate the training dataset. We use the same reward function defined in section 3.2.1 to calculate $\tilde{r}_t^k$. However, if there already exists reward signals in the training data, that can also be used as $\tilde{r}_t^k$.

Note that in equation (3) and (4), $\frac{\pi_\theta(o_t^k|\tilde{s}_t^k)}{q(o_t^k|\tilde{s}_t^k)}$ is called the importance sampling coefficient. We need the behavior policy $q(\cdot|\tilde{s}_t^k)$ to calculate this coefficient, however, the behavior policy is usually unknown. To solve this problem, Kandasamy et al. [8] proposed to train another RNN model to estimate $q(\cdot|s_t^k)$. However, the importance sampling coefficient can be very large if $q(\cdot|s_t^k)$ is very small, and this can lead to high variance of the gradients. To deal with this problem, one can clip this value, or simply set it to a constant value if $\pi_\theta(\cdot|s_t^k)$ is close to $q(\cdot|s_t^k)$ [7, 12]. Note that if we set the coefficient to a constant value of 1.0 and only use utterance-level rewards, then off-policy policy gradient reduces to optimizing cross-entropy loss – meaning that the proposed algorithm will outperform the underlying encoder-decoder if a reasonable reward function can be defined.

We update the policy parameters with a convex combination of the on-policy and off-policy gradient:

$$\nabla J(\theta) = (1 - \lambda_e)\nabla J_{\text{off-policy}}(\theta) + \lambda_e \nabla J_{\text{on-policy}}(\theta)$$

where $\lambda_e \in [0, 1]$. Our algorithm is outlined in Algorithm 1.

Algorithm 1: End-to-End Offline Dialog Policy Learning

Input: $\lambda_a, \lambda_b, \lambda_c, \lambda_d \in [0, 1]$: reward function hyper-parameters
$\lambda_e \in [0, 1]$: policy gradient coefficient

while *Not Converged* **do**
  Sample a random dialog D
  for $k = 1, ..., K$ **do**
    Use the current policy π_θ to predict the agent utterance z^k given the context $c^k = (x^1, y^1, ..., y^{k-1}, x^k)$
    Calculate $r(z^k, D)$ based on equation (1). Get $r(y^k, D)$ from the data if available, otherwise calculate $r(y^k, D)$ based on equation (1)
    Calculate $J_{\text{on-policy}}(\theta)$ and $J_{\text{off-policy}}(\theta)$ based on equation (2) and (3).
    Update θ with $\nabla J(\theta) = (1 - \lambda_e)J_{\text{off-policy}}(\theta) + \lambda_e J_{\text{on-policy}}(\theta)$
  end
end

4 Experiments

4.1 Dataset

We use the bAbI dialog task 6 dataset [3] for our experiments. The bAbI dialog task 6 was converted from the 2nd Dialog State Tracking Challenge, and is in the context of restaurant search. The goal of the dialog is to recommend a restaurant based on a user's preferences. In each dialog, the agent asked the users questions about their preferences on type of cuisine, location and price range. The users can also initiate the dialog by providing these preferences. The agent then issued an API call to a knowledge base, which returned a list of candidate restaurants and their attributes such as rating, phone number, address, etc. The restaurant with the highest rating was recommended to the user. The user can then ask further information such as the phone number or address of the restaurants. The user may update their preferences during the dialog, so there can be multiple API calls in one dialog. Although the agent utterances in the dataset are not generated by trained human agents, they are generated by a well-performed hand-crafted dialog policy. Therefore, the agent utterances can still be treated as ground truth for learning purposes.

The data contains the raw transcripts of the dialogs, which includes the agent utterances (including API calls and parameters of the API calls), user utterances, and knowledge base responses of the API calls. The train/validation/test set contains 1618, 500, and 1117 dialogs respectively. The size of the vocabulary is 1229. The average number of utterances per dialog is 14, and the average number of records returned from API calls is 40. Each record is a tuple consists of a restaurant name, an attribute name, and an attribute value.

4.2 Training

One benefit of our algorithm is end-to-end training. In our experiments, we trained on the raw data without any data pre-processing such as normalizing tokens, replacing entities with special tokens, and similar procedures. We also fed all the knowledge base responses to the encoder, so the algorithm has to learn to pick the restaurant with the highest rating. Note that the knowledge base responses are long lists of restaurants and their attributes, so this dramatically increases the length of the dialog contexts (i.e. the length of the input sequences of the encoder). The longest dialog contexts in the training data contains 1556 tokens, the average length of the contexts is 152.94. The length of the longest agent utterance in the training data is 29, and the average length of the agent utterances is 10.07. The purpose of using raw data is to evaluate the performance of the algorithm without any domain knowledge.

Model	Per Response Accuracy	BLEU	API Call			
			Precision	Recall	F_1 score	Exact Match
Memory Network	41.10	-	-	-	-	-
Attention SEQ2SEQ	47.29	57.36	33.69	**84.83**	48.23	67.28
Eric and Manning [6]	48.00	56.00	-	-	-	-
Our algorithm	**48.69**	**58.25**	**35.22**	81.34	**49.16**	**76.95**

Table 1: Evaluation on bAbI task 6. A dash indicates that the result is unavailable.

We used the same model hyper-parameters as described in Eric and Manning [6], and encoder-decoder based model, such that we can directly compare with their results. The encoder was a single layer bi-directional LSTM, and the decoder was a single layer LSTM. The word embedding size was set to 300, and the number of units in the LSTM was set to 353. The input dropout keep rate was set to 0.8. We used the Adam optimizer [9] and set the learning rate to 10^{-3}. We set the importance sampling coefficient to a constant value of 1.0, and set the maximum length of decoding sequence to 35. For the reward function, since we are evaluating the algorithm with an existing dataset instead of real humans, issuing API call too early and too late are equally bad in terms of our metrics, so we set $\lambda_a = \lambda_b = \lambda_c = \lambda_d = 0.1$. For the convex combination of on-policy and off-policy gradient, we set $\lambda_e = 0.3$. We selected the model that performed the best on the validation dataset, and reported the performance of that model on the test dataset. We report the following metrics: 1) Per Response Accuracy: the accuracy of predicting the next agent utterance. The prediction is correct only if all the predicted tokens match the corresponding ones in the dataset. 2) BLEU score of the generated utterances. 3) Precision, Recall and F_1 score (micro-averaged) of issuing API calls. A prediction is considered as true positive if an API call is correctly predicted as an API call, even if the parameters of the API call are wrong. 4) API call Exact Match: within all the true positives of issuing API calls, the accuracy of predicting every parameter correctly.

The baseline algorithm used in our experiment is an attention-based SEQ2SEQ model [1], which is trained using the same hyper-parameters as our algorithm. We also include the performance of Memory Network reported by Bordes and Weston [3], and the performance of copy-augmented SEQ2SEQ model reported by Eric and Manning [6].

4.3 Results

The experiment results are shown in Table 1, where we observe that our algorithm improved the BLEU score by 1.55% compared with the attention-based SEQ2SEQ model and by 4.02% compared with Eric and Manning [6]. This shows that our learned policy achieved better utterance-level performance. Furthermore, we can easily update the reward function to optimize other utterance-level metrics or a combination of different metrics. Our algorithm also improved the F_1 score of issuing API calls by 1.93%, and significantly improved the accuracy of API calls' parameters (exact match) by 14.37% compared with the attention-based SEQ2SEQ model. As we mentioned before, goal-oriented dialogs are usually driven by the API calls. Accordingly, better performance on API call metrics implies a better goal-oriented policy. Finally, our algorithm improved the per-response accuracy by 2.96% compared with attention-based SEQ2SEQ and by 1.44% compared with Eric and Manning [6].

5 Conclusion

In this paper, we propose an RL-based algorithm to learn goal-oriented dialog policy. The algorithm enables the following useful contributions: 1) we define a reward function such that our algorithm can optimize both utterance-level and dialog-level metrics. 2) we learn RL-based dialog policies by fully utilizing TACT datasets without interacting with real users. 3) we improve the sample efficiency of on-policy policy gradient by incorporating off-policy policy gradient. 4) we parameterize our policy with an encoder-decoder architecture, which enables end-to-end learning with no domain-specific knowledge. Compared with recently proposed methods, our algorithm achieved better performance on both utterance-level and dialog-level metrics on bAbI dialog dataset. Our algorithm excels in the scenario when there exists a large amount of TACTs, which is the common case in industry setting. Therefore, we believe this work is an important step towards building industry-quality chatbots.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *International Conference on Learning Representations*, 2015.

[2] Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. In *International Conference on Learning Representations*, 2017.

[3] Antoine Bordes and Jason Weston. Learning end-to-end goal-oriented dialog. In *International Conference on Learning Representations*, 2017.

[4] 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, 2012.

[5] Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. Towards end-to-end reinforcement learning of dialogue agents for information access. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 484–495, 2017.

[6] Mihail Eric and Christopher D Manning. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, page 468, 2017.

[7] Edward L Ionides. Truncated importance sampling. *Journal of Computational and Graphical Statistics*, 17 (2):295–311, 2008.

[8] Kirthevasan Kandasamy, Yoram Bachrach, Ryota Tomioka, Daniel Tarlow, and David Carter. Batch policy gradient methods for improving neural conversation models. In *International Conference on Learning Representations*, 2017.

[9] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[10] 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.

[11] Lihong Li, He He, and Jason D Williams. Temporal supervised learning for inferring a dialog policy from example conversations. In *Spoken Language Technology Workshop (SLT), 2014 IEEE*, pages 312–317. IEEE, 2014.

[12] Rémi Munos, Tom Stepleton, Anna Harutyunyan, and Marc Bellemare. Safe and efficient off-policy reinforcement learning. In *Advances in Neural Information Processing Systems*, pages 1054–1062, 2016.

[13] Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *Proceedings of the 16th International Conference on Machine Learning*, 1999.

[14] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics, 2002.

[15] Baolin Peng, Xiujun Li, Lihong Li, Jianfeng Gao, Asli Celikyilmaz, Sungjin Lee, and Kam-Fai Wong. Composite task-completion dialogue policy learning via hierarchical deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2221–2230, 2017.

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

[17] Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*, 2016.

[18] Stéphane Ross, Geoffrey J Gordon, and Drew Bagnell. A reduction of imitation learning and structured prediction to no-regret online learning. In *International Conference on Artificial Intelligence and Statistics*, pages 627–635, 2011.

[19] 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 *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, pages 3776–3783, 2016.

[20] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. A neural network approach to context-sensitive generation of conversational responses. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics*, pages 196–205, 2015.

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

[22] Pei-Hao Su, Milica Gasic, Nikola Mrkšić, Lina M. Rojas Barahona, Stefan Ultes, David Vandyke, Tsung-Hsien Wen, and Steve Young. On-line active reward learning for policy optimisation in spoken dialogue systems. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 2431–2441, 2016.

[23] 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.

[24] Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pages 1057–1063, 2000.

[25] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. Latent intention dialogue models. In *Proceedings of the 34th International Conference on Machine Learning*, pages 3732–3741, 2017.

[26] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M. Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, 2017.

[27] Jason D Williams and Geoffrey Zweig. End-to-end lstm-based dialog control optimized with supervised and reinforcement learning. *arXiv preprint arXiv:1606.01269*, 2016.

[28] Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 665–677, 2017.

[29] Ronald J Williams. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256, 1992.

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