Adversarial Learning of Task-Oriented Neural Dialog Models(2018)

Review

1. Introduction

a. Like apple siri와 같은 것들이 많이 생김

b. 문제점

i. Task Oriented Dialogue Model based RL 에서 reward를 사람이 직접 평가하는게 일관적이지 않고, data query 또한 많기 때문에 어려운 문제다.

1. 여러 시도가 있었지만 결국 사람이 계속 Dialogue에 대한 Scoring 해야 함

a. Ex) Gaussian process classification

c. Approach

i. 이런 문제점을 해결하기 위해서 Adversarial Learning 으로 reward signal를 estimate 하겠다

1. Reward는 user와 agent가 interactive하게 대화를 하면서 Discriminator가 Task에 관한 질문이 연관성 정도에 따라서 reward 평가

2. 결국 Dialogue의 reward를 주는 것과 같은 형상, Task관련 대화일수록 높은Reward를 준다. 즉 agent는 sum of future reward가 maximize하게끔 dialogue를 생성

3. Policy gradient based RL 사용

4. 그리고 제안된 method가 task oriented한 환경에서 restaurant위치를 얼마나 더 잘 찾아 주는지 baseline모델과 비교 해봄(결과 좋음)

ii. 추가적으로 online adversarial dialogue learning 의 covariate shift problem에 대해서 논하고, 이것을 사용자의 부분적인 feedback으로 접근하는 것을 다룬다

2. Related Work

a. Task-Oriented Dialogue Learning

i. 2017 -> end to end task oriented dialogue

1. Memory network
2. Supervised learning
3. Supervised and RL(hybrid)

b. Dialog Reward Modeling

i. CNN

ii. Gaussian process

iii. 공통점:

1. 사람들이 Score Labeling 을 해야함

c. Direct estimate

i. IRL

1. 문제점 : 학습이 expensive
2. 복잡한 Dialogue는 더 expensive하다

d. Adversarial Network

i. Text generation 에서 GAN을 처음 도입, 그 전까지는 보통 Image에 많은 연구가 있었음, 그리고 Neural Machine Translation 분야 적용

ii. Main reward function으로 Adversarial loss 사용, 하지만 사용자 Goal에 prior Knowledge 필요

iii. 이 논문에서 제시하는 것은 Task Policy를 최적화 하기위한 reward signal의 source로써 Adversarial reward 사용

3. Adversarial Learning of Task-Oriented Dialogs

a. Basic

i. User <- (conversation) -> Agent

1. User input dialogue
2. Agent select best action
3. Action (Slot-value predictions)

ii. Discriminator에서 Agent가 Task compilation을 성공할 수 있는 확률을 출력(User대화와 비슷하다면 확률이 높음) (0~1)

iii. Reward function(Discriminator) and RL 기법(agent) 순차적 학습

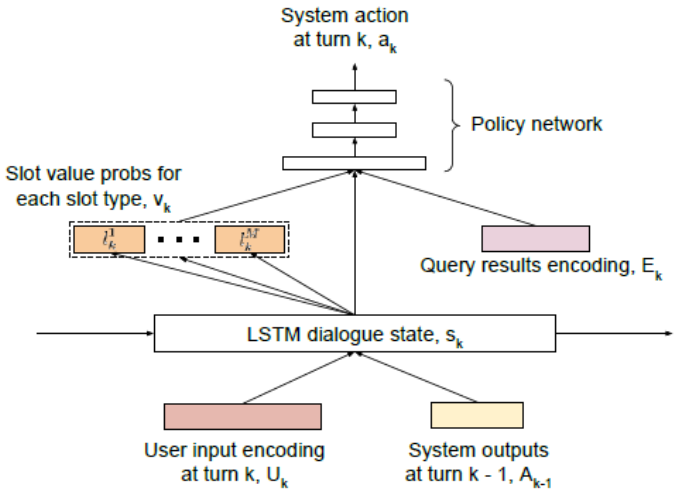

Figure 1: Design of the task-oriented neural dialog agent.

iv.

b. 3.1 Neural Dialog Agent

i. 이전 Action Ak-1, 현재 유저의 Input Dialogue Uk

ii. $$s_k = \text{LSTM}_\text{G}(s_{k-1},\ [U_k,\ A_{k-1}]) \qquad (1)$$

iii. Belief Tracking

1. State S_k에서 추적된 Slot type m에서 Goal일 확률 분포 P(lm_k)를 계속 업데이트 한다.

2. $$P(l_k^m \mid \mathbf{U}_{\leq k},\ \mathbf{A}_{<k}) = \text{SlotDist}_m(s_k) \qquad (2)$$

3. SlotDistm is a single hidden layer MLP with softmax activation over slot type m

iv. Dialog Policy

1. Policy network input

a. (1) dialogue state S_k

b. (2) probability distribution of estimated user goal slot value V_k

c. (3) information retrieved from external source E_k(query result)

2. Probability distribution over the next system action

3. $$P(a_k \mid U_{\leq k},\ A_{<k},\ E_{\leq k}) = \text{PolicyNet}(s_k, v_k, E_k)$$

4. Policy net is single hidden Layer MLP with softmax activation over all system action

c. 3.2 Dialog Reward Estimator

i. Binary classifier 사용 Dialogue 가 task를 성공했는지, 안했는지

ii. Logistic function 사용 (0~1)

iii. Input

1. encoding of the user input U_k,
2. encoding of the query result summary E_k
3. encoding of agent output A_k

iv. Final dialog representation d for the binary classifier

1. BiLSTM-Last: F, Backward last LSTM state
2. BiLSTM-max: max pooling(each dimension 의 maximum value)
3. BiLSTM-avg: 평균
4. BiLSTM-attn: attention mechanism 사용

a. $$d = \sum_{k=1}^{K} \alpha_k h_k \qquad (4)$$

b. $$\alpha_k = \frac{\exp(e_k)}{\sum_{t=1}^{K} \exp(e_t)}, \quad e_k = g(h_k) \qquad (5)$$

c. Discriminator (logistic function)

i.

$$D(d) = \sigma(W_o d + b_o) \tag{6}$$

d. 3.3 Adversarial Model Training

i. Policy Gradient

We let τ denote a state-action sequence $s_0, u_0, \ldots, s_H, u_H$. We overload notation: $R(\tau) = \sum_{t=0}^{H} R(s_t, u_t)$.

$$U(\theta) = \mathrm{E}[\sum_{t=0}^{H} R(s_t, u_t); \pi_\theta] = \sum_{\tau} P(\tau; \theta) R(\tau)$$

In our new notation, our goal is to find θ:

$$\max_\theta U(\theta) = \max_\theta \sum_{\tau} P(\tau; \theta) R(\tau)$$

1.

2. Likelihood Ratio Policy Gradient

$$U(\theta) = \sum_{\tau} P(\tau; \theta) R(\tau)$$

Taking the gradient w.r.t. θ gives

$$\begin{aligned}
\nabla_\theta U(\theta) &= \nabla_\theta \sum_{\tau} P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} \nabla_\theta P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} \frac{P(\tau; \theta)}{P(\tau; \theta)} \nabla_\theta P(\tau; \theta) R(\tau) \\
&= \sum_{\tau} P(\tau; \theta) \frac{\nabla_\theta P(\tau; \theta)}{P(\tau; \theta)} R(\tau) \\
&= \sum_{\tau} P(\tau; \theta) \nabla_\theta \log P(\tau; \theta) R(\tau)
\end{aligned}$$

Approximate with the empirical estimate for m sample paths under policy π_θ:

a.

$$\nabla_\theta U(\theta) \approx \hat{g} = \frac{1}{m} \sum_{i=1}^{m} \nabla_\theta \log P(\tau^{(i)}; \theta) R(\tau^{(i)})$$

b. Object Function

i. Agent

$$\begin{aligned}
\nabla_{\theta_G} J_k(\theta_G) &= \nabla_{\theta_G} \mathbb{E}_{\theta_G} [R_k] \\
&= \sum_{a_k \in \mathcal{A}} G(a_k|\cdot) \nabla_{\theta_G} \log G(a_k|\cdot) R_k \\
&= \mathbb{E}_{\theta_G} [\nabla_{\theta_G} \log G(a_k|\cdot) R_k]
\end{aligned} \tag{7}$$

1.

ii. Discriminator Object function

1. GAN 참고

$$\nabla_{\theta_D}\Big[\mathbb{E}_{d\sim\theta_{demo}}\left[log(D(d))\right] + \mathbb{E}_{d\sim\theta_G}\left[log(1-D(d))\right]\Big] \tag{8}$$

2.

c. 전체 알고리즘

i.

Algorithm 1 Adversarial Learning for Task-Oriented Dialog

Required: dialog corpus S_{demo}, user simulator U, generator G, discriminator D
Pretrain a dialog agent (i.e. the generator) G on dialog corpora S_{demo} with MLE
Simulate dialogs S_{simu} between U and G
Sample successful dialogs $S_{(+)}$ and random dialogs $S_{(-)}$ from $\{S_{demo}, S_{simu}\}$
Pretrain a reward function (i.e. the discriminator) D with $S_{(+)}$ and $S_{(-)}$ ▷ eq 8
for number of training iterations **do**
    for G-steps **do**
        Simulate dialogs S_b between U and G
        Compute reward r for each dialog in S_b with D ▷ eq 6
        Update G with reward r ▷ eq 7
    end for
    for D-steps **do**
        Sample dialogs $S_{(b+)}$ from $S_{(+)}$
        Update D with $S_{(b+)}$ and S_b (with S_b as negative examples) ▷ eq 8
    end for
end for

4. Experiments

a. 4.1 Dataset

i. Second Dialog State Tracking Challenge (DSTC2)

ii. restaurant search domain

iii. action 선택

1. ex(confirm(food= Italian) -> action restaurant 찾아야하니깐 => slot "Italian"

# of train/dev/test dialogs	1612/506/ 1117
# of dialog turns in average	7.88
# of slot value options	
Area	5
Food	91
Price range	3

Table 1: Statistics of DSTC2 dataset.

iv.

b. 4.2 training setting

   i. Dialog Turn 20 maximum

   ii. Mini-batch of 25 samples

c. 4.3 Result and Analysis

   i. 4.3.1 Comparison to Other Reward Type

      1. (1) Adversarial reward , (2) designed reward, (3) Oracle reward, success rate 비교

         a. (designed reward 점수 계산 방식)

            i. Dialog의 끝에 각 Informal Slot을 정확히 예측 했을 때 slot당 +1(Informal slot은 모두 +1)

            ii. 만약 모든 Informal Slot이 정확히 추적 되었을 때, 각 requestable slot당 +1(request slot 만 +1)

         b. (Oracle reward 성공하면 +1, 실패 0)

      2. 실험결과

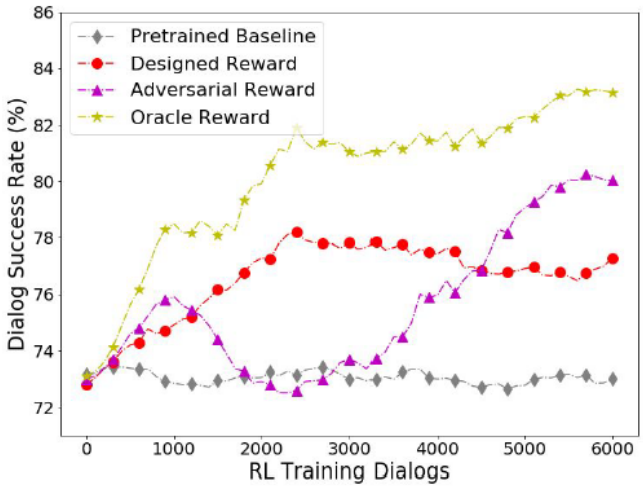

Figure 3: RL policy optimization performance comparing with adversarial reward, designed reward, and oracle reward.

a.

b. 결과를 보변 Designed reward Function은 Domain knowledge가 Dialog Success에 큰 영향을 미치지는 않는다는 것을 볼 수 있다

c. 이 Adversarial reward(BiLSTM max)가 Designed reward 보다 성능이 좋고, 대신에 학습하는 과정에서 variance가 크고, instability 하다

ii. 4.3.2 impact of Discriminator

1. Max-Pooling 굿, Attention-Pooling은 데이터의 한계로 성능이 좋지 않음

Model	Prediction Accuracy	Success Prob.	Fail Prob.
BiLSTM-last	0.674	0.580	0.275
BiLSTM-max	**0.706**	**0.588**	0.272
BiLSTM-avg	0.688	0.561	**0.268**
BiLSTM-attn	0.652	0.541	0.285

Table 2: Performance of different discriminator model design, on prediction accuracy and probabilities assigned to successful and failed dialogs.

2.

iii. 4.3.3 Impact of Annotated Dialogs for Discriminator Training

1. Model을 training 하기 위해서 dialog의 Annotating이 필요

2. Annotating된 dialog 의 sample수가 많으면 성능이 향상된다

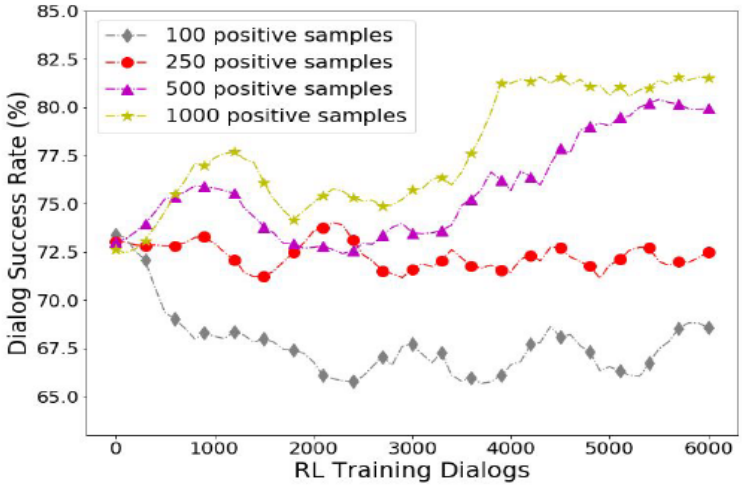

Figure 4: Impact of discriminator training sample size on RL dialog learning performance.

3.

iv. 4.3.4Partial Access to User Feedback

1. RL based interactive adversarial Learning -> covariate shift 문제가 있음
    a. Covariate shift-> input data distribution 과 test data distribution 이 다름
    b. 결국 학습 환경의 및 테스트 환경의 차이가 학습의 결과를 좋지 않게 한다는 말
2. 이 논문의 경우 pretraining 한 이후 다시 RL Training 하기 때문에 Covariate shift가 생김 -> Discriminator에서 잘못된 Reward 반환
3. 그래서 DAgger이란 dialog adversarial learning 에서 style imitation learning method 를 사용
4. User interactive learning 중에 사용자 conversation의 quality 를 feedback signal로 보낸다. 이 Feedback이 좋은 Dialog를 학습 샘플에 추가함으로써 이런 문제를 해결하고자 함

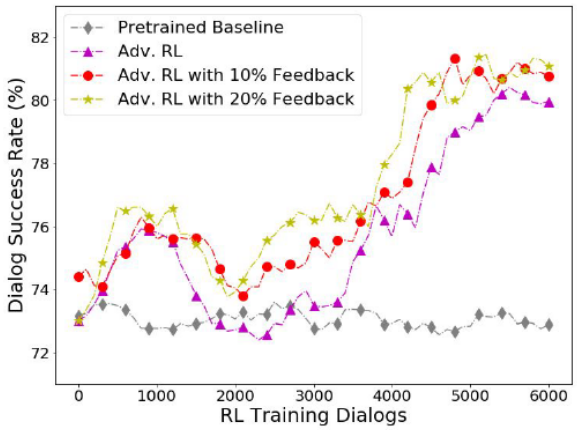

Figure 5: Addressing covariate shift in online adversarial dialog learning with partial access to user feedback.

5.

5. Conclusion

a. Reward를 task completion 할 수 있는지를 0~1사이로 맵핑, 이전 논문인 "Discriminative Deep Dyna-Q- Robust Planning for Dialogue Policy Learning"은 dialogue quality를 평가하는데 쓰였는데 같은 관점으로 봐도 무방하다고 생각됨

b. 하지만, Policy gradient를 썼다는 점이 차이가 있음, 이전 논문은 DQN

c. Policy Gradient의 기본 문제인 Variance 어떻게 잡았는지 나오지 않음 : TRPO, AC, A3C, DDPG, PPO등이 있는데 그걸 안쓴건 이상함

d. 요즘 대세는 Discriminator를 이용한 평가 방법을 바로 구해서 학습에 사용한다는 포인트

e. 하지만 역시나 Success rate 만 보여주는 것으로 보아 대화 자체는 사람이 알아먹기 힘든 듯