Dialogue policy optimization for low resource setting using Self-play and Reward based Sampling

Tharindu Madusanka, Durashi Langappuli, Thisara Welmilla,
Uthayasanker Thayasivam and Sanath Jayasena
Department of Computer Science and Engineering,University of Moratuwa ,Sri Lanka
{stharindu.16, durashi.16, welmilla.16, rtuthaya, sanath}@cse.mrt.ac.lk

Abstract

Reinforcement Learning is considered as the state of the art approach for dialogue policy optimization in task-oriented dialogue systems. However, these models demand a large corpus of dialogues to learn effectively. Training Reinforcement Learning agent with low data amount tends to overfit the agent. Although synthesizing dialogue agendas with dialogue Self-play using rule-based agents and crowdsourcing has demonstrated promising results with the low amount of samples, these methods hold limitations. For instance, rule-based agents acquire specific domain and language while crowdsourcing demands a high price and domain experts, especially in local languages. In this paper, we address these limitations by proposing a novel approach for synthetic agenda generation by acknowledging the underlying probability distribution of the user agendas and a reward-based sampling method that prioritizes failed dialogue acts. Evaluations conducted shows leveraged performance without overfitting, compared to the baseline method. Also, the reward-based sampling method improves the overall mean task success rate by an average of 11.307%.

1 Introduction

A dialogue system, or conversational agent, denotes a system that can conduct a conversation with another agent, usually a human (Perez-Marin and Pascual-Nieto, 2011). One type of conversational agent are Task-oriented conversational agents, that can help users accomplish tasks ranging from meeting scheduling to vacation planning. The structure of a task-oriented conversational agent is outlined in Figure 1. It consists of (i) a natural language understanding(NLU) module for identifying intents of user utterances (ii) a dialogue state tracker(DST) for tracking conversation state (iii) a dialogue policy learner(POL) which selects the next action based on the current state (iv) a natural language generator(NLG) for converting the agent action to a natural language response (Gao et al., 2018).

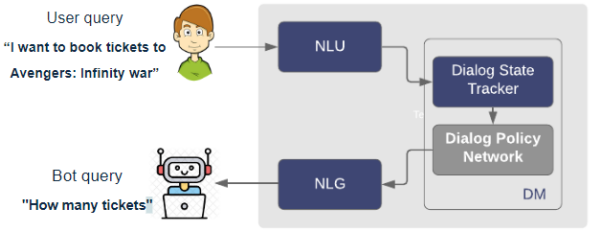

Figure 1: Components of a task-oriented conversational agent

Dialogue Policy Network(learner) plays a critical role in task-oriented conversational agents since they require logical reasoning and planning over several dialogue turns. Policy Network has been developed under rule-based methods(i.e. ontology-based, finite state machines) and model-based methods(i.e. Supervised Learning(SL) based (Bordes et al., 2016; Dinan et al., 2018), Reinforcement Learning(RL) based (Lu et al., 2019; Liu and Lane, 2018; Su et al., 2016a)). Due to enhanced data availability, dialogue policy optimization has leveraged with RL based methods. However, these state-of-the-art RL methods have barely experimented in the low resource

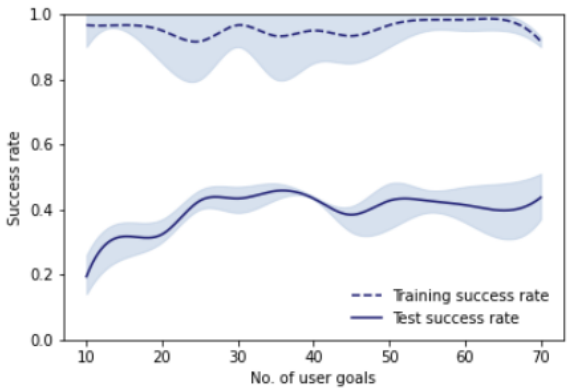

Figure 2: Training and test success rates of user goals with low amount of samples

setting because these models tend to overfit when the number of available training data is low(see Figure 2).

The accuracy of the dialogue policy network directly depends on the availability of quality dialogue samples to train. As depicts in Figure 2, the agent tends to overfit when the number of training samples is low. The dialogue Self-play approach has proposed to overcome this issue with (i) crowdsourcing(Shah et al., 2018a) or manually synthesize agendas and (ii) rule-based agents to synthesize agendas (Shah et al., 2018b). However, these methods have limitations since crowdsourcing is expensive especially considering low resource local languages, and rule-based agents are limited to a specific domain and language. Thus an alternative approach to address this issue is to synthesize agendas considering all the possibilities. But this method conduces to create unrealistic agendas besides awarding equal probability to every possible state that affects the speed of convergence of Reinforcement Learning agents.

To address these limitations, we propose a novel approach for synthetic agenda generation by acknowledging the underlying probability distribution of the user agendas. Since this methodology applies to a low amount of samples, this method can lead to an insufficient exploration of agendas by the agent. Therefore we further developed the methodology by introducing a selective sampling method based on the reward function that prioritizes the failed dialogue acts, where the agent actively decides what agendas to use. The intuition is that the agent can learn more from failed dialogues over successful ones.

The rest of the paper is organized as follows: Section 2 describes the Background on RL and data synthesis. Section 3 presents the related work for dialogue policy optimization using RL and low resource settings. Our methodology is fully described in section 4. We conduct our experiments and show the results in section 5. Finally, we conclude our work in section 6.

2 Background

2.1 Reinforcement Learning

Reinforcement Learning (RL) is a learning paradigm where an intelligent agent learns to make optimal decisions by interacting with an initially unknown environment(Sutton and Barto, 2018). The agent interacts with the environment by observing the state s_t and taking an action a_t. Depend on the action the agent receives a reward r_{t+1} and observes the state change s_{t+1}. This continues until the episode ends. The agent's goal is to maximize the cumulative reward at each step and the cumulative reward at step t is denoted by G_t

$$G_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots$$
$$= \sum_{i=1}^{\infty} \gamma^{i-1} r_{t+i}$$

where the γ is the discount factor. The action an agent takes at state s defined by the policy the agent follows. The policy is a function that maps states to actions and denoted by π. The agent's goal is to find the optimal policy denoted by π^*.

There are mainly three types of methods for finding the optimal policy. They are the value-based methods, the policy-based methods, and the actor-critic methods. In value-based methods, a value function is used to express the value of the state or state-action pair. Note the value function define with respect to the policy and the optimal value function denoted by either by V^* or Q^*. So if the optimal value function is known the optimal policy can be found by,

$$\pi^*(s) = argmax_a(Q^*(s, a))$$

The Monte-Carlo method, Q-learning, and DQN (Mnih et al., 2013) which use deep neural networks are popular value-based methods.

The policy-based methods find the optimal policy directly without using any value function. The well-known policy-based methods are Reinforce (Williams, 1992) and Proximal policy optimization (PPO) (Schulman et al., 2017). The actor-critic methods(Konda and Tsitsiklis, 2000) combine the two methods. Here, there is an actor which acts according to certain policy and a critic which tries to estimate the value function for a given state.

The dialogue policy learning is a policy optimization problem, and many researchers have used Reinforcement Learning to find the optimal policy (Pineau and Thrun, 2004; Gasic and Young, 2014; Williams and Young, 2007). Here, the policy network is the agent, and the environment is the user or the user simulator. The agent keeps track of the state of the dialogue, and the state is defined using intents, entities, and slot values. The agent takes action depending on the state. This action can be generating a dialogue act or sending an API call etc. For the action taken by the agent, a reward is given by the user simulator or user at the end of dialogue or each step. Usually, in task-oriented dialogue systems, the reward is associated with the task-success rate. The task-success rate is the measure of how much of the user's task is achieved or not.

2.2 Data synthesization

Machine learning specifically in the context of Supervised Learning can be defined as an approach that tries to get $min_\theta \sum L(\theta, D)$, where D is the data and θ is the parameters of the model. Usually, when models are trained, in each episode of training a random sample(batch), D_s is drawn from the data D used for updating model parameters. However if enough data is not available, we may need to generate data or use a different type of learning method like Meta-learning. If data is generated then this generated data, S is used for training either with D or instead of D. S can be created using a Generator function, G or manually.

$$S = G(D) \text{ and } G \sim P(D)$$

The usual method of sampling from synthetic data, S is random sampling,

$$D_s \sim random(S)$$

3 Related work

The Reinforcement Learning approaches are more suitable for modeling the policy network of the conversational agent and it has already shown very promising results in dialogue policy optimization research(Larsson and Traum, 2000; Li et al., 2009; Lucie et al., 2019; Gasic and Young, 2014). It is capable of exploring a large action state space and approximating the optimal policy. However, the Reinforcement Learning methods are rarely used for the low resource setting as these approaches require more samples with respect to the rule-based and Supervised Learning approaches.

Most of the research works in the area of using Reinforcement Learning for dialogue policy learning are mostly focused on optimizing the reward function(Su et al., 2016a; Liu and Lane, 2018), improving task success rate(Li et al., 2009), achieving personalization(Mo et al., 2018; Hengst et al., 2019) and making system end-to-end (Dhingra et al., 2016; Wen et al., 2016) or interactive(Shah et al., 2016; Liu et al., 2018).

Some research work has been done on making the Reinforcement Learning approaches more sample efficient. One such approach is using warm-start such as the imitation learning where the agent mimics an expert provided policy(Li et al., 2015), Replay Buffer Spiking(RBS)(Lipton et al., 2018) which is used with DQN and use expert generated dialogues(Henderson et al., 2008; Chen et al., 2017). Another approach for improving the RL-based approach is using an efficient method for exploration(Lipton et al., 2018). Even though this research work has made the novel Reinforcement Learning approaches more sample efficient, they still cannot be used in low resource settings as they still require large amount of training dialogues to reach a sufficient level of accuracy.

Several dialogue Self-play approaches have proposed to synthesize agendas with (i) crowdsourcing(Shah et al., 2018a) or manually synthesizing (ii) rule-based agents for synthesizing (Shah et al., 2018b) (iii) synthesize agendas considering all possibilities. However, these methods have limitations. Crowdsourcing is an expensive process and may lead to additional practical problems especially in the local language setting and rule-based agents are limited to a specific language or domain. Although these is-

sues are addressed by the last method, it generates unrealistic agendas and can affect the rate of convergence.

Our goal is to solve the overfitting problem that occurs when the dialogue policy network is implemented with the Reinforcement Learning technique in a low resource setting. Thus, we propose a combination of two techniques. Addressing current issues in the Self-play approaches for agenda synthesis in the conversational domain, we proposed a Self-play mechanism that uses underlying probability distributions of dialogue acts to synthesize new agendas. However, this proposed method may cause insufficient exploration of agendas by the agent, due low amount of data. To address this issue, we introduce a selective sampling method that prioritizes failed dialogue acts based on the reward function.

4 Proposed Methodology

Proposed methodology has two main components. The first one is the Self-play mechanism that calculate underlying probability distributions from the training data and feed dialogue acts sample from these probability distributions to the user simulator. The second component is responsible for reward based sampling technique that prioritizes failed dialogues over successful ones.

4.1 Definitions

The objective of the methodology is to synthesize data that reflect the true distribution of the data and then sample and train the agent in an efficient manner. Let training data(Agendas) is denoted by D and $P(D)$ denotes the data distribution. Let there be N number of slots and s_q denotes the q^{th} slot ($1 \leq q \leq N$). To capture the $P(D)$ we consider 3 independent discrete probability distributions.

1. $P(s_q)$ - Probability of slot s_q exists in an Agenda

2. $P(req|s_q)$ - Probability that slot s_q being requestable slot given that it exists($P(req|s_q)$ = 1 - $P(inform|s_q)$ where $P(inform|s_q)$ denotes the probability that slot s_q being informable slot)

3. $P(kb|inform, s_q)$ - Probability that the informable slot value available in the knowledge base given that slot exists in the agenda and it is informable.

We use these probabilities to capture $P(D)$. The methodology we describe and the mathematical equations are applied to all 3 probability distributions mention above. So for generalization we'll denote any of the above probability distributions with the term P. Let there are m elements in the probability distribution P. Let x_k denotes the k^{th} element in P ($1 \leq k \leq m$). $p(x_k)$ denotes the probability of x_k

4.2 Self-play Mechanism

An overview of the proposed methodology is illustrated in Figure 3. Instead of directly using the training dataset for training which leads to overfitting the model, we use the training dataset for calculating the underlying probability distribution and use that probability distribution for sampling agendas to optimize the policy network.

We add a small noise value ϵ_k for each element x_k.

$$\epsilon_k \sim \mathcal{N}(0,\, \delta^2) \tag{1}$$

where δ is chosen as a very small number. We add the noise for two purposes. One is to promote exploration by making $\forall x_k, p(x_k) > 0$, and the other is to avoid overfitting. We add that noise in a way such that,

$$0 < p(x_k) < 1 \quad \text{and} \quad \sum_{k=1}^{m} p(x_k) = 1$$

Once the probability distributions are calculated and noise is added, we use the mechanism that we created to sample dialogue acts from the probability distribution and feed it to the user simulator. This way agents can be trained without overfitting to the training dataset.

4.3 Prioritizing Failed Dialogues (Reward Based Sampling Technique)

We use a selective sampling technique that prioritizes failed dialogues over successful ones. The intuition behind the concept is that we believe the agent can learn more from the failed dialogues than successful ones. We use the reward as the indicator of the dialogues being successful or failed. We assume that the reward is negative for failed dialogues while it is positive when dialogues are successful. So the method we proposed uses the reward function and the

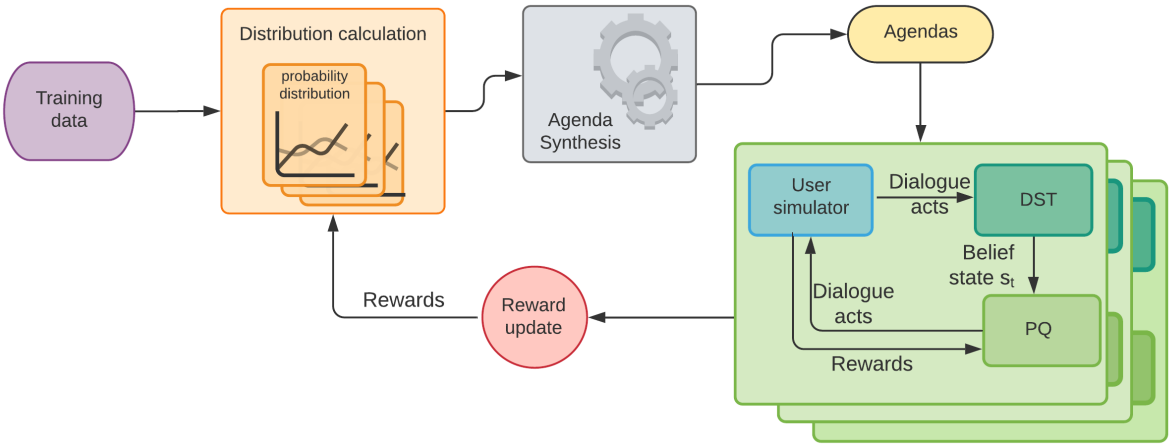

Figure 3: Our proposed Self-play framework, Distribution calculation and Agenda synthesis components responsible for synthesizing agendas, which then feed into the simulator. Reward update feed the reward signal for re-calculating distributions.

prior probabilities to recalculate the probabilities for sampling such that failed dialogues are prioritized.

So before any recalculation,

$$\sum_{k=1}^{m} p(x_k) = 1 \tag{2}$$

Let's consider element x_k. Let there be n_k agendas with element x_k in them and for each $i \leq n_k$, r_i^k denotes the reward. Since we want to prioritize the failed dialogues over successful ones we consider the negation. Also, we add the maximum possible reward to it to make sure that the resultant value is non-negative. Let $max(r)$ denotes the maximum possible reward while $min(r)$ denotes the minimum possible reward. Then we can calculate,

$$z_i^k = -r_i^k + max(r) \tag{3}$$

So that $\forall i, k,\ z_i^k \geq 0$. We can calculate the mean reward related to the element x_k by,

$$\begin{aligned} \frac{1}{n_k}\sum_{i=1}^{n_k} z_i^k &= \frac{-1}{n_k}\sum_{i=1}^{n_k} r_i^k + max(r) \\ &= w_k \end{aligned} \tag{4}$$

So w_k denotes the mean negative reward related to the element x_k. Then we can recalculate the probabilities from by considering the mean reward and previous probabilities.

$$f(x_k) = p(x_k)(\frac{w_k}{max(r)})^{\alpha} \tag{5}$$

The α is a hyperparameter which controls the influence of the reward when recalculating probability distributions. We divide by $max(r)$ for normalization. Then the resultant term $\frac{w_k}{max(r)}$ get normalized to a value between 0 and $(1 + \frac{|min(r)|}{max(r)})$. Then we can recalculate the probabilities by,

$$\begin{aligned} p(x_k) &= \frac{(f(x_k))^{\beta}}{\sum_{j=1}^{m}(f(x_j))^{\beta}} \\ &= \frac{(p(x_k)(\frac{\frac{-1}{n_k}\sum_{i=1}^{n_k} r_i^k}{max(r)} + 1)^{\alpha})^{\beta}}{\sum_{j=1}^{m}(p(x_j)(\frac{\frac{-1}{n_j}\sum_{i=1}^{n_j} r_i^j}{max(r)} + 1)^{\alpha})^{\beta}} \end{aligned} \tag{6}$$

$$\begin{aligned} p(x_k) &= \frac{exp\{(f(x_k)\beta\}}{\sum_{j=1}^{m} exp\{f(x_j)\beta\}} \\ &= \frac{exp\{(p(x_k)(\frac{\frac{-1}{n_k}\sum_{i=1}^{n_k} r_i^k}{max(r)} + 1)^{\alpha})^{\beta}\}}{\sum_{j=1}^{m} exp\{(p(x_j)(\frac{\frac{-1}{n_j}\sum_{i=1}^{n_j} r_i^j}{max(r)} + 1)^{\alpha})^{\beta}\}} \end{aligned} \tag{7}$$

Equation 6 and 7 normalize the $f(x_k)$ so that the $\sum_{j=1}^{m} p(x_k) = 1$. Equation 6 uses a division by the sum(naive normalization), while equation 7 uses the Softmax equation for normalization. In both cases, the β is a hyperparameter. So, we can use either equation 6 or 7 for recalculating probability distributions such that failed dialogues are prioritized (we can use equation 6 and 7 for reward-based sampling).

Algorithm 1 explain the flow of the process from getting the dataset to the continuous recalculation of probability distributions.

Algorithm 1: Self-play with reward based Sampling

Require: training, D
Require: hyperparametrs; α,β, and maximum reward; max(r)
calculate probability distributions from the D
add noise $\epsilon_k \sim \mathcal{N}(0,\, \delta^2)$ *for each* x_k *in each of the probability distributions* P
initialize reward buffer; R = {}
for *each training episode t* **do**
  sample agenda, A_t *from probability distribution*
  interact and train the agent
  store agenda A_t *and reward* r_t *in reward buffer* $R \cup \{A_t, r_t\}$
  if *recalculate probability* **then**
    recalculate probability distribution using either equation 6 or 7
  end if
end for

4.4 The Influence of α in Prioritizing Failed Dialogues

Consider the mean negative reward related to the element x_k, w_k and normalized mean negative reward $\frac{w_k}{max(r)}$, where max(r) is the maximum possible reward.

$$0 \leq \frac{w_k}{max(r)} \leq 1 + \frac{|min(r)|}{max(r)} \tag{8}$$

Note that if the more dialogues are successful then w_k is less than $max(r)$, while if more dialogues failed then w_k is greater than $max(r)$. Let w_i denotes a negative mean reward of a element where most dialogues succeed(more than half of dialogues succeed) while w_j denotes the negative mean reward of a element where most dialogues failed. Then,

$$0 \leq \frac{w_i}{max(r)} \leq 1 \tag{9}$$

$$1 \leq \frac{w_j}{max(r)} \leq 1 + \frac{|min(r)|}{max(r)} \tag{10}$$

So if we consider α_1,α_2, where $\alpha_1 < \alpha_2$ and α_3, where $\alpha_3 > 1$. Then,

$$(\frac{w_i}{max(r)})^{\alpha_1} > (\frac{w_i}{max(r)})^{\alpha_2} \tag{11}$$

$$(\frac{w_j}{max(r)})^{\alpha_1} < (\frac{w_j}{max(r)})^{\alpha_2} \tag{12}$$

and,

$$(\frac{w_i}{max(r)})^{\alpha_3} < \frac{w_i}{max(r)} \tag{13}$$

$$(\frac{w_j}{max(r)})^{\alpha_3} > \frac{w_j}{max(r)} \tag{14}$$

This means not only α controls the affect of the reward, but also it controls the separation between elements with majority failed dialogues and elements with majority successful dialogues.

5 Experiments and Results

We performed a set of well-designed experiments to evaluate our proposed methods. All the experiments are done in the movie booking domain. As the user simulator, we use the open-source simulator described by Li et al. (2016). The agent train by the user simulator for 200 episodes then evaluated against a test set. Each episode consists of 20 simulated dialogues followed by one epoch of training. Totally 29 slots are used and all of them are available from the beginning of the training process. To represent the dialogues we constructed 268-dimensional a feature vector.

From the dataset, we randomly sampled 30 user goals at each experiment as the test dataset. The rest of the dataset is used for training the agent. Purpose of the experiment is to measure the performance of the proposed methodology and evaluated against a baseline when number of available training samples are low. So we vary the number of training samples available for the agent in each experiment to train(or in case of self play, available for calculating probabilities). Experiments are done for different sizes of training datasets ranging from 10 to 70(we use a interval of size 5 when varying size of training samples). Each experiment-setting is done (i) without Self-play (ii) Self-play with naive normalization (iii) Self-play with softmax normalization (iv) Self-play without reward-based sampling. For each experiment setting, we plot the results as the mean of the five

Training dataset size	Without selfplay	Selfplay with naive	Selfplay with softmax	Selfplay without reward based sampling
10	0.19	0.63	**0.84**	0.58
15	0.32	0.71	**0.81**	0.60
20	0.32	0.79	**0.81**	0.66
25	0.43	0.78	**0.86**	0.76
30	0.43	0.68	**0.80**	0.73
35	0.46	0.73	**0.84**	0.68
40	0.43	0.73	**0.76**	0.68
45	0.38	0.75	**0.79**	0.76
50	0.43	**0.81**	0.72	0.68
55	0.43	**0.84**	0.79	0.76
60	0.41	**0.78**	0.76	0.73
65	0.40	**0.77**	0.76	0.68
70	0.44	**0.80**	**0.80**	0.74

Table 1: Mean test success rate for (i) without Self-play (ii) Self-play with naive normalization (iii) Self-play with softmax normalization (iv) Self-play without reward-based sampling. The mean is calculated for 5 different random seeds.

experiments. In all cases we have used a Deep-Q-Network(Mnih et al., 2013) as the agent. The results of the experiment are shown in the Figure 4.

Training details: First, the rule-based agent interacts with the simulator with 120 dialogues for the replay buffer spike. Next, the RL agent interacts with a simulator for 200 episodes each consist of 20 simulated dialogues. The resultant state-action reward pairs store in the replay buffer. Each episode followed by one epoch of training with a batch size of 16. During the training for the epoch, the agent freezes the target network parameters and then update the local Q -function.

Self-play mechanism also contains hyperparameters. We use the ϵ_i with $\delta = 0.0005$ for $P(s_q)$, $\delta = 0.01$ for $P(req|s_q)$ and $\delta = 0.01$ for $P(kb|inform, s_q)$. Also we use $\alpha = 0.5$ and $\beta = 1$ for reward based sampling with naive normalization and $\alpha = 0.5$ and $\beta = 10$ for reward based sampling with softmax normalization. α and β values are determined by grid-search. We start reward based sampling from 60^{th} episode onward.

Architectural details: All models are implemented as multilayer perceptrons(MLP's) with ReLU activations. Each model consists of 2 hidden layers each having 64 hidden neurons. The Adam optimizer (Kingma and Ba, 2014) used as the optimizer with the learning rate of 0.0005. The training batch size is 16. The target network update by soft update instead of hard update.

Results: As shown in Figure 4, our proposed methodology did not overfit and perform better than the baseline in every case. Directly training using the training data tends to overfit the model, hence a higher training accuracy is achieved, but the respective test accuracy is low. However, the test accuracy is increasing for the baseline model as the number of training samples increase. This is because as the amount of training samples increase, the model can generalize better. In most cases, reward-based sampling(prioritizing failed dialogues) method outperform the random sampling method(without prioritizing failed dialogues), especially when the number of training samples is extremely low. This is because the reward base sampling provides a better exploration of agenda space. Also, when the number of training examples available is low, reward base sampling with softmax function yields the best result. Since Softmax function takes the exponent of the probability and the mean negative reward's multiplication, the final probability calculation for elements has a higher standard deviation. This increase the degree of exploration, which is helpful when the number of training samples is low.

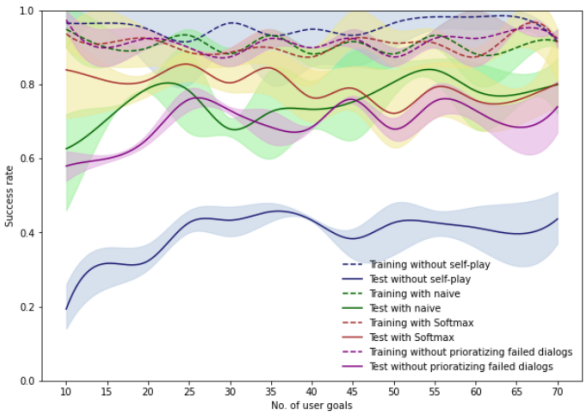

Figure 4: The mean train and test success rate for (i) without Self-play (ii) Self-play with naive normalization (iii) Self-play with softmax normalization (iv) Self-play without reward-based sampling. The mean is calculated for 5 different random seeds and curve is smoothed using interpolate spline technique.

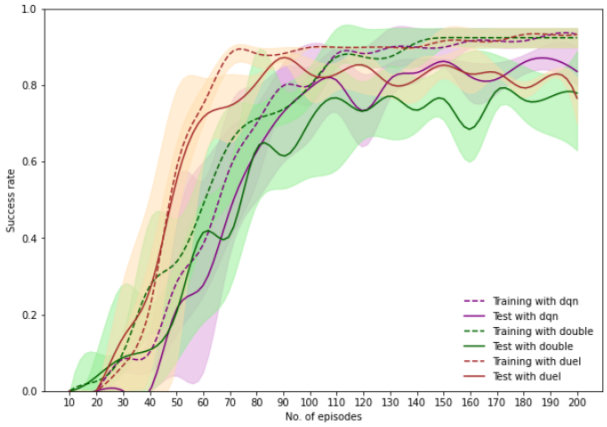

Figure 5: The mean train and test success rate for different RL methods. Kept number of training samples constant at 40 and vary the number of training episodes. The mean is calculated for 5 different random seeds and curve is smoothed using interpolate spline technique

Rate of convergence To verify that our proposed method train the model in a way such that models improve the testing success rate with their training success rate, we conduct an experiment keeping the training sample size constant and testing at different stages when training. This way, we can track how the test success rate improves with the number of episodes. In all cases, we use the reward-based sampling with naive normalization with an $\alpha = 0.5$ and $\beta = 1$.

We use the same simulator with the same movie booking dataset for this experiment as well. We randomly separate the data into train and test set, and in each run of the experiment, we use 40 samples for training and 40 samples for testing. We use the 40 training samples for calculating probability distributions, then use these probability distributions for Self-play mechanism. The agents interact with the user simulator for 200 episodes where each episode consists of 20 simulation dialogues followed by an epoch of training. After every 10 episodes, an evaluation is conducted using the test set. We start the reward-based sampling at 60^{th} episode. We experiment 5 times with each time with different train and test set that is sampled from the data. The mean train and test success rate of each of the RL method is shown in Figure 5.

The results which are shown in Figure 5 shows that our method improves the test success rate along with its train success rate. In all algorithms, our proposed method performs well without overfitting. All RL algorithms improve their performance as the number of episodes increase, and the test success rate is improving along with train success rate without an apparent lag.

6 Conclusion

We proposed a method to train RL agents for dialogue policy learning without overfitting. The methodology includes a Self-play technique that uses underlying probability distribution for agenda generation and reward-based sampling technique that prioritizes failed dialogues. We have shown that our method performs well compared to baseline as well as that the method can train a policy agent without overfitting. We also have shown that reward-based sampling method performs well in the exploration of agenda space. It also performs better than the random sampling method. We see several possible paths for future work. One of the main is using a better reward function. Also, we like to expand the work so that a full task-oriented conversational agent can be made by modeling the problem as an active learning problem with a better reward function. So an end-to-end task-oriented conversational agent can be made for low resource setting.

References

Antoine Bordes, Y-Lan Boureau, and Jason Weston. 2016. *Learning end-to-end goal-oriented dialog*. arXiv preprint arXiv:1605.07683.

Lu Chen, Xiang Zhou, Cheng Chang, Runzhe Yang, and Kai Yu. 2017. *Agent-aware dropout dqn for safe and efficient on-line dialogue policy learning*. In Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing, pages 2454–2464.

Lucie Daubigney, Matthieu Geist, Senthilkumar Chandramohan, and Olivier Pietquin. 2012. *A comprehensive reinforcement learning framework for dialogue management optimisation*. IEEE Journal of Selected Topcis in Signal Processing, 6.

Floris Den Hengst, Mark Hoogendoorn, Frank Van Harmelen, and Joost Bosman. 2019. *Wizard of wikipedia: Knowledge-powered conversational agents*. arXiv preprint arXiv:1811.01241.

Bhuwan Dhingra, Lihong Li, Xiujun Li, Jianfeng Gao, Yun-Nung Chen, Faisal Ahmed, and Li Deng. 2016. *Towards end-to-end reinforcement learning of dialogue agents for information access*. arXiv preprint arXiv:1609.00777.

Emily Dinan, Stephen Roller, Kurt Shuster, Angela Fan, Michael Auli, and Jason Weston. 2018. *Wizard of wikipedia: Knowledge-powered conversational agents*. arXiv preprint arXiv:1811.01241.

Jianfeng Gao, Michel Galley, and Lihong Li. 2018. *Neural approaches to conversational AI*. The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, pages 1371–1374.

Milica Gasic and Steve Young. 2014. *Gaussian processes for pomdp-based dialogue manager optimization*. Audio, Speech, and Language Processing,IEEE/ACM Transactions on, 22:28–40.

James Henderson, Oliver Lemon, and Kallirroi Georgila. 2008. *Hybrid Reinforcement/Supervised Learning of Dialogue Policies from Fixed Data Sets*. Computational Linguistics, 34:487–511.

Diederik P. Kingma and Jimmy Ba. 2014. *Adam: A method for stochastic optimization*. arXiv preprint arXiv:1412.6980.

Vijay R Konda and John N Tsitsiklis. 2000. *Actorcritic algorithms*. In Advances in neural information processing systems, pages 1008–1014.

Staffan Larsson and David Traum. 2000. *Information state and dialogue management in the trindi dialogue move engine toolkit*. Natural Language Engineering, 6:323–340.

Lihong Li, He He, and Jason Williams. 2015. *Temporal supervised learning for inferring a dialog policy from example conversations*. 2014 IEEE Workshop on Spoken Language Technology, SLT 2014 - Proceedings, pages 312–317.

Lihong Li, Jason D. Williams, and Suhrid Balakrishnan. 2009. *Reinforcement learning for dialog management using least-squares policy iteration and fast feature selection*. Tenth Annual Conference of the International Speech Communication Association.

Xiujun Li, Zachary C Lipton, Bhuwan Dhingra, Lihong Li, Jianfeng Gao, and Yun-Nung Chen. 2016. *A user simulator for task-completion dialogues*. arXiv preprint arXiv:1612.05688.

Zachary Lipton, Xiujun Li, Jianfeng Gao, Lihong Li, Faisal Ahmed, and Li Deng. 2018. *Bbq-networks: Efficient exploration in deep reinforcement learning for task-oriented dialogue systems*. In ThirtySecond AAAI Conference on Artificial Intelligence.

Bing Liu and Ian Lane. 2018. *Adversarial learning of task-oriented neural dialog models*. arXiv preprint arXiv:1805.11762.

Bing Liu, Gokhan Tur, Dilek Hakkani-Tur, Pararth Shah, and Larry Heck. 2018. *Dialogue learning with human teaching and feedback in end-to-end trainable task-oriented dialogue systems*. arXiv preprint arXiv:1804.06512.

Keting Lu, Shiqi Zhang, and Xiaoping Chen. 2019. *Goal-oriented dialogue policy learning from failures*. Proceedings of the AAAI Conference on Artificial Intelligence.

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

Kaixiang Mo, Yu Zhang, Shuangyin Li, Jiajun Li, andQiang Yang. 2018. *Personalizing a dialogue system with transfer reinforcement learning*. Thirty-Second AAAI Conference on Artificial Intelligence.

Diana Perez-Marin and Ismael Pascual-Nieto. 2011. *Conversational agents and natural language interaction: Techniques and effective practices: Techniques and effective practices*. IGI Global.

Joelle Pineau and Sebastian Thrun. 2004. *Spoken Dialogue Management Using Probabilistic Reasoning*. Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL).

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. *Proximal policy optimization algorithms*. arXiv preprint arXiv:1707.06347.

Pararth Shah, Dilek Hakkani-Tur, and Larry Heck. 2016. *Interactive reinforcement learning for taskoriented dialogue management*.

Pararth Shah, Dilek Hakkani-Tur, Bing Liu, and Gokhan Tur. 2018a. *Bootstrapping a neural conversational agent with dialogue self-play, crowdsourcing and online 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.

Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018b. *Building a conversational agent overnight with dialogue self-play*. arXiv preprint arXiv:1801.04871.

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

Pei-Hao Su, Milica Gasic, Nikola Mrksic, Lina RojasBarahona, Stefan Ultes, David Vandyke, TsungHsien Wen, and Steve Young. 2016b. *On-line active reward learning for policy optimisation in spoken dialogue systems*. arXiv preprint arXiv:1605.07669.

Richard S. Sutton and Andrew G Barto. 2018. *Reinforcement learning*. Cambridge, Mass: MIT Press.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. *A networkbased end-to-end trainable task-oriented dialogue system*. arXiv preprint arXiv:1604.04562.

Jason Williams and Steve Young. 2007. *Partially observable Markov decision processes for spoken dialog systems*. Computer Speech & Language,21:393–422.

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