Safe Evaluation of Dialogue Management

Layla El Asri
Microsoft Maluuba
Montreal
laelasri@microsoft.com

Adam Trischler
Microsoft Maluuba
Montreal
adtrisch@microsoft.com

Abstract

This extended abstract presents preliminary work on safe evaluation of the policy of a dialogue manager. The dialogue manager is trained through reinforcement learning with a user simulator. Safe evaluation takes into account the uncertainty over the user simulator’s behaviour during training. We show that including this uncertainty into the dialogue manager’s reward function leads to more accurate evaluation and more efficient exploration.

1 Introduction

In this work we investigate safe evaluation of fully data-driven dialogue systems trained on human-human dialogues collected in a Wizard-of-Oz setting (WOz, El Asri et al., 2017). From WOz corpora, dialogue systems may learn to perform natural language understanding, dialogue management, and natural language generation [Lemon and Pietquin, 2007] (possibly in an end-to-end architecture [Wen et al., 2016, Bordes and Weston, 2017]).

A dialogue manager is the module of a dialogue system that decides, at each turn, what to say to the user. This is often treated as a planning task amenable to reinforcement learning (RL) approaches [Singh et al., 2002, Asadi and Williams, 2016]. Within the RL framework, the dialogue manager’s behaviour is called a policy. A policy associates a state to an action or a probability distribution over possible actions. The state contains information about the dialogue such as the estimated user goal and the dialogue history. In this paper, we consider actions at the intention level: an action is represented by a dialogue act such as inform, request, or deny. The objective of the dialogue manager is to learn an optimal policy which maximizes, at each state, the expected sum of rewards that will be received after visiting this state and by following the policy. Rewards may be based on user satisfaction and task completion, for example. To learn an optimal policy from a corpus of dialogues, two options are possible: train the dialogue manager through a user simulator learned on the corpus [Fatemi et al., 2016] or train the dialogue manager directly on the data [Pietquin et al., 2011].

Training with a user simulator presents a symmetric problem: if either the simulator or the dialogue manager strays from the corpus trajectories, the other will also likely stray [Henderson et al., 2008]. Therefore, simulated dialogues might be very different from what appears in reality. The same problem occurs if, after training directly on the data, a user simulator is used for evaluation purposes: if the system strays away, the evaluation will be biased by the user simulator’s ability to output a realistic answer.

Another method to evaluate a dialogue manager is to perform off-policy evaluation [Thomas and Brunskill, 2016, Jiang and Li, 2016] which consists of evaluating a policy based on data collected with another policy. A problem there is that with limited data, off-policy evaluation tends to be biased or have high variance. Moreover, even with more data, we learn from human behaviour and humans do not explore the entire action space. Especially in the case of dialogue, we only observe certain actions for given states. Finally, for datasets collected with paid workers or volunteers, certain realistic behaviours will not be observed: for instance, users giving up on the dialogue.

In this paper we propose a new evaluation scheme that relies on user simulation. We pro-
pose to modify the dialogue system’s reward signals by adding rewards that encourage the system to stay within regions of the state space where the user simulator’s behaviour can be trusted. We perform preliminary experiments in a simulated setting and show that the evaluation of the dialogue manager is more accurate given knowledge of user-simulator uncertainty. We also show that the dialogue system’s exploration is more efficient and leads to a significantly better policy compared to the case when the user simulator’s uncertainty does not affect the rewards.

2 Method

To evaluate dialogue management, we propose learning a simulator of user behaviour. The simulator takes as input a system action $a$, updates its internal state representation $\tilde{x}$, and outputs an action $\tilde{a}$. Our goal is twofold: we want to maximize a reward function $R$ that is based, for instance, on task completion, and we also want to prevent the system from visiting unknown simulator states. To accomplish this, we propose adding intrinsic rewards to $R$. For each system state-action-state triple $(x, a, x')$, we add $\sigma(\tilde{x}) - \sigma(\tilde{x}')$ to the reward $R(x, a, x')$, where $\sigma(\tilde{x})$ is a measure of the user simulator’s uncertainty for state $\tilde{x}$. The intrinsic reward encourages the system to stay within known parts of the user simulator state space or to recover when it strays away.

In the next section, we suppose that the uncertainty is known and analyse the dialogue manager’s behaviour under this modified reward function.

3 Experiments

We use the user simulator and dialogue manager proposed by Li et al. [2016, 2017]\(^2\). The dialogues generated by these two agents are set in the movie domain: the user wants to book tickets corresponding to a few constraints (movie name, theater, time, etc.) and the agent retrieves availabilities from a database and books the tickets. At the beginning of each dialogue, a user goal is randomly drawn (e.g., movie name = Zootopia, city = Seattle, etc.). The user simulator’s behaviour is based on this goal as well as a set of rules. Examples of rules are: keep providing information on the user goal until a suitable movie is found, tell the system if it did not understand something, and so on. The dialogue system is composed of two neural networks: one that takes the user’s text utterance as input and outputs a semantic frame (language understanding) and another that takes the semantic frame and outputs an action (dialogue management). We train the dialogue system by making it interact with the user simulator. We use DQN [Mnih et al., 2015] to train the dialogue management network, with the same parameters as Li et al. [2017].

To model the case where the agent is trained on simulated data and evaluated with real users, we modify the user simulator during test time. At training time, we use the simulator as proposed by Li et al. [2016] which, when the system tries to book the wrong tickets, tells the system that it is incorrect (deny dialogue act). At test time, the user simulator exhibits this same behaviour in 80% of the wrong-booking cases, but in 20% of these cases it simply closes the conversation (closing dialogue act). The goal of this scheme is to model the following process: a user simulator was trained on human-human dialogues but some states are not well known (here, the state when the system makes the incorrect booking) and human behaviour is different from the simulator’s in these states.

We first train the dialogue manager with the reward function proposed by Li et al. [2017], i.e., $R(x, a, x') = -1$ if $x'$ is not a closing state, -20 if $x'$ is a closing state and the dialogue is not successful, or +40 if $x'$ is a closing state and the dialogue is successful.

We then train a dialogue manager with $\hat{R}(x, a, x') = R(x, a, x') + \sigma(\tilde{x}) - \sigma(\tilde{x}')$ where $\sigma(\tilde{x})$ is equal to 0 for each simulator state $\tilde{x}$ except for the state where the user can deny or close, for which $\sigma(\tilde{x})$ is 0.2 by definition.

We report averages over 10 runs of 2000 test dialogues in Table 1. Results show that the dialogue manager trained with $\hat{R}$ has a more accurate estimation of its performance, since the mean difference between training rewards and test rewards is 0.9 vs. 1.2 for the dialogue manager trained with $R$. The standard deviation of the average rewards is also significantly lower with $\hat{R}$.

In addition, the success rate of the agent trained with $\hat{R}$ is significantly higher\(^3\). As expected, the

\(^2\)The code is available at https://github.com/MiuLab/TC-Bot.

\(^3\)Student’s t-test with p-value of $9.3 \times 10^{-15}$. 
Table 1: Evaluation with and without taking into account user simulator uncertainty in training.

<table>
<thead>
<tr>
<th>Average Reward</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>55.07 ± 1.05</td>
<td>53.96 ± 1.24</td>
</tr>
<tr>
<td>$\hat{R}$</td>
<td>64.62 ± 0.56</td>
<td>63.72 ± 0.71</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Success rate (%)</th>
<th>Training</th>
<th>Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R$</td>
<td>85.2 ± 0.08</td>
<td>84.1 ± 0.09</td>
</tr>
<tr>
<td>$\hat{R}$</td>
<td>91.9 ± 0.04</td>
<td>91.1 ± 0.05*</td>
</tr>
</tbody>
</table>

dialogue manager learned to avoid the uncertain user simulator state, which led to fewer failed dialogues. The difference in success rate can also be explained by another major difference in the policies: the agent trained with $\hat{R}$ almost never chooses the action that tells the user that no result was found in the database (1 occurrence vs. 91 occurrences for the agent trained with $R$, on one run of 2000 test episodes). This action often leads to the user closing the dialogue (58 occurrences on the same run for the agent trained with $R$). An explanation of this difference is that by trying to avoid the state where the user simulator was uncertain, the agent explored other actions and learned a tradeoff between visiting uncertain states and receiving highly negative rewards.

Figure 3 shows the success rates of both policies for different closing probabilities, ranging from 20% to 100%. The dialogue manager trained with $\hat{R}$ is trained with the same rewards as before, meaning that it estimates the uncertainty at 0.2. This setting is to simulate the case when the uncertainty estimate is lower than the real uncertainty. The $\hat{R}$-agent’s success rate drops by 1.5% whereas the baseline agent’s rate drops by 4.4%. Even if the uncertainty is underestimated, as long as the agent correctly identifies which states are safer than others, the evaluation of its policy is still significantly more reliable.

This approach relies on the estimation of the uncertainty of the user simulator. In this simple case, only one state is uncertain, and that uncertainty is known, while the other states are completely certain. In reality, many states will be uncertain and uncertainty scores will range continuously from 0 to 1. Future work will consist in applying these intrinsic reward signals to cases where the uncertainty from the user simulator must be estimated, e.g., through variational inference.

4 Conclusion

In this work we modified reward signals to account for the uncertainty in user simulation when training and evaluating a dialogue manager with RL. We showed that if the uncertainty is perfectly known, the agent learns to find a good tradeoff between visiting unknown states and maximizing its reward. Future work will consist in proposing a method to estimate simulation uncertainty and confirm these results in this more realistic setting.
References


