Generative Dialogue System using Neural Network

[1]Nikki Singh, [2]Dr. Sachin Bojewar

[1]Post Graduate Student, [2]Associate Professor,
[1]Information Technology,
[1]Vidyalankar Institute of Technology, Mumbai, India

Abstract: A conversation between humans and computers is regarded as one of the most hard-core problems in computer science, which involves interdisciplinary techniques in information retrieval, Machine Learning, natural language processing, and artificial intelligence. It is an Interactive Entity which automatically generates conversations to exchange information smoothly between people with little knowledge of the computer. The challenges lie in how to respond so as to maintain a relevant and continuous conversation with humans. This research applies a generative model-based method for conversation generation. This research is for the development of conversational agent, which generates conversations using recurrent neural network and its coupled memory unit.

Index Terms **- Neural conversational model, seq2seq, LSTM, RNN, Conversational model, generative model.**

I. INTRODUCTION

A conversational agent is a piece of software that is able to communicate with humans using natural language. Ever since the birth of AI, modeling conversations remains one of the field's toughest challenges.

Advances in end-to-end training of neural networks have led to remarkable progress in many domains such as speech recognition, computer vision, and language processing. Recent work suggests that neural networks can do more than just mere classification, they can be used to map complicated structures to other complicated structures. The task of mapping a sequence to another sequence which has direct applications in natural language understanding has been used widely in conversation generation. Artificial Conversational Agent is a computer program which conducts a conversation with the user via textual methods. It uses machine learning to effectively mimic human conversation and react to written prompts to deliver a service. It is essentially a user interface which can be plugged into a number of data sources via APIs so it can deliver information or services on demand.

There are two types of models for building Artificial Conversational Agents. Namely, Retrieval based models and Generative models. Retrieval-based models use a repository of predefined responses and some kind of heuristic to pick an appropriate response based on the input and context. The heuristic could be as simple as a rule-based expression match, or a complex as an ensemble of Machine Learning Classifiers. These systems don't generate any new text, they just pick a response from a fixed set. However, Generative models don't rely on predefined responses. They generate new responses from scratch. Generative models are typically based on Machine Translation techniques, but instead of translating from one language to another, they translate from an input to an output (response). Machine Learning techniques can be used for both retrieval-based and generative models, but research seems to be moving into the generative direction. Machine Learning architectures like Sequence to Sequence are uniquely suited for generating text and researchers are hoping to make rapid progress in this area

II. LITERATURE SURVEY

Conversational agents were previously built using retrieval based methods. However, the advances in neural networks moved the direction of this research towards generative based models. The retrieval based models generally are made of statistical methods while the generative models use neural networks.

2.1 Information retrieval method:

A conversational agent can be created using information retrieval systems where the knowledge about conversation patterns in the form of user queries and their expected responses are stored in the form AIML(Artificial Intelligence Mark-up Language) files [1]. Whenever the user inputs a message or a query, it is matched with the already stored user queries in the AIML files.

This information retrieval based technique was developed for FAQs (Frequently Asked Questions).This technique can be used in the goal based systems where the knowledge required to respond to the users is limited.

2.2 Using Naive Bayesian Classifier:

A question answering conversational agent can also be created using Naive Bayesian classifier [2]. In this system, the user queries are categorized into topics and a separate category list is created in the database. The query asked by the user is fed to the system. The chat agent accepts the query. Lexical parser extracts the keywords from the query. These keywords are compared with the category list in the database. The probability that the query belongs to a particular category list is calculated using the Bayesian probability theory. This is done to reduce the search space so that the process of matching the user query with the query stored in a database is done efficiently.

2.3 Neural Network based model:

The problem of predicting responses for the conversational agent closely relates to the machine translation problem. Recurrent Neural Network Language Model (RLM) proposed in [5] uses continuous representations to estimate a probability function over natural language sentences. Shang et al. proposed to use the recurrent neural network framework for generating responses on micro-blogging websites [6].

The current state of art sentence generation model is the model of sequence-to-sequence[7]. The sequence to sequence model was firstly proposed in the field of machine translation, but it can be applied in generating the conversation as well. This model has an encoder unit and a decoder unit. The system inputs the preceding sequence to the encoder unit word by word. This model allows the sequential and recursive mapping from encoded words to its designated decoded representation.

III. THE MODEL

This approach suggested in the Neural Conversation Model by Vinyals, et al[3]. It is based on the Seq2Seq[4] framework described below diagram. It uses two LSTMs one for encoding and one for decoding. To preserve context, the input sequence is the concatenation of what has been conversed so far, and the output sequence is the reply. For example, there is a conversation between 2 persons and the one person utters ABC, and the other person replies WXYZ. This segment of conversation is trained with to produce a map from ABC to WXYZ. The encoder reads the input ABC until it gets the end of statement flag, in reverse order to generate a vector embedding. This becomes the input of the decoder to train the first word W and then, W is again put back in the decoder along with the input ABC to generate the next word.

The dialog encoder and response decoder form together a sequence-to-sequence (SEQ2SEQ model, which has been successfully used in building end-to-end conversational systems. Both encoder and decoder are recurrent neural network (RNN) models: an RNN that encodes a variable-length input string into a fixed-length vector representation and an RNN that decodes the vector representation into a variable-length output string. Encoder-decoder framework t of the model is almost identical to prior conversational SEQ2SEQ models, except that we use Multi RNN and LSTM cells. In seq2seq models sometimes encoders and decoders share weights in tasks, but do not do so in the proposed model, and also they do not share word embeddings.

3.1 Sequence-to-sequence model:

A basic sequence-to-sequence model consists of two recurrent neural networks (RNNs)[7]: an encoder that processes the input and a decoder that generates the output. This basic architecture [7] is depicted in Fig.3.1

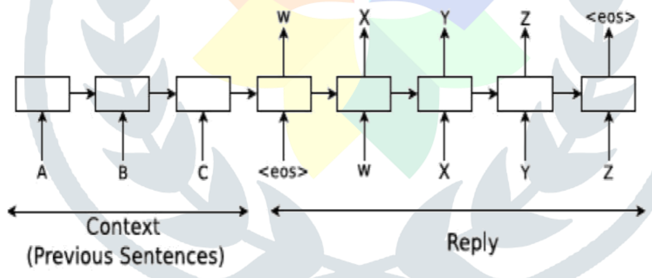

Fig 3.1: Sequence to sequence model

- Encoder - Encoder is an LSTM based neural network which takes the word embed-dings of the input and converts them to a thought vector which is given as input to the decoder. The number of hidden states of the encoder depends on the size of the input. Each LSTM cell converts a single word in the input sentence to one row of the thought vector.

- Decoder - Decoder is an LSTM based neural network whose initial state is the final state of the encoder. It takes start token as input and outputs the predicted first word of the output sequence. The word predicted by the previous cell is given as input to the next cell and then it predicts the next word in the sequence and so on. It stops when the end of sentence token is generated.

3.2 Word Embedding:

Traditional models usually treat a word as a discrete token; thus, the internal relation between similar words would be lost. Word embeddings are a standard apparatus in neural network-based text processing. A word is mapped to a low dimensional, real-valued vector. This process, known as vectorization. It captures some underlying meanings. Given enough data, usage, and context, word embeddings can make highly accurate guesses about the meaning of a particular word. Embeddings can equivalently be viewed that a word is represented as a one-hot vector and multiplied by a look-up table. Vectorization of all words using their embeddings serves as the foundation of the deep neural network

3.3 LSTM Network:

Long Short Term Memory networks are a special kind of RNN, capable of learning long-term dependencies. LSTM cells can be used in the sequence to sequence model to remember the previous words in the sentence. LSTM processes one word at a time and computes probabilities of the possible values for the next word in the sentence. The memory state of the network is initialized with a vector of zeros and gets updated after reading each word. LSTM is helpful in remembering the context of the sentence[8].

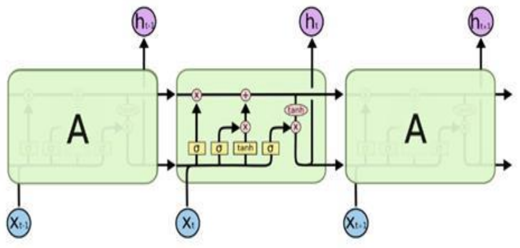

Fig 3.3: LSTM Network

In Fig. 3.3, each line carries an entire vector, from the output of one node to the inputs of others[8]. The pink circles represent pointwise operations, like vector addition, while the yellow boxes are learned neural network layers. Lines merging denote concatenation, while a line forking denotes its content being copied and the copies going to different locations. Instead of having a single neural network layer, LSTM has four interacting in a very special way

- Forget gate

Thefirst step in LSTM is to decide what information is going to be thrown away from the cell state. This decision is made by a sigmoid layer called the forget gate layer'. It looks at h_{t-1} and tx, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. A 1 represents completely keep the information while a 0 represents completely get rid of the information.

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

Where $_{it}$ is the forget gate vector, W_f is the weight matrix, h_{t1} is the previously hidden state, x_t is the input vector and b_f is the bias.

- Cell state

First, a sigmoid layer called the `input gate layer' decides which values to update.

Next, a tanh layer creates a vector of new candidate values,C_t that could be added to the state. In the next step, these two are combined to create an update to the state

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

Where i_t is the input vector, W_i is the weight matrix, h_{t1} is the previous hidden state, x_t is the input word embedding vector and b_i is the bias

$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \ + \ b_C)$$

Where C_t is the temporary cell state, W_c is the weight matrix, h_{t1} is the previous hidden state, x_t is theinput word embedding vector and b_c is the bias.

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

Where C_t is the cell state, f_t is the forget gate vector computed above, C_{t-1} is theprevious cell state, i_t is the input vector and C_t is the temporary cell state.

- Output state and Hidden state

The output will be based on our cell state, but will be a filtered version. First, we run a sigmoid layer which decides what parts of the cell state we are going to output. Then, we put the cell state through tanh to push the values to be between 1 and 1) and multiply it by the output of the sigmoid gate, so that we only output the parts we decided to.

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

Where o_t is the output vector, W_o is the weight matrix, h_{t1} is the previous hidden state, x_t is the input word embedding vector and b_o is the bias.

$$h_t = o_t * \tanh\left(C_t\right)$$

Where h_t is the current hidden state, o_t is the output state, and C_t is the current cell state.

IV. TRAINING ALGORITHM

The input will be the sequences of words. Every word is Converted to its embedding. This embedding is fed to LSTM Encoder as input. LSTM Encoder encodes the inputs into a thought vector. The thought vector is given as input to the LSTM Decoder. The decoder outputs the predicted word in the form of embedding of the output sequence. Fig.4.1 shows the flow of the training algorithm.

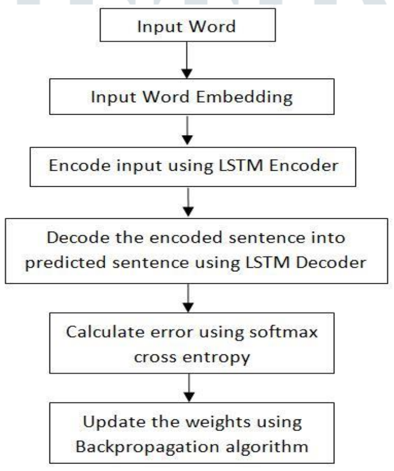
Input Word
Input Word Embedding
Encode input using LSTM Encoder
Decode the encoded sentence into predicted sentence using LSTM Decoder
Calculate error using softmax cross entropy
Update the weights using Backpropagation algorithm

Fig: 4.1 Training algorithm for conversational agent

During training, the weights of the LSTM Network are randomly initialized in the first step. Once the Decoder outputs its first prediction, the error is calculated using the cross entropy error function.

$$E(t,o) = -\sum t_i \log o_i$$

Where E(t; o) is the error function, t is the target, o is the predicted output, i is the index of the word in the input sequence.

Once the error is calculated, the derivative of every weight with respect to the error function gives the step by which we should increase or decrease the weight to minimize the error. This is called Backpropagation Algorithm. For number of iterations, the weights are kept on updating so that the error eventually reduces near to zero. Then we can say that the model is trained.

V. CONCLUSION

The work discussed in this report focuses on the implementation of the sequence to sequence framework The model is able to learn the sequence and output the expected sequence. The same model can be modified to train on the sequences of words to build a conversational agent.

This model was built using LSTM cells. Various researches have proven that the LSTM network is more effective for larger sequences as compared to the RNN network. The softmax activation function helps to map the output in the range of 0 to 1 such that their sum adds to 1. In other words, it gives the probability distribution which helps to choose which word is to be predicted next.

The remaining work will further extend current work to incorporate an algorithm to learn the queries and reply to build the conversational agent. The model can be trained on a conversational dataset to learn the dialogues

VI. REFERENCES

[1] A. Shawar, E. Atwell, and A. Roberts, “Faqchat as in information retrieval system," pp. 274-278, 2005.

[2] M. Niranjan, M. Saipreethy, and T. Kumar, “An intelligent question answering conversational agent using naive bayesian classifier," pp. 1-5, 2012.

[3] M.-T. Luong, I. Sutskever, Q. V. Le, O. Vinyals, and W. Zaremba, “Addressing the rare word problem in neural machine translation, "arXiv preprint arXiv:1410.8206, 2014.

[4] Y. Bengio, R. Ducharme, P. Vincent, and C. Jauvin, “A neural probabilistic language model," Journal of machine learning research, vol. 3, no.Feb, pp. 1137-1155, 2003.

[5] T. Mikolov, S. Kombrink, L. Burget, J. Cernocky, and S. Khudanpur, “Extensions of recurrent neural network language model," pp. 5528-5531, 2011.

[6] L. Shang, Z. Lu, and H. Li, “Neural responding machine for short-text conversation,"arXiv preprint arXiv:1503.02364, 2015.

[7] I. Sutskever, O. Vinyals, and Q. V. Le, “Sequence to sequence learning with neural networks," pp. 3104-3112, 2014.

[8] S. Hochreiter and J. Schmidhuber, “Long short-term memory," Neural computation,