Training Dialog Agents to Negotiate

Bogac Kerem Goksel Ishan Somshekar

Abstract

We explore different methods of applying reinforcement learning to train agents that can negotiate for resources using natural language dialog. We base our work off of previous work by Facebook AI Research on a dataset they collected and models they trained on this dataset. We investigate the role of Reinforcement Learning in particular within the multi-stage training process the original paper described, and also experiment with a pure RL agent that is not pretrained on the human dataset.

1. Introduction

Negotiation is an interesting task as it requires both natural language understanding, some degree of goal-based planning and natural language generation. A good negotiation agent needs to be able to understand offer it receives and be able to verbalize its own offers coherently and convincingly, but it also needs to be able to reason about the valuation of resources and plan negotiation tactics to be able to negotiate successfully. Furthermore, negotiation also offers a good multi-agent task as it is essentially competitive, but requires eventual compromise and agreement between the agents for either to receive any reward.

The original paper we base our work off of is ”Deal or No Deal? End-to-End Learning for Negotiation Dialogues”, published by Lewis et. al in 2017 (Lewis et al., 2017). In this paper, Lewis et al collect a human-to-human negotiation dataset built on a very simple negotiation game, and train an end-to-end sequence-to-sequence dialogue agent using supervised learning. They then experiment with several techniques to improve these agents including Reinforcement Learning, where they use policy gradient methods to train the parameters of their seq2seq model using the REINFORCE algorithm.

In this paper, we investigate the effectiveness of using RL in this context by analyzing its performance both in a pure RL agent that has a separate negotiation module that only generates numerical offers then uses template strings to utter in English, and on top of the pre-trained seq2seq model.

2. Background and Related Literature

While there has been work on negotiation using reinforcement learning in the past, our work is mostly about replicating the results in Lewis et al's paper, and investigating performance aspects in their work.

3. Data Collection and Processing

3.1. Task Description

Lewis et al propose the following task for testing the negotiation capabilities of agents. In this setting, there exist three distinct resource types (book, ball and hat in this instance). Each episode starts with some fixed number of these resources ”up for grabs”. Each resource type has some value to each agent, and these values are only known to the agent itself. The task for the agents is to choose an allocation of these resources between themselves that both of them can agree upon. Upon agreement, each agent's reward is the number of each item type they received times their personal value for that item. If they cannot reach agreement, neither agent receives any reward. The agents can have as many dialog turns as they want, until one of the agents decides that the conversation is over. At that point, both agents output their best guess at what was the final agreement. If their guesses at the final agreement match, they get a reward, otherwise no reward is given.

More formally, both agents get a context vector at the beginning of each episode which includes the counts of each types of item and the unit value of each item type to them. (i.e. 1 book: value 5, 3 hats: value 0, 2 balls: value 1) One of the agents is randomly selected to make the first utterance. From then on, each agent gets fed the utterance of the other agent at each dialog turn and outputs their own utterance. On top of any natural language utterances agents may output to haggle, they can also output a special ¡selection¿ token which indicates they think the negotiation is over.
Once an agent emits this ending token, the agents output their final allocation of the items, and only if their allocations agree do they actually allocate the items and each get their respective rewards (count of each item type they got times the value of that item type to them).

3.2. Dataset

Lewis et al have run this task between humans on Mechanical Turk to create a dataset of human-to-human negotiations. The dataset includes 6000 conversations of the form:

<input>2 1 2 3 3 1 </input>
<dialogue>YOU: I'd like the .... THEM: .... YOU: ... <selection></dialogue>
<output>item0=2 item1=2 item2=0 item0=0 item1=0 item2=3 </output>
<partner-input>2 0 1 2 3 5 </partner-input>

We see that each point in the training data contains information about the initial amount of resources available to the agents and the respective values of the resources to the agents (these values can be different) in the <input>and <partner-input>tags, the entire negotiation dialog in the <dialogue>tag, and the final agreed upon allocation of resources in the <output>tag.

3.3. Data Processing

We collected information about trends in the data. The distribution of total dialog lengths in words is seen below.

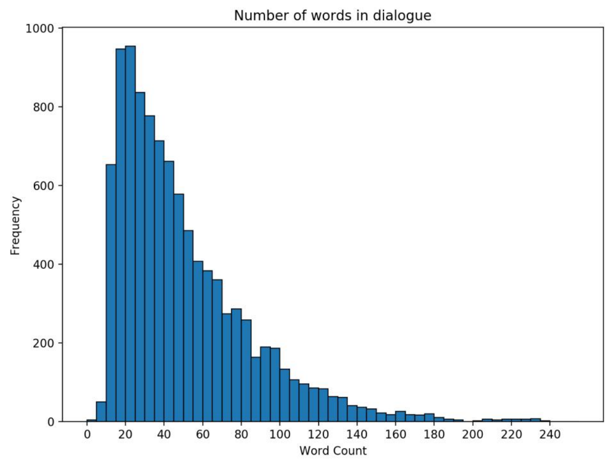

Figure 1. Frequency of the lengths of dialog in words

We see that the human dataset peaks at around 15-20 words in a negotiation before both parties find an agreement or break off negotiations. However, there is a significant tail to this data, as dialogs between 10 words and 70 words are still relatively frequent in the data.

Additionally, we also checked to see how many utterances back and forth between the two parties was necessary before they came to a decision. Here we plot the distribution of dialog lengths in utterances by either party.

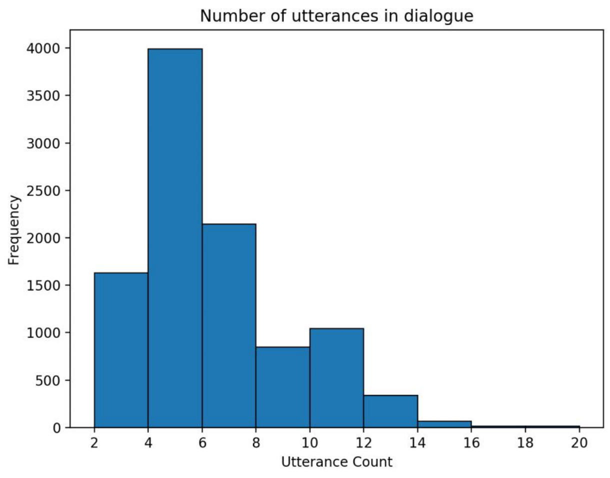

Figure 2. Frequency of the lengths of dialog in utterances

We see that the majority of negotiations are completed after each agent has spoken four times. There are some outlying data points from longer conversations because some human speakers will begin with "hello" or "how are you doing" and "I'm fine, how are you?"

4. Approach

4.1. Language Model Agent

Lewis et al base off their work on a supervised language model agent. This agent is trained on the human-to-human dataset, and it is only trained to act as a language model (i.e. predict what a human would've said next given the current state of the dialog and the initial context). As such, this agent does not incorporate the actual rewards it gets from the game at all.

The architecture of this dialog agent involves the use of a two-layer fully connected neural net to encode the initial context into a dense representation alongside one-hot word embeddings, which are used to encode the conversation history. During conversation, it concatenates the embeddings for the context and words in the conversation and feeds this into an encoder GRU (concatenating the same context embedding for each word in the conversation history). It then stores the outputs of this GRU as a representation of the conversation history.

To output utterances, it uses another GRU to go over the conversation history representations and attend over it using a fully connected attention network. Finally this conversation representation with attention is concatenated with the initial value context and is fed into a decoder network that outputs log probabilities over all possible offers that can be made.

The simple dialog agent samples from these log probabilities with an ϵ-greedy policy to output utterances, no part of this system has an explicit representation of the offers being made, so the model's parameters are being trained purely on accuracy of language.

4.2. Reinforcement Learning Agent

Lewis et al then fine tune the language model agent by framing the task as a reinforcement learning task where the language model becomes the policy parameter. They use the REINFORCE algorithm to further train the parameters of the language model after episodes of self-play against another agent.
To enable effective self-play, two agents are initialized with the language model, but one of the agents is kept fixed through self-play episodes while the other agent's policy (dialog model) is updated after every episode. In this setting, the training agent no longer considers the original dataset's language model in its optimization, and only optimizes for the task reward.
The policy gradient is more specifically applied as the following: If we set r be the final score for A, T be the dialog length, μ a running average of completed rewards, and γ a discount factor, the reward R for an action x_t is $R(x_t) = \sum_{x_t} \gamma^{T-t}(r - \mu)$

4.3. Template Filling Reinforcement Learning Agent

To test the effectiveness of a pure reinforcement learning agent in negotiation performance, we implemented the following agent that is trained only using Reinforcement Learning, but that nevertheless can consistently make sensible English utterances.
The agent does not attempt to learn the nuances of producing human language, instead focusing on learning how to negotiate for an ideal allocation of resources. Our goal was to was to explore whether end-to-end language learning was necessary from a purely negotiation optimizing standpoint. To achieve this, we constrained the policy of the agent to only outputting a numerical allocation it wanted to propose to the other side (1 book for me, 1 ball for me, 0 hats for me), and used predetermined template strings to convert these to utterances. Our agent still had to read the conversation history to be able to respond, so we kept that part as part of the policy as well. Our final architecture for this model looked like the following:

Our Template-Filling RL agent first reads the utterances of the other agent using a GRU and a single forward pass through the data. It then concatenates the context embedding to the last hidden state of this GRU. This new embedding is fed into a fully-connected layer and then a softmax layer.
At this point, the agent uses the ϵ-greedy technique to balance selecting the allocation of resources that produces the highest probability and reward and a random allocation.

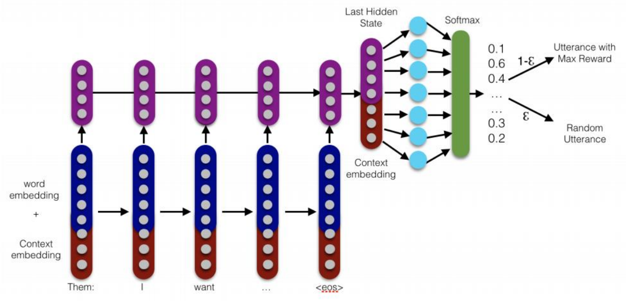

Figure 3. The architecture of our Template Filling agent

The agent then fills out a basic string template with this allocation.

The template string that we used was:
I want [] books, [] hats, and [] balls. You can have the rest.
The agent could also output Deal , No Deal, or "<selection >" to terminate the episode.

5. Experiments and Results

5.1. Initial Tests

The Template-Filling agent did not perform well at all when pitted against the language model agent; the two agents found agreement only 2% of the time. This was better than random, which would have been 0.8%, but still was not high enough to lead to consistent rewards. We felt that this was not necessarily indicative of a failure of the specific architecture, but rather a symptom of the issues that could arise from the limitations of applying reinforcement learning to tasks related to natural language dialog.
To get a better understanding of these limitations, we experimented with Lewis et al's vanilla RL agent and explored its performance.

5.2. Performance

We found interesting trends when looking at the results of the language model + RL Agent when it trained against a fixed language model agent.
As more training episodes went by, the rate of agreement increased dramatically at first to about 20%, but then began to fluctuate before falling. Since rewards are only given out when there is an agreement, the graph for reward attained for either party is almost identical with the rate of agreement.

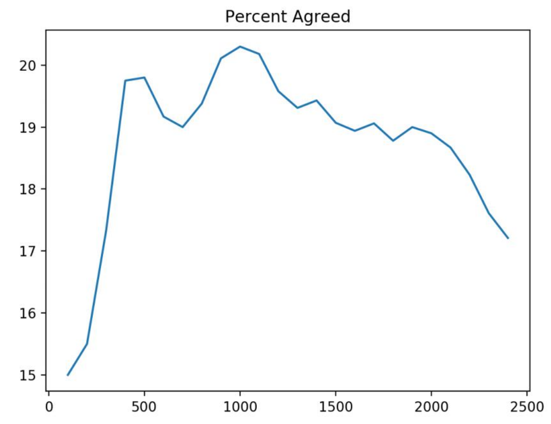

Figure 4. The percentage of negotiations agreed for RL vs Simple Dialog against training iterations

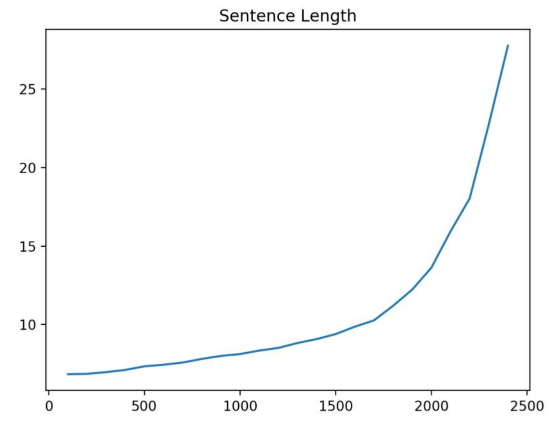

Figure 6. The average length of a single utterance against training iterations

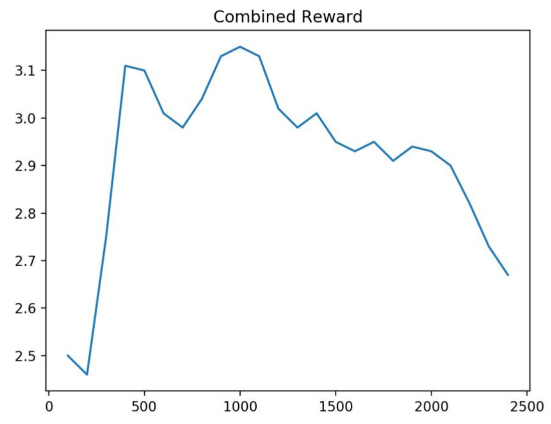

Figure 5. The combined reward attained for both agents against training iterations

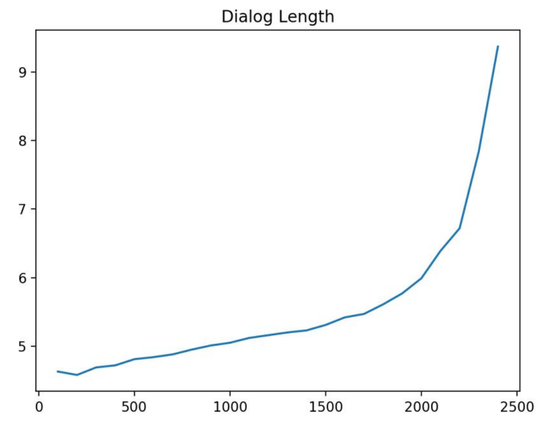

Figure 7. The average number of utterances per episode against training iterations

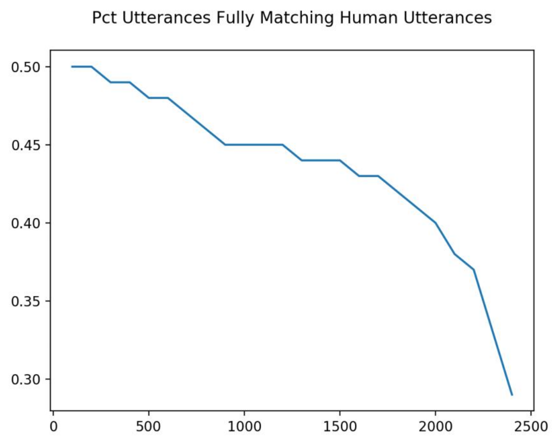

Figure 8. Similarity of human and agent produced utterances against training iterations

To further explore why this rate dropped drastically after about 2500 data points, we looked at the output negotiation sentences that the agents were producing. We see that the average number of words in a dialogue increases exponentially as we continue to train. Additionally, the number of dialog turns also increases. Both these values begin at around similar figures to the human statistics from the training data, but quickly increase until the average length of sentence and conversation are more than 3 times the human average.

Finally, we examined how these utterances that were being produced compared to actual human utterances in the data. The utterances were no longer really decipherable or understandable from a qualitative perspective (where examples like ”i am all hats i am am am want want . . . . want . . . . hats” became more and more common)

Partially due to their longer length, they were also no longer quantitatively similar to human utterances. As the RL agent was trained more and more, the more ”original” utterances it started to come up with (i.e. an utterance that wasn’t found at all in the human dataset).

While these metrics might be interpreted as the agent becoming more creative and learning to utter sentences beyond copying over utterances from the human dataset, both our qualitative analysis and the rapid fall in agreement rates suggest divergence from natural language. It is also worth noting that while divergence might be expected for the agent whose parameters are being updated, the metrics exponentially diverged for both our agents, suggesting the fixed agent was also showing divergent behavior.

6. Evaluation

This problem suffers from a common problem in RL that rewards are given only at the very end of a long episode, and only somewhat good policies can achieve any reward (as it requires being able to agree on an allocation and not ending the dialog before an agreement is found). Furthermore, since the task is somewhat cooperative, the RL agent’s performance is bounded by the competency of the fixed agent it is communicating with.

In the case of the template-filling RL agent, we believe the failure was due to both the sparsity of the rewards, and the incompatibility with the fixed agent. Since our agent ”spoke” in repetitive templates in a pattern that wasn’t common in the human dataset the fixed agent was trained on, the fixed agent in many cases could not decipher the otherwise valid English our agent was outputting. As such, the RL agent was not getting proper feedback from its environment, and the likelihood of getting a reward to positively train its policy was even lower.

We see this to a certain extent in the RL fine tuning of the language model agent as well. This case mitigates the latter problem a little bit since the RL agent is speaking in utterances that the fixed agent is trained on, and as such they can reach agreement through imitation of human dialog. However, the issue reappears as once the RL agent trains a certain amount and starts to differ from the human dataset, the other agent’s performance also decays and there’s an exponential loss of combined performance.

To help with the reward sparsity issue we experimented with adding a direct negative reward to making a proposition that was impossible given the current count of the items (which the policy was capable of since it was in the action space of the greater task where some other instantiation might have more of a certain item). This had no perceptible performance gain for our RL agent, which we believe was due to the fact that the space of the possible allocations, even with the impossible ones filtered out, was still large.

7. Future Steps

Both the original paper and our experiments have shown that while a Language Modeling based agent can produce somewhat believable dialog for negotiation, directly attempting to fine tune this network with RL is problematic as being completely free from the linguistic constraints of a language model pushes the agent to diverge from human understandable language, and the co-dependence of the agents exacerbates the problems. We have also shown that trying to keep the language aspect fixed by using very simple template-filling also fails as the other fixed agent cannot handle a robotic sounding counterpart.

One immediate solution that we implemented is to integrate the language modeling into the RL framework by adding a fluency reward (the likelihood each utterance gets from the LM). More specifically, the RL agent with the fluency reward is initialized with two separate language models that have identical weights in the beginning. However, the REINFORCE updates are only applied to the parameters of one of them. This updated language model is used to generate the utterances in the dialog. The fixed 'reference' model on the other hand is used to calculate the likelihood of the utterance of the updated model. This likelihood is then used as an immediate reward in the RL task. The balance between the game rewards and language fluency rewards can be tuned as a hyperparameter. In our experiments we observe that training this way gives a slight boost to the percentage of agent utterances that fully match a human sentence without affecting game performance when one of the agents is kept fixed. However, the real expected effect of this method is when both agents are simultaneously trained, as we're expecting such training to actually give rise to better negotiation strategies and less exploitation of a particular language model. We're still hyperparameter tuning to achieve stable training between two RL agents in this scenario.

7.1. Better Language Model and Goal Planning Separation

We believe an even better way to achieve goal-based training without diverging from human language is model-level separation of language modeling and negotiation planning aspects of the agent. We believe the first milestone for us would be to keep the language model for dialog generation, but condition the generated responses on a goal embedding, which could be outputted by a neural architecture trained using RL, while the language model aspect could still be trained using supervised learning.

Such an approach would come with the challenge of jointly training the two components, but the fact that original human dataset includes the context information and reward information might help with off-policy training, first training the RL component to output goal embeddings that push the language generation component to utter the same things the humans did, and later only training the goal embedding parameters through reinforcement learning.

Finally more recent approaches like the editor approach which picks a template sentence from a large human dataset, and then edits it word by word using some conditioning vector, could be applied successfully in this task where we want agents to be able to utter sentences similar to those from the human corpus, but with clever edits (like changing number of items offered) conditioned on a context vector (the goal of the agent, trained by RL) that they learn to train.

8. Additional Info

- Late days used: 3
- Contributions: I (Kerem Goksel) am submitting this project to 224N. Me and my partner(Ishan Somshekar) also submitted it to CS 234. Ishan and I pair programmed the template filling agent together and ran the experiments together. After we submitted for 234, I implemented the language model scoring and ran further experiments between various agent types for 224N and fixed some bugs in the original codebase.

  Unfortunately the LM-scoring experiments didn't make it to the report, but they can be found on our Codalab worksheet: https://worksheets.codalab.org/worksheets/0xf9be89b05d6b4e1d8484b4f (might be easier to run 'wsearch end-to-end-negotiator' on codalab to find it) and our GitHub: https://github.com/bkgoksel/negotiator).

  Our work is based off of the publicly released codebase by Facebook. (https://github.com/facebookresearch/end-to-end-negotiator)

  I am also in the Codalab team and was providing full support for the Codalab leaderboard throughout the quarter, both to Abi and occasionally on Piazza. I haven't kept track of actual hours spent on this but looking at internal Slack/email timestamps I estimate I spent 40 hours over the past two weeks for Codalab debugging and guidance for 224N.

References

Lewis, Mike, Yarats, Denis, Dauphin, Yann N., Parikh, Devi, and Batra, Dhruv. Deal or no deal? end-to-end learning for negotiation dialogues. *CoRR*, abs/1706.05125, 2017. URL http://arxiv.org/abs/1706.05125.