CHAT BOT USING NEURAL NETWORKS

by Pavithra Govardhanan and Keerthi Dharam, December 17, 2019

ABSTRACT

A chatbot is a computer software program that works on Artificial Intelligence (AI) and Machine Learning (ML). In this project, we developed a chatbot, which provides a genuine and accurate answer for any user query from the dataset. In this project, we used the WikiQA dataset as our reference dataset. We prepared a dataset in the JSON file format with relevant questions from the WikiQA dataset. We trained the chatbot using two types of neural networks: CNN (Convolutional Neural Network) and RNN (Recurrent Neural Network). These two models are trained with great accuracy. Our results show the relationship between the number of training times and the quality of the language model used for training our model both affect the quality of its prediction output.

Keywords: CNN, RNN, Neural Networks.

INTRODUCTION

Although technology has changed the way how humans interact with digital systems, accessibility is still largely limited. Chatbots allow for seamless interaction between software systems and humans through natural language processing, giving users an illusion of talking to another person.

Various works have been emerged in recent years related to query answering. The chatbot provides the most suited response, based on the information obtained during the understanding phase. There two different models for answering task, retrieval-based models, and generative based models.

The job of a chatbot is to be able to determine the best response for any given message that it receives. This “best” response should either be

(1) an answer for the sender’s question or,

(2) give the sender relevant information or,

(3) ask follow-up questions or,

(4) continue the conversation realistically.

The chatbot needs to be able to understand the intentions of the sender’s message, determine what type of response message (a follow-up question, direct response, etc.) is required, and follow correct grammatical and lexical rules while forming the response.

It’s safe to say that modern chatbots have trouble accomplishing all these tasks. In this work, we aim to develop a chatbot capable of maintaining a coherent conversation on popular topics in the realms of technology.

EXISTING SYSTEM

There is one chatbot called Tensorflow chatbot by Jalaj Thanaki that uses Recurrent Neural Network with Cornell Movies Dialog Dataset. It uses tensorflow 0.12.1 version and python 2.7 version which is outdated. It takes a few hours to train the model, which accounts to be one of the drawbacks of the chatbot. The other existing system is intent classification that uses Convolutional Neural Network with glove word embeddings. Its accuracy is low, around 77%. The recurrent neural network was not implemented. Both the existing systems are implemented with single neural network. Also, the input datasets are not in the pattern-response format.

PROPOSED SYSTEM

Our project is a conversational chatbot that uses two types of neural networks: CNN and RNN. We have a large dataset as our reference. The dataset has labels like tags, patterns, responses which would be very useful in pre-processing. Moreover, our chatbot’s output is not only text but also it delivers its output through voice. We can still modify the voice with volume, gender etc. We have achieved greater accuracy for both the models by performing these actions.

CHATBOT MODELS

There are two types of models namely:

Retrieval Based models

Generative Based models

For retrieval-based models, the answering process is based on reference to a predefined set of answers. Chatbots developed for clients and customers, comes under this category. The responses are based on a repository of predefined responses. Based on the input and context from the user, a certain heuristic is defined to select the appropriate response. This heuristic could be as simple as a rule-based expression match, or as complex as an ensemble of Machine Learning classifiers. [1]

The other model is generative-based models which is based on machine learning. They are not dependent on a fixed set of answers. Markov chains have originally been used for text generation.[2] Generative models do not use responses that are already defined, instead, they create new responses and generate new text. Generative based models take inspiration from Machine Translation techniques, but here instead of translating, it learns the user input and output response in the form of translations from input to output. These models tend to be harder to train due to the huge amounts of training data required.

NATURAL LANGUAGE PROCESSING

Natural Language Processing (NLP) is the ability of a computer program to understand human language. NLP is a component of artificial intelligence (AI). NLP is a subfield of computer science, information engineering and artificial intelligence concerned with the interactions between computers and human (natural) languages, how to program computers to process and analyze large amounts of natural language data. [3] Challenges in natural language processing frequently involve speech recognition, natural language understanding, and natural language generation. NLP can be used to interpret the free text and make it analyzable. Sentiment analysis is another primary use case for NLP.

For NLP tasks in deep learning, the input to models is in the form of sentences, which is given in the form of vectors. An element in the vector row is referred to as a token. Figure1 represents word embedding. Usually tokenizing is defined as using separate words, but sometimes character level tokenizing is done. These vectors represent word embeddings of each sentence (probability or frequency distributions) but sometimes are an index for the word in a dictionary (encoding).

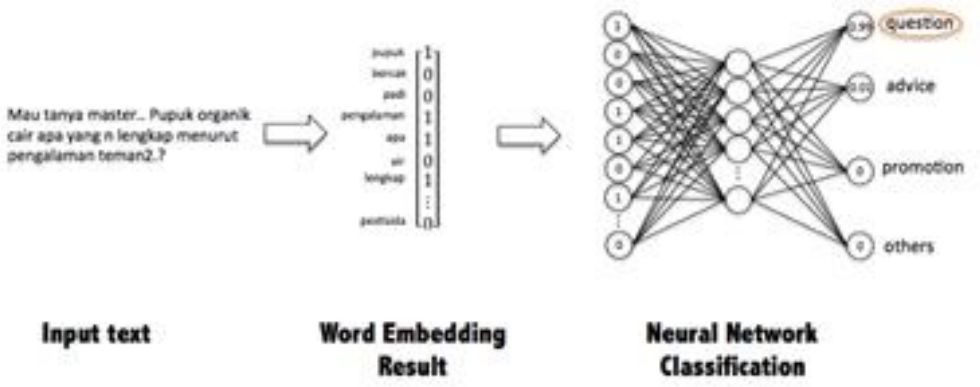

Figure 1 Word Embedding

DATASET

For training a chatbot model, the data needs to be in a question-response pair form. The conversation should have a flow and context and should not be random utterances. We used WikiQA dataset as reference, which has different columns. Among those, we used two columns questions and answers to build the dataset for our chatbot. The hierarchy of WikiQA dataset comments can be visualized as follows:

Question pattern 1

- Response 1 to question pattern 1
- Response 2 to question pattern 1
- Response 3 to question pattern 1

We want to transform this data into comment-reply pairs, so we can define input-output for the chatbot. Also, we added variations of the same question that the user can ask.

Each comment in our dataset has metadata in JSON format(training.json) containing tags, patterns, responses, and context_set. The *"tag"* is nothing but the domain of the question and answers. Eg: if the question is related to computers then the tag will be computers. The label *"pattern"* contains a different type of question that can be asked by the user for that particular domain. The label *"responses"* contains a different type of answer which will be selected by the model and the answer will be displayed to the user. We have also added a few patterns apart from those in the dataset, such as greetings, goodbye, emotions, bot profile, computers.

METHODOLOGY

The methodology involved in our project is Data collection, Data Preprocessing, Training and testing the model.

Data Collection*:* The data in training.json has tags, patterns, responses, and context_set. We obtained the training.json from the WikiQA dataset. Our dataset is a collection of question-answer pairs identified by a tag.

Data Preprocessing*:* The data which is collected is then preprocessed by various steps namely, Tokenization, Stemming, Bag of words.

Tokenization*:* Tokenization is the process of tokenizing or splitting a string, text into a list of tokens. Naturally, before any real text processing is to be done, the text needs to be segmented into words, punctuation, numbers, etc. This is important because the meaning of the text depends on the relations between words in that text.

Stemming: Stemming is the process of reducing words to their stem, base or root form. The words are reduced to root by removing inflection through dropping unnecessary characters, usually a suffix. For example: "Flying" is a word and after stemming the word becomes "Fly".Stemming is used in information retrieval systems like search engines. It is used to determine domain vocabulary.

Bag of words*:* After the initial preprocessing phase, we need to transform the text into a meaningful array of numbers. This is because the Neural network doesn't understand strings. So they are converted into numbers. This is called "a bag of words". And the process of converting strings into numbers is called "One hot encoding".

The bag-of-words describes the occurrence of words within a document. The bag-of-words model is simple to understand and implement. It is a way of extracting features from the text for use in machine learning algorithms. In this approach, we use the tokenized words for each observation and find out the frequency of each token.

Feature Extraction: The mapping of textual data to real-valued vectors is called feature extraction. One of the simplest techniques to numerically represent text is Bag of Words. Bag of

Words (BOW): We make a list of unique words in the text corpus called vocabulary. Then we can represent each sentence or document as a vector with each word represented as 1 for present and 0 for absent from the vocabulary.

For increasing the accuracy, we did hyperparameter tuning, (i.e.,) changing the parameter values such as epochs, learning rates to increase the accuracy of the model.

Hyperparameter Tuning*:* Hyperparameters for the deep neural network is difficult as it is slow to train a deep neural network and there are numerous parameters to configure. Various hyper parameters are,

Learning rate: The learning rate controls how much to update the weight in the optimization algorithm. We can use a fixed learning rate, gradually decreasing learning rate, momentum-based methods or adaptive learning rates, depending on our choice of optimizers such as SGD, Adam, Adagrad, AdaDelta or RMSProp.

Number of epochs: Number of epochs is the number of times the entire training set pass through the neural network. We should increase the number of epochs until we see a small gap between the test error and the training error.

Batch size: Mini batch is usually preferable in the learning process of convnet. A range of 16 to 128 is a good choice to test with. We should note that convnet is sensitive to batch size.

Activation function: Activation function introduces non-linearity to the model. Usually, the rectifier works well with convnet. Other alternatives are sigmoid, tanh and other activation functions depending on the task.

Number of hidden layers and units: It is usually good to add more layers until the test error no longer improves. The trade-off is that it is computationally expensive to train the network. Having a small number of units may lead to underfitting while having more units are usually not harmful with appropriate regularization.

Weight initialization: Initialize the weights with small random numbers to prevent dead neurons, but not too small to avoid zero gradients. Uniform distribution usually works well.

Dropout for regularization: Dropout is a preferable regularization technique to avoid overfitting in deep neural networks. The method simply drops out units in the neural network according to the desired probability. A default value of 0.5 is a good choice to test with.

NEURAL NETWORKS

Neural networks are the core of deep learning. It is a field that has practical applications in many different areas. Today neural networks are used for image classification, speech recognition, object detection, etc. There are various types of neural networks among which we used CNN and RNN because of the promising results.

The basic neural network look like the one shown in Figure2

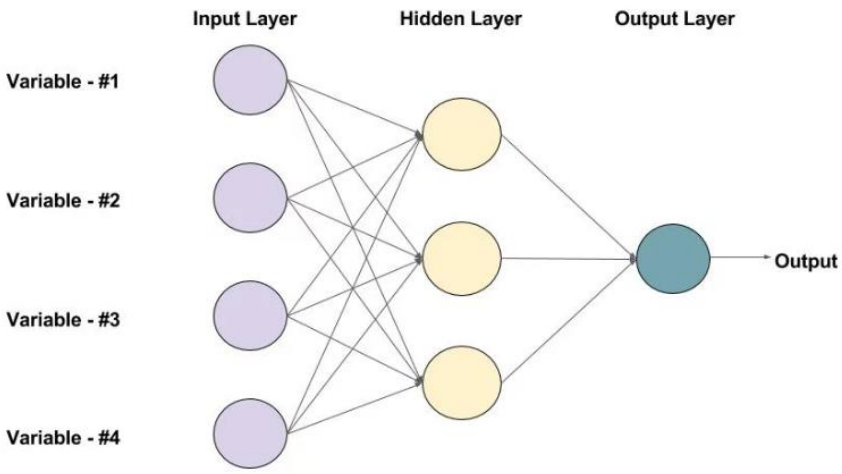

Figure 2 Structure of Basic Neural Network

CONVOLUTIONAL NEURAL NETWORK (CNN)

A Convolutional Neural Network (CNN) is a variation of (Multilayer Perceptron) MLP with at least one convolutional layer. The convolutional layer reduces the complexity of the network by applying a convolution function on the input and passing the output to the next layer, analyzing a part of the data (sentence/image) at a time. Since the complexity is reduced with CNN, the network can be much deeper and handle more complex data.

Keras provides an implementation of the convolutional layer called a Conv1D shown in Figure3. Convolution layer is the first layer to extract features. Filter size and kernel size must be provided as parameters to the Conv1D layer. Stride is the number of pixel shifts over the input matrix. There are three types of padding: full, same and valid.

Among these three we used “valid” padding is default one. Rectified Linear Unit (ReLU) activation function is used for a non-linear operation. ReLU’s purpose is to introduce non-linearity in our ConvNet.

Here, we are adding one Conv1d layer followed by the max-pooling layer to reduce the spatial dimensions of the output volume.

After this, a Flatten layer is added to convert the pooled feature map to a single column that will be passed to the fully connected layer.

Finally, this is followed by a fully-connected layer with a softmax activation function with an optimizer being “Adam” optimization.

The model is thus compiled and saved.

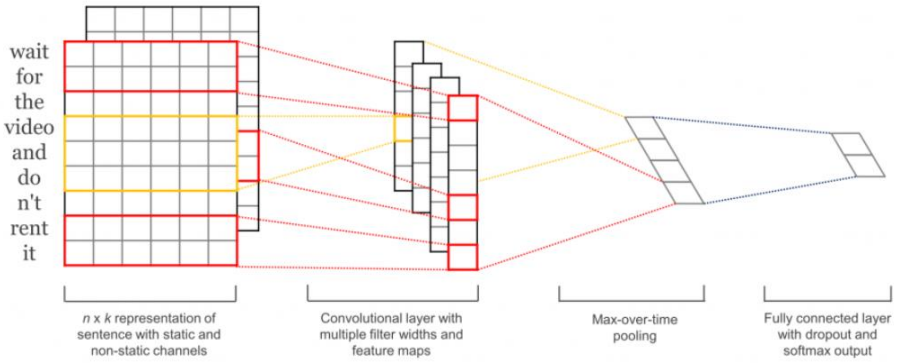

Figure 3 Convolutional Neural Network

RECURRENT NEURAL NETWORK (RNN)

A Recurrent Neural Network (RNN) is designed to preserve the previous neuron state. This allows the neural network to retain context and produce output based on the previous state. This approach makes RNNs desirable for chatbots as retaining context in a conversation is essential to understand the user. RNNs are extensively used for NLP tasks such as translation, speech recognition, text generation and image captioning.

Simple RNN (shown in Figure4) processes batches of sequences, like all other Keras layers. This layer takes inputs of shape (batch_size, timesteps, input_features). Simple RNN can be run in two different modes: it can return either the full sequences of successive outputs for each timestep (a 3D tensor of shape (batch_size, timesteps, output_features)), or it can return only the last output for each input sequence (a 2D tensor of shape (batch_size, output_features)). These two modes are controlled by the return_sequences constructor argument.

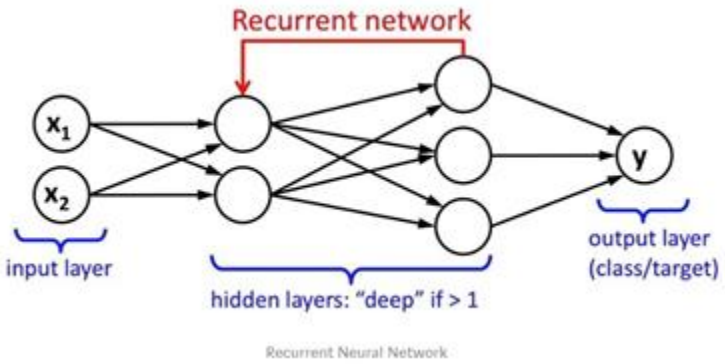

Figure 4 Recurrent Neural Network

First, the preprocessed data is sent to the embedding layer, where the bag of words task is completed. Then the SimpleRNN layer takes filter size, activation function, and return_sequence as its parameters. After this, a Flatten layer is added to convert the features to a single column that

will be passed to the fully connected layer. Finally, this is followed by a fully-connected layer with a softmax activation function with an optimizer being “Adam” optimization.

TEAM CONTRIBUTION

Overall, we completed this project dividing the work equally between us. The work contributed by each of us is listed below:

Keerthi Dharam:

The WikiQA dataset obtained from the internet and gathered relevant details of the reference dataset to form a new dataset for our chatbot. The dataset consists of labels like tags, patterns, and responses.

After Pre-Processing, the model was trained with Convolutional Neural Network(CNN). I added Embedding layer and one Conv1D layer with filter size being 128 with ‘ReLU’ activation function. After this step, a MaxPooling layer was added. A flattened layer was included after this step and at last two dense layers were added with softmax activation function. For fitting the model, hyper-parameter tuning like changing epochs, learning rate etc was done to achieve better accuracy.

Pavithra Govardhanan:

After the data collection, I did the pre-processing. The pre-processing of the dataset includes Tokenization, stemming and bag of words. Also, successful training and testing of the RNN model was done.

After Pre-Processing, the model was trained with Recurrent Neural Network (RNN). I added two Embedding layers followed by a single SimpleRNN layer with filter size being 128 and returned sequences being True. After this step, a flattened layer was included. At last, two consecutive dense layers were added with softmax activation function. For fitting the model, hyper-parameter tuning like changing epochs, learning rate etc was done to achieve better accuracy.

After training the models, both of us did the prediction part. The trained models were saved and the pickle files were saved using “. pickle” extension. The predicted answers were stored in variable called results. Then the result having the highest probability if found and displayed as the response to the user. Both of us, figured out how to deliver the response in voice format. We found; it was achievable using pyttsx3 package.

MODEL PERFORMANCE AND RESULTS

We have trained out chatbot using two types of neural networks, Convolutional Neural Network (CNN) and Recurrent Neural Network (RNN).

The output of our chatbot is shown in Figure 5.

```
Start talking with the bot (type quit to stop)
You: hi
Good to see you again

You: what is your name
I'm Chatbot

You: what is a computer
An electronic device capable of performing calculations at very high speed and with very high accuracy.

You: operating system
Android and iOS are operating systems for mobile devices.

You: how a rocket engine works
A rocket engine, or simply rocket, is a jet engine that uses only stored propellant mass for forming its high speed propulsive
jet

You: worth of american eagle
One ounce is $1634 worth

You: how to treat burns
Burn injuries are unique and require specialized treatment
```

Figure 5 Project Output

The specialty of our project is that the response from the chatbot is not only in text format but also the speech response. The response from the system is conveyed both in text and in speech.

The accuracy of two models during training were promising for audio classification and it was found to be 99% for CNN and 89 % for RNN. This accuracy was improved by changing the hyper parameters. For example, when 'tanh' activation function did not perform well. However, there was an improvement in accuracy when 'ReLU' activation function was used.

The comparison of accuracy is given below. **According to our project, CNN was found to be the best neural network to train on.**

The accuracy of CNN(Figure6) was found to be : 99%

```
In [9]: print("Model saved successfully...")

        scores = cnn_model.evaluate(train, test, verbose=1)
        print("Accuracy:", scores[1])

        Model saved successfully...
        465/465 [==============================] - 0s 545us/sample - loss: 0.0084 - acc: 0.9991
        Accuracy: 0.9990616
```

Figure 6 Accuracy of CNN model

The accuracy of RNN(Figure7) was found to be : 89%

```
In [42]: print("Model saved successfully...")

         scores = rnn_model.evaluate(train, test, verbose=1)
         print("Accuracy:", scores[1])

         Model saved successfully...
         465/465 [==============================] - 1s 2ms/step
         Accuracy: 0.8903225806451613
```

Figure 7 Accuracy of RNN

GRAPH RESULTS

CNN Results:

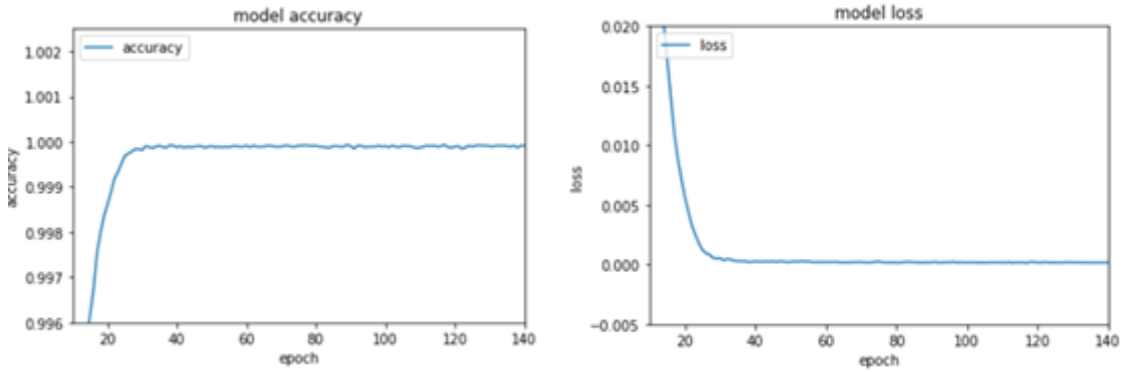

Figure 8 Graph results of CNN(accuracy and loss)

Approximately at 30 epochs, the accuracy reached 1 from 0.9 at 10 epochs. The accuracy started with 0.2 from the initial epochs and then gradually increased to 0.9. The model trained quite successfully using cnn very quickly. This can be attributed mainly due to the fact that the dataset that we used is quite less but also the convolutional layers helped in training. The loss also decreased gradually from 0.02 at the initial epochs and reached zero at 30 epochs.

RNN Results:

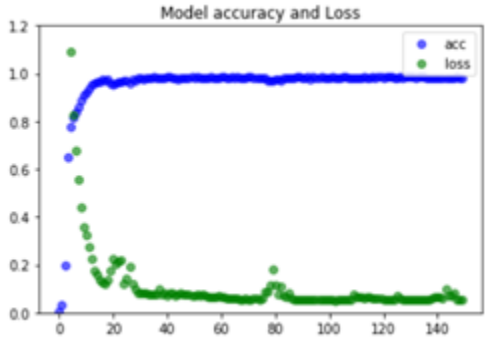

Figure 8 Graph result of RNN(accuracy and loss)

Initially, the accuracy was very less but after 6 epochs it shoots to 0.6 and then gradually increases and reaches a maximum of 0.9 after 10 epochs. Here, the maximum accuracy is reached faster which attributes to the content of the dataset and the kind of dataset that we used. Also, the loss is very high at the beginning and there are some anomalies in its behaviour after 20 epochs and also after 80 epochs. But then, it reduces to the minimum and stagnates there.

TRAINING SETUP

The model was built using Python tools including Keras, Pandas, NumPy and Jupyter Notebook.

FUTURE ENHANCEMENT

Overall, this project was interesting and exciting to work on. This project gave us a lot of insights on how these chatbot can be explored and used in different applications.

There are various possibilities on how this project can be extended in future. Currently, this system is designed on some of the relevant information in WikiQA dataset. In future, we would plan on developing an intuitive User Interface (UI) for the chatbot using Flask interface. The negatives of the dataset are that the dataset is not very huge enough to train using a CNN model and give highest accuracy as CNN has now been used for text preprocessing too. So, we plan to increase the dataset. We would try and implement other algorithms and also use deep neural networks in our project.

REFERENCES

[1] Gaikwad S, ‘Chatbots with Personality Using Deep Learning’, *San Jose State University SJSU ScholarWorks,* 2019.

[2] Bhagwat, Vyas Ajay, "Deep Learning for Chatbots" (2018). Master's Projects. 630. [Online Available]: https://scholarworks.sjsu.edu/etd_projects/630

[3] Anbang Xu, Zhe Liu, Yufan Guo, Vibha Sinha, Rama Akkiraju, 'A New Chatbot for Customer Service on Social Media', 2017.

[4]Agado, 'Personality for Your Chatbot with Recurrent Neural Networks', (Online Available) https://towardsdatascience.com/personality-for-your-chatbot-with-recurrent-neural-networks-2038f7f34636

[5] Kovalevskyi, 'Own ChatBot Based on Recurrent Neural Network'(Online Available) https://blog.kovalevskyi.com/rnn-based-chatbot-for-6-hours-b847d2d92c43

[6](OnlineAvailable)https://github.com/TapanBhavsar/Contextual_chatbot_tensorflow/blob/master/train_file.py

[7]Loy J, 'How to build your own Neural Network from scratch in Python', (Online Available) https://towardsdatascience.com/how-to-build-your-own-neural-network-from-scratch-in-python-68998a08e4f6

[8] (Online Available) WikiQA Corpus: https://www.microsoft.com/en-us/download/confirmation.aspx?id=52419