Chatbot

By Cheng Yuan, Haoqing Sun, Zhen Liu
University of Central Florida

Abstract

A chatbot is a machine conversation system which interacts with human users via natural conversational language. This paper presents a program to learn from spoken transcripts of movies and Twitter's comments; we discuss the problems which arose during learning and testing. Two main goals were achieved from the automation process. One was to generate a better version chatbot by comparing models based on two types of datasets. The second achievement was the ability to learn a very large number of categories within a short time, saving effort and errors in doing such work manually.

1. Introduction

Human machine conversation as a technology integrates different areas where the core is language, and the computational methodologies facilitate communication between users and computers using natural language. [1]A related term to machine conversation is the chatbot, a conversational agent that interacts with users turn by turn using natural language [2]. Different chatbots or human-computer dialogue systems have been developed using text communication starting from ELIZA that simulates a psychotherapist [3], then PARRY which simulates a paranoid patient.

In this paper, the data sets are 'Cornell Movie-Dialogs Corpus' and conversation extracted from twitter.With these datasets, we trained the chatbot by recurrent neural network. Basic RNNs [4] are a network of neuron-like nodes, each with a directed connection to every other node. Each node has a time-varying real-valued activation. Each connection (synapse) has a modifiable real-valued weight. With the model RNN, we could be able to train the chatbot in order to make it works.

Natural language processing is also applied to this work. In the training session, first step is pre-processing of natural language which could include different varieties of human language. Second step is implementing language feature extraction method to get keywords. Final step is building neural network model based on keyword of training set.

By applying the model RNN with LSTM units to the chatbot, we expect it can reply meaningfully and clearly.

The remainder of the paper is structure as follows. In Section 2 we will elaborate on the construction of RNN and LSTM, also we will describe the external software we used, whilst in Section 3 we describe datasets we use and the dataset preprocessing. Section 4 will show the

result of our chatbot system by showing cross-entropy loss and testing conversation with our system and end with conclusion and further work in Section 5.

2. Background & External Software

There will be three main parts: third-party libraries we used, Recurrent neural networks(RNN), Long Short-Term Memory networks(LSTM).

2.1 External Software

The external software we used are Tensorflow and Tensorlayer. The purpose by using two of libraries is to implement LSTM deep learning method. The specific feature will be described in 2.2 and 2.3.

2.2 Recurrent Neural Networks

As we know, neural network is a powerful machine learning system which has been used in a huge amount of field. With the development of machine learning and neural network, we hope to make machine learn as human especially learn from past experience and memory. However, traditional neural network cannot finish this job well.

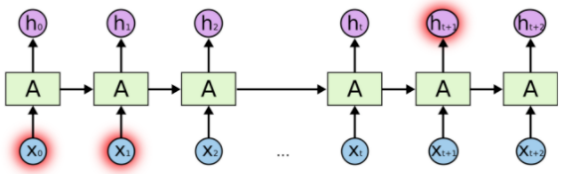


Figure 1 Long short-term memory

Recurrent neural networks can competent the job that learning from past experience and memory. The chain-like unrolled recurrent neural network reveals that recurrent neural network is related to sequence and list. As we mentioned, one of the advantage of RNN is that it can inform things from previous steps. But there is a limitation of the 'memory' of RNN: it can only gather short-term's information, which means when the gap between two relevant information is very large, RNN become unable to learn to connect the information like figure 1 shows.

2.3 Long Short-Term Memory networks

Though in theory, RNN is capable to handle such "long-term" problem. In practice, RNN don't seem to be able to learn "long-term" information [5]. Under this condition, long short-term network which is a special kind of RNN and capable of learning long-term dependencies.

LSTM can fix this problem. The most different part between standard RNN and LSTM is the structure in core layer. As the figure 2 shown, the repeating module in RNN will have a very simple structure, such as a single tanh layer. To fix the long-term memory problem, LSTMs add four interacting layers instead of only one simple layer.

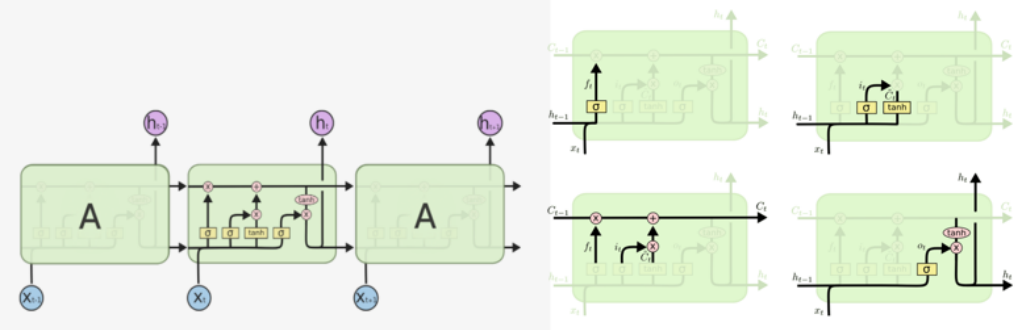


Figure 2 LSTM structure

The first gate is "forget gate layer" which contains sigmoid function and it will used to decide what information is going to be thrown away from the cell gate. ›

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f$$

The second gate is "input gate layer" which have two parts. The first part is another sigmoid layer which values what we will update. Then a tanh layer will create a vector of new candidate values which will be added into the cell state.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_f$$

$$\widetilde{C_t} = \tanh\,(W_C \cdot [h_{t-1}, x_t] + b_{C)}$$

After these two gates, we can update the cell state. We multiply the old state by f_t, forgetting the information what we have decided to throw. Then we add $i_t * \widetilde{C_t}$, the new candidate values, scaled by how much we decided to update each state value.

$$C_t = f_t * C_{t-1} + i_t * \widetilde{C_t}$$

Finally, "output gate layer" will decide what we're going to output. Same with the first two gates, a sigmoid layer will decide what parts of the cell state will be output. Then the cell state will pass through a tanh gate and multiply the result with the output of the sigmoid layer.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o)$$

$$h_t = o_t * \tanh\,(C_t)$$

3. Data Source and Preparation

We use 'Cornel Movie-Dialogs Corpus' and conversation extracted from twitter as our dataset. All datasets are in the form of one question and one answer.

3.1 Corpus

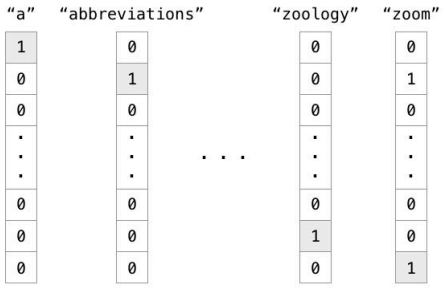


Figure 3 Corpus one-hot vector example

To train a chatbot model, first we need a corpus which contains words and their index which will be used to transform natural language to machine language. For example, we can set 'a' = $[1, 0, 0, \ldots, 0]$, 'b'= $[0, 1, 0, \ldots, 0]$. In our project, we build our words corpus based on conversation datasets. Though this corpus will miss some words which may lead to a bad answer from a particular question, it covers most of popular words in our daily conversation. An informative corpus can cover all words which we use in daily life, however, it will produce huge-dimension one-hot vectors. Despite this, the dimension of the word index vector is still over-sized for our RNN model. So, we need to do word embedding which is a popular method in natural language processing. This method will consider the relevance between words and it will map the original vector to a new lower-dimensional space.

3.2 Training Dataset

As we mentioned, for the comparison of different corpus, we use two dialogs corpus: 'Cornel Movie-Dialogs Corpus' and conversation extracted from twitter as our datasets. The ratio of training set and testing set is 0.15 for this project. A total training set in twitter dataset is 91,296 and we set batch size to 32 as a group of training samples. The total step in this project will train 2,852 per epoch. In order to avoid model over-fitting or under-fitting, we set 50 as our total epoch number to train the model. During the training process, we put some sample sentences consist of the most common daily conversation to observe the performance of our model as training epoch continues, such as "happy birthday to you", "help me to do the exam" and "NY is so cold now", to get the replying by using LSTM current model parameters in every epoch per 200 steps. By adding these sample sentences, it shows that the parameters in the model are gradually adjusting.

4. System implementation

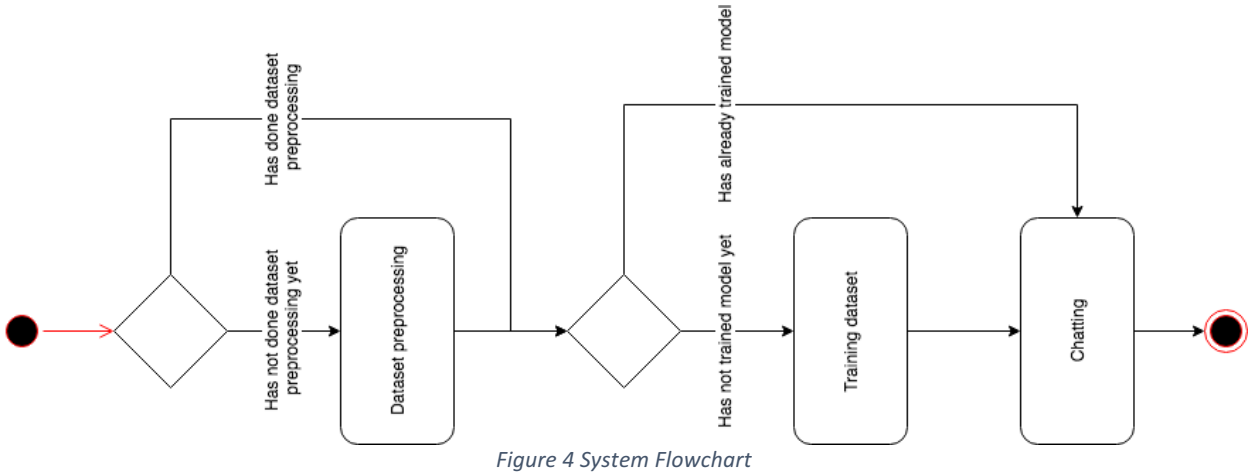


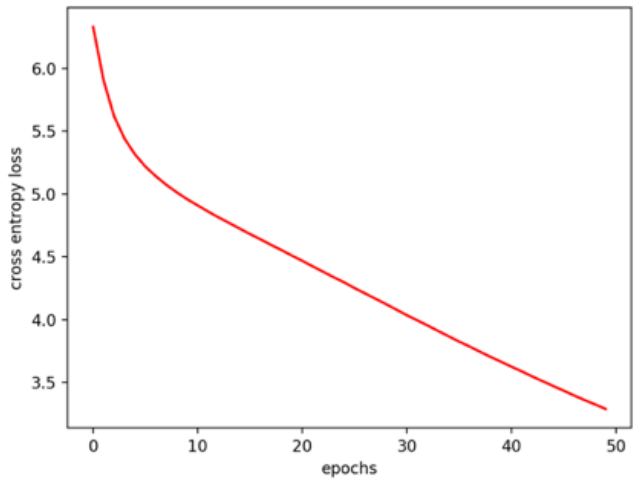
Figure 4 System Flowchart

The system can be separated into three parts, i) dataset preprocessing ii) dataset training and iii) chatting with our system. At the beginning of the stage, system will detect the dataset preprocessing files which are saved by processing the dataset preprocessing function. If dataset preprocessing has been done once, the system will save the preprocessing matrix into "npy" file and "pkl" file. After preprocessing stage, again the system will detect the parameter file, "npz", if system has trained once. The reason why we saved these file in every stage is because it sometimes cost too much time. For example, in the training stage it will cost at least 15 to 25

days without graph card and will cost 5 to 6 hours with graph card. The graph card we used for our project is TITAN X (Pascal). The final stage is chatting stage, in this stage the training parameter result file will be used to put into neural network model then when user types some sentence/question into system, the system will reply some meaningful sentences by using this neural network model, we set 5 replying answer every time when user types one sentence/question to show more result once.

5. Results

5.1 Training Result



Query > happy birthday to you
> i i you
> i i
> i you
> you you the
> i i you
> i you you
> you i you
> you you the
> i i
> i you you
Query > help me to do the exam
> i i
> i i the
> i i
> you i
> i i
> i you
> i you the the
> you i
> you you
> the i the
Query > ny is so cold now
> i i the the
> i you
> i you
> i i the
> you you the the
> i i the the
> i i the
> i i
> i i the
> you you

Query > happy birthday to you
> thank you so much gorgeous
> thank you
> thank you
> thank you so much
> thank you
> thank you
> thank you so much
> thank you
> thank you
> thank you so much
Query > help me to do the exam
> ill be there in a few weeks
> you can do me in my life
> you can me too
> you can me too
> you can me in person
> you can do me in my life
> you can do me in my mentions
> ill try to see it in the morning
> you are so cute
> you can me in person
Query > ny is so cold now
> i need to get a new phone
> im so excited for your loss
> i need a new pair
> im so excited for your response
> i have a feeling
> i need to get some more
> im so excited for your loss
> i have a feeling i need to go back
> its so good
> i need a few hours

Figure 5 (1) Cross Entropy Loss. (2) Training process

During the training, the system is showing the average cross-entropy loss value in every epoch to illustrate the training processing. As the Figure 5(1) at the beginning of the epochs, the cross-entropy loss value is high but gradually diminished. It means that after many recurrences training in LSTM method, the result gradually being better.

The left parts of figure 5(2) is at beginning of the epochs, we can see that system replies the message which is no meaning and having some English grammar problem. However, the right part of figure 5(2) is showing that after training several epochs, the system replies the message which some of them can be considered a meaningful replying and most of them don't have English grammar problem.

5.2 Testing Results

For this testing result, we use our system to test several questions in order to get the meaningful and suitable answer. In figure 6, it shows part of our test, some of the answers from our system are suitable but still have some answers which replies no sense. The problem for

these no sense replying is because the dataset we used is not large enough to deal with all questions. As a result, the system sometimes will get wrong meaning of the question.

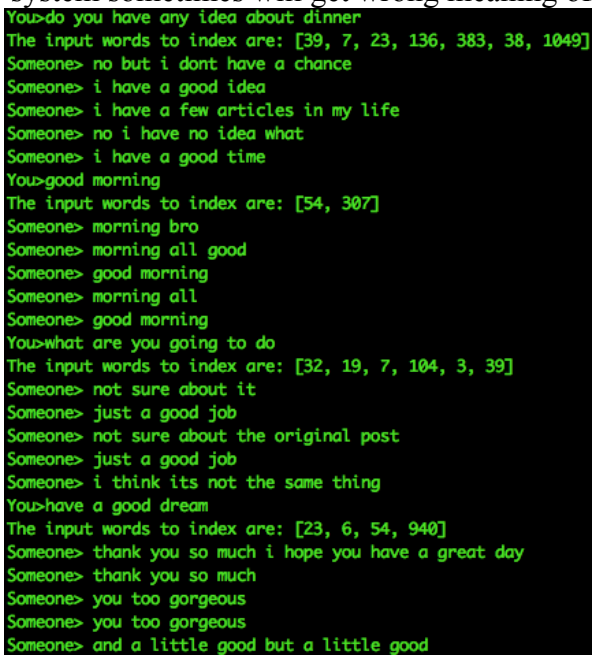


Figure 6 Several testing questions and answers

6. Conclusion & Further work

We have built a simple chatbot system based on a special recurrent neural network – long short-term network. We capitalize conversation corpus extracted from twitter as our datasets. For the natural language processing part, we use NLP related methods which we learned from courses and other academic papers to preprocess our dataset. Furthermore, we develop our own word index corpus and do embedding to generate the input feature for our LSTM model. We use tensorflow and tensorlayer to build our basic sequence to sequence which contains encoding and decoding process chatbot system. As the results shown, our chat robot system is able to generate normal answers when you input some sentences which is daily used.

For the further work, first we need more powerful datasets which could cover more conversation in different field which may be mentioned in daily chat. Then, we need to update structure of our neural network, for example, updating the core layer in our LSTM such as using GRU and other new recurrent neural networks. What's more, we will adjust parameters in our model such as testing different epochs to avoid underfitting or overfitting. Other further works like building a simple user interface may be implemented in the future.

Reference

[1] Bayan Abu Shawar and Eric Atwell. A chatbot system as a tool to animate a corpus. *International Computer Archive of Modern and Medieval English Journal,* pages 5-24, 2005.

[2] Ankil Shah, Bhargav Jain , Bhavin Agrawal , Saurabh Jain ,and Simon Shim. Problem solving chatbot for data structures. *Computing and Communication Workshop and Conference ,* IEEE,2018.

[3] Joseph Weizenbaum. ELIZA — a computer program for the study of natural language communication between man and machine.*Communications of the ACM - Special 25th Anniversary Issue* , pages 23-28, 1983.

[4] Alex Graves , Abdel-rahman Mohamed , Geoffrey Hinton.Speech recognition with deep recurrent neural networks.*Acoustics, Speech and Signal Processing,*2013.

[5] Tsung-Hsien Wen, David Vandyke,and Nikola Mrksic, .A network-based end-to-end trainable task-oriented dialogue system.*European Chapter of the Association for Computational Linguistics*,2016.