ISSN: 2586-7652 (Print) Vol. 03, No. 02, September 2020
ISSN: 2635-7607 (Online)
International Journal of Advanced Engineering
Source: http://ictaes.org
Manuscript received: July 24, 2020; Revised: August 28, 2020; Accepted: September 10, 2020

Deep Learning Chatbot based on RNN and Seq to Seq Model

Suzen Sen Shrestha[1], Suman Kafle[2]

[1]*Cosmos College of Management and Technology, Lalitpur, Nepal*
[2]*Nepal Information Technology Center(NITC), Kathmandu, Nepal*

suzenshrestha98@gmail.com[1], suman99pro@gmail.com[2]

Abstract

A chatbot is a computer program that conducts a conversation via auditory or textual methods. It uses sophisticated NLP systems. They are basically used on the website and instant messaging networks. Deep Learning is the application of Artificial Neural Networks (ANNs) to learning task that contain more than one hidden layer. A recurrent neural network (RNN) is a class of artificial neural network where connections between units form a directed cycle. TensorFlow is an open source software library for numerical computation using data flow graphs.

Keywords: *NLP systems, ANNs, RNN, Tensorflow*

1. INTRODUCTION

A chatbot is an AI agent that can participate in a conversation with a user. Most are equipped with a messenger type interface with an input from a user and an output from the chatbot. The chatbot displays this information based on a set of rules set by the human operator who can route responses based on keyword matching. It processes the users input and outputs a reply based on what the user has just sent. It could be a greeting, conversation topic, or even an image. Most basic chatbots work by matching a user's input with a predefined set of dialog. For example, a user saying “Thank you” will result in the chatbot saying “You’re Welcome”. The predefined set of dialogs can be set up to imitate a normal conversation between two people. Problems can arise when a user says something the chatbot does not recognize, an example could be the user meaning to say “Thank you”, but instead says “Thanks a lot”, this can confuse the chatbot as it will be looking to match the “Thank you” input with “Welcome”. This leads to a lot of manual work by trying to define every combination of a user saying “Thanks”. Modern chatbots are more complex and feature natural language processing that can learn from user inputs. They can access APIs to get information users such as news, - 10 - weather, time etc. They can even process orders and make bookings entirely through a chatbot interface [1][4].

2. Software design

Corresponding Author : Suman Kafle
Author’s affiliation : Nepal Information Technology Center (NITC), Kathmandu, Nepal
Email: suman99pro@gmail.com
Copyright © ICT-AES

2.1Front End Design:

The Graphical User Interface is attained using a Python API called Tkinter using the following steps:

• A Tkinter Window consists of a chat box for display and a text box for accepting input question.

• A send button is used which when pressed triggers two events.

• The first is that the question given as input is displayed in the chat box.

• The second event causes a method call at the python backend which transfers the input to the backend for processing [9] [12].

• The output feedback from the backend is sent using the call and displayed on the chat box [18].

• A series of different input is processed the same way as above.

2.2 Backend design:

A Recurrent Neural Network is a straightforward adaptation of the standard feed-forward neural network to allow it to model sequential data. At each timestamp, the RNN receives an input, updates its hidden state, and makes a prediction. The RNN's high dimensional hidden state and nonlinear evolution endow it with great expressive power, enabling the hidden state of the RNN to integrate information over many timestamps and use it to make accurate predictions [9] [10].

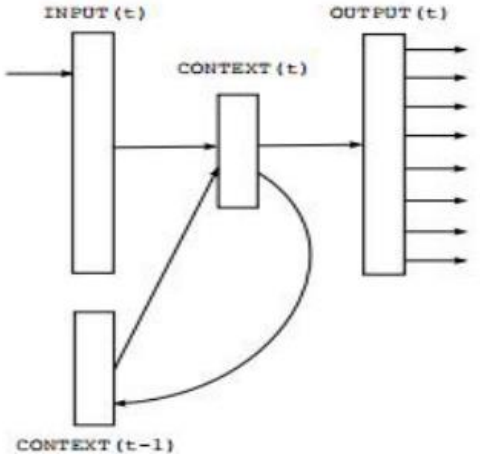


Figure 1. Simple Recurrent Neural Network

2.3 System Design:

Figure 2(below) shows an RNN being unrolled (or unfolded) into a full network [2]. By unrolling we simply mean that we write out the network for the complete sequence. For example, if the sequence we care about is a sentence of 5 words, the network would be unrolled into a 5-layer neural network, one layer for each word.

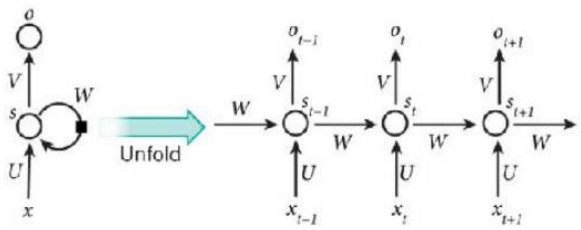


Figure 2. Algorithm Design

The formulas that govern the computation happening in a RNN are as follows [16]:

• x_t is the input at time step t. For example, could be a one-hot vector corresponding to the second word of a sentence [3].

• s_t is the hidden state at time step t. It's the "memory" of the network. s_t is calculated based on the previous hidden state and the input at the current step:

s_t = f($U x_t + W s_{t-1}$). The function f usually is a nonlinearity such as tanh or ReLU. s_{-1}, which is required to calculate the first hidden state, is typically initialized to all zeroes.

•o_t is the output at step t. For example, if we wanted to predict the next word in a sentence it would be a vector of probabilities across our vocabulary: o_t = softmax(V_(s_t)).

3. METHODOLOGY

We have used the standard Cornell Movie-Dialogue Corpus as a Dataset. We have used python 2.7 with TensorFlow 0.12 for seq2seq for picking/generating responses. To test our hypothesis, we will train our chat bot on filtered and minimally filtered data and compare the responses [3].

3.1 Data Collection

The bot comes with the script to do the pre-processing for the Cornell Movie-Dialogs Corpus, created by the wonderful Cristian Danescu-Niculescu-Mizil and Lillian Lee at, guess where, Cornell University. This is an extremely well-formatted dataset of dialogues from movies. It has 220,579 conversational exchanges between 10,292 pairs of movie characters, involving 9,035 characters from 617 movies with 304,713 total utterances. The corpus is distributed together with the paper "Chameleons in Imagined Conversations: A new Approach to Understanding Coordination of Linguistic Style in Dialogs.", which was featured on Nature.com. It is a fascinating paper that highlights several cognitive biases in conversations that will help you make your chatbot more realistic. The pre-processing is pretty basic.

3.2 Solution Structure

We have a seq2seq model that will construct two RNNs for the input message and the reply message, and train them to maximize the probabilities in the thought vector of the reply given the input. Padding will be added and bucketing used based on requirements of the seq2seq model for both the input and output to be fixed-lengths. The LSTM cells for each word can be tested with Attention.

LSTM: Long-Short-Term-Memory (LSTM) is a variant of RNN which help preserve the error that can be back propagated through time and layers. By maintaining a more constant error, they allow recurrent nets to continue to learn over many time steps (over 1000), thereby opening a channel to link causes and effects remotely. LSTMs contain information outside the normal flow of the recurrent network in a gated cell. Information can be stored in, written to, or read from a cell, much like data in a computer's memory. The cell makes decisions about what to store, and when to allow reads, writes and erasures, via gates that open and close. Unlike the digital storage on computers, however, these gates are analog, implemented with element-wise multiplication by sigmoid, which are all in the range of 0 to 1. Those gates act on the signals they receive, and similar to the neural networkâĂŹs nodes, they block or pass on information based on its strength and import, which they filter with their own sets of weights. Those weights, like the weights that modulate input and hidden states, are adjusted via

the recurrent networks learning process. That is, the cells learn when to allow data to enter, leave or be deleted through the iterative process of making guesses, back propagating error, and adjusting weights via gradient descent [1][2].

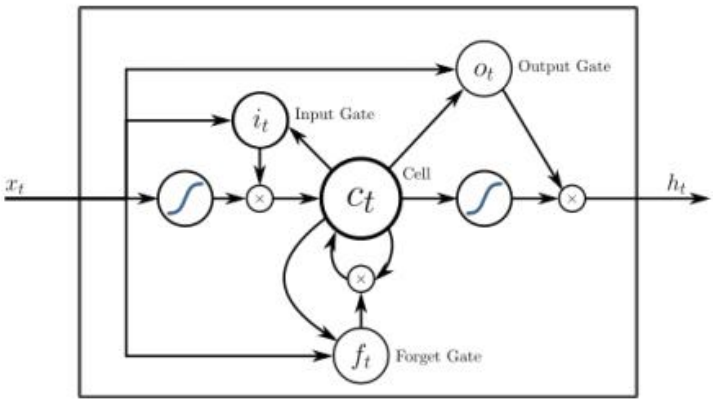


Figure 3. A LSTM Block

GRU: A gated recurrent unit (GRU) is basically an LSTM without an output gate, which therefore fully writes the contents from its memory cell to the larger net at each time step. [9]

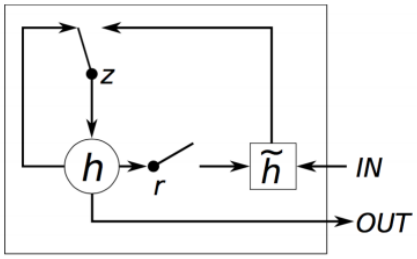


Figure 4. A Gated Recurrent Unit Block

The input will be a minimally filtered dataset of Movie Corpus. It will select dialogue that are part of a conversation, targeting textual message only and limiting to a three-tier conversation for now. The output will be a message reply.

In this paper consists of two parts: the movie corpus, and the neural network model. For the latter, we have two implementations: the seq2seq version and the simple RNN version. The files are described as

Movie Corpus: The data file under the research consist of the standard Cornell movie dialog corpus including encoders and decoders.[7][8]. Further this file consists of test.enc, test.dec, train.enc and train.dec.

Seq2Seq Model: Sequence-To-Sequence model introduced in Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation has since then, become the Go-To model for Dialogue Systems and Machine Translation. It consists of two RNNs (Recurrent Neural Network): An Encoder and a Decoder. The encoder takes a sequence (sentence) as input and processes one symbol(word) at each timestep. Its objective is to convert a sequence of symbols into a fixed size feature vector that encodes only the important information in the sequence while losing the unnecessary information. You can visualize data flow in the encoder along the time axis, as the flow of local information from one end of the sequence to another [5].

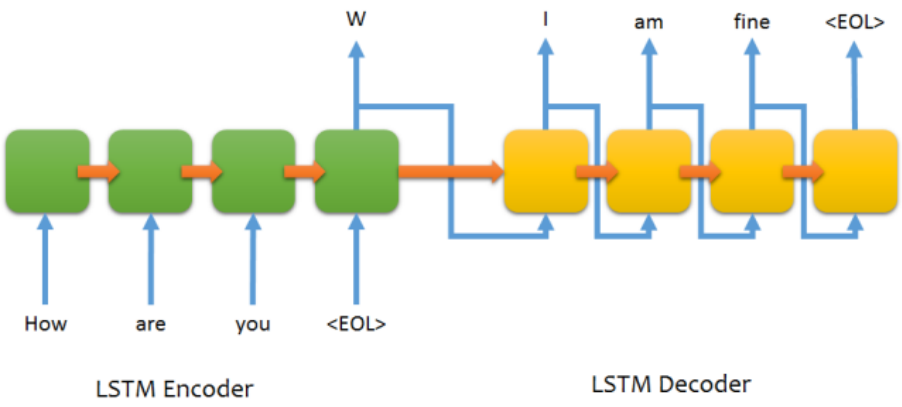


Figure 5. Seq2Seq Model

4. CONCLUSION

We relied on a completely open-ended dataset with 0 filtration of the quality of sentences that went in. Therefore, we would see conversations where the topic was barely noticeable, thus to our model these were nothing but noise. Often, we observed conversations where the conversations would be about two different topics at the same time, and this was not useful to our model as well. The example conversation below highlights both of these issues. Therefore, the optimal dataset may have been one where conversations are single-topiced, flow smoothly, and start and end with clear announcements.

REFERENCE

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. arXiv preprintarXiv:1409.0473(2014).

[2] Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009.Curriculum learning. In Proceedings of the 26th annual international conference on machine learning. ACM, 41–48.

[3] Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint Copying and Restricted Generation for Paraphrase.. In AAAI. 3152–3158.

[4] Yllias Chali and Sadid A Hasan. 2015. Towards topic-to-question generation. Computational Linguistics41, 1 (2015), 1–20.

[5] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. 2014.Empirical evaluation of gated recurrent neural networks on sequence modeling. arXiv preprint arXiv:1412.3555(2014).

[6] Kevin Clark, Urvashi Khandelwal, Omer Levy, and Christopher D Manning. 2019.What Does BERT Look At? An Analysis of BERT's Attention . arXiv preprintarXiv:1906.04341(2019).

[7] Guy Danon and Mark Last. 2017. A Syntactic Approach to Domain-Specific Automatic Question Generation .arXiv preprint arXiv:1712.09827(2017).

[8] Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In Proceedings of the ninth workshop on statistical machine translation. 376–380.

[9] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. arXivpreprint arXiv:1810.04805(2018).

[10] Xinya Du and Claire Cardie. 2018. Harvesting paragraph-level question-answer pairs from wikipedia.arXiv preprint arXiv:1805.05942(2018).

[11] Bottou, L. 2010. Large-scale machine learning with stochastic gradient descent. In Proceedings of COMPSTAT'2010, pages 177–186. Springer(2010)