# Friendly Companion Chatbot

Shailendra Kumar Gupta
2016CSB1059@iitrpr.ac.in
Indian Institute of Technology
Ropar, Punjab

Shreyanshu Shekhar
2016CSB1060@iitrpr.ac.in
Indian Institute of Technology
Ropar, Punjab

Mattela Nithish
2016CSB1042@iitrpr.ac.in
Indian Institute of Technology
Ropar, Punjab

Abhinav Jindal
2016CSB1026@iitrpr.ac.in
Indian Institute of Technology
Ropar, Punjab

## ABSTRACT

A chatbot implementation using the combined technique of s2s and RL models. RNN s2s model give semantically correct results but lack the capability to make the conversation more interactive which is improved by policy gradient RL technique after training RNN s2s model and using that for the RL implementation.

## KEYWORDS

RNN, LSTM, reinforcement learning

**ACM Reference Format:**
Shailendra Kumar Gupta, Shreyanshu Shekhar, Mattela Nithish, and Abhinav Jindal. 2019. Friendly Companion Chatbot. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

## 1 PROBLEM STATEMENT AND MOTIVATION

To build a friendly companion chatbot that understands the user's query and responds to them accordingly. There are lot of reason behind building a chatbot as project.

- Automated customer services
- Use of human resource in other qualitative tasks
- Easy to use
- Very low development cost
- Cost and time efficient

## 2 LITERATURE OVERVIEW

We have used three literatures as guides for this project. Following subsections explains the basic work carried out in these literatures.

The paper [1] by Ilya Sutskever et. al. implements the main task of translation from English to French using sequence to sequence model. The method uses a multilayered Long Short-Term Memory (LSTM) to map the input sequence to a vector of a fixed dimensionality, and then another deep LSTM to decode the target sequence from the vector. The WMT'14 dataset was used to train the model. They used BLEU(bilingual evaluation understudy) score to evaluate the quality of translation from English to French. The LSTM also learned sensible phrase and sentence representations that are sensitive to word order and are relatively invariant to the active and the passive voice. At last they tried to train the model with reversed source input(word by word), which has improved the LSTM's performance because doing so introduced many short term dependencies between the source and the target sentence which made the optimization problem easier.

The next paper [2] by Subhashini Venugopalan et. al. uses the sequence to sequence model to map a video with a sentence description. They used LSTM for sequence to sequence model, which have demonstrated state-of-the-art performance in image caption generation.They used M-VAD and MPII-MD dataset containing movie descriptions. They applied convolutional neural network(CNN) to input images and provided the output of the top layer as input the LSTM unit. The target output sequence of words are represented using one-hot vector encoding. Quantitative evaluation of the models are performed using the METEOR metric.

The last paper [3] by Jiwei Li et. al. focuses on improving the responses by the chatbot agent by using reinforcement learning techniques. The basic neural network models offer great promise for generating responses for conversational agents, but tend to be shortsighted, predicting utterances one at a time while ignoring their influence on future outcomes. This paper tries to integrate these goals by applying deep reinforcement learning to model future reward in chatbot dialogue. The model simulates dialogues between two virtual agents, using policy gradient methods to reward sequences that display three useful conversational properties: informativity, coherence, and ease of answering (related to forward-looking function). The model was evaluated on diversity, length as well as with human judges, showing that the proposed algorithm generates more interactive responses and manages to foster a more sustained conversation in dialogue simulation.

## 3 PRELIMINARIES

In this section we will talk about the dataset and the preprocessing done on data before feeding the them into the model.

### 3.1 Definitions

One hot vector is a vector representation of a word of length equal to size of vocabulary. It contains 1 at the index represented by that word and 0 at all other indices. Policy gradient is a RL technique in

*Conference'17, July 2017, Washington, DC, USA*
© 2019 Association for Computing Machinery.
ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

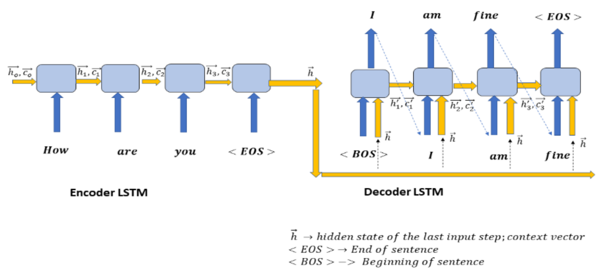


**Figure 1: Sequence to sequence model:**
**Source** https://subscription.packtpub.com/book/big_data_and_business_intelligence/9781788996921/8/ch08lvl1sec98/building-a-sequence-to-sequence-model

which a given policy is used to evaluate the utility values for states which is then used to update the policy and so on.

## 3.2 Dataset

We have used Cornell Movie-Dialogs Corpus which is a collection of fictional conversations extracted from movie scripts. This dataset contains around 3 Lakhs utterances. Two main files that are used from the dataset is 'movie_lines.txt' containing the utterances from movies and 'movie_conversations.txt' containing list of lines which are related as single conversation in movie scripts.

## 3.3 Preprocessing

From 3 Lakhs utterances we have selected around 1,60,000. First we extracted list of all the lines that are related. This information is present in movie_conversation.txt file. This file only has line codes. The information of line code and text is present in movie_lines.txt file. So we extracted all the line codes and their corresponding text. After this we prepared the dictionary of words using tokenizer object of Keras.preprocessing. This object has all word2index and index2word information. For every consecutive pair lines that are related will be a question and answer respectively. Using conversations list and lines, We prepared question and answer set and converted them into sequence of numbers where every word is replaced by its corresponding index which is stored in tokenizer object. We also padded the data to length of max(sentence) present in the dataset. We divided question and answer pair into training and validation instances with split ratio=0.9.

## 4 PROPOSED METHODS

In this section we will discuss about the different methods we have used to build the chatbot. First we will talk about sequence to sequence model to build a simple neural network based chatbot. Next we will talk about reinforcement learning methods helped to improve the responses generated by the chatbot.

## 4.1 Sequence to Sequence Model

This section talks about sequence to sequence model and some improvements in the sequence to sequence model.

*4.1.1 Basic Encoder and Decoder model.* This is a basic sequence to sequence model we developed to build the chatbot. This model include one encoder network that takes a sequence as input and generates a context vector that is used by the decoder network to generate the output sequence. The input sequence that is supplied to encoder is generated using the input sentence and vocabulary dictionary. The decoder produces one hot vector encoding as the output which is interpreted to generate the output sequence and further output sentence using vocabulary dictionary.

*4.1.2 Basic model with Embedding.* In the previous basic model the size of input word vector supplied to encoder for processing is equal to the size of number of words in vocabulary which is quite large due to which the training time of the model is very high, hence we used an embedding layer in the encoder network. The embedding layer embeds the information of each large sized word vector into a small sized word vector which will be fed as new input to the encoder reducing the training time of the model. The other reason of using embedding layer is to pack the information into small vectors in order to improve the training because the vocabularies are vast and a given word or document would be represented by a large vector comprised mostly of zero values which are somewhat not good for training a model. Hence the embedding layer is included in both encoder and decoder network.

*4.1.3 Methods to improve the models.* In this section of the document we will discuss some methods which are used to improve the performance of the model by using intelligence techniques while training the model.

- Teacher forcing

It is a method for training RNN models quickly and efficiently. It uses the output of the previous timestamp as one of the input current of decoder LSTM cell.

- Bucketing

As we discussed in the preprocessing techniques that each sentence will be converted to a fixed length vector which has the size equal to the length of the largest sentence in the dataset. Now if we use this representation for training and length of largest sentence is around 30 then there will be a lot of padding in case of sentences which are very small i.e. having 3-5 words. So the main information of the sentence is diluted by the padded zeros, hence making the agent to generate improper responses. So in order to overcome this problem we used the concept of bucketing in which sentences are divided into buckets such that each bucket have a attribute length which decides what length of sentences can be kept in that bucket. Also corresponding to each bucket an encoder-decoder model is built preserving the parameters except the length of LSTM network(which depends on the length attribute of each bucket). In this way agent is trained on different length of sentences, overcoming the problem of padding in sentences.

- Attention Mechanism

While training the model the encoder encodes the information of the input sequence into a fixed length context vector which is used by the decoder network to decode the output sequence. So in cases where the length of the sentence is very large the information encoded in the context vector will not be good enough for decoding because as the sentence is very large and each word is given same importance, so the words which do not have any semantic importance in the sentence will also contribute to the meaning of sentence equally to those words which are more important in conveying the semantic of the sentence. In such cases the main information of the sentence will get dilute due to these unimportant words, hence the agent will not be able to generate proper responses. So to overcome this problem in our model we used the concept of Attention Mechanism in which weights are assigned to each value of the context vector depending upon their importance in the sentence. These weights are also trained with the model.

## 4.2 Reinforcement Learning

As already pointed reinforcement techniques are used to model the agent in order to overcome the problem of short-sightedness in sequence to sequence model, to predict future outcomes, to produce positive and meaningful responses, etc. to improve the agentâĂŹs responding sense. We will be using policy gradient method instead of the Q-learning approach because Q-learning because we can initialize the encoder-decoder RNN using parameters from our s2s model that already produce some acceptable results, Q-learning directly estimates the future expected reward of each action, which can differ from the s2s model by great magnitude, thus making those parameters inappropriate for initialization. The basic method for training our RL is simulating 2 agents to converse with each other with the use of parameters explained below.

*4.2.1 State.* A state **s** of the RL model is the tupple [P(i),Q(i)], where P(i) is the response of one agent and Q(i) the response of the other.

*4.2.2 Action.* An action **a** for a state [P(i),Q(i)] is defined as the response of the first agent i.e. P(i+1) moving it to state [Q(i),P(i+1)].

*4.2.3 Policy.* The policy for policy gradient technique is taken as the LSTM encoder-decoder model defined by its parameters.

*4.2.4 Reward.* The reward function for our RL technique is taken as the linear combination of 3 factors.The rewards include ease of answering, information flow and semantic coherence. The weights assigned to these are 0.25,0.25.0.5 respectively due to more significance of the third factor. The factor as explained below.

*4.2.5 Ease of Answering.* The responses given by the agents should be easy to respond. Therefore we introduce a factor in the reward function that minimizes the probability for a dull response. We penalize the response that leads the other agent to give the dull responses. Dull response refers to statements like ’I don’t know’, ’I am not sure’, etc. These responses doesn’t keep the conversation going therefore, we need to avoid them.

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

$P_{seq2seq}(s|a)$ denotes likelihood output by s2s model. $N_S$ refers to the cardinality of dull responses i.e. the number of dull responses and $N_s$ refers to number of tokens in that response.

*4.2.6 Information Flow.* We does not want the same thing to keep on repeating in the conversation. So while training when we are making two agents to converse, we want them to contribute new information at each turn to keep the dialogue moving and avoid repetitive sequences. We therefore propose penalizing semantic similarity between consecutive turns from the same agent. To implement this in model we check the similarity between the encoder output at time i and i+1 by the first agent. The reward is given by the negative log of the cosine similarity between them as follows:

$$r_2 = -logcos(h_{p_i}, h_{p_{i+1}}) = -logcos\Big(\frac{h_{p_i} \cdot h_{p_{i+1}}}{\|h_{p_i}\|\|h_{p_{i+1}}\|}\Big) \tag{2}$$

Here $h_{p_i}$ refers to the encoder output for the $p_i$. Here we are negating the similarity between the 2 responses so that new information is contributed.

*4.2.7 Semantic Coherence.* Apart from the high rewarded response we also need that a response generated is adequate i.e. a response which has high reward but is grammatically incorrect or not coherent should be avoided. We therefore consider the mutual information between an action and previous turns in the history to ensure the generated responses are coherent and appropriate using the below formula:

$$r_3 = \frac{1}{N_a} logP_{seq2seq}(a|q_i, p_i) + \frac{1}{N_{q_i}} logP^{backward}_{seq2seq}(q_i|a) \tag{3}$$

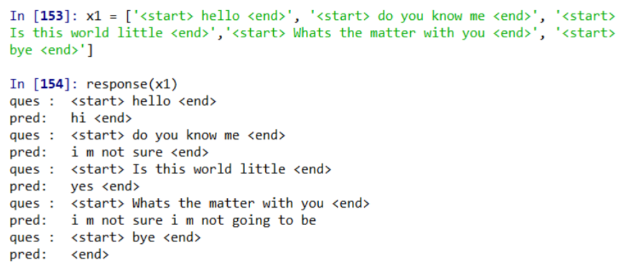
$P_{seq2seq}(a|q_i, p_i)$ refers to the probability of a response a given an previous conversation instant and $P^{backward}_{seq2seq}(q_i|a)$ refers to the backward probability of generating the previous dialogue utterance Q(i) based on response a.

## 5 EXPERIMENTATION

First we tried to use all the sentences in the dataset in training and testing. Since we need to have a fixed length of question and answer while training, we padded all the sequences to maximum length of sentences. Here we observe that the maximum length of sentences is around 300 and most the question and answers have length of around 8 to 30. If we consider only those question and answers whose length is around 10 we have very few question and answer pairs. So to maintain a balance between information loss due padding and number of question-answer pairs, We considered max length of sentence to be 30. Considering all the words in the dataset, The size of the vocabulary is more than 1,00,000. This value is very large. We experimented with the vocabulary size based on the occurrence of the words in the text. After that we also experimented on the number of latent dimensions that the embedding layer will have.

## 6 RESULTS AND DISCUSSION

The final model has latent dims of embedding as 256 as it was giving better response. Evaluating model on accuracy is not a very good idea as we were using exact match loss function while training the seq2seq model. We tested the model with some questions which are very often asked with a friendly bot. As we see dull responses are quite common in this and sometimes the results are totally unexpected. This is basically due to limited dataset and less epochs for training.



```
In [153]: x1 = ['<start> hello <end>', '<start> do you know me <end>', '<start>
Is this world little <end>','<start> Whats the matter with you <end>', '<start>
bye <end>']

In [154]: response(x1)
ques :  <start> hello <end>
pred:   hi <end>
ques :  <start> do you know me <end>
pred:   i m not sure <end>
ques :  <start> Is this world little <end>
pred:   yes <end>
ques :  <start> Whats the matter with you <end>
pred:   i m not sure i m not going to be
ques :  <start> bye <end>
pred:   <end>
```

**Figure 2: Output of sequence to sequence model**

## 7 CONCLUSION

We integrate the strengths of neural SEQ2SEQ systems and reinforcement learning for chat bot responses. We see that s2s produces semantically correct responses with appropriate responses for many queries but still fails to extend the conversation or make it more interactive. The frequency of dull responses is quite high for this model. We also expect to observe that RL improvises our s2s model significantly leading to much responsive and interesting chat as compared to just s2s model. The reward function for the RL plays a great role in producing these results.Despite the fact that our model uses very simple heuristics for capturing these global properties, the model generates more diverse, interactive responses that lead to a more sustained conversation.

## REFERENCES

[1] Sequence to Sequence Learning with Neural Networks. Ilya Sutskever, Oriol Vinyals, Quoc V. Le https://arxiv.org/abs/1409.3215
[2] Sequence to Sequence – Video to Text. Subhashini Venugopalan, Marcus Rohrbach, Jeff Donahue, Raymond Mooney, Trevor Darrell, Kate Saenko https://arxiv.org/abs/1505.00487
[3] Deep Reinforcement Learning for Dialogue Generation Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, Dan Jurafsky https://arxiv.org/abs/1606.01541