Retrieval vs Generative ChatBot

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 Chatbot which conducts domain specific conversations designed to convincingly simulate human conversational behavior. Chatbots are typically used in dialog systems for various practical purposes. This paper has implementation for both generative based and retrieval based Chatbot and it explores the pros and cons between the two models while using Recurrent Neural Network as the underlying base architecture.

1 Introduction

Chatbots are artificially intelligent computer systems that converse with humans using natural language. It is the new form of storytelling by being available to the consumers at the right time, right place and with the right information. The most common domain of today's chatbots are information retrieval, personal assistance and e-commerce. It would not be fair to talk about Chatbots without mentioning Joseph Weizenbaum who in 1966 created the first ever chatbot system, ELIZA which imitated the language of a psychotherapist form only 200 lines of code. These models have ever since been growing and now cover almost every sector of technology.

There are two types of chatbot models, Retrieval based and Generative Model. The retrieval based model uses a predefined set of responses and some kind of heuristic to pick an appropriate response based on the input and the context. This system does not generate any new text. Generative models on the other hand do not rely on predefined responses and generates new responses from scratch. They are based on translation but instead of translating from one language to another it translates from an input to an output (response). Both the models have their own pros and cons. The advantage of generative based model is that it imitates a better human behavior and can handle unseen cases where appropriate predefined responses are not available. On the other hand, the advantage of retrieval-based model is that it renders lesser grammatical mistakes and very relevant useful answers. This paper implements both the models and scrutinizes the performance.

The conversation domain of any Chabot is a very important aspect; it can be either an open domain or a closed domain. In an open domain model the conversation can take any turn and its does not necessarily have a well-defined aim or intention. The infinite number of topics and the fact that a certain amount of knowledge is required to create reasonable responses makes this a hard problem. In a closed domain, however, the conversation is expected to revolve around a common topic and the system is trying to achieve a very specific goal, technical customer support or shopping assistants are few such examples. We have worked on a closed domain model and implemented deep learning techniques to function a chatbot. One way of looking into the two different models is that a retrieval based model has a stand for a correct or a wrong answer and can be seen as an unsupervised learning whereas for a generative model the answers cannot be judged to be correct or wrong, more like a supervised learning.

2 Dataset Definition

The Chatbot is aimed to be capable of having regular humanoid social conversations and therefore the data taken is largely composed of naturally occurring dialogues. We wanted to train the model as good as it could be and in the effort we decided to go for two separate datasets. The first dataset has general conversations between people which was helpful to give the model more human like behavior, however, so that we could analyze whether the results were in the correct direction the second dataset we took was a technical support dataset which provided a more specific vocabulary.

Cornell Movie Dialogs Corpus. It is a large metadata-rich collection of fictional conversations extracted from raw movie scripts. The collection procedure started from raw publicly available movie scripts. The pairs of characters that interact were identified and their conversations separated 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. The dataset contains conversational exchanges between thousands pairs of movie characters and hence has data files consisting of information about each of the movie, their titles, character details and conversation metadata and finally the dialogue lines. Since we were mostly interested in the chats therefore we took two specific data files which retained the conversation context and the lines. According to the paper [12] 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.

Description	Value
Number of Movies	617
Pairs of movie characters	10,292
Characters	9,035
Conversational exchanges	220,579
Total utterances	304,713

Table 1: Cornell Movie Dialogs Corpus details

Ubuntu Dialogue Corpus. It consists of almost one million two-person conversations extracted from the Ubuntu chat logs, used to receive technical support for various Ubuntu-related problems. The conversations have an average of 8 turns each, with a minimum of 3 turns. All conversations are carried out in text form (not audio). The dataset is orders of magnitude larger than structured corpuses such as those of the Dialogue State Tracking Challenge. It is on the same scale as recent datasets for solving problems such as question answering and analysis of microblog services, such as Twitter, but each conversation in this dataset includes several more turns, as well as longer utterances.

Description	Value
# dialogues (human-human)	930,000
# utterances (in total)	7,100,000
# words (in total)	100,000,000
Min. # turns per dialogue	3
Avg. # turns per dialogue	7.71
Avg. # words per utterance	10.34
Median conversation length (min)	6

Table 2: Ubuntu Dialogue Corpus details

2.1 Data Preprocessing

The movie dialogue corpus has list of conversations which are 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). The Ubuntu Data corpus is consists of context and responses so that no such bifurcation is required.

The data is cleaned for any format inconsistencies by removing unnecessary characters and altering the format of words. 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. Dictionary is created for word to integer mapping including special tokens (<PAD>, <UNK>, <EOS>). 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.

3 Architecture

3.1 Recurrent Neural Network

A Recurrent Neural Network is a class of artificial neural network where connection between the units form a directed cycle. They perform the same task for every element in the sequence with the output being dependent on the previous computations. RNN's have a "memory" which captures information about what has been calculated so far which was a necessary requirement, as for a chatbot to reply correctly to any sentence, it needs to have the previous context.

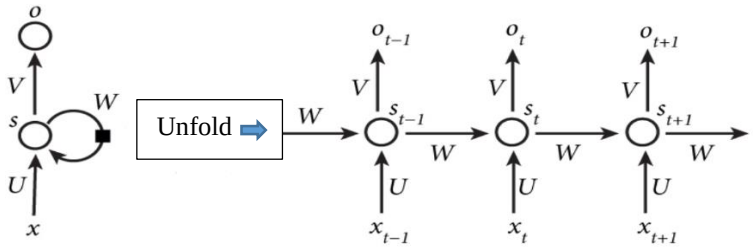


Figure 1: Recurrent Neural Network

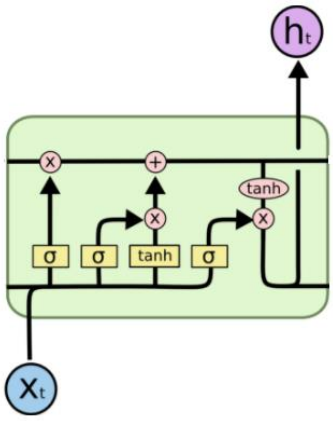
The Figure 1 shows an RNN being unfolded into a full network. For instance, in the model if the sentence has 5 words the network would unroll into 5 layers of neural network. A state s_t at any time step t is calculated as

$$s_t = f(s_{t-1}, x_t) \text{ , f is tanh function}$$
$$s_t = \tanh(Ux_t + Ws_{t-1})$$

3.1 Long Short Term Memory Unit (LSTM)

Long Short Term Memory Networks usually called LSTM's are a special kind of RNN, capable of learning long term dependencies. The model required a memory mechanism which would

store previous values of any ongoing conversation and respond accordingly, and therefore this LSTM memory mechanism was used which is specially designed for long-tern dependency problem. The LSTM keeps the summary of the whole context till the current timestamp until made to forget and hence can summarize whole necessary context. The only drawback experienced while using LSTM was the large amount of computations which made It expensive and the extra memory adds extra weights which makes it harder to train.



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

$$i_t = \sigma\,(W_i \,.[h_{t-1}\,,\, x_t] + b_i)$$

$$\tilde{C}_t = \tanh(W_c \,.\, [h_{t-1}\,,\, x_t] + b_c)$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

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

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

3.3 Word Embedding

The dataset to be fed to the network was a collection of words, but a machine does not understand English words, it understands numbers, this problem statement is well handled by Natural Language Processing. Word embedding is the mapping processed from a word to a corresponding vector of real numbers. It is a transformation from discrete values/scalars to dense real value vectors. During 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 farther 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.

W("morning") = (0.2, -0.4, 0.7, ...)

4 Algorithm

While the underlying architecture remains the same for both the Retrieval based and Generative Based Model, their deep learning techniques differ. Recurrent Neural Network with LSTM as the basic building block is the best choice for a dialog system and hence has been kept the same.

4.1 Retrieval Based Model

This model had a repository of pre-defined responses it could use, the input to the model was a context c and a potential response r, the output of the model was a score for the response. The model working was more like a classification problem where scores for multiple responses were calculated and the one with the highest score was chosen as the final response.

4.1.1 Dual Encoder LSTM

The Deep Learning Structure built for the model was a Dual Encoder LSTM Network. Among

the various other networks that would work well for building a dialog system this network has been reported to perform most decently.

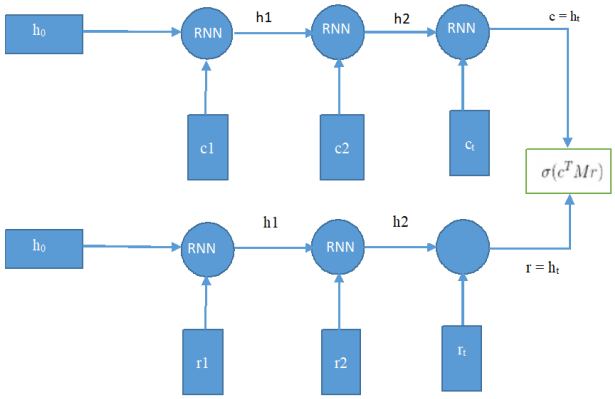


Figure 2: Dual Encoder LSTM Model

4.1.2 Training

Both the conversational context and the response text were split by words and each word was embedded into a vector. The context and the response were fed into the same recurrent neural network word by word. A vector representation got generated by the RNN which as mentioned above is a numeral mapping of the context(c) and response(r). An embedding size of 50 was taken for these vector representations. The context vector c then gets multiplied with a matrix M that is learned during the training to predict the response(r). Since c had the size 50 therefore this matrix had the size 50x50 and the multiplication result was another 50x50 matrix which was interpreted as the generated response. The dot product of the predicted response and actual response gave the similarity measure for the two vectors. The value of this product decided the score given to the response, a large value meant a higher score while a smaller value would result in a lesser score. The sigmoid function was then applied to the score to convert it into a probability value.

4.1.3 Results

Question >i set up my hd such that i have to type a passphrase to access it at boot how can i remove that password

Response >backup your data and re instal without encrypt

Question >i just instal ubuntu on my laptop , and i notice i get poor batteri life

Response >there be not ani program that will magic improve your batteries

Question >i can not mount my drive

Response >and post the url that it output here

4.2 Generative Based Model

This model does not rely on pre-defined repository rather generates responses from scratch.

The generative models are typically based on Machine Translation techniques, and the best known technique if the sequence-to-sequence modelling.

4.2.1 Sequence-to-Sequence Model

The sequence-to-sequence model used in this paper 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.

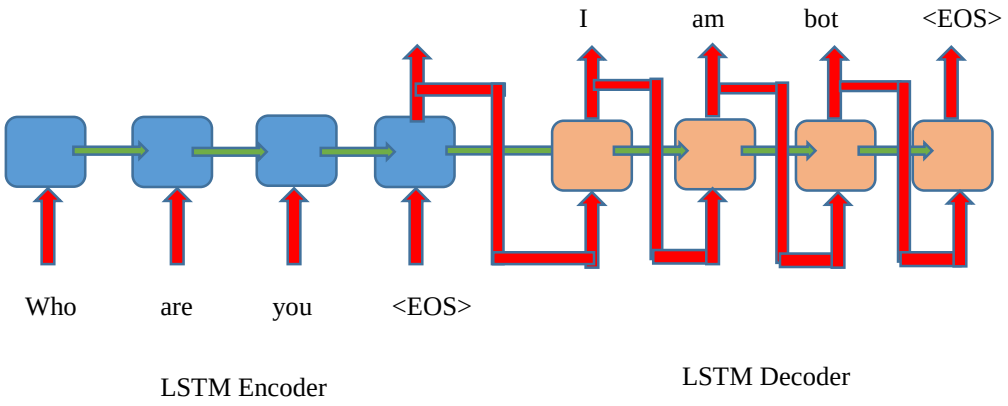


Figure 3: Sequence-to-Sequence Model

4.3.1 Training

Training begins by feeding batch of questions and answers to the model. The questions are fed to encoder which returns a variable sized vector. This vector is not directly fed to decoder but divided into batches of same size as that of encoder and the tag <Go> is appended to each batch so that decoder knows 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 similarly for both training and testing while the function of decoder varies for training and testing, during testing it predicts the response by using the learned weights.

The LSTM cell as mentioned earlier in this paper has computational drawbacks and hence we trained the model using GRU cells. The idea behind GRU is very similar to LSTM and it has memory retaining applications however since is has fewer parameters the training was a bit faster with a better generalization as it requires lesser data to generalize. Another very important performance enhancement technique is the attention mechanism. In the sequence-to-sequence modelling the final encoder state is fed into the decoder. This last state of the encoder contains mostly information from the last element of the encoder sequence. Therefore, the context is biased towards the end of the encoder sequence, and might miss important information at the start of the sequence. Instead of compressing the entire input sequence into a fixed representation this mechanism holds onto all states from the encoder and gives the decoder a weighted average of the encoder states for each element of the decoder sequence. This made the results better.

5 Cost Function

There were two separate cost functions for both retrieval and generative models. The problem addressed in the retrieval based model was a classification problem where scores were rendered to the responses and the response with the highest score was picked. Cross-entropy loss function is one of the common cost functions used for classification problems and we have used it to train the neural network. To understand it more for our model, considered a context-response pair and its actual true label to be p (which can either be a 1 for correct response or 0 for an incorrect response). Let's consider the predicted probability for the pair to be p`. Cross entropy is calculated as:

Loss = -p * ln(p`) - (1-p) * ln(1-p`)

The intuition behind this formula is simple. If p=1 Loss = -ln(p`), which penalizes a prediction far away from 1, and if p=0, Loss = -ln(1-p`), which penalizes a prediction far away from 0.

In the generative based model, softmax function was used to calculate the discrete probability distribution over the classes.

6 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.

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

$$t = t + 1$$

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

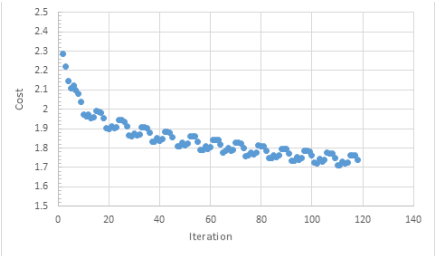
$$m_t = beta1 * m_{t-1} + (1 - beta1) * g$$

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

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

7 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

8 Conclusion

Chatbot is a brilliant AI engine which is now engulfing almost every technology sector. This paper demonstrated the two giant models today used to build a chatbot. Both the model had their positives and negatives, while the generative based model generated entertaining answers for the most random questions, the retrieval based model responded with quite good accuracy for the questions related to the dataset. For the negatives, the generative model rendered grammatical mistakes more often and the retrieval model responded with irrelevant answers for the out of domain questions. The paper aimed to show the best possible implementations for the two models and their results.

The results were impressive but still not satisfactory. The dataset taken was comprised of naturally occurring dialogues and most of the words in the vocabulary were eliminated due to the scarcity of the word occurrence which hindered the performance. Availability of an efficient processing machine can help improve the system better by enabling more training on further larger data.

Acknowledgments

We are extremely grateful to Professor Sargur Srihari for elucidating all the necessary concepts in Deep Learning. 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/.

[12] Cristian Danescu-Niculescu-Mizil and Lillian Lee: Chameleons in Imagined Conversations:

A new Approach to Understanding Coordination of Linguistic Style in Dialogs.

[13]Stackflow https://stackoverflow.com/questions/40442098/saving-and-restoring-a-trained-lstm-in-tensor-flow

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

[15] Siraj Raval. https://www.youtube.com/watch?v=ElmBrKyMXxs