Generative Chat-Bot

Nitika Chaudhary
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
nitika@buffalo.edu

Raj J. Patel
Department of Computer Science
University at Buffalo
Buffalo, NY 14214
rajjaysu@buffalo.edu

Abstract

We built a Chat-bot to answer the domain specific questions or to talk in human like way Chatbots are also called Conversational Agents or Dialog Systems. The Chatbot implemented in this paper is generative based. The basic algorithm used for developing Chatbot in this paper is Long Short Term Memory which a version of Recurrent Neural Network.

1 Introduction

The term bots are shortening for robot and refers to a software that behaves like a human in one or the other way. In this case, it is talking. Companies like Facebook, Apple, Google and Microsoft are developing chatbots that have natural conversations indistinguishable from humans. Most of them claim to be using NLP and Deep Learning techniques. We have also used NLP and Deep Learning Technique called Long Short Term Memory (LSTM), which is a Recurrent Neural Network (RNN) in this paper. There are two types of chatbot models, Retrieval based and Generative Model. The retrieval based model use a predefined responses and some kind of heuristic to pick an appropriate response based on the input and the context. This system does not generate new any new text. Generative models on the other hand do not rely on pre-defined responses and generate new responses from scratch. They are based on translation but instead of translating from one language to other it translate from an input to output (response).

Both has its own pros and cons. The advantage of generative based model is that it looks more like human like and can handle unseen cases where appropriate predefined response is not available. On the other hand, the advantage of retrieval-based model is that it is very less likely to give silly responses with grammatical mistakes and is easy to train. We have used generative-based approach in this paper. The domain can be of two type: open domain and closed domain. In open domain, setting the user can take the setting anywhere, which makes it a hard problem. In a closed domain setting the space of possible output and inputs are limited because a system is trying to achieve very specific goal. The users can still take the conversation anywhere they want, but the system is not expected to answer it. The domain for the chatbot in this paper is closed one.

The model can be trained for any domain specific data and can used for answering the questions related to that domain. Our focus is to build model that can be used for answering questions similar to technical support or other type of support required by various companies to answer the common problems of their customers in human like way. There are some common challenges while developing conversational agents. The system needs to understand the linguistic context in order to give sensible answers. The responses regenerated has to be consistent for semantically similar questions or context. There has to be some ideal way to evaluate the model. The ideal model will be retrieval based closed domain model. As it is not possible to develop retrieval based open domain model because we cannot hand craft enough responses required for that. In addition, in corporate companies grammatical errors and offensive errors can be very costly and can drive users away. Therefore, we decided to go with retrieval based closed domain chatbot.

2 Dataset Definition

The data set in use is Cornell Movie Dialogue Corpus. Cristian Danescu-Niculescu-Mizil and Lillian Lee created the dataset in the paper *Chameleons in imagined conversations: a new approach to understanding coordination of linguistic style in dialogs*. It is a large collection of fictional conversations extracted from raw movie scripts. The collection procedure started from raw publicly available movie scripts. In order to collect the metadata necessary for this study and to distinguish between two script versions of the same movie, each script was automatically matched with an entry in movie database provided by IMDB. The movies with votes less than 5 on IMDB were removed and total of 617 unique titles with metadata including genre, release year, IMDB rating and number of IMDB votes and cast distribution were left. Then the pairs of characters were identified that interact and separated their conversations automatically using simple data processing heuristics. The pairs that exchanged less than five conversations were removed and data set was left with around ten thousand exchanges. After that, each character was matched with the list of cast distribution and the gender of each cast was determined as per the gender of actor involved in the script. According to the paper this is the largest dataset of imaginary conversations. We have used movie dialogue corpus to make it little open and interesting but it should perform better when trained and tested on more narrow domain. The numerical details of the final dataset used for this paper is mentioned in following table.

Table 1: Dataset Details

Description	**Value**
Number of Movies	617
Pairs of movie characters	10292
Conversational exchanges	220579
Total utterances	304713

2.1 Preprocessing

The original data collection has a list of lines and was cleaned rather extensively to help the model produce better responses. Series of preprocessing steps were performed on the corpus to generate the most useful representation of data.

Original dataset

```
['L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ They do not!',
 'L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ They do to!',
 'L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I hope so.',
 'L984 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ She okay?',
 "L925 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Let's go."]
```

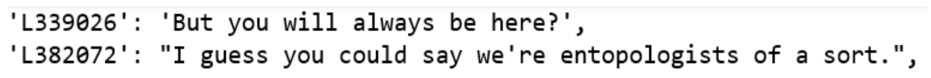
Data snippet of movie_lines

```
["u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L194', 'L195', 'L196', 'L197']",
 "u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L198', 'L199']",
 "u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L200', 'L201', 'L202', 'L203']",
 "u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L204', 'L205', 'L206']",
 "u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L207', 'L208']"]
```

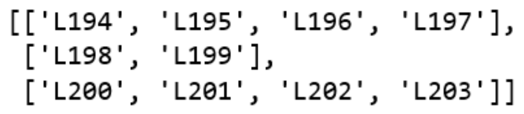
Data snipper of movie_conversations

2.1.1 Extracting lines and conversations

The above figure displays how each line in the dataset has the userId, lineId or movieId appended to its beginning. Since we took just the lines and the conversation context into consideration hence they were extracted out so as to have the following output filtered.

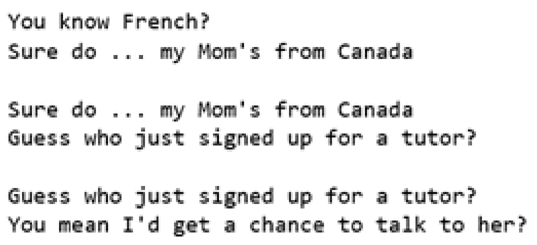
Movie lines 

```
'L339026': 'But you will always be here?',
'L382072': "I guess you could say we're entopologists of a sort.",
```

Conversations: 

```
[['L194', 'L195', 'L196', 'L197'],
 ['L198', 'L199'],
 ['L200', 'L201', 'L202', 'L203']]
```

2.1.2 Sorting lines into questions and answers

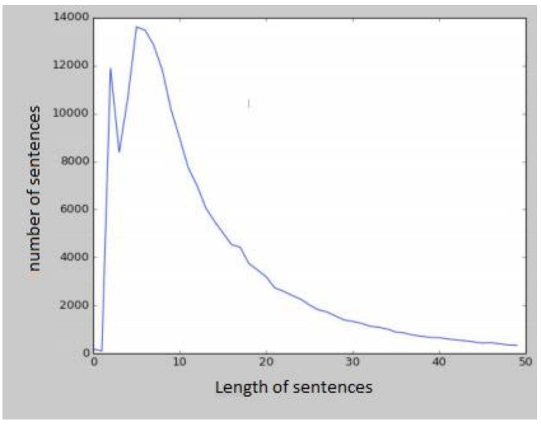


```
You know French?
Sure do ... my Mom's from Canada

Sure do ... my Mom's from Canada
Guess who just signed up for a tutor?

Guess who just signed up for a tutor?
You mean I'd get a chance to talk to her?
```

The list of lines was bifurcated into questions and answers using a line neighbor strategy, such that each line except the last line in a conversation was a question and the following line was the answer. For example, if A, B, C, D are lines then the question answer pairs would be (A, B), (B,C) and (C,D).

2.1.3 Cleaning and Filtering Data

The data is cleaned for any format inconsistencies by removing unnecessary characters and altering the format of words, also the entire line is changes to lowercase characters. Any sentence in the data which is too short or too long tends to bring unconventional results, this is because for each batch during training it has to be made sure that all the sentences have the same fixed length and the outlier lengths would generate the need of excessive padding and hence are avoided in the model training. Therefore, a length range is maintained from 5 to 20 and all the sentences falling out of this range are discarded.



2.1.4 Creating Dictionary

The data set is now composed of clean discrete words but since our model processes information and computations through numbers and we will be using embedding therefore each word is associated with an integer so as to create a final dictionary of word to integer, this is the vocabulary map. After mapping each word to an integer the frequency of the words is verified to be above a threshold value and if not they are removed from the vocabulary.

2.1.5 Adding tokens to the Dictionary

Since our model uses sequence to sequence modelling the vocabulary needs to have four unique tokens:

- <PAD>: During training, the data is fed into the network in batches. All the lines in these batches have to be of the same width for the network to do its calculations. Since the lines are of varied length hence padding is done to make them all even.
- <EOS>: This is another necessity of batching as well, but on the decoder side. It allows us to tell the decoder where a sentence ends, and it allows the decoder to indicate the same thing in its output as well.

- <UNK>: There are words in the dataset which occur less frequently than others, these words were discarded from the dictionary, any such word is replaced by this token. This improves the resource efficiency while training the model on real data.
- <GO>: This is the input to the first time step of the decoder to let the decoder know when to start generating output.

3 Algorithm

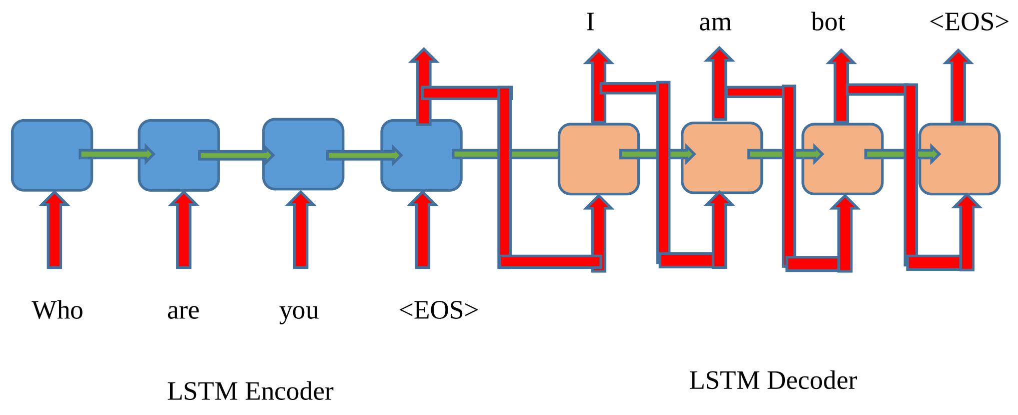
3.1 Word Embedding

Before we pass data to machine, we need to make sure that the machine understands it properly and there comes the role of Natural Language Processing. Embedding is a transformation from discrete values/scalers to dense real value vectors. In preprocessing, we assigned a unique integer to each word in the vocabulary. However, that is not enough machine needs to understand which words are closer and which are far in more than one dimensions and in real value vectors. So each word can be seen as a fixed length vector point in low dimensional space. Each dimension represent a particular spectrum of the vocabulary, which is out of human's understanding. So humans cannot understand the relation between words 'dog', 'bowl' and 'fruit' but embedding makes it possible for machine to relate those words. One challenge in word embedding is that by optimizing to model statistical properties of the training data, we may inadvertently be picking up prejudices and stereotypes implicit in the data.

3.2 Sequence-to-Sequence Model

The sequence-to-sequence model used in this paper is consists of two recurrent neural network. One is the encoder and the other is decoder. The encoder processes the input by mapping a variable length source sequence (input) to a fixed length vector. The decoder generates the output by mapping the vector representation back to a variable-length target sequence (output). The two recurrent neural networks are trained jointly to maximize the conditional probability of the response given the context. The combined model is shown in figure below.

Figure 1: Sequence-to-Sequence Model



Each box in the figure above represents a cell of the RNN, we have taken LSTM(Long Short-term Memory cell). The Recurrent Neural Network is a natural generalization of feedforward neural networks to sequences. Given a sequences of inputs $(x_i, \ldots . x_T)$, a standard RNN computes a sequence of outputs $(y_i, \ldots, y_T)$ by iterating the following equation:

$$h^{(t)} = g^{(t)}(x^{(t)}, x^{(t-1)}, \ldots, x^{(2)}, x^{(1)})$$
$$= f^{(t)}(h^{(t-1)}, x^{(t)};\ \theta)$$

There will be cases where in order to predict last word we need more context for example int the sentence "I grew up in France….. I speak fluent *French.* Using only recent information the last word is some language which is correct but it is not precise. But if the whole context is taken into consideration then the last word would be French. While RNN could work since it is provided with all the relevant information, but it would be difficult to consider long term dependencies. The LSTM keep the summary of whole context till the current timestamp unless made to forget and hence can summarize whole necessary context. Therefore we have used LSTM in this paper because in order to answer any question it is important to summarize the whole context and based on that response is generated. The drawbacks of using LSTM over RNN is that LSTM has more math to do and hence is expensive. And the extra memory adds extra weights which makes is harder to train.

The process of training begins by feeding batch of questions and answers to the model. The questions are feeded to encoder which return a variable sized vector. This vector is not directly feeded to decoder. This vector is first divided into batches of same size as that of encoder. And the tag <'Go'> is added at the end of each batch so that decoder can know when to end each batch. The same embedding procedure is repeated before feeding the batched output of encoder to the decoder.

The weights are initialized based on truncated normal distribution which works similar to normal distribution except the values more than two standard deviations from the mean are discarded and redrawn. Biases are initialized as zeros. Fully connected weights matrix is used for training the decoder. The encoder works same for training and testing. While the function of decoder is, separate for training and testing.

3.2.1 Training

There are two types of attention available in seq2seq function of Tensorflow, "bahdanau" and "luong". The bahdanau pays attention to the parts of the input rather than relying on a single vector. For every step, the decoder can select a different part of the input sentence for consideration. The luong on the other hand considers the whole input as a single vector and importance to parts is not given. We have used the bahdanau attention model because it considers the whole sentences and pays attention to parts of it, which is necessary sometimes. For example, consider the two very large sentences with only one word different well the word may be of great importance say if the word negates the whole sentence. For this reason, we have used bahdanau attention model.

3.2.2 Cost

Simple softmax function is used to calculate the discrete probability distribution over the classes. In addition, we have used weighted average cross entropy (perplexity) to calculate the error in order to update the weights. The weighted average cross entropy is similar to cross entropy except weight cost is calculated across batches or sequences of timestamps and divided by the total cost of the batch or across the sequence.

3.2.3 Optimization

We have used Adam optimizer provided by the Tensorflow. Adam is an algorithm for first-order gradient-based optimization of stochastic objective functions, based on adaptive estimates of lower-order moments. The update rule for variable with the gradient g uses an optimization described as below though the understanding of the method is beyond the scope of this paper.

$$intialized\ as: m_0 = 0,\ v_0 = 0, t = 0$$

$$t = t + 1$$

$$lr_t = learning\ rate * \frac{sqrt(1 - beta2^t)}{1 - \ beta1^t}$$

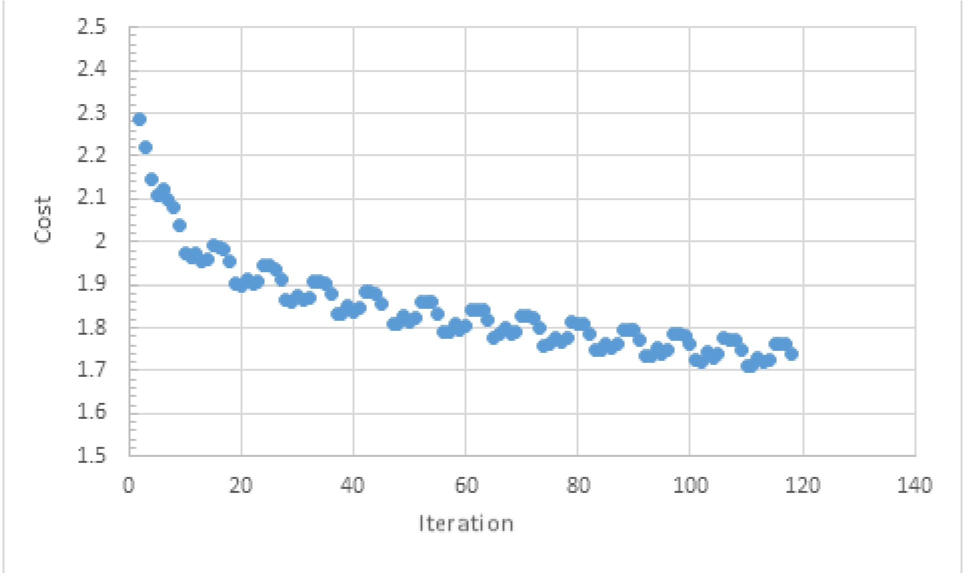
$$m_t = beta1 * m_{t-1} + (1 - beta1) * g$$

$$v_t = beta2 * v_{t-1} + (1 - beta2) * g * g$$

$$variable = variable - \frac{lr_t * m_t}{\sqrt{v_t} + epsilon}$$

4 Experiments and Results

The graph for the cost over the iterations is as shown in figure



Few outputs of the experiments are as follows:

Question > i am sorry

Response> about what

Question> where are you going

Response> i do not know

5 Conclusion

This model demonstrates that it is possible to encode certain aspects of the personality and identify of a character in the chatbot. The movie dialogues scraped from movies are not of very high quality and heuristics were used to separate conversations, which affected the performance of the bots. Moreover, the dataset used is from more than five hundred movies, which makes its domain wide. If the model is trained from dataset with narrow domain, for example dataset for answering the customer's questions can provide much better performance.

There is lots of chances of having grammatically wrong and silly answers in this model as it is generative model while retrieval based models may not be able to answer the unseen questions. Therefore, the future work we are planning to do is develop Generative Adversarial Network, which will be combination of both discriminative (Retrieval based) and generative based which will compete with each other make each other better.

Acknowledgments

We are extremely grateful to Professor Sargur Srihari for elucidating all the necessary concepts related to Probability Graphical Models. We would also appreciate the efforts taken by the TAs of this class for proofreading the paper and furnishing critical reviews for the same.

References

[1] Ilya Sutskever, Oriol Vinyals. & Quoc V. Le. Sequence to Sequence Learning with Neural Networks.

[2] Kyunghyun Cho Bart van Merrienboer Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares Holger Schwenk. & Yoshua Bengio. Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translation.

[3] Suryiadeepan Ram. http://suriyadeepan.github.io/2016-06-28-easy-seq2seq/.

[4] Standford. http://web.stanford.edu/class/cs20si/lectures/slides_13.pdf.

[5] Diederik P. Kingma, Jimmy Lei Ba. ADAM: A Method For Stocastic Optimization.

[6] Zachary C. Lipton, John Berkowitz, Charles Elkan. A Critical Review of Recurrent Neural Networks for Sequence Learning.

[7] Tensorflow. https://www.tensorflow.org/tutorials/seq2seq.

[8] Sean Robertson. https://github.com/spro/practical-pytorch/blob/master/seq2seq-translation/seq2seq-translation.ipynb.

[9] http://docs.w3cub.com/tensorflow~python/tf/contrib/seq2seq/sequence_loss/.

[10]https://www.tensorflow.org/versions/r1.0/api_docs/python/tf/contrib/seq2seq/dynamic_rnn_decoder.

[11] Denny Britz. http://www.wildml.com/2016/04/deep-learning-for-chatbots-part-1-introduction/.