Deep Learning Dialog Generation

Spencer Ottarson
Computer Science and Engineering
Michigan State University
East Lansing, MI
Ottarso5@msu.edu

James Mariani
Computer Science and Engineering
Michigan State University
East Lansing, MI
Mariani4@msu.edu

Abstract

The advancement of Natural Language Processing (NLP), Artificial Intelligence (AI), and Deep Learning have led to interesting and unique fields that combine techniques used in all three individually. One of the areas gaining traction is human to machine communication. Human to machine communication revolves around creating systems that can respond similarly to how a human would respond, often in the form of a ChatBot. This paper creates a ChatBot using NLP, AI, and Deep Learning techniques.

1 Introduction

The intersection of Natural Language Processing (NLP) and Artificial Intelligence (AI) has yielded many interesting and rewarding fields of study. One of the most intriguing areas is the idea of human to machine communication, sometimes in the form of a ChatBot.

A ChatBot is a computer program that can conduct a conversation with a human and is often designed to simulate how a real human would respond in a given situation.

ChatBots can be used for many purposes, one of the most common is for simple entertainment. In addition, many companies are also using ChatBots to help with customer engagement. Different companies use ChatBots to promote their products, help customers order products, provide support for products etc.

Many big names in the world of computer science have invested time and resources into ChatBots, including Google, Amazon, Samsung and Microsoft. The world of ChatBots is also becoming more and more pervasive, seeing as most people walk around with a ChatBot in their pocket or home in the form of Alexa, Siri, the Google Assistant, etc. In this project, we develop a ChatBot using deep learning techniques.

2 Background Information

2.1 Neural Networks

In recent years, the popularity of artificial neural networks, or multilayer perceptrons, has grown tremendously. Though neural networks were first proposed as early as 1940, they were considered impractical due to their computational complexity. It was not until the work by (Hinton et al., 2006) which led to an efficient algorithm of training that neural networks gained the following they have today.

A typical feed-forward neural network consists of an input layer, any number of hidden layers, and an output layer.

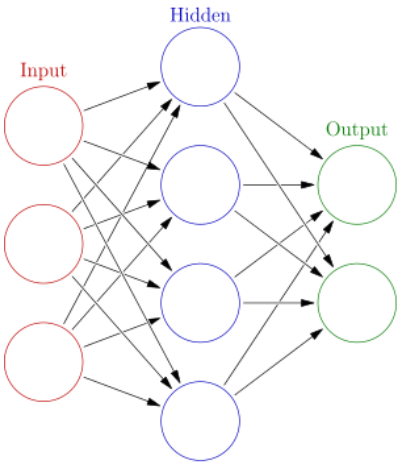

Figure 1: A single hidden layer neural network

Figure 1 shows a simple one hidden layer neural network. The input is represented by an *n*-diminesional vector of the data. The input vector is multiplied by a matrix of weights to get the hidden layer, which is then multiplied again by a matrix to get the output layer.

Training a neural network is about finding the weights of each of the connections from one layer to the next. This is typically done using back-propagation (Rumelhart et al., 1988) combined with an optimizer, such as gradient descent (Bottou, 2010), which minimizes the difference between expected output and the output on training data. Neural networks can contain any number of hidden layers, and the activation function from one layer to the next can be any differentiable function.

2.2 Recurrent Neural Networks

Traditional feed forward neural networks achieve very good results for numerous problems, but they have some drawbacks. The networks require an input of fixed size, and produce output of a fixed size. Problems in natural language usually have inputs and outputs of varying length, such as words or sentences. Recurrent neural networks (RNNs) allow operations over sequences of vectors.

The main difference between a traditional neural network and an RNN is that each layer takes 2 inputs: the output of the previous layer as well as the its own output from the previous information. An RNN is thus stateful, allowing information gained from one input to be stored, and influence the output of the next output. Intuitively this makes sense to use with language processing, as the meaning of a sentence is a result of all of the words in the sentence in sequence. An example RNN structure is shown in figure 2.

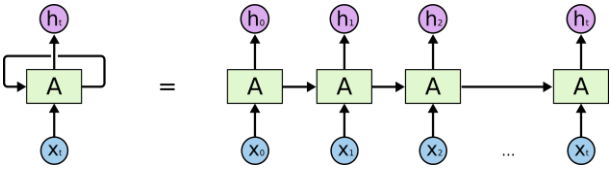

Figure 2: Recurrent Neural Network Structure

2.3 LSTMs

In theory, RNNs should work well for processing variable length sequences such as sentences. In practice, unfortunately, results are generally not very good. Although RNNs keep state information from one input to the next, they tend to lose that information very quickly and do not capture long term dependencies. A special kind of RNN, called a Long-Short Term Memory (LSTM) network (Hochreiter and Schmidhuber, 1997) was designed explicitly to solve this problem.

LSTMs add another vector of information that is passed between iterations of the RNN, referred to as the *cell state*.

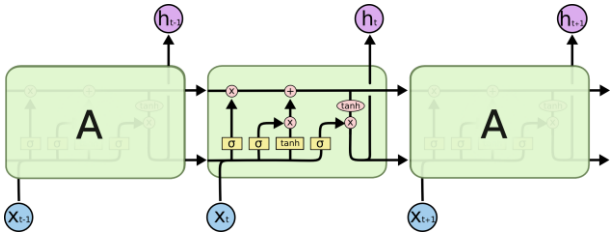

Figure 3: Long-Short Term Memory Model

Figure 3 show the basic design of an LSTM. The cell state line runs through the node at every iteration, with only minor modifications each time. At each iteration, some new information may be stored in the cell state, some information may be forgotten, and some information is output from the cell state. Information in the cell state is passed

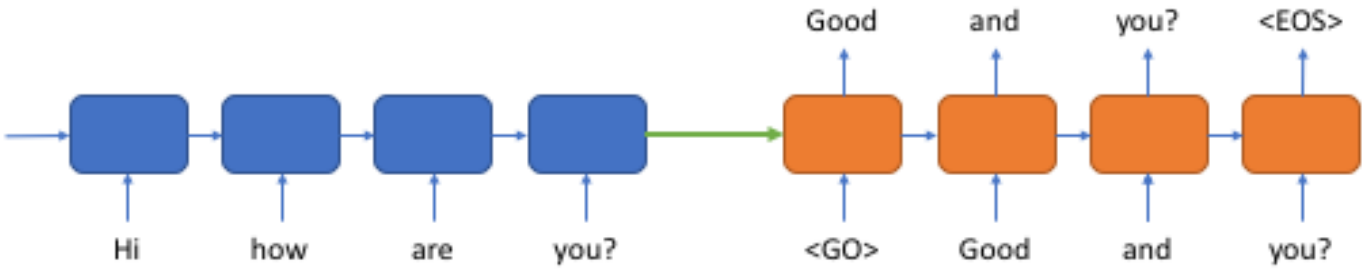

Figure 4: Sequence-to-Sequence Model

through each iteration, and in practice is very efficient at keeping around data for long periods of time.

2.4 Sequence-to-Sequence Models

In a general sense, sequence-to-sequence models are encoders and decoders. A generic sequence-to-sequence model is shown in Figure 4. In a sequence-to-sequence system the encoder and the decoder are two separate RNNs. The blue nodes on the left of figure 4 represent the encoder, which is an RNN. The orange nodes on the right represent the decoder, and this is a completely separate RNN.

The encoder takes input, such as a sequence of words: word1, word2, word3, etc. Most of the outputs of the encoder RNN are not used at all, and the only useful result of the encoder is its final state. This final state is a vector representation of the entire inputted sentence. This vector representation is then used as the initial state of the decoder RNN.

The vector representation given by the encoder is then fed as input into the decoder. As mentioned earlier, the decoder is just an RNN. It starts with a special <GO> tag and, unlike the encoder, this time the actual output is used. The output is word1, which is then fed back into the model to get the second output, and so on, ultimately ending in a <EOS> tag. An example of this can be seen in the decoder section of figure 4.

The main issue with the sequence-to-sequence model is that it is purely one input leading to one response. This is based on training data and is completely probabilistic. There is no long-term memory, and really no short-term memory either. It is simply one input to one output.

3 Related Work

Previous work has been conducted by many groups studying how to create a conversational agent through deep learning methods. The work done by (Sutskever et al., 2014) was revolutionary in figuring out how to use LSTMs to map an input sequence to an output sequence. In terms of a conversation, a sentence can be considered the input sequence of words. The predicted output can be used as the response. This model was implemented as a conversational agent and described in (Vinyals and Le, 2015). (Shang et al., 2015) also described a neural network based response machine which achieved quality results.

The biggest problem with some of these initial models was that information is not kept around throughout the conversation, they only consist of a single response to a single sentence. In many cases this leads to inconsistent and strange responses, such as the following:

message:	How old are you?
response:	16 years old
message:	What is your age?
response:	18

Some work has already been done to mitigate these issues, but it is still a large and difficult to solve problem. The research by (Li et al., 2016) developed a system in which different models were trained to capture the characteristics of different speakers. The network used a combination of a general model and the individual models to generate responses.

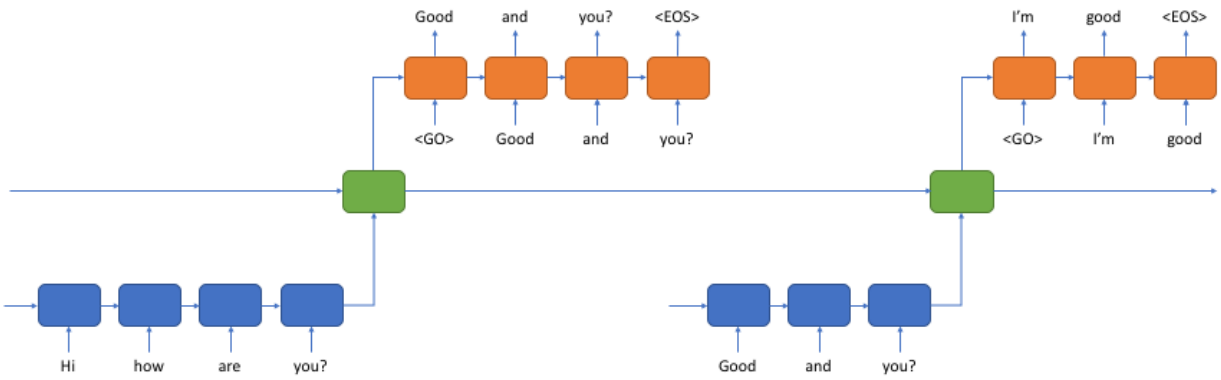

Figure 5: Our Proposed Model

Another approach is to use a hierarchy of neural network models to capture different information, as shown by (Serban et al., 2015). They used a hierarchical model to capture both conversation information and sentence information.

As mentioned earlier, one of the main issues with existing approaches is that none of the linguistic context is saved. This leads to exchanges that output inconsistent responses. In addition to this, existing methods fall short because no intention is captured, and it they usually respond only to a single utterance. Our system will try to mitigate these problems.

4 Our Approach

The main goal of our system is to create a ChatBot using deep learning techniques that can save some of the linguistic context of a conversation. The main idea behind the system that we developed is that a sentence is a sequence of words, and a conversation is a sequence of sentences.

4.1 Design Overview

The main strength of RNNs is that they are good at modeling sequences of arbitrary length. We have discussed the concept that a sentence is a sequence of words, and a conversation is a sequence of sentences. Therefore, one RNN is used to capture the sequence of words, and a separate RNN is used to capture the sequence of sentences. Figure 5 shows the basic structure of our model.

The encoder on the far left is almost exactly the same as discussed earlier in the sequence-to-sequence model section. The input into the encoder is the sequence of words, and the output is a vector representation of the sentence. The original sequence-to-sequence model on its own would then take the vector representation of the sentence as input into the decoder. Our system, however, introduces another RNN in the middle which takes the encoded sentence representations as input and gives another vector representation of the conversation state as output. This middle RNN is shown as the green nodes in figure 5.

In the original sequence-to-sequence model, the output of the encoder is fed as input into the decoder, but our implementation actually takes the output of the middle RNN and uses that as the input into the decoder. It could be the case that the output from the encoder and the middle RNN are the same, but it can also use the previous sentences and context to gain more information and give a better representation of the conversation as a whole, instead of just the single input into the encoder. The encoder and the decoder are not aware that there is a middle RNN that is collecting this data and aggregating it. The encoder still produces a vector as output, and all the decoder sees is a vector coming in as input.

4.2 Design Specifics

The specifics used in our implementation of our deep learning dialog generation system are described in this section.

Our system was trained using sentence triples. This means that training was done based on a sequence of sentence1=>response1=>sentence2. This can be visualized in figure 5. Sentence1 is used as the input into the first encoder, response1 is used as the expected output of the first decoder. Response1 is then again used as the input into the

second encoder, and sentence2 is used as the expected output of the second decoder. There will be further discussion of training on triples in the next few sections of the paper.

In our decoders, the output of each node in the decoder is a one-hot encoded vector, which is then used as the input to the next node of the decoder.

The number of layers of the neural network that we used was 3, and we used a hidden size of 1024. Hidden size is the vector output size from each layer.

We used bucket sizes of 9, 13, 18, and 24, and these are used for padding on the inputs and the outputs.

We had an overall vocab size of 18193, which is equal to 18192 unique words in addition to "unknown." In order for a word to be counted as unique, it had to appear 5 times or more in the training data. So we had 18192 unique words that were used more than 5 times, and every word that was not used 5 times or more then became "unknown." There is a more detailed description of the dataset in the next subsection.

4.3 Dataset

The dataset that was used for training purposes in our system was the Cornell Movie Dialog Corpus. We initially were using the Ubuntu Dialog Corpus because it is absolutely massive. We ended up switching to the Movie Dialog Corpus for one main reason. The Ubuntu Dialog Corpus is conversations taken from a Ubuntu message board where people were discussing technical issues, trouble shooting problems, and other related discussions. We decided that because of the mostly technical nature of the conversations, and the heavy use of acronyms we were going to switch to the Movie Dialog Corpus, which represented a more natural, flowing conversation.

The Cornell Movie Dialog Corpus is data that has been extracted from raw movie scripts. There are over 220000 conversational exchanges, over 10200 pairs of movie characters, and over 300000 utterances in the data set.

5 Evaluation

This section of the paper will show example outputs from our system and give a brief explanation and analysis. A more thorough discussion of the results and outcomes can be found in section 6.

When deciding how to evaluate our model we considered many possible ways of quantifying success or failure. Ultimately, there is no ideal way to quantify the responses that a ChatBot gives when prompted. We decided to evaluate our system qualitatively based on a few different goals. The first, and main goal, is whether or not our system can capture linguistic context. Did we develop a system that can save information from previous exchanges and use it to help construct future responses. The second goal that we wanted to look at is how many turns is this information held.

The first example, shown below is an example of the system when the sequence-to-sequence model was used by itself with no extra middle layer. As can be seen by the exchange, it does not make much sense. There is clearly room for improvement upon a simple sequence-to-sequence system.

```
====================================
HUMAN ++++ hello
BOT ++++ where ? ? ? ? . . . . .
HUMAN ++++ what is your name?
BOT ++++ nothing . . . . . .
HUMAN ++++ i am spencer
BOT ++++ you ? ? ? . . . . . .
HUMAN ++++ yes
BOT ++++ you ? ? ? ? ? ? ? ? ? ?
HUMAN ++++ no
BOT ++++ you ? ? ? . . . . . .
HUMAN ++++ stop
BOT ++++ what . . . . . .
HUMAN ++++ what is your name?
BOT ++++ nothing . . . . . .
HUMAN ++++ what are you called?
BOT ++++ i . . . . . .
HUMAN ++++ what?
BOT ++++ you . . . . . .
====================================
```

As seen in the previous example, when asked the same question, the ChatBot only using sequence-to-sequence gives an identical response. This makes logical sense because every output is based solely on the immediate input. In the next example, below, our system has been implemented and has begun, but not yet completed training. When asked the same question twice, we got two different responses. Two of these instances are highlighted in the exchange below.

==========
HUMAN ++++ hello
BOT ++++ how you wanted to do ?
HUMAN ++++ hello
BOT ++++ i thought hi .
HUMAN ++++ hi
BOT ++++ i thought i had you to die .
HUMAN ++++ what?
BOT ++++ i ' m lonely .
HUMAN ++++ why are you lonely
BOT ++++ i like you , i can ' t .
HUMAN ++++ how are you?
BOT ++++ fine .
HUMAN ++++ how are you?
BOT ++++ fine . . . you are .
==========

The actual responses to what was said do not make perfect sense, but it is important to show that we got two different responses to the same question.

The exchange below shows a promising result regarding saving of linguistic context.

==========
HUMAN ++++ What day is it?
BOT ++++ it ' s tuesday .
HUMAN ++++ What is today?
BOT ++++ Tuesday.
HUMAN ++++ what color is the sky?
BOT ++++ - - five .
==========

The example above shows an instance of relative success, but for every good result, we also got bad results, an example is shown below.

==========
HUMAN ++++ What time is it?
BOT ++++ nine - thirty .
HUMAN ++++ What is the time?
BOT ++++ morning .
==========

The exchanges below are examples of general use of our system as a ChatBot.

==========
HUMAN ++++ my name is spencer
BOT ++++ your name .
HUMAN ++++ yeah
BOT ++++ okay .
HUMAN ++++ is spencer
BOT ++++ i love you , sam .
HUMAN ++++ i love you too
BOT ++++ love me .
HUMAN ++++ i do
BOT ++++ need you been love to her ?
HUMAN ++++ nope
BOT ++++ want you met ?
==========

==========
HUMAN ++++ hi
BOT ++++ hi .
HUMAN ++++ What was the civil war?
BOT ++++ it was great in the same people .
HUMAN ++++ When was the civil war?
BOT ++++ why were you done it ?
HUMAN ++++ Give me the definition of life
BOT ++++ no !
HUMAN ++++ what is the meaning of life
BOT ++++ the most human criminal in the death .
HUMAN ++++ Hello
BOT ++++
HUMAN ++++ Hello
BOT ++++ i got to eat some action .
HUMAN ++++ Hi
BOT ++++ what are you doing ?
HUMAN ++++ talking to you
==========

==========
HUMAN ++++ what are your plans?
BOT ++++ plans .
HUMAN ++++ fun
BOT ++++ we could bring up the way .
HUMAN ++++ what is my name
BOT ++++ what do you want to remember ?
HUMAN ++++ what is my name?
BOT ++++ what business is it now ?
HUMAN ++++ wrong
BOT ++++ what ?
HUMAN ++++ wrong!
BOT ++++ huh ?
HUMAN ++++ You are wrong
BOT ++++ what ?
HUMAN ++++ What day is it?
BOT ++++ it ' s tuesday .
HUMAN ++++ What is today?
BOT ++++ Tuesday.

HUMAN ++++ what color is the sky?
BOT ++++ - - five .
==========

As you can see, the results from general use of the ChatBot are not great. A few of the responses from the ChatBot seem to make some sense, but overall most responses are gibberish.

Analysis of these exchanges, thoughts on why we got the results we did, and ways to possibly fix the system are discussed in sections 6 and 7.

6 Discussion of Results

This section of the paper will be focused on a discussion of why we got the results that we did, and possible ways of improving this system.

The results of using this system were both unexpected, and disappointing. There are some instances where the ChatBot is successful in interpreting linguistic context, as evidenced by one of the exchanges above, but there are also many times where this is not the case.

In addition to the failure to capture context, many of the responses given by the ChatBot were generally nonsensical and not correct. This is caused in large part by how we are training the system. Training using triples is not ideal, and it is the main cause of the failure of our system. We discussed, and tried to implement a system in which we could train on four and five sentence sequences, but doing so introduced an exponential increase in time needed for training. In the end, it was not feasible to train on sequences larger than three.

While a large portion of the results were negative, there are also a few positives that can be taken from our work. The first positive is that in some cases our system was able to successfully interpret the linguistic context and respond accordingly. That is a huge achievement and proof that our system can be viable given more training time.

The second positive was shown in the second example. In that example, our system responded differently to the same input sentence. This shows that our system is indeed making different choices of response based on previous sentence and responses. If our system was not taking into consideration past responses, then the same question would get the exact some response every time.

Our system has shown that what we are trying to do is possible. Given the time and resources to be able to train on sentence sequences of four or larger, our system will continue to improve.

7 Future Work

There are two main areas to focus on when it comes to future work. The first has already been heavily discussed, and it involves training the system on sentence sequences of larger than 3. This will give the system more context and allow it to make better decisions on responses.

The second area of future work involves training the system for a certain goal. At the moment, our system is training using random weights, but goals can be trained by giving the starting weights different values based on any scenario you want to train.

8 Conclusion

This paper discussed issues and approaches related to developing a ChatBot using deep learning techniques. Many deep learning models and techniques were discussed, and a proposed deep learning chatbot was implemented and tested.

The proposed model involved taking a standard sequence-to-sequence model, and adding a second RNN as a middle layer. The main idea comes from the fact that sentences are just a sequence of words, and a conversation is just a sequence of sentences. The original RNN handles the sequence of words, and the middle layer RNN deals with the overall context of the conversation.

The discussed method was implemented and the results were somewhat unexpected. The system performed far below expectation, but there were still some positives to be taken from the results. There were certain cases where it is clear that linguistic context is saved and utilized. There is also definitive proof that the system makes different decisions on output if there are previous sentences in the sequence to give some context to the conversation as a whole.

The main issue with the system implemented in this paper is the fact that training takes place using sentence triples, instead of the more accurate sequences of four, five, or even six sentences. This falls under the future work that will be taken.

Acknowledgments

We would like to thank Dr. Chai and Michigan State University Department of Computer Science and Engineering

References

L´eon Bottou. 2010. Large-scale machine learning with stochastic gradient descent. *In Proceeding of COMPSTAT'2010*, pages 177-186. Springer

Geoffrey E Hinton, Simon Osindero, and Yee-Whye The. 2006. A fast learning algorithm for deep belief nets. *Neural computation*, 18(7): 1527-1554.

Sepp Hochreiter and Jurgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735-1780.

Jiwei Li, Michel Galley, Chris Brockett, Georgios P Spithourakis, Jianfeng Gao, and Bill Dolan. 2016. A persona-based neural conversation model. *axXiv preprint arXiv:1603.06155*

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1988. Learning representations by back-propagating errors. *Cognitive modeling*, 5(3):1.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808*.

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. arXiv preprint arXiv:1503-02364.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information procession systems*, pages 3104-3112.

Priol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.