Chatbot Based on Movie Dialogue

Shuo Liu, Jiarong Yu, Yi Ding, Yu Xiao
Department of Computer Science
Georgetown University
Washington, D.C.
{sl1539, jy576, yd137, yx151}@georgetown.edu

***Abstract*—Building a chatbot based on movie dialogue is a common but challenging NLP task. Our goal is to generate movie dialogue styled responses to random questions. There are several ways to approach this task. In this project, we experimented with both retrieval-based model and generative model. In addition to these models, we tried to use the policy gradient method with pre-defined reward functions introduced in [1] to improve the the quality of response. We studied possible ways of evaluating performance of chatbots. We then compared the results of chatbots trained using these different models.**

Code is available on https://github.com/dy11/ChatBot

***Index Terms*—chatbot, generative language model, NLP**

I. Introduction

We built a Chatbot based on the movie dialogs dataset to produce interactive conversations and we mainly experimented with three models. We started with the retrieval-based model which calculated sentence score using pre-trained word2vec model and then retrieving similar sentences (high similarity score) from dataset. Then, we built the generative model, Sequence to Sequence (SEQ2SEQ), with Keras framework. The SEQ2SEQ model contains two LSTM layers acting as the encoder and the decoder. In addition, based on the SEQ2SEQ model introduced in Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation, we improved the model using reinforcement learning, in particularly, the policy gradient method with pre-defined reward functions. We evaluated and compared the results of different models. We adapted recall, precision and F1 score to evaluate the retrieval-based model. For generative model, evaluation was more challenging. We used BLEU score to evaluate the general performance of our models and manually evaluated a small portion of the results to get a more precise result.

II. Related Work

The retrieval-based model is the traditional way to build the chatbot. There are many different ways to retrieve similar sentence from dataset. Mueller and Thyagarajan [2] implemented a model that applied to assess semantic similarity between sentences, where they exceed state of the art, outperforming carefully handcrafted features and recently proposed neural network systems of greater complexity. Yan et al. [3] showed an architecture for IR-based chatbot with unstructured documents. This paper also showed that leverage unstructured documents, instead of Q-R pairs, to respond to utterances. And a learning to rank model with features designed at different levels of granularity is proposed to measure the relevance between utterances and responses directly.

The SEQ2SEQ model is one of the most effective way to achieve chatbot functionality. It is proposed by Sutskever et al. [4] that the SEQ2SEQ model could effectively handle problems whose inputs and targets are encoded with vectors of different dimensionality. The idea is to use one LSTM to read the input sequence, one time step at a time, to obtain large fixed-dimensional vector representation, and then to use another LSTM to extract the output sequence from that vector. Cho et al. [5] also supports that the RNN Encoder-Decoder model could learn a semantically and syntactically meaningful representation of linguistic phrases.

Deep reinforcement learning methods are developing fast and drawing more and more attention from researchers of different application fields. Li et al. [1] studied possibility of applying one of the deep reinforcement learning methods on the task of dialogue generation. The model proposed in [1] specifically addresses problems of previous dialogue generation models: (1) the answer from machine is a response only to the previous user entered sentence, rather than the dialogue history; and (2) the method generates answers that grammatically right yet "dull", *e.g.*, "I don't know what you are talking about" or "I don't understand". Deep reinforcement learning methods can train the model to optimize a pre-defined reward function. By defining reward functions penalizing these two issues above, we can train a dialogue generation model that is able to avoid these problems. Also, reinforcement learning can evaluate the action of a model in an "episodic" point of view. The target of training can be set to maximize returns (sum of rewards of steps) in an episode, in this case, a dialogue, instead of maximize reward of a single step, in this case, one sentence as an answer.

For evaluation, we can consider using BLEU Papineni et al. [6] to evaluate the Chatbot. And we also apply the evaluation methods from SQuAD [7].

III. Methodology

A. Retrieval-Based Model

For the retrieval-based model, we searched for the sentence from the dataset that is most similar to the input sentence and retrieved the corresponding answer as the output. For example, for the input sentence "Do you have a boyfriend?", the system would find the most similar question in the dataset "Do you have a girlfriend?", and return its correspond answer

"I am alone in the world". If a sentence does not have a response (*e.g.*, the last sentence in a conversation), the system would select the second most similar sentence as the output. To find the most similar sentence of the input sentence, we used Google pre-trained word2vec b5 [8] to calculate sentence vector and determined the distance between sentences. There are two ways to calculate sentence vector. The first way is to average all words in the sentence and the other way is to average words besides stopwords in the sentence. Then we calculated cosine distance between the input question and each sentence in the dataset.

Retrieval-based model performed well for daily conversations as our model is trained on the movie dialogues and similar sentences/questions can be found in the dataset. However, for rare sentences that never shown in the dataset, the retrieved-based model had limited effects. We adapted the Cornell movie-dialogs corpus as our dataset.

B. SEQ2SEQ *Model*

The SEQ2SEQ takes an input as a sequence of words (sentence or sentences) and generates an output sequence of words using the recurrent neural network (RNN) Sutskever et al. [4]. The encoder uses deep neural network layers and converts the input words to corresponding hidden vectors. Each vector represents the current word and the context of the word. Similarly, the decoder takes the hidden vector generated by encoder as input, its own hidden states and current word to produce the next hidden vector and finally predict the next word.

We used the Keras framework to build a LSTM-based SEQ2SEQ model in this project. As shown in Figure 1, there is an embedding layer that reduce dimensions, a LSTM layer serves as the encoder and a LSTM layer serves as the decoder.

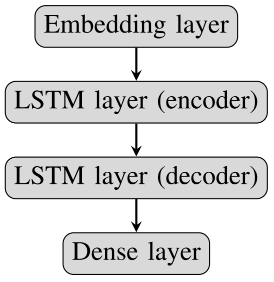

Fig. 1. Structure of a SEQ2SEQ model

C. Reinforcement Learning Method

In the reinforcement learning method, we adapt the dialogue generation model proposed in [1]. We consider the conversation is generated by two agents: one machine (our model) m and one user u talking with each other. During a conversation, a alternating sequence of sentences is generated by the two agents: u_1, m_1, u_2, m_2, …, u_i, m_i. Every sentence generated by the model is an action, which is taken according to a policy defined by an encoder-decoder language neural network.

The optimization goal of training is to maximize the expected future reward using policy gradient. Under this goal, it is possible to utilize a set of pre-trained parameters produced by the MLE objective as initialization, and further optimize the parameters toward the goal of policy gradient.

1) Action: An action a is a sentence as the response to the conversation to generate. If there is no limit to the length of a sentence, the action space is infinite.

2) State: A state here is denoted by the previous two dialogue turns $[m_i, u_i]$. The two sentences of a state is transformed to a vector representing the concatenation of these sentences as input of the LSTM encoder model.

3) Policy: A policy takes the form of an LSTM encoder-decoder (*i.e.*, $p_{RL}(m_{i+1}|m_i, u_i)$) and is defined by its parameters.

4) Reward: Each action a results in a reward r. The reward function consists of three major parts:

Ease of answering

A return generated by machine should be easy to respond to. On the other hand, dull responses are difficult to be answered, and may lead to the end of a conversation. We manually produced a set $\mathbb{S}$ of dull responses, and then use the negative log likelihood of responding to a generated sentence with a dull response. The reward function is defined as in 1.

$$r_1 = -\frac{1}{N_{\mathbb{S}}} \sum_{s \in \mathbb{S}} \frac{1}{N_s} \log p_{\text{seq2seq}}(s|a), \tag{1}$$

where $N_{\mathbb{S}}$ denotes the cardinality of $\mathbb{S}$ and N_s denotes the number of tokens in the dull response s. Notice that many of other dull responses that are not listed in $\mathbb{S}$ are likely fall into similar regions in the vector space computed by the model, therefore the system that is less likely to produce the listed sentences should also be less likely to generate other similar dull response.

p_{seq2seq} represents the likelihood output of SEQ2SEQ models, although this likelihood, learned based on the MLE objective, is different from $p_{RL}(m_{i+1}|m_i, u_i)$, learned by policy optimization.

Information flow

We would like our agent generate new response that contributes new information to the conversation. We therefore penalize semantic similarity between consecutive turns from a same agent. Let h_{m_i} and $h_{m_{i+1}}$ denote representations obtained from the encoder for two consecutive turns m_i and m_{i+1}. The reward is defined by the negative log of the cosine similarity between these representations:

$$r_2 = -\log \cos \frac{h_{m_i} \cdot h_{m_{i+1}}}{||h_{m_i}||||h_{m_{i+1}}||}. \tag{2}$$

Semantic coherence

For a chatbot, we also need the responses generated to be grammatically right and coherent. We consider mutual information between an action a and previous turns in the history to ensure the generated responses are coherent and appropriate:

$$r_3 = \frac{1}{N_a} \log p_{\text{seq2seq}}(a|m_i, u_i) + \frac{1}{N_{u_i}} \log p^{\text{backward}}_{\text{seq2seq}}(u_i|a),$$

where $p_{\text{seq2seq}}(a|m_i, u_i)$ denotes the probability of generating response a given the previous dialogue turns $[m_i, u_i]$, and $p^{\text{backward}}_{\text{seq2seq}}(u_i|a)$ denotes the backward probability of generating the previous dialogue turn u_i based on response a. $p^{\text{backward}}_{\text{seq2seq}}$ is trained in a similar way of standard SEQ2SEQ models, just with swapping of the inputs and outputs.

Finally, the reward function of an action a is the weighted sum of these three:

$$r(a, [m_i, u_i]) = \lambda_1 r_1 + \lambda_2 r_2 + \lambda_3 r_3, \quad (3)$$

where $\sum \lambda_i = 1$.

5) Training Procedure: The training of this model consists of two steps. First, we build a SEQ2SEQ model for initialization of the reinforcement learning model. The difference is the input of the model is also changed to a concatenation of two sentences.

After finishing training of the SEQ2SEQ model, we optimize our policy model p_{RL} by maximizing the expected future reward:

$$J_{RL}(\theta) = \mathbb{E}_{p_{RL}(\{a_i\})} \left[\sum_{i=1}^{T} R(a_i, [m_i, u_i]) \right], \quad (4)$$

where T is the number of turns of the conversation, and a_i is the i-th action taken by the model. The gradient updates is made by

$$\nabla J_{RL}(\theta) \approx \sum_i \nabla \log p(a_i|m_i, u_i) \sum_{i=1}^{T} R(a_i, [m_i, u_i]). \quad (5)$$

D. The Methods of Evaluation

To evaluate a generative model is still an unresolved problem. In this section, we are using several methods to evaluate the results.

1) BLEU with smooth methods: The method is based on the N-gram model. For each N-gram unit, the denominator is the number of units in the candidate but the numerator is quite different. For a unit item in the candidate, the count in the numerator is counted as $count = \min(count, \max(\text{reference count}))$. Where count means the number of occurrences of item in the candidate and the reference count means the number of occurrences of item in a reference. Since there are N N-grams units, and the accuracy decreases exponentially with the increase of N, the geometric mean weight is used, and apply the sentence brevity penalty.

If we consider that the reply of an input sentence is a kind of translation of it and make an assumption that different version of replies has the almost same meaning.

2) Recall, precision, F_1: The method was provided by SQuAD [7] - F_1 score. It allows some variances between the generated answer and the gold answer. This F_1 score evaluates the harmonic mean of precision and recall. However, there is nothing we can do when the generated answer has not any connection with the gold answer. And this happens a lot in our project.

3) Common words: The method was modified by exact match evaluation which is also provided by SQuAD [7]. It is a binary measure which is the percent of how the answer we generated similarly to the gold one. But it quite meaningless in our project since the score is nearly 0. Then I try to use common words as an evaluation method instead of using exact match.

4) Human evaluation: We also evaluated the result by manual inspection. We classify the results into 4 categories. There are perfect, fluent English but not match to question, related with question but not fluent English and gibberish.

For retrieval model and generative model, all of those methods have been applied. For the reinforcement learning model, due to the goal of it, generating a longer conversation, is different from the first two models, generating a relevant answer to an input, we only did human evaluation on its selected results.

IV. EXPERIMENTS

A. Dataset

The Cornell movie-dialogs dataset [9] contains a large metadata-rich collection of fictional conversations extracted from raw movie scripts. There are 220,579 conversational exchanges between 10,292 pairs of movie characters and involves 9,035 characters from 617 movies. The dataset contains 83,098 conversation which can split to 304,713 query-response pairs.

The dataset contains 83,098 conversation which were splitted to 221,616 query-response pairs. After removed long sentences (longer than 20 tokens) and short sentences(shorter than 2 tokens), there were 138332 query-response pairs left for training. In addition, we found out that the 5000 most frequent words each occurred about more than 10 times in the vocabulary, thus we only encoded 5000 most frequent words to distinct index.

B. Data Preprocessing

There are mainly three steps of preprocessing. First, we broke long conversations between movie characters into a series of question and answer pairs based on the sentence tags. Second, we cleaned dialogues by removing the apostrophe in contractions, punctuations, and tokenized sentences. Third, we removed long sentences (longer than 20 tokens) and short sentences(shorter than 2 tokens). In addition, for SEQ2SEQ model, we encoded 5000 most frequent words as distinct index and the rest of infrequent words as 1, and paddled word sequences.

C. Experiment Settings

1) Retrieval model: We used the Google pre-trained word2vec model which contains 300-dimensional vectors for 3 million words and phrases trained on Google news dataset. Therefore we could easily calculate cosine distance between any two words and we could also generate sentence vector by using this word2vec model.

2) SEQ2SEQ *model:* To train the model, we used mini batch size (64) with 256 hidden units to train for 100 epochs. We encoded 5000 most frequent words with distinct index as major features. We chose '`categorical cross entropy`' as the loss function and '`adam`' as the optimizer. After the training, the model could generate good responses for short input. However, for long questions, the model could not always yield semantically related answers. Since the training time was extremely long, we only fine-tuned on batch size and number of epochs.

3) Reinforcement learning model: We first train a word2vec word embedding using our dataset, with a length of vector of 300. Then we train a SEQ2SEQ model. Because this SEQ2SEQ model is actually different from the SEQ2SEQ model used above, we cannot use the previous parameters. Due to limited time, we only trained this model for 30 epochs. Then we trained the reinforcement learning model for another 30 epochs to get the final RL model.

D. Empirical Results

In this section, we evaluate the experimental result with 2000 input sentences in our dataset by our evaluation metrics and with 20 input sentences in our dataset using human evaluation. The results are shown in Table I and II. The four results from SEQ2SEQ model are trained using 4 different parameter settings: (1) batch size = 128, number of epochs = 1; (2) batch size = 128, number of epochs = 30; (3) batch size = 64, number of epochs = 60; and (4) batch size = 64, number of epochs = 100, where number of hidden units = 256 and maximum vocabulary size = 5000 are fixed.

TABLE I
AUTOMATIC EVALUATION RESULTS ON THE RETRIEVAL MODEL AND THE SEQ2SEQ MODEL.

	BLEU	recall	precision	F_1	common words
retrieval	0.067	0.597	0.545	0.425	0.911
SEQ2SEQ1	0.003	0.023	0.032	0.023	0.141
SEQ2SEQ2	0.014	0.166	0.172	0.137	0.623
SEQ2SEQ3	0.016	0.161	0.207	0.143	0.652
SEQ2SEQ4	0.008	0.058	0.120	0.068	0.041

The results of our generative model are not good. There are several possible reasons: (1) According to our evaluation methods, there is only one correct answer for each question. But this is not true. Moreover, our correct answer is not that correct. (2) Since our data comes from movie dialogues, the input sentence will contain much more information than the sentence itself. Those information can not be caught by our model.

TABLE II
HUMAN EVALUATION RESULTS ON THE RETRIEVAL, SEQ2SEQ, AND RL MODEL.

	Perfect	Fluent	Match	Gibberish
SEQ2SEQ3	30%	25%	0%	45%
SEQ2SEQ4	30%	35%	15%	20%
RL	0%	30%	5%	65%

The performance of retrieval model is better than generative model in our evaluation metric. One of the possible reason is that all of the output of the retrieval model are from the movie dialogues in the same dataset. And dialogues are similar to dialogues than the sentence we generated. To eliminate the similarity, we need a relatively standard test set for automated testing.

Besides, the reason I did not manually evaluate the first and second version of SEQ2SEQ models is that the most of their output are sentences like "I do not know" and "I am sorry". In our evaluation criterion, they can answer almost all the questions.

V. CONCLUSION

In this course project, we build prototypes of chatbot using different models with the same movie dialogue dataset as training data. Our aim is to enable the chatbot generate casual response to user's input sentence. The three models we tried are a retrieval model, a supervised learning model SEQ2SEQ, and a reinforcement learning based model. We also studied on how to evaluate results of these model. We conducted automatic evaluation and human evaluation on the results from the three models using different training parameters, and compared their performance. The retrieval model outperforms other models under BLEU scores, yet human evaluation indicates the SEQ2SEQ model can produce better response to given questions. The reinforcement learning model can produce longer conversations with certain connection among sentences, yet its performance can still be improved given more training time. Based on different usage, all the three methods can be good candidate to build a chatbot in real-world.

VI. LIMITATION AND FUTURE WORK

First, the dataset itself caused some confusions as the movie dialogues are heavily based on the context. Different characters, movie genre, era would result in different conversational styles. However, our model could not capture these differences.

For Retrieval-Based model we could optimize similar sentence selection by training a LSTM model to select more suitable sentence for input query. Since there may response multiple sentences, we could apply an architecture for Multi-turn Response Selection [10].

Besides, We have made some assumptions when we were evaluating the generative model. If we considered the generated sentence is a translation of the input. Then for using BLEU to evaluate our model, we need to make an assumption that different versions of reply have almost the same meaning. We all know that both of the assumptions are incorrect. But with those assumptions, we can at least evaluate the model automatically.

An interesting unsolved problem about our project is on how to measure the quality of models. We were trying to evaluate our model manually and by computing the evaluation metric. But either of them is not good enough. From my perspective, it is impossible to evaluate the model like this without a standard test set. If we want to judge a result, we need to know what is right. The biggest backfire of our evaluation part is our gold label is not that gold. Thus, with a standard test set, our evaluation metric will make more sense. Just like the test set in [7].

In our experiments, we have already know that the training of a deep reinforcement model for language generation might take a rather long period of time to train. With suitable equipment, we should try different parameters and epochs to get a better performance. Also, though the authors of [1] stated in their work that policy gradient is more suitable method for training rather than Q-learning, it would still be interesting to see if other reinforcement learning methods can improve performance, such as modifications to policy gradient like policy gradient with baseline or actor-critic methods.

References

[1] J. Li, W. Monroe, A. Ritter, M. Galley, J. Gao, and D. Jurafsky, “Deep reinforcement learning for dialogue generation,” *arXiv preprint arXiv:1606.01541*, 2016.

[2] J. Mueller and A. Thyagarajan, “Siamese recurrent architectures for learning sentence similarity,” in *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[3] Z. Yan, N. Duan, J. Bao, P. Chen, M. Zhou, Z. Li, and J. Zhou, “Docchat: An information retrieval approach for chatbot engines using unstructured documents,” in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, vol. 1, 2016, pp. 516–525.

[4] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks,” in *Advances in neural information processing systems*, 2014, pp. 3104–3112.

[5] K. Cho, B. Van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, “Learning phrase representations using rnn encoder-decoder for statistical machine translation,” *arXiv preprint arXiv:1406.1078*, 2014.

[6] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, “Bleu: a method for automatic evaluation of machine translation,” in *Proceedings of the 40th annual meeting on association for computational linguistics*. Association for Computational Linguistics, 2002, pp. 311–318.

[7] P. Rajpurkar, R. Jia, and P. Liang, “Know what you don’t know: Unanswerable questions for squad,” *arXiv preprint arXiv:1806.03822*, 2018.

[8] “Google pre-trained word2vec model.” [Online]. Available: https://code.google.com/archive/p/word2vec/

[9] C. Danescu-Niculescu-Mizil and L. Lee, “Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs.” in *Proceedings of the Workshop on Cognitive Modeling and Computational Linguistics, ACL 2011*, 2011.

[10] Y. Wu, W. Wu, C. Xing, M. Zhou, and Z. Li, “Sequential matching network: A new architecture for multi-turn response selection in retrieval-based chatbots,” *arXiv preprint arXiv:1612.01627*, 2016.