Artificial Intelligence Self Learning Chatbot

Authors

Vinaya Chinti

Shivani Ghatge

Rutuja Shinde

Course Project Professor's Name

Dr. Lindi Liao

Course Name

Introduction to Natural Language Processing

Course Number and Section

AIT 590

Date

05/03/2020

GEORGE MASON UNIVERSITY

TITLE: Artificial Intelligence Self Learning Chatbot

ABSTRACT

Conversational based modeling is an important task in Natural Language Processing and the field of Artificial Intelligence. A chatbot that works as a conversational agent is a software designed to communicated with humans using language processing abilities and machine learning. Designing a smart chatbot has been one of the most challenging aspects in the world of AI and Natural Language Processing. Previously chatbots had been designed using handwritten rules like regular expressions. Today this approach is not feasible and does not suffice the requirements in any domain as there are many rules required to be written for a chatbot to function as a human in a conversation.

In this project, we have designed a neural network-based AI chatbot that uses LSTM as its training model for both encoding and decoding. The chatbot works like an open domain chatbot that can answer day-to-day questions involved in human conversations. Words embeddings are the most important part of designing a neural network-based chatbot. Glove Word Embedding and Skip-Gram models have been used for this task. The model was trained end-to-end without any handcrafted rules. The huge conversational dataset used for training the Long Short-Term Memory (LSTM) recurrent neural network is the Cornell Movies dataset. The chatbot self learns from the training data and answers questions that are not domain specific.

Keywords--- Seq2Seq, chatbot, LSTM, Glove, Skip-Gram, neural network.

INTRODUCTION

Chatbots are the new age software-based friends of humans used for informative and friendly conversations. They are a computer-based program that simulates human conversations through either voice commands or via texts or both. Initial chatbots were rule-based but today they are AI and Machine Learning driven. In this project, we are implementing an Intelligent chatbot using a generative-based method that simplifies the conversations. Rule-based chatbots give a response based on just regular expressions and technically it's difficult to incorporate all regular expressions for conversations. We aim to solve this problem of rule-based chatbots and build an intelligent chatbot that self learns from the large corpus of data given to it. AI-based chatbots are interesting as they interact with humans just like any other human and in a much friendly way to answer all our queries. The project, however, is very challenging as the chatbot requires a huge corpus of conversational data for training and learning along with deep learning for which regular hardware and software used are not enough.

The history of chatbot dates to 1950 when The Turing Test was theorized by Alan Turing stating that a truly intelligent machine that can converse just like humans could be built based on text-only conversations. This idea was the foundation for the creation of chatbots. In1960's first chatbot in the history of AI was invented named ELIZA. She is a psychotherapist chatbot designed by Joseph Weizenbaum which uses regular expressions which is a pattern matching technique and a substitution methodology to simulate the conversations. In 1995 another chatbot named ALICE was invented which used natural language processing for conversations. It was used by developers to use AIML(AI Markup Language) to create their chatbots that used ALICE. The term 'Chatterbot' was coined first in 1994. A voice-based chatbot named Jabberwacky was later created by a British programmer Rollo Carpenter. This was the transition phase from text-based chatbots to voice-based. The advanced version is the Mitsuku chatbot which contains all the AIML files from the chatbot ALICE. It is an 18-year-old female chatbot that can reason with specific objects for example if the question is "Can we eat the house?", then her reply would be "No" as she looks up the properties of the subject "house" and finds the value of "made from" which is "brick" and hence the answer is No. She can also play games and do magic tricks on user requests.

The advancement of AI-based chatbots has been made for a few decades where each version is an improved version of the previous one, with added functionalities. They now have multiple uses such as timesheets (Nika), expenses(Acebot), Customer Service (Twyla), FAQ bots (QnA), and feedback(Wizu). 2006 was the start of the golden era for chatbots as IBM's Watson was created which used Machine Learning and Natural Language Processing for learning from a huge database for conversations. 2010 saw the invention of Siri an intelligent personal assistant which is a part of Apple's IOS who uses NLP to fulfill user requests. Google Now, Alexa and Cortana were invented in 2012 and 2015 respectively who are voice-based chatbots who receive voice commands for processing. 2016 was the year for bot-based messengers where they conversed with users. Microsoft's Tay in 2016 can mimic the speech and habits of a teenage girl but was shut down as it became paranoid and posted offensive tweets.

Chatbots are of two types names *vertical* and *horizontal* types. Vertical chatbots are closed-domain chatbots focused on applications – this would be the case with a chatbot for a customer service department in the telecom industry. A horizontal chatbot is a general and open-domain bot like Siri, Alexa, or Google Assistant. We aim to design a vertical chatbot.

RELATED WORK

For this project, we studied several papers on generative dialogue systems, Sequential models, Word Embeddings, and Natural Language Processing. Many papers and tutorials studied for the understanding of the Data pre-processing and building sequential models. The inspiration and idea for this project are taken from Inkawhich, M. (2017) et.al [1] tutorial, the tutorial is about building a generative chatbot using the deep learning techniques. The author has explained the technique to design a chatbot and implemented different deep learning models in pyTorch[1]. In terms of language modeling and deep learning has made huge advances for the chatbots. The most important takeaway from the paper [13], the chatbot model uses a sequence to sequence model. The architecture has two Recurrent Neural Networks one to encode a variable sequence to get vector representation and other to decode the vector into a variable-length sequence.

The paper [15], demonstrates the problem that traditional chatbots are dependent on human handwritten rules but with the future advances in Neural Conversational Model, the neural network models can be used to train chatbots. The Long Short-term Memory (LSTM) recurrent neural network is a better model as compared to vanilla Recurrent Neural Network (RNN) because vanilla RNN suffers from vanishing gradient problem. The paper [4] presents the implementation of Language models by using the Pre-trained word Embeddings Global (Glove). The implementation is done using Keras.

Richard Krisztian Csaky (2017) et.al [14], has mentioned the history of the chatbots and early approaches used for designing a chatbot. Later, he has mentioned about the different approaches to conversational, these approaches are hierarchical model, Task-Oriented Dialog system, Reinforcement Learning, and Encoder-Decoder Models. From the experiment, they concluded that the transformer model didn't perform well as compared to sequence-to-sequence, because the parameter space of sequence-to-sequence model is trained by much bigger than the base variant of the Transformer. Other papers [20][21] provides brief information about the use of bidirectional LSTM on the same Cornell Movie Dialogue Corpus. They speak about using TensorFlow in python for building a sequence-to-sequence model with LSTM, a bi-directional RNN encoder, and a decoder with an attention mechanism.

OBJECTIVES

- Overcome the pitfalls of handwritten rule-based chatbots
- Improve the fluency of conversations more like humans
- Build an open domain chatbot
- Increase robustness and efficiency
- Build a smart self-learning Chatbot
- Incorporate advance techniques like deep learning neural network models
- Train on a large corpus of Conversational data.
- Incorporate word embeddings so that the model learns better.

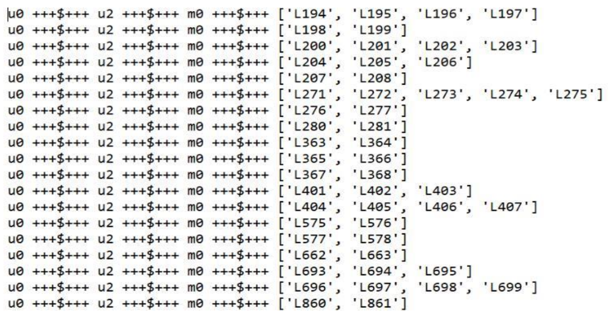
SELECTED DATASET

To build a self-learning chatbot, we have selected the following corpora to train the neural network**.**

Cornell Movies Dialogue Corpus: It consists of 220579 conversational exchanged between 10292 pairs of movie characters involves 9035 characters from 617 movies. It has in total of 304713 utterances.
Data Source: https://www.cs.cornell.edu/~cristian/Cornell_Movie-Dialogs_Corpus.html

We have two files for the data:

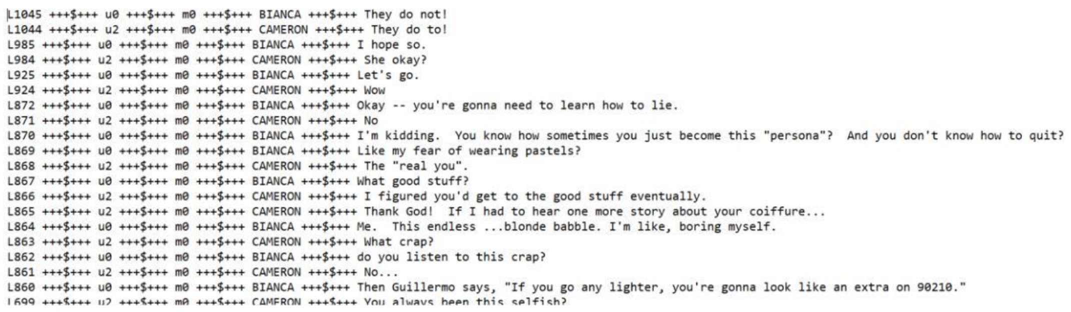
1. movie_conversations.txt:



```
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L194', 'L195', 'L196', 'L197']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L198', 'L199']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L200', 'L201', 'L202', 'L203']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L204', 'L205', 'L206']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L207', 'L208']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L271', 'L272', 'L273', 'L274', 'L275']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L276', 'L277']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L280', 'L281']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L363', 'L364']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L365', 'L366']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L367', 'L368']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L401', 'L402', 'L403']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L404', 'L405', 'L406', 'L407']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L575', 'L576']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L577', 'L578']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L662', 'L663']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L693', 'L694', 'L695']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L696', 'L697', 'L698', 'L699']
u0 +++$+++ u2 +++$+++ m0 +++$+++ ['L860', 'L861']
```

Figure 1: Snapshot of the movie_conversations file

2. movie_lines.txt:



```
L1045 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ They do not!
L1044 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ They do to!
L985 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I hope so.
L984 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ She okay?
L925 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Let's go.
L924 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ Wow
L872 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Okay -- you're gonna need to learn how to lie.
L871 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ No
L870 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ I'm kidding.  You know how sometimes you just become this "persona"?  And you don't know how to quit?
L869 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Like my fear of wearing pastels?
L868 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ The "real you".
L867 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ What good stuff?
L866 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ I figured you'd get to the good stuff eventually.
L865 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ Thank God!  If I had to hear one more story about your coiffure...
L864 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Me.  This endless ...blonde babble. I'm like, boring myself.
L863 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ What crap?
L862 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ do you listen to this crap?
L861 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ No...
L860 +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++ Then Guillermo says, "If you go any lighter, you're gonna look like an extra on 90210."
L699 +++$+++ u2 +++$+++ m0 +++$+++ CAMERON +++$+++ You always been this selfish?
```

Figure 2: Snapshot of the movie_lines file

Two files that we have used for generating the dataset are the movie_conversations.txt and the movie_lines.txt.

Dataset Schema:

(1) movie_lines.txt

1) LineID

2) CharacterID

3) MovieID

4) Character name

5) Conversation text

(2) movie_conversations.txt

1) characterID of the first character involved in the conversation

2) characterID of the second character involved in the conversation

3) movieID of the movie in which the conversation occurred

4) list of utterances that make the conversation. They are the lineID's and this lineID's should match with the lineID's in the movie_lines.txt. The lineID's are given in chronological order.

PROPOSED SYSTEM

1) SYSTEM ARCHITECTURE

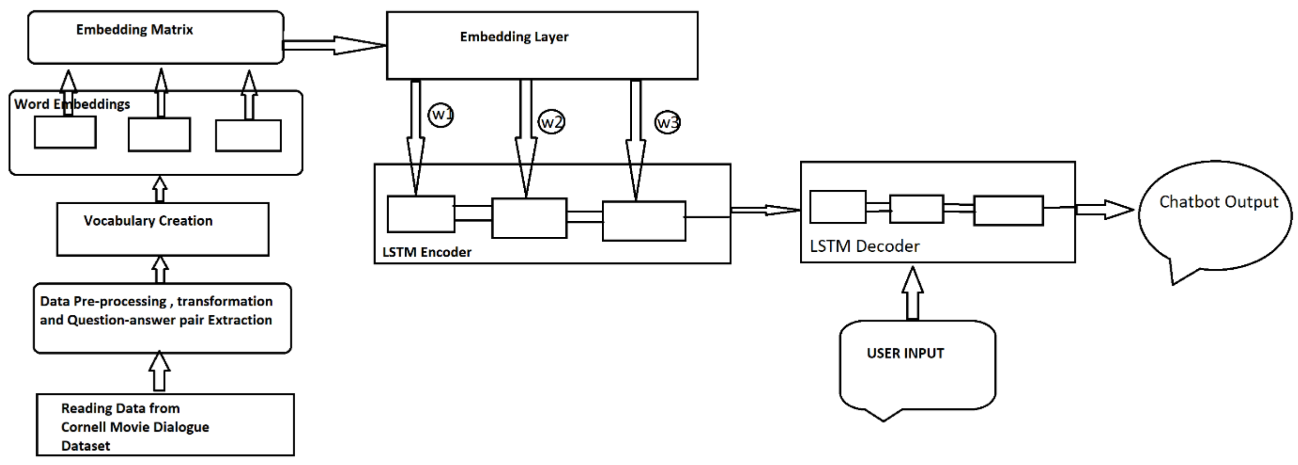

Embedding Matrix
Embedding Layer
Word Embeddings
w1
w2
w3
Vocabulary Creation
LSTM Encoder
LSTM Decoder
Chatbot Output
Data Pre-processing , transformation and Question-answer pair Extraction
USER INPUT
Reading Data from Cornell Movie Dialogue Dataset

Figure 3: System Architecture

Following are the components of the system:

(1) **Reading Data:** The Cornell Movie Dialogue Corpus dataset is used as the input dataset for the model. Two data files movie_lines.txt and movies_conversations.txt are read.

(2) **Data Pre-processing, Transformation, and Extraction**: In this step, the question and answer pairs are extracted from the two files that are read. They are then transformed, and a filter is added to these pairs. The pairs with a length between 2- 20 are only filtered and taken for further processing as question-answer conversation pairs. The pairs data is then split into the train, and validation sets. The training set is 80% of the data and the validation set is 20 % of the data. Data from both the files namely movie_lines.txt and movie_conversations.txt has been mapped using the LineID. Other unnecessary data such as ‘ +++$+++ u0 +++$+++ m0 +++$+++ BIANCA +++$+++’ has been removed from both the data documents. The mapped conversations are further divided into question and answer sets.

(3) **Vocabulary Creation**: Vocabulary is created that consists of the encodings and decodings. Each word is assigned a unique value. The encoding is a dictionary that consists of a word as the key and its unique number as value. The decoding is a dictionary that consists of the unique number as key and word as the value. The first position of the decoding has the word START which tells the decoder later form where to start the decoding.

(4) **Word Embedding**: Word embeddings are created using Glove and Skip-Gram for the words in the sentences of the data. Text cannot be given as an input to the neural network.

We need to develop embeddings of the text so that they can be given as input to train the neural networks. Embeddings represent the vector form of the data. They are the continuous vector representations of the discrete data. The embedding affects the performance of the neural networks to a great extent. To improve performance, we have used Global vectors and Skip-grams. We have used the Glove embedding with 50 dimensions of the pre-trained model and the embeddings are loaded from the file 'glove.6B.50d'. The embedding matrix is created using Glove and Skip-Gram methods for the conversation pairs. The embedding matrix is used as weights that are assigned to the model. We have built the Skip-Gram model and trained it using the encoding dictionary created earlier and creates an encoding matrix of 100 dimensions. The word embedding is created by passing the question-answer pairs to this skip-gram model.

(5) **Model Creation and Fitting**: The LSTM model is used for both encoder and decoder. The model is trained on a batch size of 64 and an epoch of 100. The training data is 30,000 from which 24,000 is training and 6000 is the validation set. The embedding matrix created using GLOVE and skip-gram which is given as weights to the embedding layer. The model will use these weights to learn the words. The optimizer used is Adam, and the loss is categorical cross-entropy This model is then saved after training so that it can be directly used without any need for re-training.

(6) **User Input**: This is the input sentence that the user will give the chatbot. The input received from the user needs to be converted to the decoder dimensions which is 20 in our case. The model's prediction method is used to predict the answer to the user's question. The output generated is in vector format and is converted back to words using the decoding's formed initially.

(7) **Chatbot Output**: This is the chatbot's answer to user questions.

2) <u>NLP / DATA ANALYTIC APPROACHES</u>

Existing Algorithms

The chatbot is developed by using the LSTM model. Following is the description for the existing algorithms that are used for development.

Recurrent Neural Network

Generative chatbots are built by using neural networks. They require a large amount of training data so that they can learn from the data and then use these learning to chat with human beings. The Recurrent neural networks are the neural networks in which the output of one step is given as an input to the next step. This is one of the advantages of the RNN over the traditional neural networks. This makes them useful in making predictions using the context in the sentence. In chatbot as well the previous words matter a lot and this is achieved in the RNN's using the hidden layer. Recurrent neural networks are good for modeling of sequence data. They have a hidden layer along with the input and output layer. The input vectors are taken, and output vectors are produced which are influenced by the weights applied to the input and the hidden state vectors representing the context based on prior inputs or outputs. This leads to formations of different outputs depending on the previous inputs in the series. An RNN can be considered as multiple copies of the same network, each passing a message to a successor. The chain-line nature of the RNN is intimately related to sequences and lists. RNN is used extensively for speech recognition, language modeling, translation, image captioning, etc. The RNN's suffer from the problem of short-term memory and cannot remember long sequences of data. To overcome this, Bi-directional LSTM has been used.

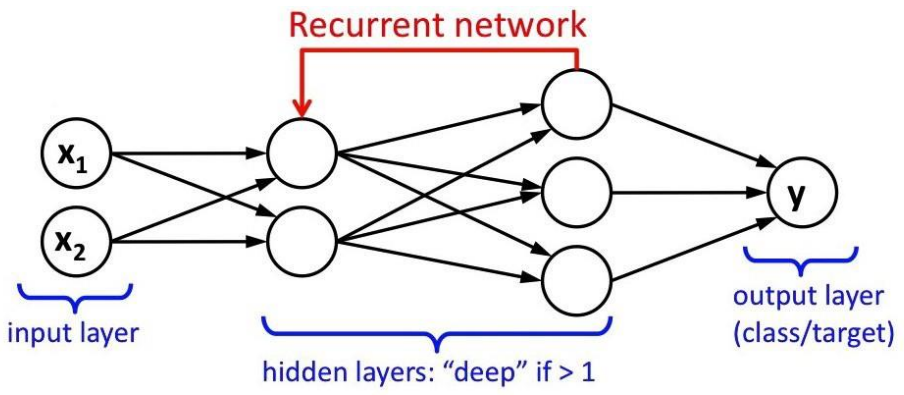


Figure 4: Recurrent Neural Network [28]

Long short-term memory(LSTM)

It is used for processing long sequences of data by remembering the previous words in the sentence. It is designed to avoid the long-term dependency problem faced in RNN. Remembering information for long periods is practically the default behavior of the LSTM. They also have a chain-like structure, but the repeating module has a different structure. The cell state is the key of LSTM and it is used for transferring the information all down the sequence chain. The gates use the sigmoid activation function which is in the sigmoid layer and it outputs numbers between zero and one that describes how much of each component should be let through it. The LSTM's make use of the forget gate, the input gate, and the output gate. The gates are used to update or forget the information. The LSTM(Long short-term memory) RNN architecture forms the core of the chatbot. They are best suited for classification, processing, and making predictions. It is a type of deep learning model used for learning the order dependence in sequence prediction problems. LSTM models are used for language modeling, speech recognition, and machine translation. LSTM can be used as a bi-directional or unidirectional model. Bi-directional LSTM is better than using a single LSTM as the input is run through two LSTM's, one which reads from left to right and one which reads the data from right to left and then it concatenates the output. For each word, the first LSTM learns the effect of the previous words and the second LSTM learns the effect of the future words. It provides slightly better results than a single LSTM. The Encoder-Decoder LSTM are very effective for seq2seq prediction problems. The architecture comprises of two models , one for reading the input sequence and encoding it into a fixed-length vector while the second for decoding the fixed length vector and outputting the predicted sequence.

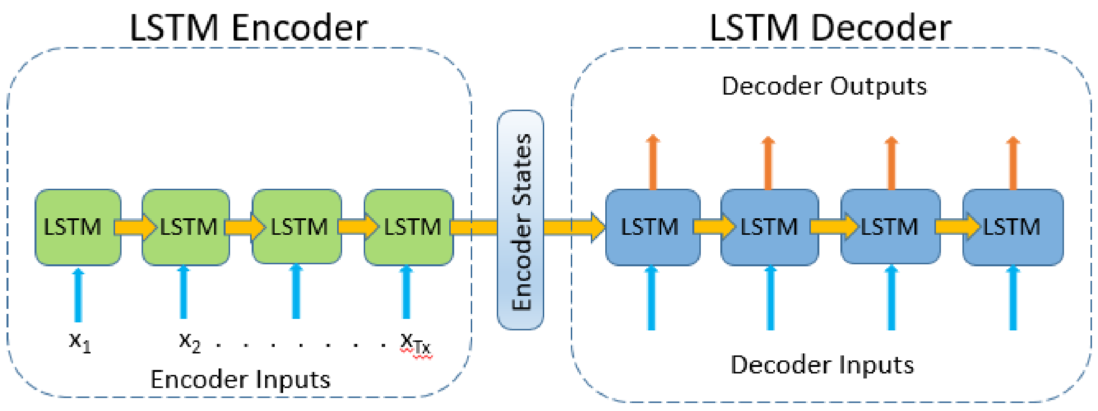


Figure 5:Long Short-Term Memory (LSTM) [29]

New Algorithms(Own) :

The LSTM model is used. The following process is followed to get the answer to the question asked by the user.

1) The movie_conversations.txt and movie_lines.txt are read and preprocessing is done on the data. The conversations are extracted from the data.
2) The conversations are mapped to question answers pairs.
3) Word Embeddings are generated using a glove and skip-gram.
4) LSTM and Bidirectional LSTM models are trained using the embedding matrix formed from the glove and skip-gram as weights.
5) For creating the encoder and decoder LSTM model is used.
6) The model created is fit on training and validated using the validation dataset.
7) Answer to the question is predicted using the trained model.

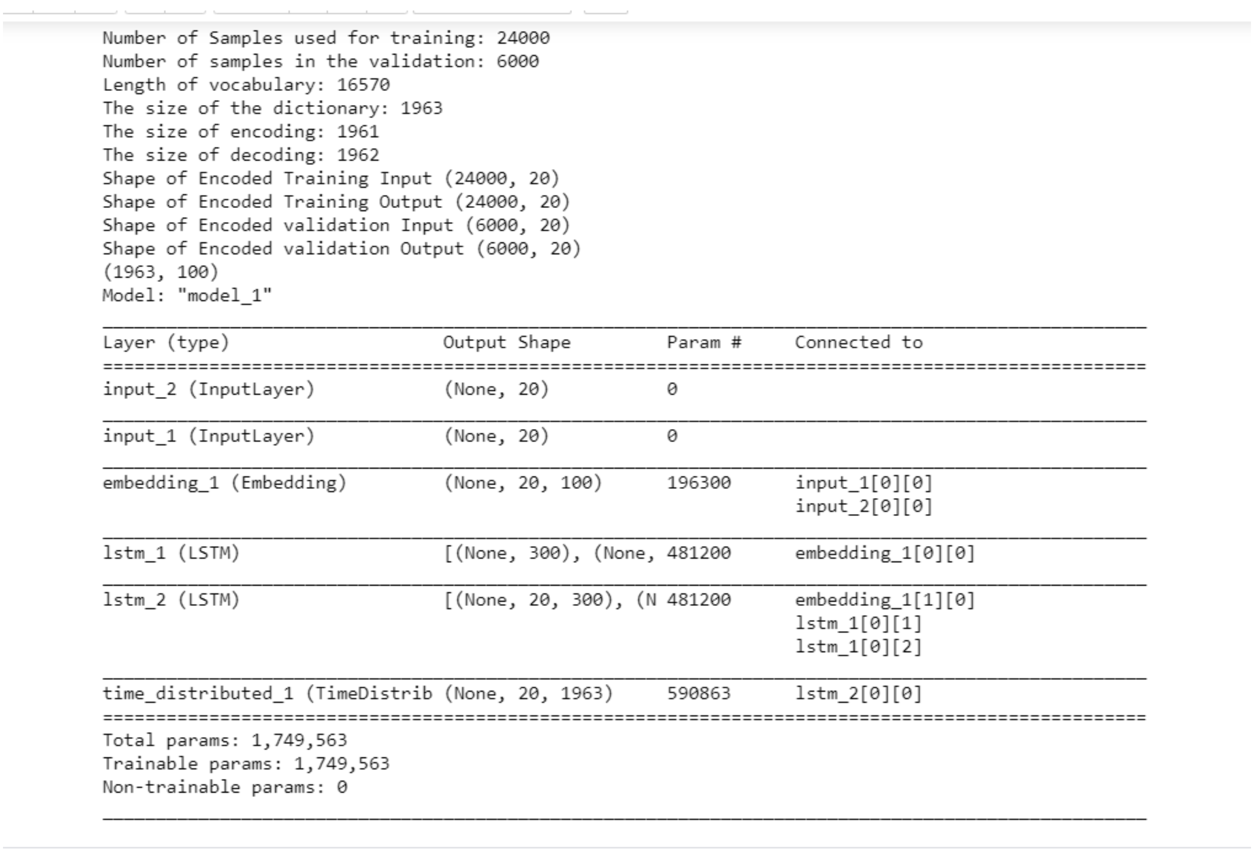
The existing LSTM model is used for building the encoder and decoder. The weights to the embedding layer are assigned using GLOVE pre-trained model and skip-gram. Skip-gram embedding is created using the word2vec function which is trained on our vocabulary.

3) SOFTWARE -HARDWARE DEVELOPMENT PLATFORMS:

Hardware: Laptops with configuration Intel I7 8th gen processor, 512 SSD, and 16 GB RAM.

Software: Jupyter Lab, Python, Kera, NumPy

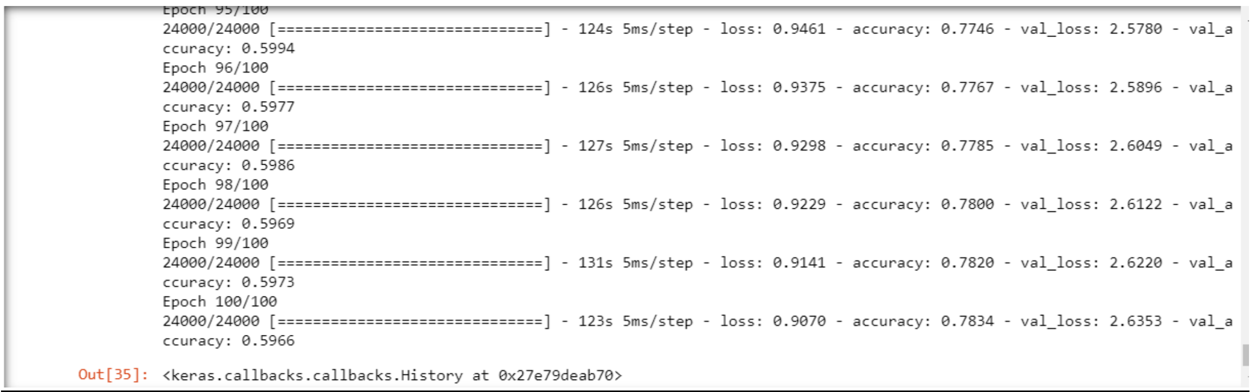
EXPERIMENTAL RESULTS AND ANALYSIS



```
Number of Samples used for training: 24000
Number of samples in the validation: 6000
Length of vocabulary: 16570
The size of the dictionary: 1963
The size of encoding: 1961
The size of decoding: 1962
Shape of Encoded Training Input (24000, 20)
Shape of Encoded Training Output (24000, 20)
Shape of Encoded validation Input (6000, 20)
Shape of Encoded validation Output (6000, 20)
(1963, 100)
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 20)           0
__________________________________________________________________________________________________
input_1 (InputLayer)            (None, 20)           0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 20, 100)      196300      input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 300), (None, 481200      embedding_1[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 20, 300), (N 481200      embedding_1[1][0]
                                                                 lstm_1[0][1]
                                                                 lstm_1[0][2]
__________________________________________________________________________________________________
time_distributed_1 (TimeDistrib (None, 20, 1963)     590863      lstm_2[0][0]
==================================================================================================
Total params: 1,749,563
Trainable params: 1,749,563
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 6: Summary of Model built using Skip-gram

Figure 6 shows the summary of the model built using LSTM and the skip-gram embeddings. It can be seen from the figure that the encoder and the decoder both use the LSTM models. The dimension of the embedding layer is (20,100) as the embeddings generated using the skip-gram have 100 dimensions. Even the size of the training and the validation data which is 24000 and 6000 can be seen in *Figure 4.* The size of the vocabulary used is 1963.

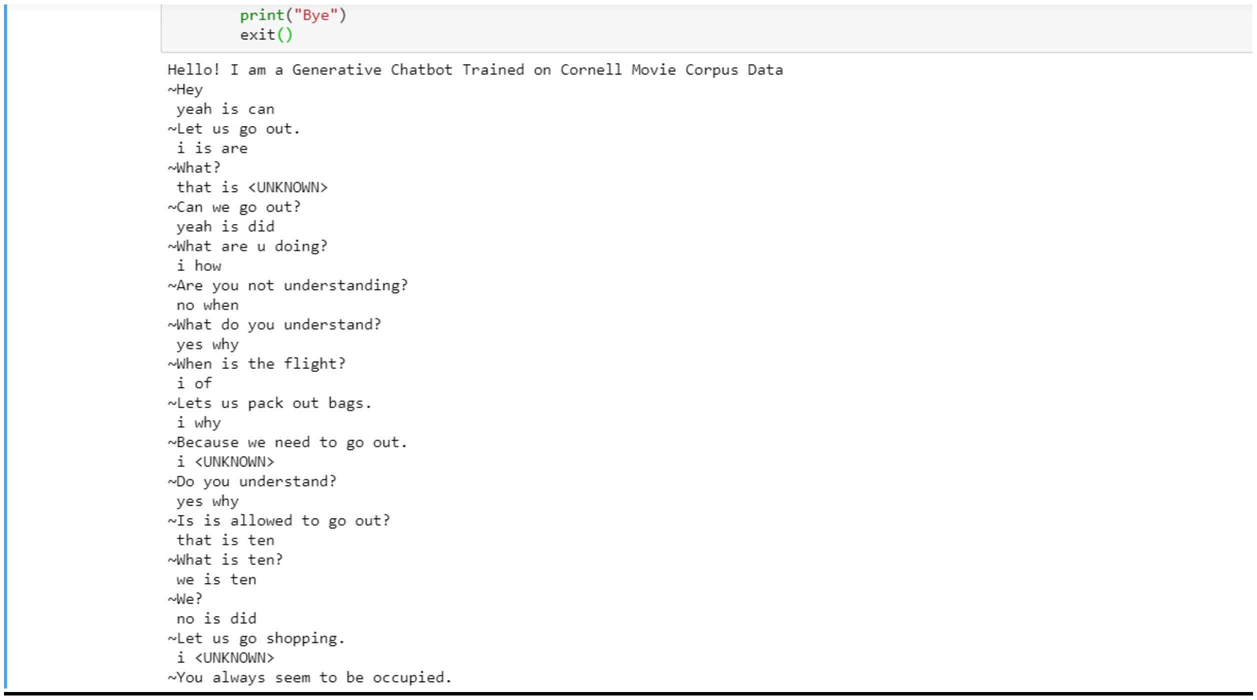


```
Epoch 95/100
24000/24000 [==============================] - 124s 5ms/step - loss: 0.9461 - accuracy: 0.7746 - val_loss: 2.5780 - val_a
ccuracy: 0.5994
Epoch 96/100
24000/24000 [==============================] - 126s 5ms/step - loss: 0.9375 - accuracy: 0.7767 - val_loss: 2.5896 - val_a
ccuracy: 0.5977
Epoch 97/100
24000/24000 [==============================] - 127s 5ms/step - loss: 0.9298 - accuracy: 0.7785 - val_loss: 2.6049 - val_a
ccuracy: 0.5986
Epoch 98/100
24000/24000 [==============================] - 126s 5ms/step - loss: 0.9229 - accuracy: 0.7800 - val_loss: 2.6122 - val_a
ccuracy: 0.5969
Epoch 99/100
24000/24000 [==============================] - 131s 5ms/step - loss: 0.9141 - accuracy: 0.7820 - val_loss: 2.6220 - val_a
ccuracy: 0.5973
Epoch 100/100
24000/24000 [==============================] - 123s 5ms/step - loss: 0.9070 - accuracy: 0.7834 - val_loss: 2.6353 - val_a
ccuracy: 0.5966

Out[35]: <keras.callbacks.callbacks.History at 0x27e79deab70>
```

Figure 7: Model Fitting using Skip-Gram Word Embeddings

Figure 7 shows that the model is fitted on the data and is trained for 100 epochs. The accuracy of the model on the validation dataset is 59.66% and its accuracy on the training data is 78.34%.

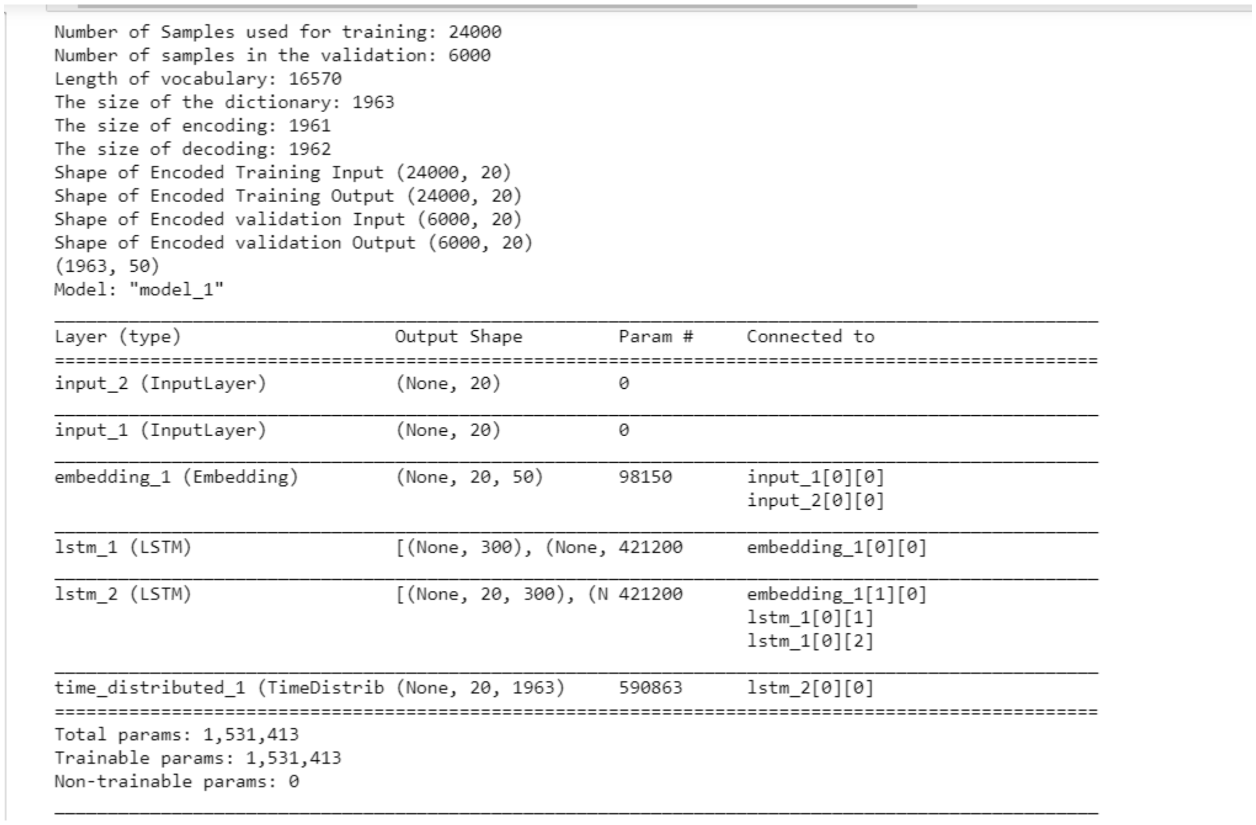


```
        print("Bye")
        exit()
```

```
Hello! I am a Generative Chatbot Trained on Cornell Movie Corpus Data
~Hey
 yeah is can
~Let us go out.
 i is are
~What?
 that is <UNKNOWN>
~Can we go out?
 yeah is did
~What are u doing?
 i how
~Are you not understanding?
 no when
~What do you understand?
 yes why
~When is the flight?
 i of
~Lets us pack out bags.
 i why
~Because we need to go out.
 i <UNKNOWN>
~Do you understand?
 yes why
~Is is allowed to go out?
 that is ten
~What is ten?
 we is ten
~We?
 no is did
~Let us go shopping.
 i <UNKNOWN>
~You always seem to be occupied.
```

Figure 8: Chat with the model trained using Skip-gram embeddings

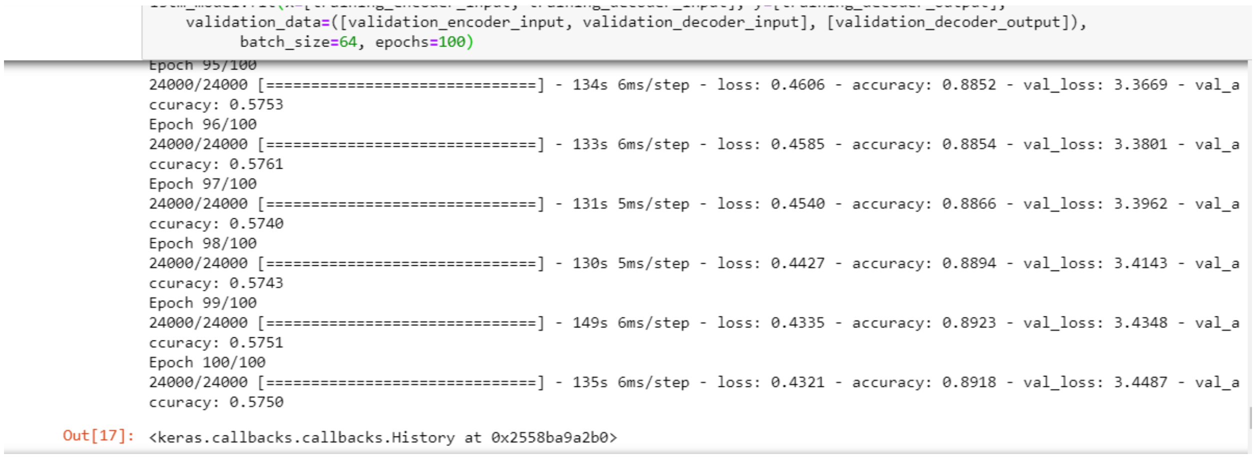
Figure 8 shows the chat with the model trained by using the user input. As the model is trained is on a very small amount of data the answers provided by the chatbot are a little vague.



```
Number of Samples used for training: 24000
Number of samples in the validation: 6000
Length of vocabulary: 16570
The size of the dictionary: 1963
The size of encoding: 1961
The size of decoding: 1962
Shape of Encoded Training Input (24000, 20)
Shape of Encoded Training Output (24000, 20)
Shape of Encoded validation Input (6000, 20)
Shape of Encoded validation Output (6000, 20)
(1963, 50)
Model: "model_1"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_2 (InputLayer)            (None, 20)           0
__________________________________________________________________________________________________
input_1 (InputLayer)            (None, 20)           0
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, 20, 50)       98150       input_1[0][0]
                                                                 input_2[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, 300), (None, 421200      embedding_1[0][0]
__________________________________________________________________________________________________
lstm_2 (LSTM)                   [(None, 20, 300), (N 421200      embedding_1[1][0]
                                                                 lstm_1[0][1]
                                                                 lstm_1[0][2]
__________________________________________________________________________________________________
time_distributed_1 (TimeDistrib (None, 20, 1963)     590863      lstm_2[0][0]
==================================================================================================
Total params: 1,531,413
Trainable params: 1,531,413
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 9: Summary of Model built using Glove Embeddings

Figure 9 shows a summary of the model building by using the glove embeddings. The size of the data used for training and the validation is the same as that of the model built using skip-gram. The only change is in the dimensions of the embedding layer. The dimensions of the embedding layer are (20,50) as we have used glove embedding with the 50 dimensions.



```
    validation_data=([validation_encoder_input, validation_decoder_input], [validation_decoder_output]),
          batch_size=64, epochs=100)
```

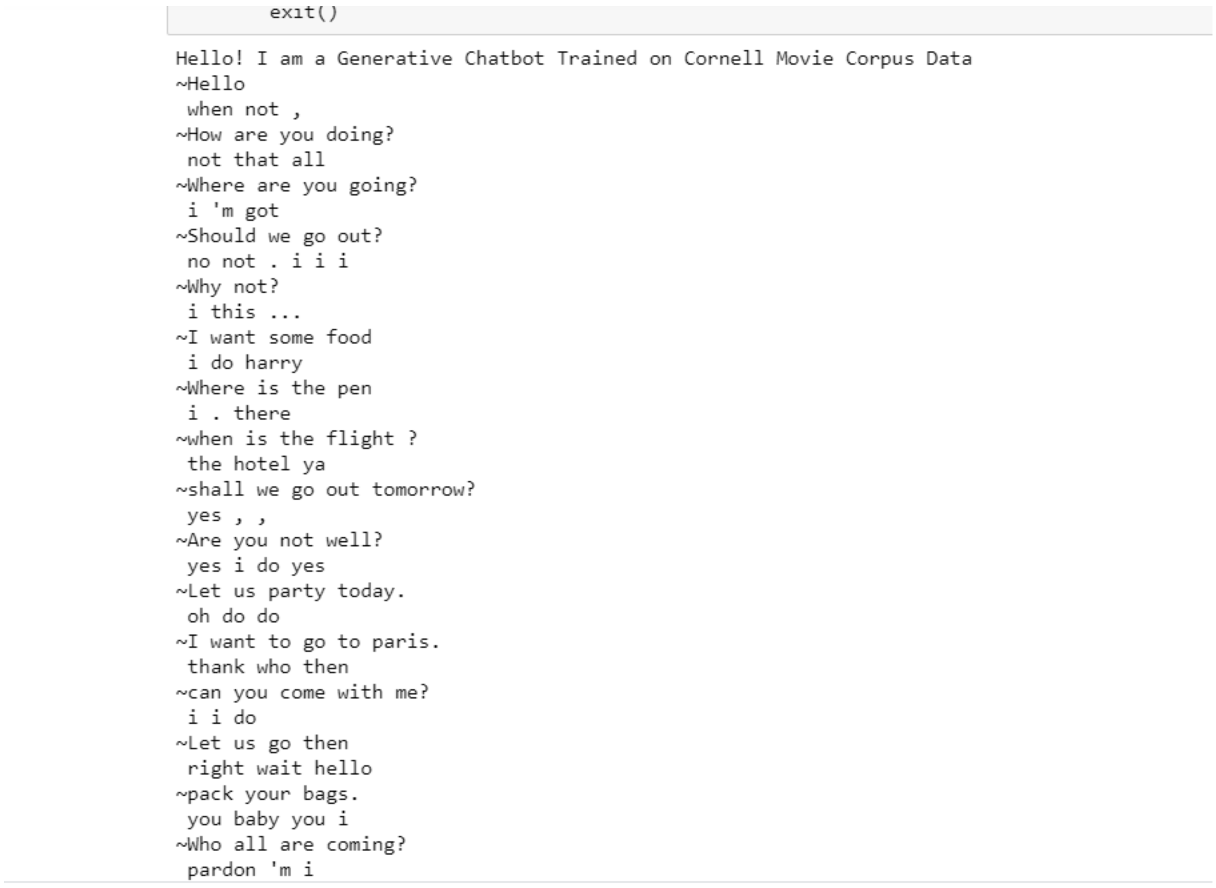
```
Epoch 95/100
24000/24000 [==============================] - 134s 6ms/step - loss: 0.4606 - accuracy: 0.8852 - val_loss: 3.3669 - val_a
ccuracy: 0.5753
Epoch 96/100
24000/24000 [==============================] - 133s 6ms/step - loss: 0.4585 - accuracy: 0.8854 - val_loss: 3.3801 - val_a
ccuracy: 0.5761
Epoch 97/100
24000/24000 [==============================] - 131s 5ms/step - loss: 0.4540 - accuracy: 0.8866 - val_loss: 3.3962 - val_a
ccuracy: 0.5740
Epoch 98/100
24000/24000 [==============================] - 130s 5ms/step - loss: 0.4427 - accuracy: 0.8894 - val_loss: 3.4143 - val_a
ccuracy: 0.5743
Epoch 99/100
24000/24000 [==============================] - 149s 6ms/step - loss: 0.4335 - accuracy: 0.8923 - val_loss: 3.4348 - val_a
ccuracy: 0.5751
Epoch 100/100
24000/24000 [==============================] - 135s 6ms/step - loss: 0.4321 - accuracy: 0.8918 - val_loss: 3.4487 - val_a
ccuracy: 0.5750

Out[17]: <keras.callbacks.callbacks.History at 0x2558ba9a2b0>
```

Figure 10: Model Fitting using GLOVE Word Embeddings

Figure 10 shows that the model is fitted on the data and is trained for 100 epochs. The accuracy of the model on the validation dataset is 57.50% and its accuracy on the training data is 89.18%.

When compared with the model built using skip-gram, the model built using glove embeddings have less accuracy on the validation data and more accuracy on the training data.



```
        exit()

Hello! I am a Generative Chatbot Trained on Cornell Movie Corpus Data
~Hello
 when not ,
~How are you doing?
 not that all
~Where are you going?
 i 'm got
~Should we go out?
 no not . i i i
~Why not?
 i this ...
~I want some food
 i do harry
~Where is the pen
 i . there
~when is the flight ?
 the hotel ya
~shall we go out tomorrow?
 yes , ,
~Are you not well?
 yes i do yes
~Let us party today.
 oh do do
~I want to go to paris.
 thank who then
~can you come with me?
 i i do
~Let us go then
 right wait hello
~pack your bags.
 you baby you i
~Who all are coming?
 pardon 'm i
```

Figure 11: Chat with the model trained using GLOVE embeddings

Figure 11 shows the chat with the model trained by using the user input. As the model is trained is on a very small amount of data the answers provided by the chatbot are a little vague. The results obtained are better and more relevant when compared to the model built using the skip-gram embeddings. This is because we have used the pre-trained model to generate the glove embeddings.

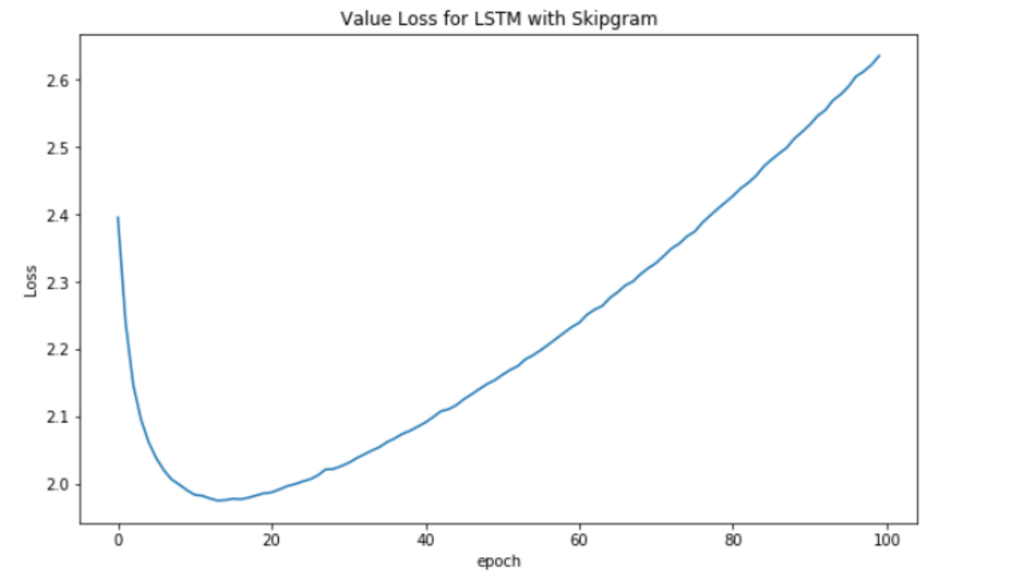


Figure 12: Skip-gram Loss

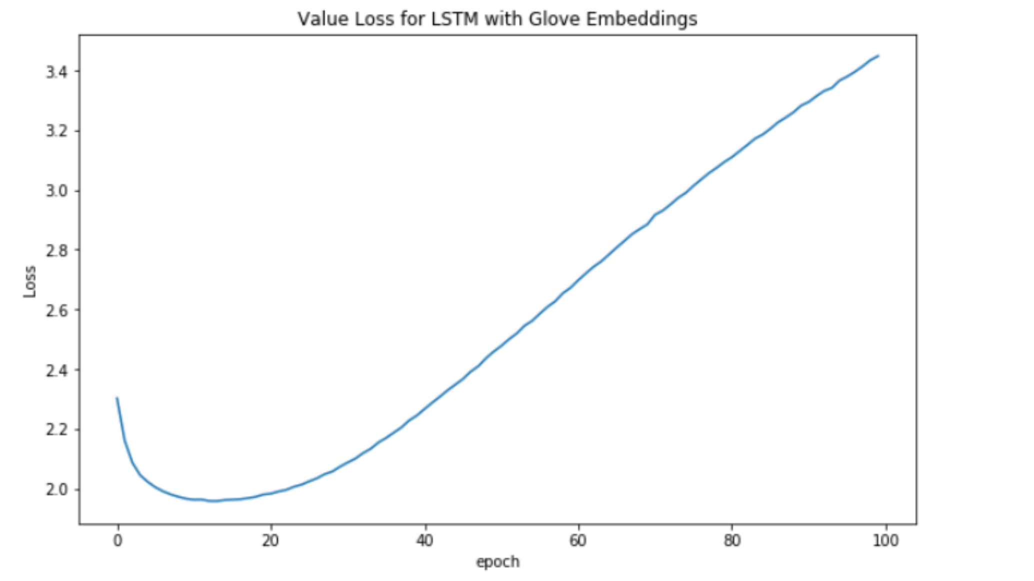


Figure 13: Glove Embedding Loss

When the loss for both the functions is compared, we can see that the loss initially drops and then goes on increasing as the number of epochs increasing for both the models. The loss for the model built using glove embedding is more when compared to the loss of the model built using skip-gram embeddings.

LESSONS LEARNED

Deep learning along with the fundamentals of chatbot building have been the key topics of study in this project. Chatbots learn from scratch using deep learning where human-to-human dialogues are the data used for its training. The more the data is supplied to the chatbot, the better is its effectiveness. Recurrent neural networks are good for modeling of sequence data. Due to its feedback nature, an RNN learns from its past input data and during training while generating outputs. They have a hidden layer along with the input and output layer. The input vectors are taken, and output vectors are produced which are influenced by the weights applied to the input and the hidden state vectors representing the context based on prior inputs or outputs. This leads to formations of different outputs depending on the previous inputs in the series. The LSTM(Long short-term memory) RNN architecture used forms the core of the chatbot. They are best suited for classification, processing, and making predictions. It is a type of deep learning model used for learning the order dependence in sequence prediction problems. LSTM models are used for language modeling, speech recognition, and machine translation. LSTM can be used as a bi-directional or unidirectional model. Bi-directional LSTM is better than using a single LSTM as the input is run through 2 LSTM’s, one which reads from left to right and one which reads the data from right to left and then it concatenates the output. For each word, the first LSTM learns the effect of the previous words and the second LSTM learns the effect of the future words. It provides slightly better results than a single LSTM.

Word embeddings are the most important and popular representation of the document vocabulary when building a chatbot. It captures the semantic and syntactic similarity and relations between the words. They are the vector representation of the words. Word2Vec is the most popular technique used for word embeddings. Chatbots built using neural networks do not understand human language or words. Thus, the input data must be converted into numerical i.e. vector format before feeding it to the neural network for training and predictions. Word2Vec is used for this conversion. GLOVE and Skip-Gram are one of the most popular word embeddings used. Skip-Gram uses the center word to predict the surrounding words. Glove embedding is a pre-trained model used for word embedding. It does not rely on local statistics i.e. the context information of the words but it incorporates the word occurrence to obtain the word vectors. GLOVE which means Global Vectors captures both the local and global statistics of a corpus to generate word vectors. For chatbot, building GLOVE proves to be a very useful technique due to these features and functionalities. All these functionalities used for building a chatbot are built on a very popular library called Keras which is used for neural networks. Keras is an open-source, high-level library used for developing neural networks and deep learning models. The core functionality of Keras is to build a neural network, train it, and then use it for predictions. It is easy and accessible for anyone with basic programming knowledge. It is an interface that can run on top of different deep learning frameworks like TensorFlow, Theano, etc.

FUTURE WORK

The major constraint in building the system was the memory as the models were trained on the local system. The model was trained using very fewer data. To enhance the accuracy of the model, the model can be trained on a large amount of data so that it can have more vocabulary and can learn from more words and sentences. Context and prediction mechanism can be added so that the bot can analyze the context and predict the answer to the questions

CONCLUSION

AI-based self-learning chatbots require machine learning and neural network rather than regular expressions to talk fluently like a human in a conversation. When we compare the model trained using LSTM with skip-gram embedding with the model trained using LSTM with glove embeddings we observe that the accuracy of the model with skip-gram is more. Irrespective of the accuracy the model trained with glove gives better answers as compared to the skip-gram this is because we have used pre-trained glove embedding. Further, the model was trained on a comparatively small amount of data so the performance and fluency of a conversational chatbot can be improved by training it on more data.

REFERENCES:

[1] Inkawhich, M. (2017). Chatbot Tutorial. Retrieved April 1, 2020, from https://pytorch.org/tutorials/beginner/chatbot_tutorial.html

[2] Zornoza, J. (2019, July 18). Deep Learning for NLP: Creating a Chatbot with Keras! Retrieved April 12, 2020, from https://towardsdatascience.com/deep-learning-for-nlp-creating-a-chatbot-with-keras-da5ca051e051

[3] Jethwani, T. (2019, October 18). Using Glove Word Embeddings with Seq2Seq Encoder Decoder in Pytorch. Retrieved April 13, 2020, from https://leakyrelu.com/2019/10/18/using-glove-word-embeddings-with-seq2seq-encoder-decoder-in-pytorch/

[4]Makarenkov, V., Rokach, L. and Shapira, B., 2017. Language Models With Pre-Trained (Glove) Word Embeddings. [ebook] Department of Software and Information Systems Engineering Ben-Gurion University of the Negev. Available at: https://arxiv.org/pdf/1610.03759.pdf [Accessed 16 April 2020].

[5] TensorFlow. (2020, April 1). Word embeddings: TensorFlow Core. Retrieved April 16, 2020, from https://www.tensorflow.org/tutorials/text/word_embeddings

[6] Pellarolo, M. (2018, April 6). How to use Pre-trained Word Embeddings in PyTorch. Retrieved April 12, 2020, from https://medium.com/@martinpella/how-to-use-pre-trained-word-embeddings-in-pytorch-71ca59249f76

[7] Theiler, S. (2020, January 5). Basics of Using Pre-trained GloVe Vectors in Python. Retrieved April 12, 2020, from https://medium.com/analytics-vidhya/basics-of-using-pre-trained-glove-vectors-in-python-d38905f356db

[8] Oliinyk, H. (2018, March 6). Word embeddings: exploration, explanation, and exploitation (with code in Python). Retrieved April 14, 2020, from https://towardsdatascience.com/word-embeddings-exploration-explanation-and-exploitation-with-code-in-python-5dac99d5d795

[9] Bednarski, M. (2018, March 19). Implementing word2vec in PyTorch (skip-gram model). Retrieved April 14, 2020, from https://towardsdatascience.com/implementing-word2vec-in-pytorch-skip-gram-model-e6bae040d2fb

[10] Cambridge Spark. (2018, November 9). Tutorial: Build your own Embedding and use it in a Neural Network. Retrieved April 14, 2020, from https://blog.cambridgespark.com/tutorial-build-your-own-embedding-and-use-it-in-a-neural-network-e9cde4a81296

[11] Doshi, S. (2019, March 17). Skip-Gram: NLP context words prediction algorithm. Retrieved April 14, 2020, from https://towardsdatascience.com/skip-gram-nlp-context-words-prediction-algorithm-5bbf34f84e0c

[12] Osipenko, A. (2019, May 30). Building ChatBot - Weekend of a Data Scientist. Retrieved April 15, 2020, from https://medium.com/cindicator/building-chatbot-weekend-of-a-data-scientist-8388d99db093

[13] Roca, M., 2019. A Generative Dialogue System For Reminiscence Therapy. [ebook] Barcelona: telecom BCN. Available at https://imatge.upc.edu/web/sites/default/files/pub/xCaros.pdf [Accessed 16 April 2020].

[14] Cs ´ aky, R., 2017. Deep Learning Based Chatbot Models. [ebook] BUDAPEST. Available at: https://arxiv.org/pdf/1908.08835.pdf [Accessed 16 April 2020].

[15] Shah, J. and Mohammed, D., n.d. Chatbot Analytics Based On Question Answering System: Movie Related Chatbot Case Analytics. Thunder Bay, Canada: Department of Computer Science Lakehead University.

[16] Introduction to Recurrent Neural Network. (2018, October 3). Retrieved from https://www.geeksforgeeks.org/introduction-to-recurrent-neural-network/

[17] Nguyen, M. (2019, July 10). Illustrated Guide to LSTM's and GRU's: A step by step explanation. Retrieved April 19, 2020, from https://towardsdatascience.com/illustrated-guide-to-lstms-and-gru-s-a-step-by-step-explanation-44e9eb85bf21

[18] Koehrsen, W. (2018, October 2). Neural Network Embeddings Explained. Retrieved April 19, 2020, from https://towardsdatascience.com/neural-network-embeddings-explained-4d028e6f0526

[19] Monsters, D. (2017, September 26). 7 types of Artificial Neural Networks for Natural Language Processing. Retrieved April 19, 2020, from https://medium.com/@datamonsters/artificial-neural-networks-for-natural-language-processing-part-1-64ca9ebfa3b2

[20] Dhankhar, P., 2018. *RNN And LSTM Based Chatbot Using NLP*. [ebook] Delhi: International Journal of Innovations in Engineering and Technology (IJIET). Available at: http://ijiet.com/wp-content/uploads/2019/10/32.pdf [Accessed 1 May 2020].

[21] Ali, A., 2019. *Conversational AI Chatbot Based On Encoder-Decoder Architectures With Attention Mechanism*. [ebook] Karachi: Artificial Intelligence Festival 2.0, NED University of Engineering and Technology. Available at: https://www.researchgate.net/profile/Amir_Ali56/publication/338100972_Conversational_AI_Chatbot_Based_on_Encoder-Decoder_Architectures_with_Attention_Mechanism/links/5dfe1dff4585159aa48e8008/Conversational-AI-Chatbot-Based-on-Encoder-Decoder-Architectures-with-Attention-Mechanism.pdf [Accessed 1 May 2020].

[22] Wizu. (2018, June 4). A Visual History Of Chatbots. Retrieved May 1, 2020, from https://chatbotsmagazine.com/a-visual-history-of-chatbots-8bf3b31dbfb2

[23] Arya, M. (2019, April 23). A brief history of Chatbots. Retrieved May 1, 2020, from https://chatbotslife.com/a-brief-history-of-chatbots-d5a8689cf52f

[24] Colah's blog. (2015, August 27). Understanding LSTM Networks. Retrieved May 1, 2020, from https://colah.github.io/posts/2015-08-Understanding-LSTMs/

[25] Word2Vec Tutorial - The Skip-Gram Model. (2016, April 19). Retrieved May 3, 2020, from http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model/

[26] Chollet, F. (n.d.). The Keras Blog. Retrieved May 3, 2020, from https://blog.keras.io/using-pre-trained-word-embeddings-in-a-keras-model.html

[27] (n.d.). Retrieved May 3, 2020, from https://keras.io/layers/recurrent/

[28] Chatterjee, C. C. (2019, July 25). Implementation of RNN, LSTM, and GRU. Retrieved May 1, 2020, from https://towardsdatascience.com/implementation-of-rnn-lstm-and-gru-a4250bf6c090

[29] Battula, Manohar. "Time Series Forecasting with Deep Stacked Unidirectional and Bidirectional LSTMs." Medium, Towards Data Science, 4 Mar. 2019, https://towardsdatascience.com/time-series-forecasting-with-deep-stacked-unidirectional-and-bidirectional-lstms-de7c099bd918