[Final Paper] Deep Reinforcement Learning with Hierarchical Recurrent Encoder-Decoder for Conversation

Heejin Jeong and **Xiao Ling**

Final Project Paper for CIS700-007 Spring 2017

Introduction

In recent years, numerous neural network based methods have been introduced for automatic dialog generation. Sutskever et al. presented a sequence to sequence learning model with deep neural networks (SEQ2SEQ) consisting of two multi-layered Long Short-Term Memory (LSTM) and showed that it outperformed a standard SMT-based system on an English to French translation task (**?**). Vinyals et al. applied this sequence to sequence framework to conversational modeling and their model predicted the next sentence given the previous sentence (**?**). Unlike other models that had been used widely, this model achieved better performance requiring much fewer hand-crafted rules.

However, since the SEQ2SEQ model was originally proposed for machine translation tasks, it does not capture previous conversations when it is applied to a conversation task. Being able to address previously mentioned topics or information is essential in conversation. In order to solve this issue, Serban et al. extended the hierarchical recurrent encoder-decoder (HRED), proposed by Sordoni et al. (**?**), to the dialog domain (**?**). Although they used only triple utterances for implementation, they were able to show their proposed model outperformed both n-gram based models and baseline neural network models. Another issue of the SEQ2SEQ model is that it tends to be short-sighted since the model does not consider its future outcomes. Addressing this issue, Li et al. proposed a novel approach for a dialog generation task combining a policy gradient optimization method and the SEQ2SEQ model (DRL-SEQ2SEQ) (**?**). The policy gradient optimization method is one of policy-based methods in Reinforcement Learning (RL), and it updates its policy parameters in order to maximize a learner's expected discounted future total reward. Thus, we can expect the model would generate an outcome more likely to continue the current conversation.

While the HRED model does not consider its future outcomes, the DRL-SEQ2SEQ model cannot address its conversation history other than a few immediate previous utterances. In this project, we studied the HRED model for dialog generation in open domain (**?**) and DRL-SEQ2SEQ. We initially aimed to replace SEQ2SEQ model in DRL-SEQ2SEQ with HRED in order to utilize the advantage of each model. However, due to the time constraint, we were not able to combine both models but tried implementing them on Open-Subtitle Dataset.

Hierarchical Recurrent Encoder Decoder

Deep Reinforcement Learning for Dialog Generation

In the RL framework, there is a learner in a certain state s and the learner executes an action a to an environment based on her action policy $\pi^{(a)}$. Then, the environment gives a feedback in a form of a reward, r, and the learner updates her optimal policy π^* in order to maximize her expected discounted future total reward:

$$\mathbf{E}[r_t + \gamma r_{t+1} + \cdots + \gamma^{T-t} r_T | s_t, \pi^*]$$

where T is time horizon and γ is a discount factor which controls the relative importance of the immediate reward for the learner. If $\pi^{(a)}$ differs from π^*, the learning method is categorized to *off-policy*, and if $\pi^{(a)} = \pi^*$, the method is *on-policy*. In order to apply RL to a certain learning task, we need to define an appropriate learning system first. In this project, we followed the learning system defined in the DRL-SEQ2SEQ paper (**?**).

Learning System for Dialog Generation

The learning system consists of two agents where they take turns talking with each other. Unlike the notations in the paper, we will use u_i as a notation for the i-th utterance of a conversation regardless which agent spoke it in order to make sure that the learning model is trained with both agents.

A state is defined by the previous two dialog turns, $s_t = [u_{t-1}, u_t]$. Specifically, the concatenated vector of the two utterance vectors is used as an input of the SEQ2SEQ encoder. An action a is defined as a dialog utterance to generate. The action space, $\mathscr{A}$, is considered as an infinite space, but we constrained the maximum length of the generated utterance, L, and the number of available vocabularies, $|V|$ (V is a set of available vocabulary). Therefore, $|\mathscr{A}| = L \times |V|$ in our model. The RL policy, p_{RL}, uses the form of a SEQ2SEQ encoder-decoder defined by its parameters.

Reward

Li et al. defined three reward functions and used the weighted sum of the rewards as a total reward (**?**). The first

reward function measures the ease of answering a generated turn (or action) using the log likelihood of responding to that action with a dull response. They defined dull responses with 8 turns including "I don't know what you are talking about", "I have no idea", etc. The function is:

$$r_{1,t} = -\frac{1}{N_{\mathscr{D}}} \sum_{d \in \mathscr{D}} \frac{1}{|d|} \log p_{seq2seq}(d|a_t) \quad (1)$$

where $\mathscr{D}$ is a set of dull responses and $p_{seq2seq}$ is a probability distribution with parameters learned by SEQ2SEQ. $p_{seq2seq}$ is different from p_{RL}.

The second reward function penalizes semantic similarity between consecutive turns from the same agent. They used cosine similarity to measure:

$$r_{2,t} = -\log \cos(h_{u_{t-1}}, h_{a_t}) = -\log \cos \frac{h_{u_{t-1}} \cdot h_{a_t}}{||h_{u_{t-1}}||_2 ||h_{a_t}||_2} \quad (2)$$

where h_{u_t} is an SEQ2SEQ encoder output with the input u_t.

The last reward function addresses semantic coherence considering mutual information:

$$r_{3,t} = \frac{1}{|a_t|} \log p_{seq2seq}(a_t|u_{t-1}, u_t) + \frac{1}{|u_t|} \log p^{backward}_{seq2seq}(u_t|a_t) \quad (3)$$

where $p^{backward}_{seq2seq}$ the probability distribution with parameters learned by SEQ2SEQ where sources and targets are swapped. Therefore, this backward SEQ2SEQ model has to be separately trained before using the RL rewards.

Model

Any RL method falls into one of policy-based, value-based, or actor-critic RL areas. Among policy-based RL methods, a policy gradient optimization (PGO) using REINFORCE algorithm (**?**) has been widely used. Li et al. also used the REINFORCE algorithm for learning the RL model. The training procedure of the RL model is divided into three steps as shown in the Fig.**??**. First, we train a SEQ2SEQ model and a SEQ2SEQ backward model. Next, we train a mutual information model learned by PGO maximizing the semantic coherence (Eq.**??**). The policy parameters of the model are initialized by the parameters of the pre-trained SEQ2SEQ model, and its objective function is:

$$J(\theta) = \mathbf{E}[m(\hat{a}, [u_{t-1}, u_t])] \quad (4)$$

where $m(\hat{a}, [u_{t-1}, u_t])$ is equal to $r_{3,t}$ in the Eq.**??**. The gradient is estimated using the likelihood ratio trick in PGO:

$$\nabla J(\theta) = m(\hat{a}, [u_{t-1}, u_t]) \nabla \log p_{RL}(\hat{a}, [u_{t-1}, u_t]) \quad (5)$$

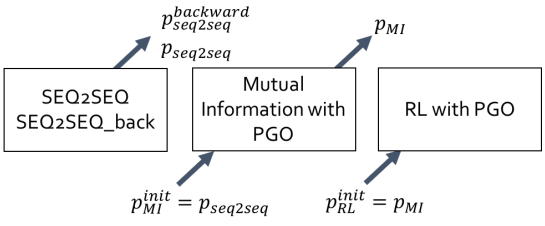

Figure 1: Training Procedure of the proposed Reinforcement Learning Model

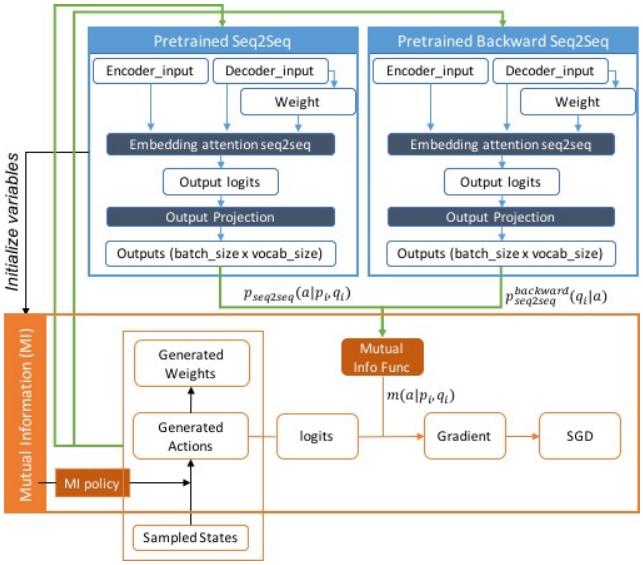

Figure 2: SEQ2SEQ and MI model Code Architecture

Then, we update the parameters in the encoder-decoder model using stochastic gradient ascent. In addition to the standard PGO method, Li et al. adopted a curriculum learning strategy proposed by Ranzato et al. (**?**) and a baseline strategy. The baseline strategy is common in implementing PGO method since PGO tends to have high variance, and the strategy helps to reduce the variance.

Experiment

Dataset

We first trained the modified SEQ2SEQ model on the CALLHOME American English Speech corpus (LDC97S42), consisting of 120 of 30-minute phone conversations between native English speakers.The size of vocabulary we used is 8038. However, in most of the conversation datasets, one speaker talks a long sentence or several sentences in a row and the other speaker answers with one or two words such as *"mhm"*,*"okay"*, and *"yeah"* as shown in Table.**??**. In fact, about 20% of the utterances of the dataset is *"mhm"*. As a results, the trained model after 44200 steps generated a short answer for long input utterances and a long answer for short input utterances (Table.**??**). Therefore, we concluded that this dataset may not have been appropriate for learning a dialog generation model, and used OpenSubtitles dataset (http://opus.lingfil.uu.se/OpenSubtitles.php) with 50005 vocabulary size. The DRL paper also used this dataset, but unlike the paper, we didn't extract *"i don't know what you are talking about"*. Instead, we refined abbreviated words such as *"i'm"* → *"i am"*, "you're" → *"you are"*, "i've" → *"i have"*, "wanna" → *"want to"*, "gonna" → *"going to"*, "don't" → *"do not"*, etc. The tokenization part was mostly done by Xiao Ling.

Table 1: Example Conversation in the CALLHOME corpus

A:	who is in um someone not that they have problems but someone who's like an okay student but kind of on the borderline you know like maybe not a great homelife and we would ha i got paired up
B:	uh-huh
A:	with someone at um lipsmack i forget the school it was actually in port richmond um breath i forget the name of the school
B:	really
A:	hensfiel no it was in the philadelphia school system and it it was a middle school
B:	mhm

Implementation

Although it is not common to explain code structures in an academic paper, we explicitly mentioned functions and files in tensorflow we used as well as those written by us in this section. For the SEQ2SEQ model, we applied Sequence to sequence with attention mechanism and used GRU cells instead of LSTM cells. We used 3 layers and 512 units for all models. We implemented models in tensorflow version 1.0.1. We used `embedding_attention_seq2seq()` function in `seq2seq.py` and related functions/codes written in tensorflow. We also utilized `seq2seq_model.py` in tensorflow version 0.12.x as a bottom line.

The `seq2seq_model.py` is structured as follow (see Fig.**??**): when the model is initialized, three main lists of placeholders are constructed - encoder, decoder, and weight. The length of the lists are encoder size, decoder size, and decoder size, respectively. Each element of a list is a tensor with a shape [batch size, None]. Variables are also built and initialized. In each step of training, batch size data are sampled in a given training dataset and converted into a proper format in the function `get_batch()` by operations such as padding, adding GO id, sizing, etc. Then, these are fed to the placeholders. Output logits are computed through `embedding_attention_seq2seq()` function and losses are computed using a softmax loss function. Then, variables in the encoder-decoder recurrent neural network (RNN) are updated.

The mutual information model requires a pre-trained SEQ2SEQ model and a pre-trained backward SEQ2SEQ model for its initialization and for computing mutual information score. The model first builds three placeholder lists - concatenated states (e.g. u_{t-1}, u_t), the most recent state (e.g. u_t) and weights for the backward model. Instead of sampling several action candidates given an input from the policy probability distribution p_{RL} as presented in the paper, we sampled a batch of inputs (size of 64) and selected the best action of each sampled state based on their probability outcomes. These states (or inputs), actions, and weights become inputs to the pre-trained SEQ2SEQ and the pre-trained backward SEQ2SEQ model, and we obtain log probabilities from each model. Then, the mutual information score and the gradient in Eq.**??** are computed. According to the paper (**?**), stochastic gradient descent (SGD) was applied to the objective function **??**. However, since SGD is for minimization and the objective function has to be maximized, we used SGA instead to update variables of the encoder-decoder RNN. In this project, we implemented mutual information model without the curriculum and the baseline strategies.

Results

The training results of the backward SEQ2SEQ model after 109000 steps are presented in Table.**??**. Since it is a backward model, the bot generates the most likely utterance that will have the input utterance as its response. The examples in the Good section show good performance of the model. The generated responses in the Bad section do not make sense. The model tends to generate certain responses a lot such as "i am going to go to the bathroom", or to repeat all or some words in an input utterance. The training results of the SEQ2SEQ model with concatenated utterances after 107000 steps are presented in Table.**??**. Each row of the table is one conversation, since the model consider two previous sentences. However, from the results, it was not hard to see that the model responses considering both two previous utterances. Also, it generates "i am sorry" or "no" many times. For example, for almost any human input utterance starting with "can you", it generates "no".

The results of the mutual information model were not good at all. First of all, since the mutual information score is averaged by the number of words in a sentence and each term is negative (log probability), we could see that the shorter a response is, the higher its score is. Also, it tends to generates lots of repeated words in a sentence. The example is shown in **??**. Therefore, we added an additional reward for the number of words, N_a of an output sentence a and penalties for the repeated words and for "i am sorry" or "no" response:

$$\begin{aligned} J(\theta) = \mathbf{E}[-m(\hat{a}, [u_{t-1}, u_t]) - c_l N_a \\ + c_r \sum_{1 \le i < N_a} \mathbb{1}(w_i^{(a)} == w_{i+1}^{(a)}) \\ + c_d \mathbb{1}(a == \text{"no" or "i am sorry"})] \end{aligned} \tag{6}$$

where c_l, c_r, and c_d are constant weights and $w_i^{(a)}$ is the ith word in a sentence a. However, those additional reward and penalties didn't improve the overall quality of responses. The Table.**??** shows an example of after 20 steps and the Table.**??** shows an example after training for 540 steps. Since we considered only the next word in the repeated-words penalty, the generated response repeat a set of words. Also, the outcomes tend to be unnecessarily long.

Discussion

In this project, we implemented HRED and DRL-SEQ2SEQ model with the OpenSubtitle Dataset. Unlike the results presented in the original paper of DRL-SEQ2SEQ (**?**), our mutual information model generated poor responses and we could see that the model quickly diverges to wrong outcomes during training. There are several possible reasons for this poor performance. First, we did not use the curriculum strategy, but the strategy may play a critical role in RL learning since the reward functions do not perfectly describe quality of responses. Second, we used 0.5 for the learning rate of

Table 2: Backward Seq2Seq Model Examples

Good	**human:**	i do not know man
	bot:	what do you think
	human:	that s amazing
	bot:	wow
	human:	that works best for me
	bot:	i will be right back
	human:	I love you too
	bot:	i love you
	human:	i know i am hot
	bot:	i am a little nervous
	human:	i am done with you
	bot:	i am a fucking bitch
Bad	**human:**	i am going to go to the bathroom
	bot:	i am going to go to the bathroom
	human:	this is weird
	bot:	i am going to be a great man
	human:	okay
	bot:	i am going to go to the bathroom
	human:	you look gorgeous
	bot:	i am going to go see the kitchen
	human:	fine
	bot:	i am fine

Table 3: Seq2Seq Model Examples

human:	this is bad
bot:	this is a joke
human:	what joke
bot:	i do not know
human:	what are you doing
bot:	nothing
human:	you are a loser
bot:	you are a real joke you know
human:	what is a real joke
bot:	i do not know
human:	how old are you
bot:	eleven
human:	you are so young boy
bot:	i am sorry
human:	it's totally okay
bot:	i am sorry
human:	what are you up to this weekend
bot:	i am sorry
human:	where are you from
bot:	where are you going
human:	i am going home
bot:	i am going to get you out of here
human:	thank you but where
bot:	what is this
human:	nothing
bot:	i am sorry

SGD (the default value in the tensorflow seq2seq model), but it may have been too large because the performance changed very quickly. Lastly, not using the baseline strategy may also have affected the performance since it is known that the strategy reduces the variance of PGO.

Table 4: Mutual Information Model Results

The highest probability response among actions in one batch after 10 steps
bot: i am flattered flattered truancy truancy truancy truancys truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy truancy soviets soviets soviets soviets soviets soviets soviets soviets soviets
The lowest probability response in one batch after 10 steps
bot: i am

Table 5: Mutual Information Model with additional reward/penalty

The highest probability response among actions in one batch after 20 steps
bot: i have loveyou
The lowest probability response in one batch after 10 steps
bot: yeah

Table 6: Mutual Information Model with additional reward/penalty after 540 steps

human:	how old are you?
bot:	000 toppy * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid *
human:	how are you doing?
bot:	fiancee pyramid * didit * didit * didit * didit * didit * didit * didit * didit * didit * didit * didit * didit *
human:	okay?
bot:	fiancee pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid * didit pyramid *