[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 (Ilya Sutskever 2014). Vinyals et al. applied this sequence to sequence framework to conversational modeling and their model predicted the next sentence given the previous sentence (Vinyals and Le 2015). 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. (Sordoni 2015), to the dialog domain (Serban 2015). 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) (Li 2016). 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 (Serban 2015) 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 OpenSubtitle Dataset.

Hierarchical Recurrent Encoder Decoder

In this section explain the hiearchical recurrent neural net (HRED) model proposed by (Serban 2015) and (Sordoni 2015), the problem they posed and how HRED solves these problems. In the original paper by (Sordoni 2015), HRED was proposed as a model that could generate context-senstitive queries, then (Serban 2015) adapted it to train an end-to-end history-aware conversation model.

Recurrent Neural Net

Now we wish to review recurrent neural net (RNN) in some detail. We denote a dialogue as a sequence of M utterances U_m: $D = \{U_1, \ldots, U_M\}$ between two speakers, each utterance contains N_m tokens: $U_m = \{w_{m,1}, \ldots, w_{m,N_m}\}$. ($Serban2015$) allowed the random variable $w_{m,n}$ to range over the vocabular and "speech acts", although in our case we only considered words. Next they defined a distribution P with parameter θ over the set of all possible dialogues of any length, and factorized P by:

$$\begin{aligned} P_\theta(U_1, \ldots, U_M) &= \prod_{m=1}^{M} P_\theta(U_m | U_{<m}) \\ &= \prod_{m=1}^{M} \prod_{n=1}^{N_m} P_\theta(w_{m,n} | w_{m,<n}, U_{<m}), \end{aligned}$$

where $U_{<m} = \{U_1, \ldots, U_{m-1}\}$. In other words, the conditional probability of the current word is only a function of previous words in the utterance and previous utterances. Next, (Serban 2015) represents P_θ with:

$$P_\theta(w_{n+1} = v | w_{\leq n}) = \frac{\exp(g(h_n, v))}{\sum_{v'} \exp(g(h_n, v'))},$$

where $h_n \in \mathbb{R}^{d_h}$ is a hidden state computed by a recurrent neural net:

$$\begin{aligned} h_n &= tanh(Hh_{n-1} + I_{w_n}) \\ g(h_n, v) &= O_{w_n}^T h_n, \end{aligned}$$

where $I \in \mathbb{R}^{d_h \times |V|}$ is the input word embeding. Again note that the hidden state is only a function of the past.

Hierarchical Recurrent Neural Net

HRED augments the basic RNN model by explicitly learning a transition function over the hidden dynamic of the conversation or "session". Serban and Sordoni learned this transition function using a session level RNN. Specifcially, this RNN predicts the next utterance by:

$$P(U_m|U_{<m}) = \prod_{n=1}^{N_m} P(w_n|w_{<n-1}, U_{m-1}) = \frac{\exp(o_v^T w(d_{m,n-1}, w_{m,n-1}))}{\sum_k \exp(o_k^T w(d_{m,n-1}, w_{m,n-1}))},$$

where $w(d_{m,n-1}, w_{m,n-1}) = H_o d_{m,n-1} + E_o w_{m,n-1} + b_o$, so that the next utterance is a function of all pervious utterances. In summary, in HRED each utterance is a function of the previous utterance, and each word in the utterance is a function of all previous words in the utterance. A session level RNN learns the transition between utterances vector, and sentence level RNN learns the transition between words.

Learning

HRED is trained end-to-end by maximizing the log-likelihood of the whole session S:

$$Loss(S) = \sum_{m=1}^{M} logP(U_m|U_{<m}) = \sum_{m=1}^{M}\sum_{n=1}^{N_m} logP(w_{m,n}|w_{m,<n}, U_{<m}).$$

We train the model by dividing the data set into sessions with four utterances each, the vocabulary size was set to be 50,005, and the maximum sentence length was 50 tokens.

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 (Li 2016).

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 (Li 2016). 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:

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

where $p_{seq2seq}^{backward}$ 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 (Williams 1992) 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.3). 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.3. The gradient is estimated using the likelihood ratio trick in PGO:

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

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. (Marc' Aurelio Ranzato 2016) 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

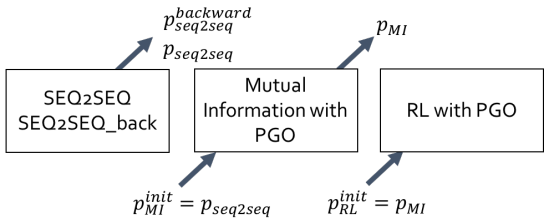

Figure 1: Training Procedure of the proposed Reinforcement Learning Model

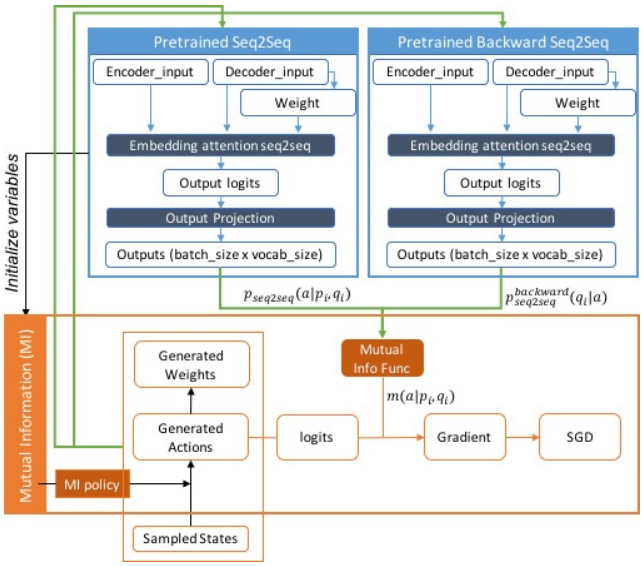

Figure 2: SEQ2SEQ and MI model Code Architecture

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

Table 2: Example Conversation in the OpenSubtitles corpus

A:	a proud rebellious son who was sold to living death in the mines of libya before his thirteenth birthday
B:	there under whip and chain and sun he lived out his youth and his young manhood dreaming the death of slavery 2 000 years before it finally would die
A:	back to work
B:	get up spartacus you thracian dog
A:	come on get up
B:	my ankle my ankle

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.2. 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.

Instead, we used the OpenSubtitles corpus (http://www.opensubtitles.org/) for both HRED and DRL. The OpenSubtitles corpus is a bilingual parallel corpus composed of movie subtitles. Originally constructed for machine translation, it has 20,400 files with 149.44 million tokens and 22.27 million sentence fragments. We used the 28 documents with English counterparts. The corpus is preprocessed by removing all ASCII symbols and adding consecutive speaker turns in an ad-hoc manner, that is to say we defaulted the first line in the script to be that of speaker one, the second line to be speaker two, and so on. Next, the corpus was normalized by case-folding, white space stripping, and converting all tokens to lower cases. Proper nouns and numbers were kept as is.

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.

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.2): 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.5 are computed. According to the paper (Li 2016), stochastic gradient descent (SGD) was applied to the objective function 4. 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: Backward SEQ2SEQ

The training results of the backward SEQ2SEQ model after 109000 steps are presented in Table.3. 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.4. 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".

Table 3: 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

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 5. 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.6 shows an example of after 20 steps and the Table.10 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.

Results: HRED

In this section we discuss the results for HRED. It was trained for 5 days on a GeForce GTX TITAN X GPU. The results after five hours are displayed in table 8. We keep the example brief and show an elucidating exmaple: after five

Table 4: 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

Table 5: 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 truancy truancy soviets soviets soviets soviets soviets soviets soviets soviets soviets soviets
The lowest probability response in one batch after 10 steps
bot: i am

Table 6: 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

hours the model appears to have acquired a basic grammar, but have a tendancy to output repeated words. Next we direct our attention to table 8, where we show the results after training for 72 hours. We observed a marked decrease in sentences with repeated words, and the table shows one example of a response that appears to have some "semantic" relationship with the question asked. We ultimately trained the models for 5 days, and the results are displayed in table 10. In general, the results after 5 days do not appear qualitatively better than results after just 72 hours. In particular, we see no evidence that HRED is tracking "some" conversation state.

Table 7: 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 *

Table 8: HRED after 5 hours

human:	to be quite honest it is not very clear for me either?
bot:	what let s a bitch and i m know you s gone black to you in you in you to you to you out of it to black to move to me *

Table 9: HRED after 5 hours

human:	do you have any weapons explosives. any history of violent conduct
bot:	a very interracial couples you re not gonna die you that s complicated

Table 10: HRED after 5 hours

human:	i can t remember anything
bot:	i got where the target and make preparation on froebe in jill
human:	i m a guinea pig
bot:	morning the problem your brutality

Discussion

In this project, we implemented HRED and DRL-SEQ2SEQ model with the OpenSubtitle Dataset. We trained HRED for five days but saw no dramatic qualitative improvements after two day. Futhermore, there is no qualitative evidence that HRED is keeping track of conversational state. There are several possible reasons for this. First the data set may not have been appropriately preprocessed for our task, looking at the last three sentences of table 2, it is clear that the second to last two sentences are spoken by the same person. Future

work would certainly need to be much more careful in their annotation of speakers if they choose to use this data set. Furthermore, since HRED is hypothesized to improve chatbot performance by keeping track of a session state, future data sets should ensure some session state exists by inspection. For example looking through the movie corpus, often times it is not clear there is such a state, especially if there are narrator interjections in the script, or scene cuts. Finally, the implementation of HRED may also limit the model's ability to learn full conversations. Since the current implementation of HRED assumes the entire session must be passed in as one vector, in practice we had to limit our sessions to four turns each, and each utterance could be at most 50 tokens long. This is a very stringent criteria in practice, since a conversation with explicit state, ie: greeting, middle and finally the good-bye is certainly longer than four turns. On a more general note, the notion that there could be enough data to place a distribution over the set of all conversations of arbitrary length may be unrealistic unless we are considering the most stringent definition of "conversation" possible. Restricting HRED to a limited domain (ie by topic) may alleviate such issues.

Now we turn our attention to DRL-SEQ2SEQ, unlike the results presented in the original paper of DRL-SEQ2SEQ (Li 2016), 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 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.

References

Ilya Sutskever, Oriol Vinyals, Q. V. L. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, 3104–3112. Association for Computational Linguistics.

Li, J. 2016. Deep reinforcement learning for dialogue generation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 1192–1202. Association for Computational Linguistics.

Marc' Aurelio Ranzato, Sumit Chopra, M. A. 2016. Sequence level training with recurrent neural networks. In *International Conference on Learning Representations*.

Serban, I. V. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Association for the Advancement of Artificial Intelligence*.

Sordoni, A. 2015. A hierarchical recurrent encoder-decoder for generative context-aware query suggestion. In *Conference of Information Knowledge and Management*.

Vinyals, O., and Le, Q. V. 2015. A neural conversational model. In *Proceedings of the 31st International Conference on Machine Learning*.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8(3-4):229–256.