Self-improving Chatbots based on Reinforcement Learning

Self-improving Chatbots based on Reinforcement Learning

Elena Ricciardelli, Debmalya Biswas
AI Center of Excellence
Philip Morris International
Lausanne, Switzerland
`firstname.lastname@pmi.com`

Abstract

We present a Reinforcement Learning (RL) model for self-improving chatbots, specifically targeting FAQ-type chatbots. The model is not aimed at building a dialog system from scratch, but to leverage data from user conversations to improve chatbot performance. At the core of our approach is a score model, which is trained to score chatbot utterance-response tuples based on user feedback. The scores predicted by this model are used as rewards for the RL agent. Policy learning takes place offline, thanks to an user simulator which is fed with utterances from the FAQ-database. Policy learning is implemented using a Deep Q-Network (DQN) agent with epsilon-greedy exploration, which is tailored to effectively include fallback answers for out-of-scope questions.

The potential of our approach is shown on a small case extracted from an enterprise chatbot. It shows an increase in performance from an initial 50% success rate to 75% in 20-30 training epochs.

Keywords: Reinforcement Learning, Chatbots, NLP

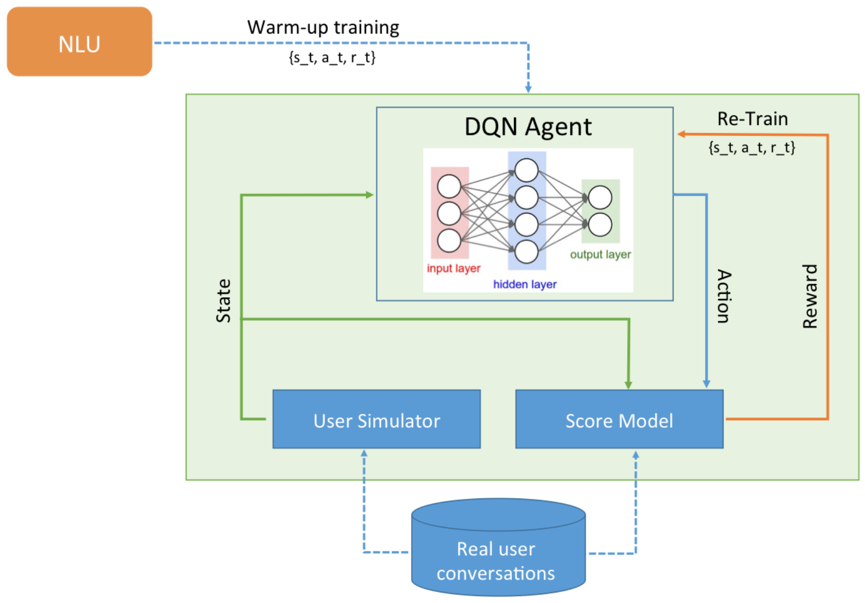

Figure 1: Architecture of the RL model used in this work. The DQN agent is initially trained offline in a warm-up phase on the NLU. The score model is also trained offline with the data from real user conversations. In the RL loop, the user state (user utterance) is provided by the user simulator, the action (chatbot response) is provided by the DQN agent and the reward is provided by the score model. Each tuple (s_t, a_t, r_t) feeds the experience replay buffer, which is used to re-train the DQN after $n_{episodes}$ episodes, which is a tunable parameter.

1 Introduction

The majority of dialog agents in an enterprise setting are domain specific, consisting of a Natural Language Understanding (NLU) unit trained to recognize the user's goal in a supervised manner. However, collecting a good training set for a production system is a time-consuming and cumbersome process. Chatbots covering a wide range of intents often face poor performance due to intent overlap and confusion. Furthermore, it is difficult to autonomously retrain a chatbot taking into account the user feedback from live usage or testing phase. Self-improving chatbots are challenging to achieve, primarily because of the difficulty in choosing and prioritizing metrics for chatbot performance evaluation. Ideally, one wants a dialog agent to be capable to learn from the user's experience and improve autonomously.

In this work, we present a reinforcement learning approach for self-improving chatbots, specifically targeting FAQ-type chatbots. The core of such chatbots is an intent recognition NLU, which is trained with hard-coded examples of question variations. When no intent is matched with a confidence level above 30%, the chatbot returns a fallback answer. For all others, the NLU engine returns the corresponding confidence level along with the response.

Several research papers [2,3,7,8] have shown the effectiveness of a RL approach in developing dialog systems. Critical to this approach is the choice of a good reward model. A typical reward model is the implementation of a penalty term for each dialog turn. However, such rewards only apply to task completion chatbots where the purpose of the agent is to satisfy user's request in the shortest time, but it is not suitable for FAQ-type chatbots where the chatbot is expected to provide a good answer in one turn. The user's feedback can also be used as a reward model in an online reinforcement learning. However, applying RL on live conversations can be challenging and it may incur a significant cost in case of RL failure. A better approach for deployed systems is to perform the RL training offline and then update the NLU policy once satisfactory levels of performance have been reached.

2 Reinforcement Learning Model

The RL model architecture is illustrated in Figure 1. The various components of the model are: the NLU unit, which is used to initially train the RL agent in a warm-up phase; the user simulator, which randomly extracts the user utterances from the database of user experiences; the score model trained on the user's conversation with feedback and the RL agent based on a Deep Q-Network (DQN) network.

2.1 Dialog System

We apply the reinforcement learning approach on a FAQ-type chatbot. At the core of the chatbot, there is an intent[1] recognition NLU, which is trained with hard-coded examples of question variations. For this work, we have chosen the open-source NLU from Rasa[2], using the TensorFlow pipeline. However the RL approach is independent from the NLU chosen and for systems in production it can easily be extended to NLU engines such as IBM Watson or Amazon LEX.

2.2 Real User Conversations

We used user feedback obtained during the development an actual internal chatbot for our work. The scope of the chatbot was to answer employee queries related to office building facilities, HR policies and benefits, etc. All the 10 users participating in the test phase were informed that their feedback would be used to improve the chatbot performance. The testers provided a (binary) feedback after each conversation turn, thus rating the utterance-response tuples. The historical data thus contains quadruples of the following format: (utterance, response, NLU confidence level and feedback). By removing non valid conversations (i.e. those lacking or with invalid feedback) we end up with 550 user conversations, triggering about 120 intents. Although we have tested the score model on all the conversations, the RL model has been applied only on a subsample of 155 conversations, triggering the top 5 intents. On this subset, the user's satisfaction is 50%.

Table 1: Example of conversation from the database, as well as the score provided by the model and by the user

Utterance	Response	Model score	User feedback
What is Interact?	I can help you mainly with GS service-related questions on topics like transportation (your mobility plan/benefit car), business travel, hospitality and events, gym, mailroom etc.	0.30	0
where can I find my salary payment	I was getting bored, but now you are here.	0.26	0
Have a nice day	Bye bye! Thanks for the chat!	0.81	1

2.3 Reward Function: the Score Model

Evaluating chatbot performance is a long-standing issue in computational linguistic. Automatic metrics borrowed from machine translations (e.g. [6]) do not perform well on short sentences (e.g. [4]), such as the chatbot utterance-response tuples. On the other hand, human rating of chatbots is by now the *de-facto* standard to evaluate the success of a chatbot, although those ratings are often difficult and expensive to gather.

To evaluate the correctness of chatbot responses, we propose a new approach which makes use of the user conversation logs, gathered during the development and testing phases of the chatbot. Each user had been asked to provide a binary feedback (positive/negative) at each chatbot turn. In order to use the user feedback in an offline reinforcement learning, we have developed a score model, capable of modeling the binary feedback for unseen utterance-response tuples. In a supervised fashion, the score model learns how to project the vector representations of utterance and response in a linearly transformed space, such that similar vector representations give high score. As for the vector representation of sentences, we compute sentence embedding through the universal sentence encoder [1], available through TensorFlow Hub [3]. To train the model, the optimization is done on a squared error (between model prediction and human feedback) loss with L2 regulation. To evaluate the model, the predicted scores are then converted into a binary outcome and compared with the targets (the user feedbacks). For those couples of utterances having a recognized intent with both a positive feedback and a NLU confidence level close to 1, we perform data augmentation, assigning low scores to the combination of utterance and fallback intent.

A similar approach for chatbot evaluation has been suggested by [4]. The authors model the scores by using a labelled set of conversations, that also include model and human-generated responses, collected through crowdsourcing. Our approach differs from the above authors in that it just requires a labelled set of utterance-response tuples, which are relatively straightforward to gather during the chatbot development and user testing phases.

[1]An intent is defined as the user's intention, which is formulated through the utterance

[2]https://rasa.com/

[3]https://www.tensorflow.org/hub

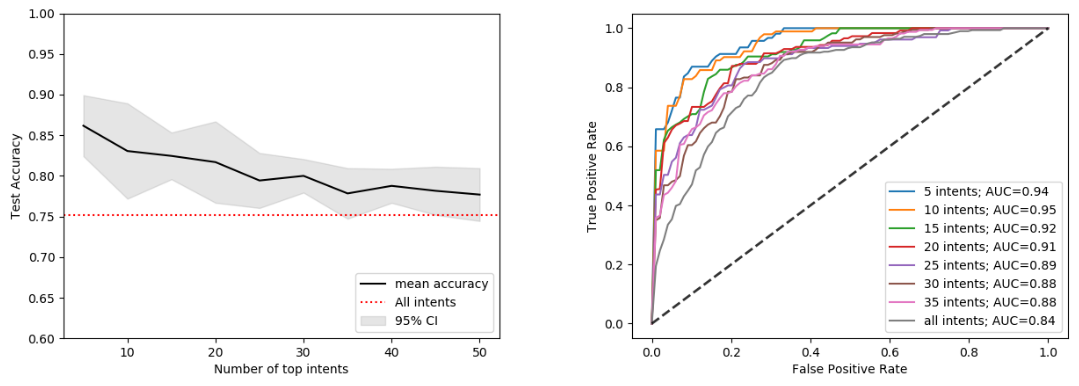

Figure 2: Performances of the score model. Left-hand panel: cross-validated test set accuracy with 95% confidence interval for different sub-samples having different number of intents. The horizontal red line indicates the performances for the entire sample. Right-hand panel: ROC curves for the different subsamples.

2.4 Policy Learning with DQN

To learn the policy, the RL agent uses a Q-learning algorithm with DQN architecture [5]. In DQN, a neural network is trained to approximate the state-action function $Q(s_t|a_t, \theta)$, which represents the quality of an action a_t provided a state s_t, and θ are the trainable parameters. As for the DQN network, we have followed the approach proposed by [3], using a fully-connected network, fed by an experience replay pool buffer, that contains the one-hot representation of utterance and response and the corresponding reward. An one-hot representation is possible in this case as we have a finite possible values for utterances (given by the number of real users's question in the logs) and responses (equal to the number of intents used on out test-case, 5). In a warm-up phase, the DQN is trained on the NLU, using as a reward the NLU confidence level. The DQN training set is augmented whenever a state-action pair has a confidence above a threshold, by assigning zero weight to the given state and all the other available actions. Thus, at the starting of the RL training, the agent performs similar to the NLU unit.

During RL training, we use an ϵ-greedy exploration, where random actions are explored according to a probability ϵ. We use a time-varying ϵ which facilitates the exploration at the beginning of the training with $\epsilon_{t_0} = 0.2$ and $\epsilon_t = 0.05$ during the last epoch. To speed-up the learning when picking random actions, we also force higher probability to get a "No intent detected", as several questions are actually out of the chatbot scope, but they are erroneously matched to a wrong intent by the NLU. During an epoch we simulate a batch of conversations of size $n_{episodes}$ (ranging from 10 to 30 in our experiments) and fill an experience replay buffer with the tuple (s_t, a_t, r_t). The buffer has fixed size and it is flushed the first time when the agent performance increases above a specified threshold. In those episodes where the state-action tuple gets a reward greater than 50%, we perform data augmentation by assigning zero reward to the assignment of any other action to the current state.

3 Model Evaluation

3.1 Score Model Evaluation

To evaluate the model, we select subsets of conversations, triggering the top N intents, with N between 5 and 50. The results of the score model are summarized in Figure 2, showing the cross-validated (5-*fold* CV) accuracy on the test set and the ROC curve as a function of the number of intents. For the whole sample of conversations, we obtain a cross-validated accuracy of 75% and an AUC of 0.84. However, by selecting only those conversations triggering the top 5 intents, thus including more examples per intent, we obtain an accuracy of 86% and an AUC of 0.94. For the RL model evaluation, we have focussed on the 5 intents subsets; which ensures that we have the most reliable rewards.

3.2 Reinforcement Learning Model Evaluation

The learning curve for the RL training is shown in Figure 3. In the left-hand panel, we compare the RL training with the reward model with a test done with a direct reward (in interactive way), showing that the score model is giving similar performances to the reference case, where the reward is known. Large fluctuations in the average score are due to a limited batch size ($n_{episodes} = 10$) and a relatively large ϵ. We also show the success rate on a test set of 20 conversations, extracted from the full sample, where a "golden response" is manually provided for all the utterances. The agent success rate increases from an initial $\sim 50\%$ to 75% in only $\sim$30 epochs, showing the potential of this approach. In the right-hand panel, we show the results using $n_{episodes} = 30$, showing similar performances but with a smoother learning curve.

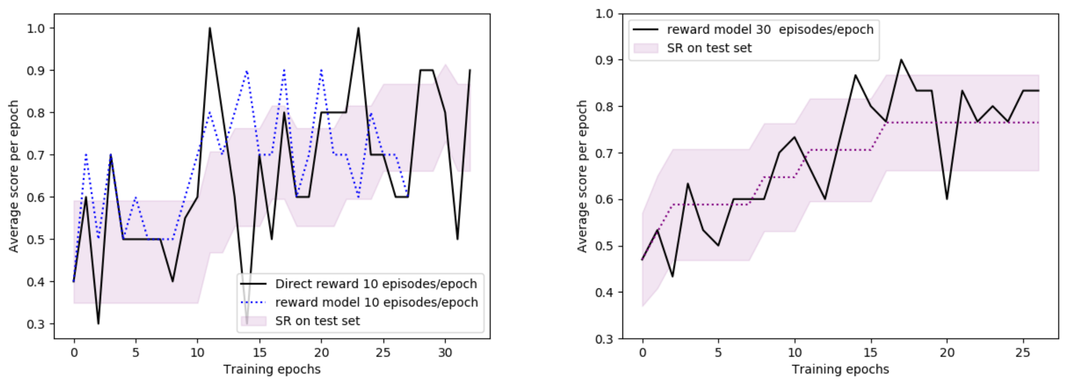

Figure 3: Learning curves showing the DQN agent's average score (continuous black line) per training epoch and success rate (purple shaded area) based on a labelled test set of 20 conversations. Left-hand panel: learning curves for direct RL with interactive reward (black line) and the reward model (blue dotted line), using 10 episodes per epoch. Right-hand panel: learning curves for the model reward, using 30 episodes per epoch.

4 Conclusions

In this work, we have shown the potential of a reinforcement learning approach in improving the performance of FAQ-type chatbots, based on the feedback from a user testing phase. To achieve this, we have developed a score model, which is able to predict the user's satisfaction on utterance-response tuples, and implemented a DQN reinforcement model, using the score model predictions as rewards. We have evaluated the model on a small, but real, test case, demonstrating promising results. Further training on more epochs and including more data, as well as extensive tests on the model hyper-parameters are in progress. The value of our approach is in providing a practical tool to improve large-scale chatbots (with a large set of diverse intents), in an automated fashion based on user feedback.

Finally, we notice that although the reinforcement learning model presented in this work is suitable for FAQ-type chatbots, it can be generalised to include the sequential nature of conversation by incorporating a more complex score model.

References

[1] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St. John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, Yun-Hsuan Sung, Brian Strope, and Ray Kurzweil. Universal sentence encoder. *CoRR*, abs/1803.11175, 2018.

[2] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[3] Xiujun Li, Yun-Nung Chen, Jianfeng Gao, and Asli Celikyilmaz. End-to-end task-completion neural dialogue systems. In *8th International Joint Conference on Natural Language Processing*, 2017.

[4] Ryan Lowe, Michael Noseworthy, Iulian Vlad Serban, Nicolas Angelard-Gontier, Yoshua Bengio, and Joelle Pineau. Towards an automatic turing test: Learning to evaluate dialogue responses. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1116–1126. Association for Computational Linguistics, 2017.

[5] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518:529 EP –, 02 2015.

[6] 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*, ACL '02, pages 311–318, 2002.

[7] Baolin Peng, Xiujun Li, Jianfeng Gao, Jingjing Liu, and Kam-Fai Wong. Deep dyna-q: Integrating planning for task-completion dialogue policy learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2182–2192. Association for Computational Linguistics, 2018.

[8] Iulian Vlad Serban, Chinnadhurai Sankar, Mathieu Germain, Saizheng Zhang, Zhouhan Lin, Sandeep Subramanian, Taesup Kim, Michael Pieper, Sarath Chandar, Nan Rosemary Ke, Sai Mudumba, Alexandre de Brébisson, Jose Sotelo, Dendi Suhubdy, Vincent Michalski, Alexandre Nguyen, Joelle Pineau, and Yoshua Bengio. A deep reinforcement learning chatbot. *CoRR*, abs/1709.02349, 2017.