ConvoBot: A Conversational Bot via Deep Q-Learning and Query Simulation

Vidush Mukund
Department of Electrical Engineering
Stanford University
Palo Alto, CA 94305
vmukund@stanford.edu

Brian Wilcox
Department of Electrical Engineering
Stanford University
Palo Alto, CA 94305
wilcoxee@stanford.edu

Abstract

This paper looks at combining advances in deep reinforcement learning with natural language processing techniques to train a conversational chatbot. The primary aims are to build a deep Q network (DQN) chatbot model, a novel architecture for user simulation, and finally integrating the two components into a reinforcement learning (RL) pipeline for training the DQN chatbot by conversing with the user simulator in a database query setting. While prior work has defined rule-based user simulators as well as simplistic chatbot agents, our primary contributions will be in training a DQN chatbot as well as developing a novel architecture for user simulation, the State-Action2Action model (SA2A) that aims to help with addressing cases where there is a lack of available data. Deploying a DQN chatbot rather than a simple single-layer neural network has shown improvements both in average reward as well as success rate in correctly responding to user queries. The SA2A model has also shown promise as an effective user simulation module.

1 Introduction

Generally, conversational chatbots are difficult to train as they require well-structured dialogue formats and significant overhead to make sure that knowledge is learned by agents that do not directly know the truth queries. A non-goal oriented approach is practically infeasible as it would require universal knowledge and the memory buffer for immediately learned knowledge would not be sufficient for any kind of complex query. This requires the scope of the problem to be reduced to a specific goal. Due to time constraints and the above reasons the group decided to take a goal-oriented approach.

Additionally, a database query approach is used to help train the agent on the correct action it should take. In a goal-oriented setting, this helps to guarantee that the truth information used for training is preserved and success can be measured based on if the agent provides the correct response query. Otherwise, trying to parse semantic meaning of different kinds of dialogues would lead to issues of trying to measure success in the chatbot agent.

While much work has been done in deploying deep learning architectures for natural language processing (NLP)[12] and in using reinforcement learning techniques in training agents to play video games [8], we are looking at combining these two approaches in addressing the conversation generation problem. With this project, there were two major goals at hand. The first goal of this project is to expand upon past work in using generative chatbots to help 'teach' a reinforcement learning agent how to engage in a dialogue environment [7, 12]. The second goal is to take a NLP-based approach to the user agent.

Winter 2019 CS224n: Natural Language Processing with Deep Learning

In our approach, the user agent is modeled by the SA2A architecture as seen in Figure 5 for conversational response generation and is trained on a corpus of conversation dialogue [3]. This user agent is then used to train a DQN agent as a chatbot via the end-to-end dialogue system.

2 Related Work

In NLP, one of the commonly used approaches for translating between two language representations is the seq2seq encoder-decoder framework that utilizes recurrent networks to capture temporal information [9]. This approach is common for going from one language to another or from a query to an answer (assuming the answer is known). Recent work with the Transformer model has presented a non-recurrent approach to sequence generation. This newer approach, in part, inspired our own development of the SA2A model.

In deep reinforcement learning, the success of the DQN for games [8] has inspired deep reinforcement learning approaches in other domains. Recent updates to this work has been in the use of deep recurrent Q networks (DRQN) which take history of states into consideration to better capture the dynamics [4].

Goal-Oriented Chatbots is one area of NLP that has received a lot of attention lately. The domains that are explored for these chatbots are quite wide ranging from the transcripts of a television show [2] to web/chat room transcripts [1]. In "Goal-Oriented Chatbot Dialog Management Bootstrapping with Transfer Learning", researchers try to apply transfer learning from one goal-oriented domain to another [5]. In "Training a Goal-Oriented Chatbot with Deep Reinforcement Learning", the author builds a basic pipeline for a rule-based user/agent dialogue with simplistic deep learning model [3]. Similarly, researchers have used deep reinforcement learning and dialogue pipelines for task-completion dialogue tasks. [6].

3 Approach

3.1 Methods

3.1.1 Conversational Pipeline

Our first step was to recapitulate the findings of "Training a Goal-Oriented Chatbot with Deep Reinforcement Learning" [3]. The purpose of goal-oriented (GO) chatbot is to produce a dialogue response given a state that represents the conversation history. This original model consists of four primary components, an RL Chatbot Agent, a Dialogue State Tracker (DST), the User, and an Error Model Controller as shown in Figure 2. We replaced the simplistic baseline vanilla Agent with a DQN Agent. The DQN model architecture is shown in Figure 1.

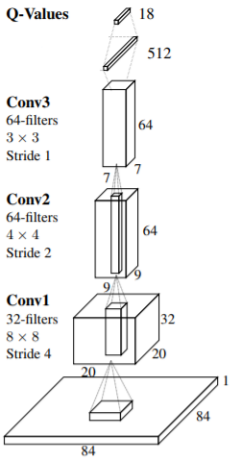

Figure 1: DQN Architecure [4]

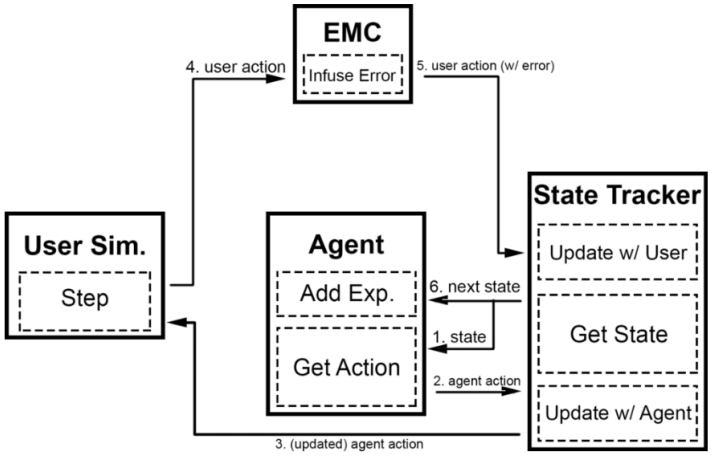

Figure 2: Our Training Loop [3]

Actions are encoded as dictionaries with keys being 'intent', 'inform slots', and 'request slots' as seen in Figure 3. The 'intent' field contains whether or not this action is to inform, request, thank (i.e. confirm that a request was met), or terminate the conversation. The inform and request slots contain the information the Agent wishes to communicate to the User or vice versa.

```
{'intent': 'request', 'inform_slots': {'city': 'seattle'}, 'request_slots': {'theater': 'UNK', 'starttime': 'UNK'}}
{'intent': 'inform', 'inform_slots': {'moviename': 'the witch'}, 'request_slots': {}}
{'intent': 'done', 'inform_slots': {}, 'request_slots': {}}
```

Figure 3: Examples of User/Agent Actions [3]

The state is constructed from the information the last User action as well as the last Agent action. This is to inform the Agent on its past action in order to promote the choice of the most optimal next action. The round number is also encoded, so as to encode a certain urgency in answering queries by the User. Finally, the state includes information about the current informs and how many items in the database match those current informs.

In this training loop, the current state is passed as input to the Agent to get the predicted action (Figure 3: Step 1). This action is then passed to the DST module so that the conversation history can be updated as well as adding database query information to the agent action (Step 2). This action is then passed to the User, who then makes a rule-based response in the GO chatbot scenario (Step 3). Our implementation of the User agent can make this action decision via the SA2A model described in the next section. This module also calculates the reward and success information. The User then outputs an action which is passed to the EMC for error calculation (Step 4). The user action and error are then passed to the DST to save the user's action in the conversation history (Step 5). Finally, the next state is passed to the Agent to complete the (state, action, reward, next state) experience tuple (Step 6).

The reward is a piece-wise constant function defined by the success of the Agent. If the Agent is able to successfully provide the correct response to the User agent, it gets a positive reward as a function of the maximum number of rounds (max_{rounds}) of the User agent as shown below. Similarly, it will get a negative reward based on the same quantity if it fails to answer the question correctly. The use of the max_{rounds} in reward calculation promotes queries to be answered with greater urgency.

$$Reward(success) = \begin{cases} 2 * max_{rounds} & \text{if } success == 1 \\ -max_{rounds} & \text{if } success == 0 \end{cases}$$

In the case of training in the pipeline, max_{rounds} is set to 20. If the maximum number of rounds is reached without success or success is reached, the Agent finished that training run and resets state to generate a new user action. Given this criteria, the maximum achievable average reward is 40.

The training block is relatively straight forward. After a certain period of cycles, defined as training frequency, the Agent is trained on its memory of experience. If the success rate of a period is greater than or equal to the current best success rate and surpasses a predefined threshold, then the memory is emptied in order to force the agent to learn from experienced from the newer and better model. The model weights are then copied to the target model as with all DQNs. The model is then trained on the current memory buffer of experiences.

The performance of the vanilla Agent in the original pipeline is considered to be the baseline of this model for the reinforcement learning component of this project. The performance of the DQN model implementation reflects the results of the model implemented by the group. The oracle is considered to be the rule-based policy approach where prior knowledge about the answers of the user questions is used. Performance is measured here in terms of average reward and average success rate.

3.1.2 State-Action2Action Model

The group devised a novel NLP model called State-Action2Action (SA2A) which takes in state information of a dialogue sequence and complex input actions in the form of raw text to generate output actions as seen in Figure 5.

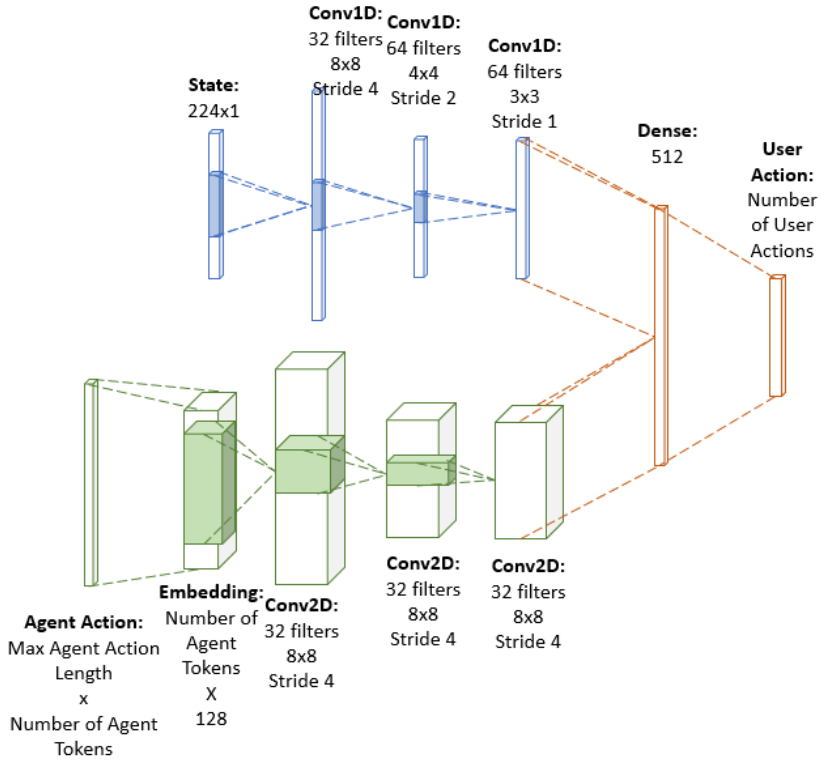

Figure 4: State-Action2Action Model

To train the generative User module, the SA2A approach is trained offline on data collected from a training sequence of the pipeline that uses the rule-based User module. In essence, this supervised learning approach utilizes the behavior of the rule-based User module to train the SA2A User.

This architecture takes in the Agent action and the current conversational state as inputs to generate one output, the User action. Notably, the conversational state is in a 1D vector representing the state and the Agent action is a 2D vector of the one-hot encoded Agent action. The dimensions of the Agent action are the maximum agent action length by the number of total unique agent tokens. Each input learns its own set of weights until they are concatenated together and passed to a final dense layer to generate the model outputs.

4 Experiments

4.1 Data

The pipeline was trained on a database of movie theater tickets represented as a dictionary where the keys are the indices of the tickets and the values are also dictionaries which contain the movie information.

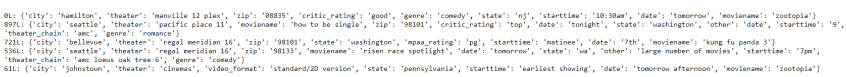

Figure 5: Movie Database Examples

While the DQN Agent trains, the User action (post error infusion), conversational state, and Agent action were recorded and put into string format. Once training is completed, the vectors are constructed from each of three strings in preparation for the SA2A User model. This dataset consists of what is necessary for training of the NLP model, with the Agent action and state vectors as inputs and the User action vector as the output. This dataset consisted of 799,991 sets of state, Agent action, and User action tuples.

4.2 Evaluation Method

Two evaluation metrics were used for the DQN agent, success accuracy as well as reward value. For the SA2A model, precision, recall, and micro-average f1 score were use to evaluate the approach.

4.3 Experimental Details

Using an EC2 instance on AWS, the group trained a DQN Agent as well as simpler vanilla Agent. The pipeline with the basic vanilla Agent serves as the baseline for the RL component.

For the DQN architecture, we used 3 convolution layers as defined by Figure 1. Also, hidden layers used relu activation functions with a linear activation for the output layer. For hyperparameters, the group used a learning rate was $1e^{-3}$, a batch size of 16, a γ value of 0.9 and an ϵ value of 0 for ϵ-greedy search. The vanilla Agent used a single hidden fully connected layer with size 80 and the same hyperparameters as the DQN Agent. Higher epsilon values were tried to promote exploration, but the epsilon value of 0, i.e. a greedy policy, performed the best. Mean squared error was used as the loss, and the Adam optimizer was used to minimize this loss. Both models were trained for 40,000 episodes with a 1000 episode warm-up to fill the experience replay memory.

The SA2A model was constructed as shown in Figure 5. Hidden layers used relu activation layers with a softmax activation layer on the output. RMSprop was used to optimize the categorical cross-entropy loss. The model was trained for 5 epochs.

4.4 Results

The test results for the baseline show a success rate in answering queries correctly of ≈ 0.83 and an average reward of ≈ 21.07 as shown in Table 1. The DQN agent performs marginally better with a success rate of of ≈ 0.85 and an average reward return of ≈ 22.7. We also see an improvement in the standard deviation, with tighter bounds for the DQN agent. This improvement is to be expected as the DQN model has far greater capacity than the simpler vanilla model. We don't expect a significant improvement in performance due to the smaller size of the dataset.

Model	Average Reward	Average Success
Vanilla	21.073 ± 27.499	0.8296
DQN	22.702 ± 25.992	0.8502

Table 1: DQN versus Vanilla Results

In regards to the test results for the SA2A model, the approach has a high micro-average f1 score of ≈ 0.94 as well as high precision and recall scores as seen in Table 2. There are in total 1039

unique actions the User may take and the dataset, due to the nature of acquisition via simulation, is unbalanced. It is promising to see the model perform well even with imbalances in data.

F1 Score	Recall	Precision
0.944	0.884	0.913

Table 2: SA2A Model Metrics

5 Analysis

The results shown in Table 1 suggest that the DQN model is more scalable for the chatbot agent than the vanilla implementation. This suggests that the chatbot pipeline benefits from a DQN implementation and could benefit from more complex implementations in the future.

The training of the SA2A showed strong results in terms of f1 score, precision, and recall. It would help to have a baseline to compare this to but since the default user simulator is our baseline in this case, we are comparing to perfect accuracy. Overall, our model can generalize well for this dataset but it would be worthwhile to explore this approach for other methods like action2action directly.

In regards to a qualitative analysis of our SA2A results, we captured three samples that suggest certain features of out SA2A model.

```
True User Action:{intent:inform,request_slots:{},inform_slots:{'city': 'portland'}}
Model User Action:{intent:inform,request_slots:{},inform_slots:{'city': 'los angeles'}}
```

Figure 6: Error Example 1

Figure 6 shows that the model was able to capture the nature of the inform action being related to the city in which the movie was to be seen. However, the model outputted the incorrect city, Los Angeles rather than Portland. This error could be due to an unbalanced dataset, with more examples including Los Angeles as the city rather than Portland.

```
True User Action:{intent:inform,request_slots:{},inform_slots:{'starttime': '10:00 pm'}}
Model User Action:{intent:inform,request_slots:{},inform_slots:{'starttime': '9:30 pm'}}
```

Figure 7: Error Example 2

Figure 7 also shows a similar error to the previous figures. This error to could be due to an unbalanced dataset, with more examples including 9:30 p.m. as opposed to 10:00 p.m. Another possible issue is that the model has never seen 10:00 p.m. as a possible valid action, since the entire dataset to train the SA2A model was collected from running the chatbot training pipeline.

```
True User Action:{intent:inform,request_slots:{},inform_slots:{'date': 'tuesday'}}
Model User Action:{intent:inform,request_slots:{},inform_slots:{'theater': 'amc lowes oak tree 6'}}
```

Figure 8: Error Example 3

Figure 8 shows that the model was able to consistently follow formats and aliases but it does not produce the exact same result as shown by discrepancies in start time. This shows issues with the model not being able to fully inform the user for the correct query and may be a failing due to convergence to a local optima.

6 Conclusion

This paper has shown the effectiveness of using a DQN agent for a conversation chatbot in a database query setting as well as the first steps into designing an end-to-end training pipeline with rule agnostic user simulation. We have shown the benefits of using deep Q learning in a data base query setting with our DQN agent as well as the greater generality afforded by a model like the SA2A model for user simulation.

6.1 Limitations

The primary limitation of the current pipeline is the reliance on a structured rule-based approach. This has led to difficulties in integrating the new SA2A architecture within the larger pipeline framework. In particular, the lack of perfect accuracy would cause infinite feedback loops where the user is incapable of breaking out of the same query. This would confuse the DQN into oscillating between weights that give no meaningful information and causing very poor success rates.

A second limitation is the assumption of a finite User action space for the User agent simulator. This limitation was chosen in part to the simpler training of the model, but it can be remedied however with a generative component, such as an RNN, LSTM, or Transformer. Finally, the smaller dataset allows even a simplistic model to fit it well, but our approach allows for this pipeline to scale to larger and more complex datasets.

6.2 Future Work

The goal-oriented chatbot we trained still uses a 'vanilla' DQN agent in conversation action decisions. After replacing this component with a DRQN, we found that training took much longer than the DQN model. This is probably due to the exponential growth in the state space due to the state space now including past states. Our first step is to compress the state space representation to help the DRQN agent explore more of the state space when training. Additionally, other variations of the DQN architecture should be explored, particularly Dueling DQNs and Double DQNs [10, 11].

Another major change is to switch to a larger dataset with more natural conversational queries and step away completely from a simplistic rule-based approach. This will require us to replace the User Simulation which is currently a rule-based approach to one that uses an NLP approach, such as TC-Bot [6]. Though we have trained the SA2A model for just this purpose, we have yet to integrate this module in the larger pipeline.

In evaluating the potential of the SA2A model, the group would like to train the model on other conversational datasets that take into account state to evaluate the potential of this model for goal-oriented chatbots and general question-answer based NLP tasks. Based on the initial results, the group finds this approach meriting further exploration.

7 Additional Information

7.1 Mentor

Christopher Manning

7.2 Sharing Project

CS234: Reinforcement Learning

References

[1] Nltk web and chat text. `http://www.nltk.org/book/ch02.html`. Accessed: 2019-02-12.

[2] Friends chatbot. `https://github.com/npow/friends-chatbot`. Accessed: 2019-02-12.

[3] Max Brenner. Goal-oriented chatbot with deep rl. `https://towardsdatascience.com/training-a-goal-oriented-chatbot-with-deep-reinforcement-learning-part-i-introduction-and-dce3af21d383`, 2018. Accessed: 2019-02-12.

[4] Matthew J. Hausknecht and Peter Stone. Deep recurrent q-learning for partially observable mdps. *CoRR*, abs/1507.06527, 2015. URL `http://arxiv.org/abs/1507.06527`.

[5] Vladimir Ilievski, Claudiu Musat, Andreea Hossmann, and Michael Baeriswyl. Goal-oriented chatbot dialog management bootstrapping with transfer learning. *CoRR*, abs/1802.00500, 2018. URL `http://arxiv.org/abs/1802.00500`.

[6] Machine Intelligence and Understanding Laboratory. User simulation for task-completion dialogues. `https://github.com/MiuLab/TC-Bot`, 2017. Accessed: 2019-02-12.

[7] Mahipal Jadeja, Neelanshi Varia, and Agam Shah. Deep reinforcement learning for conversational AI. *CoRR*, abs/1709.05067, 2017. URL http://arxiv.org/abs/1709.05067.

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

[9] Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. Sequence to sequence learning with neural networks. *CoRR*, abs/1409.3215, 2014. URL http://arxiv.org/abs/1409.3215.

[10] Hado van Hasselt, Arthur Guez, and David Silver. Deep reinforcement learning with double q-learning. *CoRR*, abs/1509.06461, 2015. URL http://arxiv.org/abs/1509.06461.

[11] Ziyu Wang, Nando de Freitas, and Marc Lanctot. Dueling network architectures for deep reinforcement learning. *CoRR*, abs/1511.06581, 2015. URL http://arxiv.org/abs/1511.06581.

[12] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017. URL http://arxiv.org/abs/1708.02709.