JARVIS, A GENERALISED CHATBOT

Thesis submitted to

Visvesvaraya National Institute of Technology, Nagpur

in partial fulfilment of requirement for the award of degree of

Bachelor of Technology

(Computer Science & Engineering)

By

Anshul Pardhi (BT13CSE010)

Bala Raveen Nadar (BT13CSE055)

Shubham Sharma (BT13CSE080)

Under the guidance of

Prof. Manish Kurhekar

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology,

Nagpur – 440010

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology, Nagpur – 440010.

2016-17

Certificate

This is to certify that the thesis entitled - “JARVIS, A GENERALISED CHATBOT”, is a bona fide work done by **Anshul Pardhi**, **Bala Raveen Nadar & Shubham Sharma** under the guidance of **Prof. Manish Kurhekar** in the **Department of Computer Science & Engineering, Visvesvaraya National Institute of Technology, Nagpur**, for the partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology** in “**Computer Science & Engineering**”.

Prof. Manish Kurhekar
Project Guide
Professor
Department of Computer
Science & Engineering
VNIT, Nagpur.

Prof. U. A. Deshpande
Head of Department
Professor
Department of Computer
Science & Engineering
VNIT, Nagpur.

Department of Computer Science & Engineering

Visvesvaraya National Institute of Technology, Nagpur – 440010.

2016-17

DECLARATION

We, hereby declare that the thesis titled – "JARVIS, A GENERALISED CHATBOT", submitted herein has been carried out by us in the Department of Computer Science and Engineering of Visvesvaraya National Institute of Technology, Nagpur under the guidance of Prof. Manish Kurhekar. The work is original and has not been submitted earlier as a whole or in part for the award of any degree/diploma at this or any other Institution/University.

Anshul Pardhi
(BT13CSE010)

Bala Raveen Nadar
(BT13CSE055)

Shubham Sharma
(BT13CSE080)

ACKNOWLEDGEMENTS

We take this opportunity to acknowledge our deep sense of gratitude to our project guide Prof. Manish Kurhekar, Department of Computer Science & Engineering, VNIT Nagpur for his impeccable guidance. We are also thankful to him for his constant motivation and the encouragement he provided to us throughout the project and also his innovative solutions to be the problems that we faced during this tenure. We sincerely thank him for providing each and every facility required for the successful completion of the project.

We also express our gratitude to Prof. U. A. Deshpande, Professor and Head, Department of Computer Science & Engineering, VNIT, Nagpur for his generous guidance, motivation and allowing us to take this project.

We are also indebted to the faculty of the Department of Computer Science & Engineering at VNIT, Nagpur. Last but not the least, we express our thanks to our parents and all our friends for their constant help, support and encouragement.

ABSTRACT

This project aims to build a generative chatbot, which will generate appropriate responses to any text input by the user. It uses the concepts of Machine Learning and Natural Language Processing for the response generation. For the implementation part, Google's open-source Machine Learning library, TensorFlow is used. This project is based on a "Neural Conversation Model" and "Sequence-to-Sequence Models". While Recurrent Neural Networks (RNNs) are used to generate outputs based on previous inputs, Long Short-Term Memory (LSTM) networks are used for storing those previous inputs. Facebook Messenger is used as the interface where the user types the input in the form of a chat message to our created Facebook Page and in turn, gets a response from the chatbot.

The chatbot provides results which grammatically sound correct. It is able to hold the conversation along with appropriate responses to some user inputs based on the training provided.

CONTENTS

ACKNOWLEDGEMENTS……i

ABSTRACT……ii

CONTENTS……iii

LIST OF FIGURES……v

1. INTRODUCTION.....1

1.1 Motivation

1.2 Types of Chatbots

2. CONCEPTS……4

2.1 Concepts in Neural Networks

2.1.1 Artificial Neural Network

2.1.2 Feed Forwarding

2.1.3 Backward Propagation

2.1.4 Recurrent Neural Networks

2.2 Sequence to Sequence Architecture

3. DEEP NEURAL NETWORKS……...8

3.1 Recurrent Neural Networks

3.2 LSTM Networks

3.2.1 Introduction to LSTM

3.2.2 Concepts of LSTM

4. AN INTRODUCTION TO TENSORFLOW……..17

4.1 What is Tensorflow?

4.2 More about Tensorflow

4.2.1 Data Flow Graph

4.2.2 Activation Function

4.3 Advantages of Tensorflow

5. IMPLEMENTATION IN TENSORFLOW……..19

5.1 Introduction

5.2 Dataset and Trained Model

5.3 Model

6. DEPLOYMENT ON FACEBOOK MESSENGER……...22

6.1 Setting Up

6.2 Messenger Architecture

6.2.1 Facebook Chat Platform

6.2.2 Web Application

6.2.3 Bot Engine

6.3 Workflow

6.4 Screenshots

7. CONCLUSION AND FUTURE SCOPE…….....27

7.1 Result and Conclusion

7.2 Future Scope

REFERENCES……..29

LIST OF FIGURES

1.1 Chatbot Conversation Framework.....3
2.1 Feed Forward Neural Network.....5
2.2 Sequence to Sequence Architecture.....7
3.1 RNN Cell.....8
3.2 Unrolled RNN.....9
3.3 Long-term Dependency issue with RNN.....10
3.4 Repeating Module in Standard RNN.....11
3.5 Repeating Module in LSTM.....11
3.6 Notations.....12
3.7 Cell state in LSTM.....12
3.8 Gate.....13
3.9 Forget Gate.....14
3.10 Update Gate.....14
3.11 Resultant Tensor.....15
3.12 Output Tensor.....16
5.1 Sequence to Sequence Model.....20
6.1 Messenger Architecture.....22
6.2 Screenshot 1.....25
6.3 Screenshot 2.....26

CHAPTER – 1
INTRODUCTION

Chatbots or conversational agents are artificially intelligent computer programs that can simulate a conversation with a human, just like a real person. They are one of the most exciting aspects of Artificial Intelligence and Machine Learning. Improvisation and development in the field of chatbot by researchers is a constant process. However, these chatbots being specific to a particular domain and following hand-crafted rules, often don't work well for other domains and as a result, don't get a large audience.

Development of chatbots using machine learning follows a statistical approach. Supervised learning helps to build chatbots that can learn from labelled datasets and modeling sequences of variable length. Language Translation involves such architectures. Today's computational capabilities allows to perform matrix operations using modern Graphical Processing Units (GPUs) on large scale efficiently.

Our project revolves around developing a conversational bot following a Neural Conversational Model and sequence to sequence architecture using machine learning capabilities.

1.1 Motivation

The idea of chatbot is revolving around since a long time. However, lack of efficient computational capabilities to operate on large dataset tethered its growth in past. Recent advancements in technology supports processing large datasets and extracting patterns to compute the suitable outputs as response by the chatbot. This motivates us to learn and understand the concepts and apply the same to build a generalised chatbot.

1.2 Types of Chatbots

Chatbots can be broadly classified as:

A. On the basis of responses generated:
   1. Retrieval-based chatbots, which are systems that contain pre-trained responses stored in the databases and use parsing techniques on user input to find context which is then applied to select a pre-defined response.
   2. Generative-based chatbots, which do not work on pre-defined responses and rather perform analysis on the user input and later use this knowledge to generate responses based on previous training on a large dataset using deep neural networks.

B. On the basis of conversations:
   1. Open domain, carries generative and retrieval based chatbots with the knowledge being open to several domains or any domain for generating or retrieving outputs over the given input.
   2. Closed domain, also carries generative and retrieval based chatbots with the knowledge being specific to one particular domain for generating or retrieving outputs over the given input.

Following figure suffices the above-mentioned details.

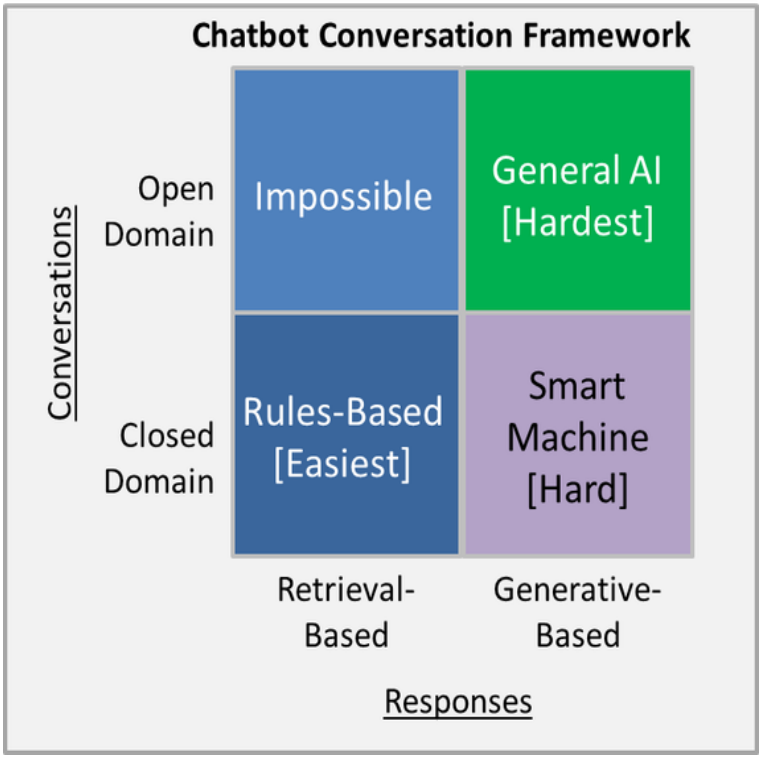


Figure-1.1 Chatbot Conversation Framework

From figure 1.1, it is evident that closed domain retrieval-based chatbots are easiest to build while open domain generative-based chatbots are hardest to build. Our project idea revolves around closed domain generative based chatbots which are the smartest of all the chatbots

CHAPTER – 2

CONCEPTS

2.1 Concepts in Neural Networks

2.1.1 Artificial Neural Network

Artificial neural network is a structure which establishes set of connections between input neurons called as sensors and output neurons called as effectors that is layered and well-organized. An artificial neural network starts off gathering input which resembles to observing the world, processing the data through the neurons and completes with output neuron effectors containing values for decision making.

2.1.2 Feed Forwarding

A feed forward neural network is considered an inspiration from the biological classification algorithms which consists of units, simple neurons, that have been organized in layers wherein units residing in a layer has connections with all the units from the previous layers. The connections between layers resemble to knowledge of a network and may have different weights.

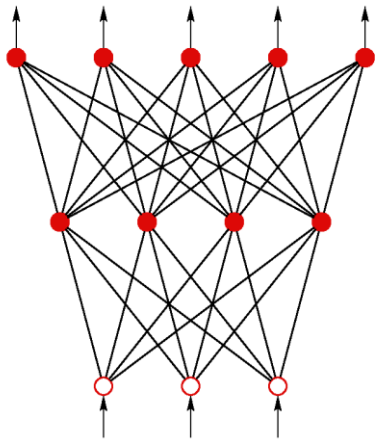


Figure-2.1 Feed Forward Neural Network

The process starts with the data entering at inputs which then passes layer by layer through the network until it reaches at the outputs. Normal Operations where it acts as a classifier, feedback is not present between the layers. This corresponds to why they are called as feedforwarding.

2.1.3 Backward Propagation

Each time the results are obtained, error measures in results, with expectations, which helps to learn the network by modifying the behavior of its neurons so as to get improved results when it sees the same values. This process is called as back propagation. This process helps to obtain optimal values for the weights and biases of the network which then supports better prediction close to the expected outputs.

2.1.4 Recurrent Neural Networks

The above mentioned neural networks are useful for cases where relevance among the input text data is inconsiderable. Each input is independent to the other and together supports to train the model. Chatbots hold conversations based on the user input text where order among the words of the input has significance when feeded to the network loses its relevance. Also, feedforward neural network has a deterministic behavior and they can be modeled to converge where the inputs drive the outputs and targets can be predefined and later work backwards to the inputs. This leads us to Recurrent Neural Networks (RNN) which has the capability to use previous outputs along with current input to generate current outputs. The outputs work as the inputs in recurrent neural network which then leads to self-reinforcing feedback cycles.

Multilayer RNN models are at the core of deep neural networks for chatbots. This RNN, then further, is utilized in the sequence-to-sequence architecture based model for developing a generalized chatbot.

Later on, LSTM (Long Short Term Memory) model is preferred as an extension to RNN so as to come over the problems of vanishing gradient, exploding gradient, etc.

2.2 Sequence to Sequence Architecture

Our project builds the chatbot using the sequence to sequence architecture which uses two LSTM cells, viz. the encoder and the decoder.

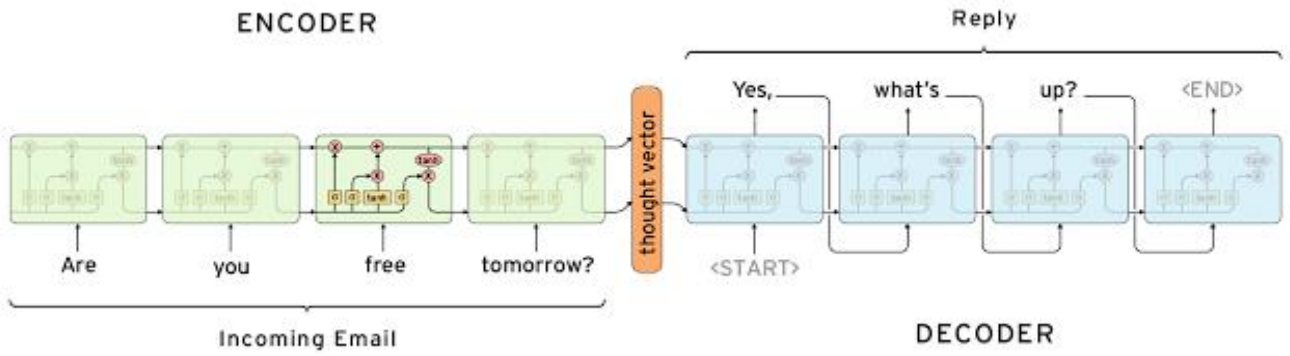


Figure-2.2 Sequence to Sequence Architecture

The encoder processes user input text using a multi-layered LSTM cell that accepts two inputs, user-text input word and output from the previous iteration as knowledge. This knowledge and user-text input word are then analyzed to understand the relevance among the words of user-text input. After processing of the user input, the decoder, which is another LSTM cell, starts with the prediction of words in sequence to form a sentence based on previous predicted word and sampling data from the vocabulary.

Each LSTM cell has three gates, viz. forget gate, update gate and result gate to optimize the training.

CHAPTER – 3

DEEP NEURAL NETWORKS

3.1 Recurrent Neural Networks

Humans don't start their thinking from the scratch every second. We understand each word based on our understanding of previous words. We don't throw everything away and start thinking from the starch again. Our thoughts have persistence.

Traditional neural networks can't do this, and it seems like a major shortcoming. For example, imagine we want to classify what kind of event is happening at every point in a movie. It's unclear how a traditional neural network could use its reasoning about previous events in the film to inform later ones. Recurrent neural networks address this issue. They are networks with loops in them, allowing information to persist.

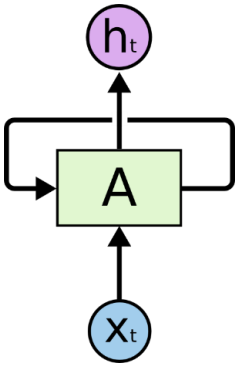


Figure-3.1 RNN Cell

In figure 3.1, ‘A’ represents the core of RNN cell which accepts input X_t and produces a value h_t along with passing the information from one step to the next. This recurrent neural network turns out to be not far different from an artificial neural network. It can be imagined as multiple copies of the same network that passes a message from the current step to the next step.

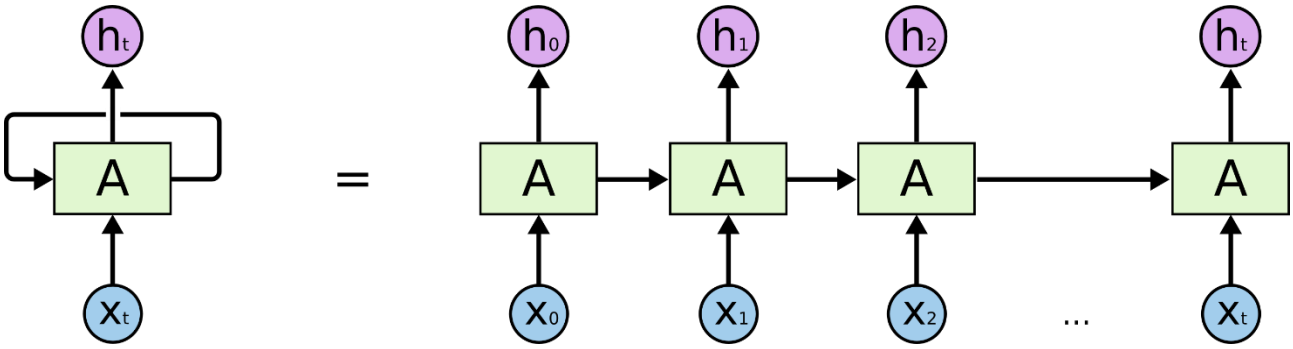


Figure-3.2 Unrolled RNN

This unrolled RNN itself relates to sequences and lists.

Unfortunately, standard RNN models work with small gap in relevance of words to predict. As that gap grows, RNNs find it difficult to connect the information which is better understood by the below figure.

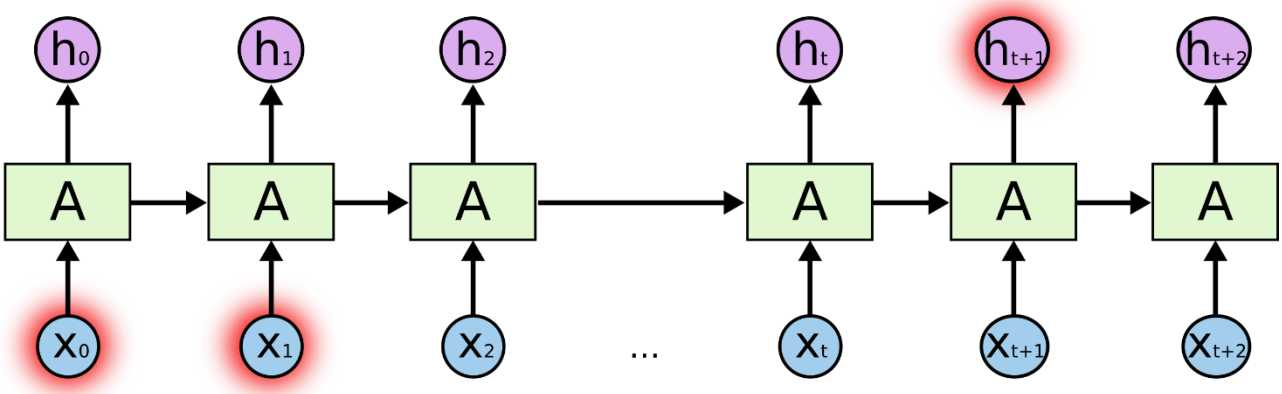


Figure-3.3 Long-term Dependency issue with RNN

In theory, RNNs are absolutely capable of handling such “long-term dependencies”. Sadly, in practice, RNNs don’t seem to be able to learn them. Also, there are issues related to exploding and vanishing gradient.

3.2 LSTM Networks

3.2.1 Introduction to LSTM

LSTM stands for Long Short Term Memory. These networks are capable of learning long-tern dependencies and are explicitly designed to avoid such problem. LSTM shows a default behavior of remembering information for long periods of time, not like they struggle to learn.

The recurrent neural networks are a chain of repeating modules. These modules have a simple structure in standard RNNs and consisting of a single tanh layer structure as shown in the figure below.

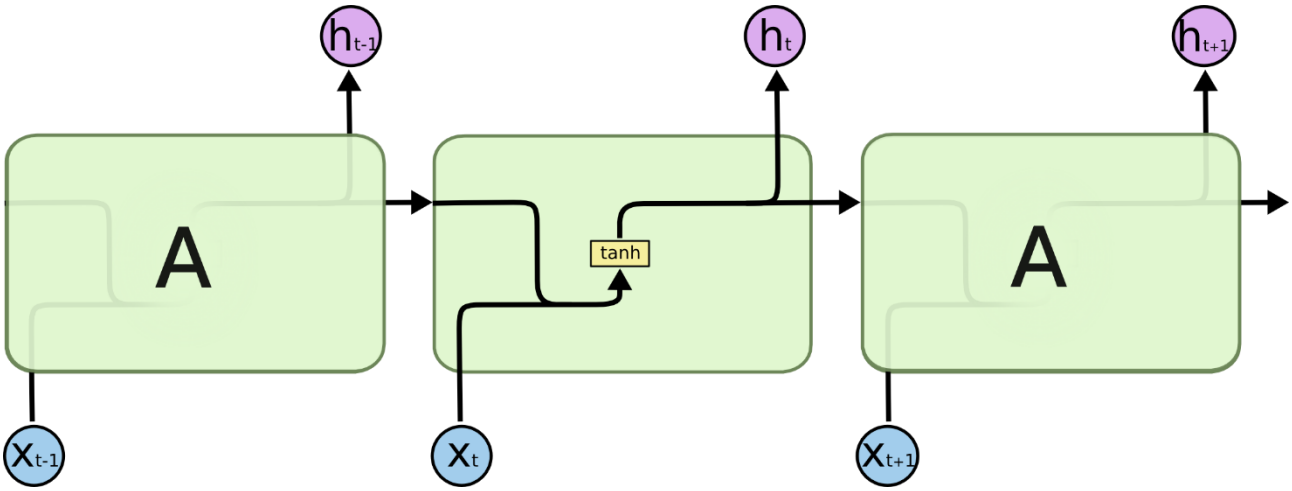


Figure-3.4 Repeating Module in Standard RNN

LSTMs also possess this chain, but with different structure of repeating module. There are four neural network layer interacting in LSTM. Details will be discussed in the next section.

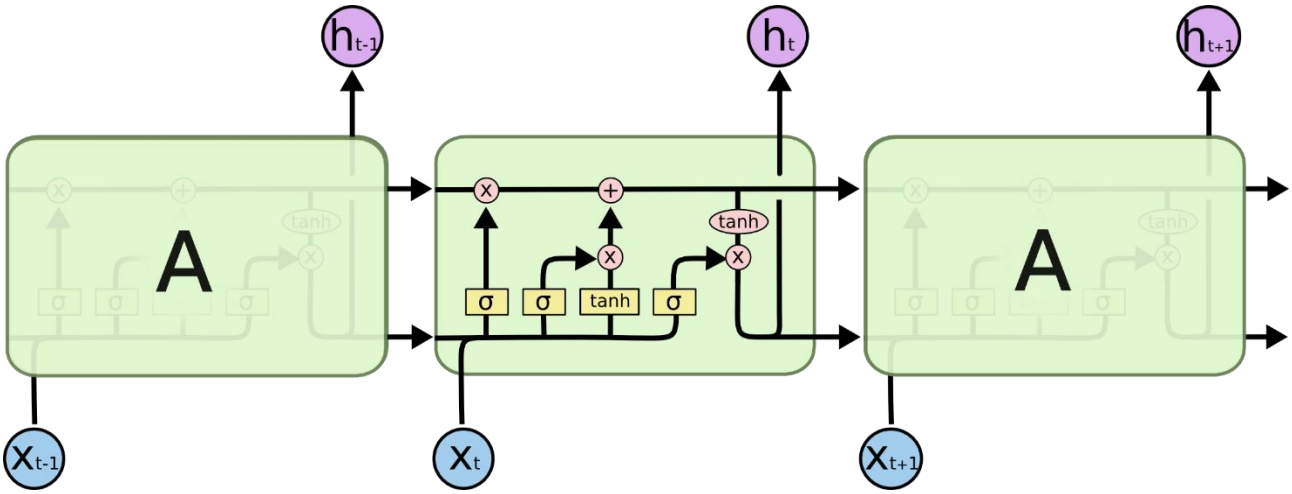


Figure-3.5 Repeating Module in LSTM

Following are the notations which we will be used further:

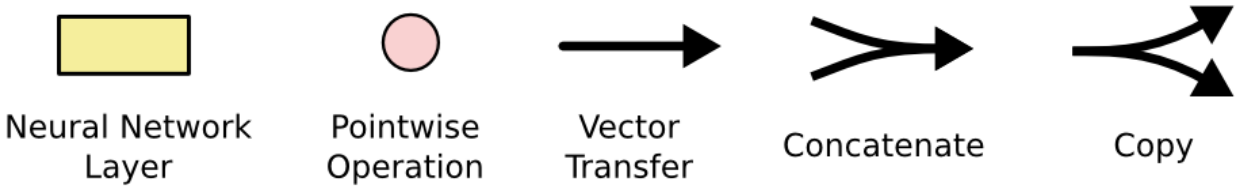


Figure-3.6 Notations

In figure 3.6, yellow boxes are learned neural layers, pink circles represent vector addition, line carries vector from the output of one node to the inputs of others, two converging vectors represent concatenation and two diverging vectors represent copying.

3.2.2 Concepts of LSTM

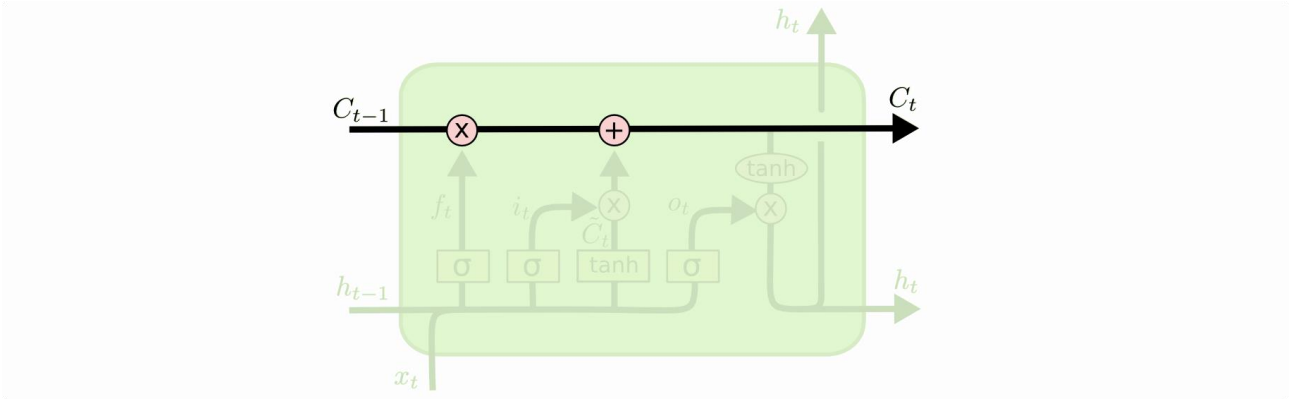


Figure-3.7 Cell state in LSTM

Cell state is the key to the LSTMs. In figure 3.7, the horizontal line is a cell state.

There are structures knows as gates in LSTMs which allows to add or remove information to the cell state. The cell state is kind of like a conveyor belt. It runs straight down the entire chain, with only some minor linear interactions. It's very easy for information to just flow along it unchanged.

The LSTM does have the ability to remove or add information to the cell state, carefully regulated by structures called gates.

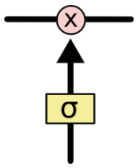


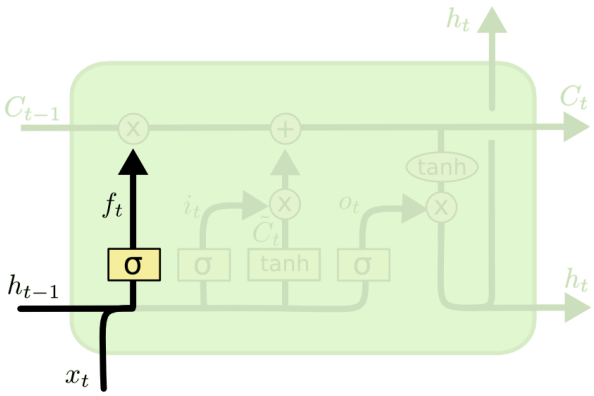
Figure-3.8 Gate

Pointwise multiplication and sigmoid net combines to form gates. The value produced by sigmoid net corresponds to how much should be let through each component and can be understood from the above figure.

The sigmoid layer outputs numbers between zero and one, describing how much of each component should be let through. A value of zero means "let nothing through," while a value of one means "let everything through!"

The first step in our LSTM is to decide what information we're going to throw away from the cell state. This decision is made by a sigmoid layer called the "forget gate layer." It looks at h_{t-1} and x_t, and outputs a number between 0 and 1 for each number in the cell state C_{t-1}. 1 represents "completely keep this" while 0 represents "completely get rid of this".

Let's go back to our example of a language model trying to predict the next word based on all the previous ones. In such a problem, the cell state might include the gender of the present subject, so that the correct pronouns can be used. When we see a new subject, we want to forget the gender of the old subject.

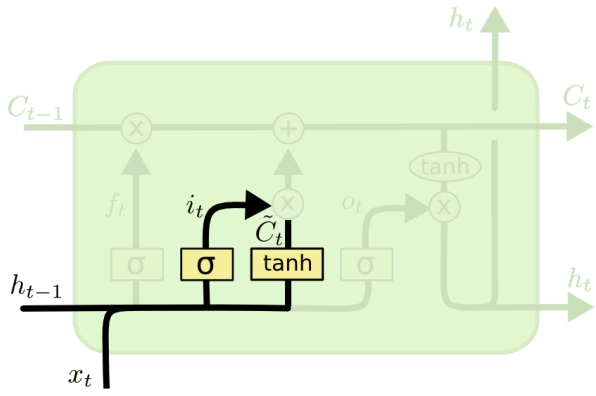


$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] \; + \; b_f\right)$$

Figure-3.9 Forget Gate

The next step is to decide what new information we're going to store in the cell state. This has two parts. First, a sigmoid layer called the "input gate layer" decides which values we'll update. Next, a tanh layer creates a vector of new candidate values, $\tilde{C}_t$, that could be added to the state. In the next step, we'll combine these two to create an update to the state.

In the example of our language model, we'd want to add the gender of the new subject to the cell state, to replace the old one we're forgetting.



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

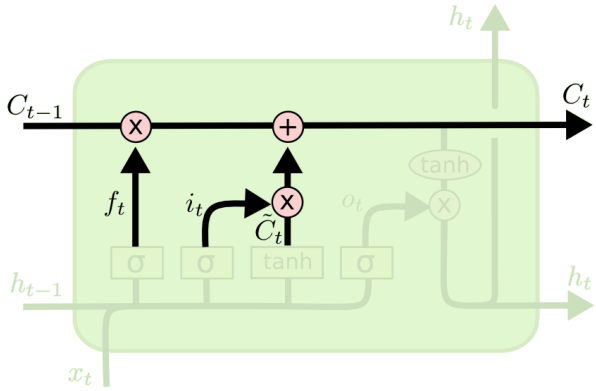
$$\tilde{C}_t = \tanh(W_C \cdot [h_{t-1}, x_t] \; + \; b_C)$$

Figure-3.10 Update Gate

It's now time to update the old cell state, C_{t-1}, into the new cell state C_t. The previous steps already decided what to do, we just need to actually do it.

We multiply the old state by f_t, forgetting the things we decided to forget earlier. Then we add $i_t * \tilde{C}_t$. This is the new candidate values, scaled by how much we decided to update each state value.

In the case of the language model, this is where we'd actually drop the information about the old subject's gender and add the new information, as we decided in the previous steps.



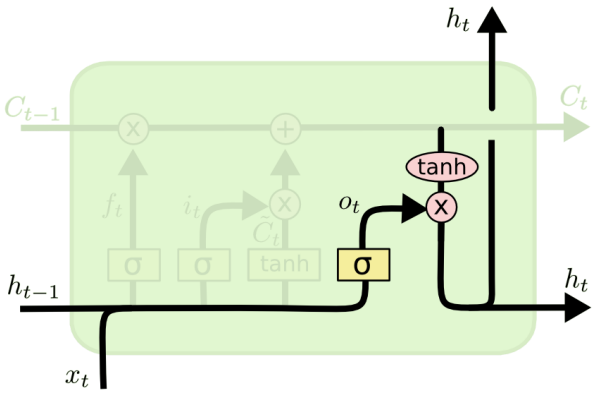
$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t$$

Figure-3.11 Resultant Tensor

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

For the language model example, since it just saw a subject, it might want to output information relevant to a verb, in case that's what is coming next. For

example, it might output whether the subject is singular or plural, so that we know what form a verb should be conjugated into if that's what follows next.



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

Figure-3.12 Output Tensor

CHAPTER - 4

AN INTRODUCTION TO TENSORFLOW

4.1 - What is Tensorflow?

Tensorflow is an open source machine learning library developed by the Google Brain Team. It's an extremely versatile library. It was originally created for tasks that required heavy numerical computations. As a result, Tensorflow's focus later deviated towards machine learning and deep neural networks. Due to a C++ backend, Tensorflow is able to run faster than pure Python code. Tensorflow uses a data structure called a data flow graph.

4.2 - More about Tensorflow

4.2.1 - Data Flow Graph

Tensorflow's structure is based on the execution of a data flow graph. A data flow graph has 2 basic units:

- Node - It represents a mathematical operation.
- Edge - It represents a multi-dimensional array, also known as a tensor.

The main function of a data flow graph is to build a graph and then execute using the 'run' and 'eval' operations. After the graph is built, in order to drive computation, an inner loop is written. Inputs are fed to the nodes through variables or placeholders.

Each data flow graph is made up of operations. To run these operations, the data flow graph is launched into a session. The session translates the operations and passes them to a device containing a CPU or a GPU for execution.

4.2.2 - Activation Function

Deep neural networks are capable of more complex behavior than their shallow counterparts. Each node or neuron processes input using an activation function. Different activation functions include the hyperbolic tangent, binary step and the logistic function. The choice of the activation function strongly affects the behavior and functioning of the neural network.

4.3 - Advantages of Tensorflow

- Tensorflow provides both a Python and a C++ API for developers. The Python API is better and it’s easier to use.
- Because of its flexible architecture, Tensorflow compiles way faster than other deep learning libraries. It supports CPUs, GPUs and even distributed processing in a cluster.
- A developer can scale up and develop models faster with different implementations.
- Tensorflow has built-in support for deep learning neural networks, so it is easy to assemble a network, assign parameters and run the training process.
- Tensorflow also has a collection of simple trainable mathematical functions that are useful for neural networks.

CHAPTER – 5

IMPLEMENTATION IN TENSORFLOW

5.1 Introduction

Chatbot uses Tensorflow for deep learning and has been trained on Reddit comments data. Model has been designed with RNN cells, specifically LSTM cells, for encoder and decoder of the sequence-to-sequence architecture.

5.2 Dataset and Trained Model

The chatbot has been trained on Reddit comments data. Original Reddit dataset consists of ~1.65 billion comments which is Terabytes in quantity. Our chatbot is trained on 5 GB of comments using GPU. The file is a series of JSON blocks delimited by new lines. Each JSON block has properties such as author, comment, etc. Each comment has an ID and is arranged in sequential order. During training on this dataset using our model, the weights and biases are calculated which fit the dataset responses. Losses are propagated for improvement in next iteration.

5.3 Model

We start with incorporating an LSTM cell to our model using **rnn_cell.BasicLSTMCell()** command in Tensorflow. This cell is then used to create a multi-layer RNN cell using **rnn_cell.MultiRNNCell()** command for improving the performance and better train the model.

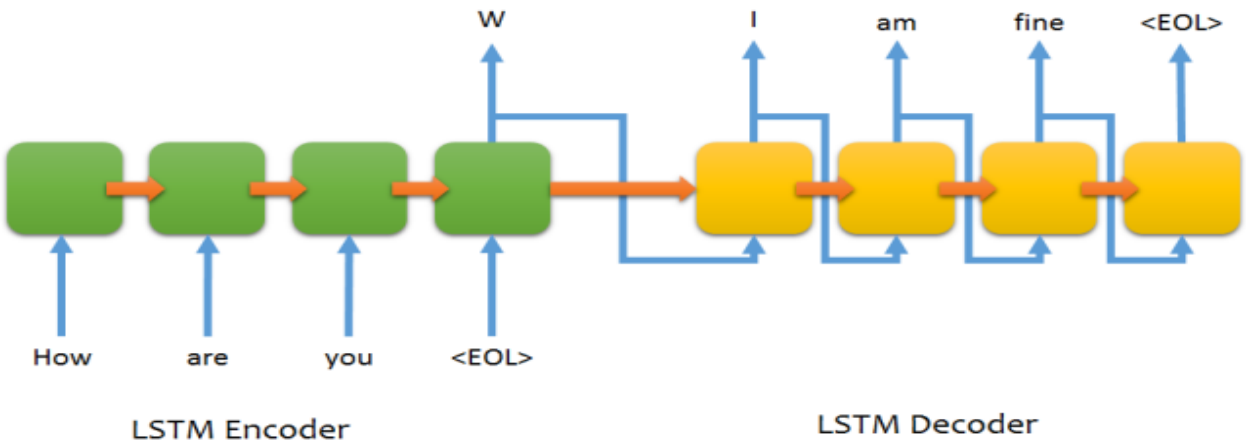


Figure-5.1 Sequence to Sequence Model

This LSTM cell accepts input and previous output to generate next output for the encoder and the process continues. This is a very general idea of how our encoder processes the user input text to develop knowledge to be submitted to the service decoder.

The command **seq2seq.rnn_decoder()** constructs the outputs and states which is a list of tensors. This is also the step where the weights and biases for the LSTM are calculated for optimal result generation. The output is a 2-D matrix where the rows correspond to a batch and the columns correspond to a

sequence word predicted where column size is sequence length specified through argument during training.

Logit, which is unnormalized output value, is then calculated using [**output * w + b**] followed by normalization to convert the logits to probability values between 0 and 1 using Tensorflow's **softmax()** function.

Finally, the losses are calculated. The command **seq2seq.sequence_loss_by_example()** returns log-perplexity which corresponds to comparison between logit and one-hot target. Then, cost has been calculated from the loss tensor. The cost is a floating-point tensor which the optimizer seeks to optimize. Optimization is then applied with current learning rate and summary is created.

Using the above model, the chatbot generates a sentence as response, sequence by sequence. Initially, the user input is parsed and tokenized. Each word of the user-text is then feeded in sequence to the encoder of the sequence-to-sequence model.

The encoder starts accepting words of user input and they are then provided as input to each iteration of the LSTM cell of encoder. Then, decoder starts to generate response. In each iteration, the LSTM cell of decoder sample's data from the vocabulary list based on the previous output as shown in the seq2seq diagram and predicts the probability for each word in the sampled data to be present in current iteration. Higher probability word is taken for response and the process repeats until final response is generated.

CHAPTER – 6

DEPLOYMENT ON FACEBOOK MESSENGER

6.1 Setting Up

1. Get Facebook Developer access to host the bot on to the Messenger platform.
2. Install Flask, gunicorn for transferring messages between Facebook's server and the chatbot.
3. Install ngrock to set up a local server on which the chatbot will be running.

6.2 Messenger architecture

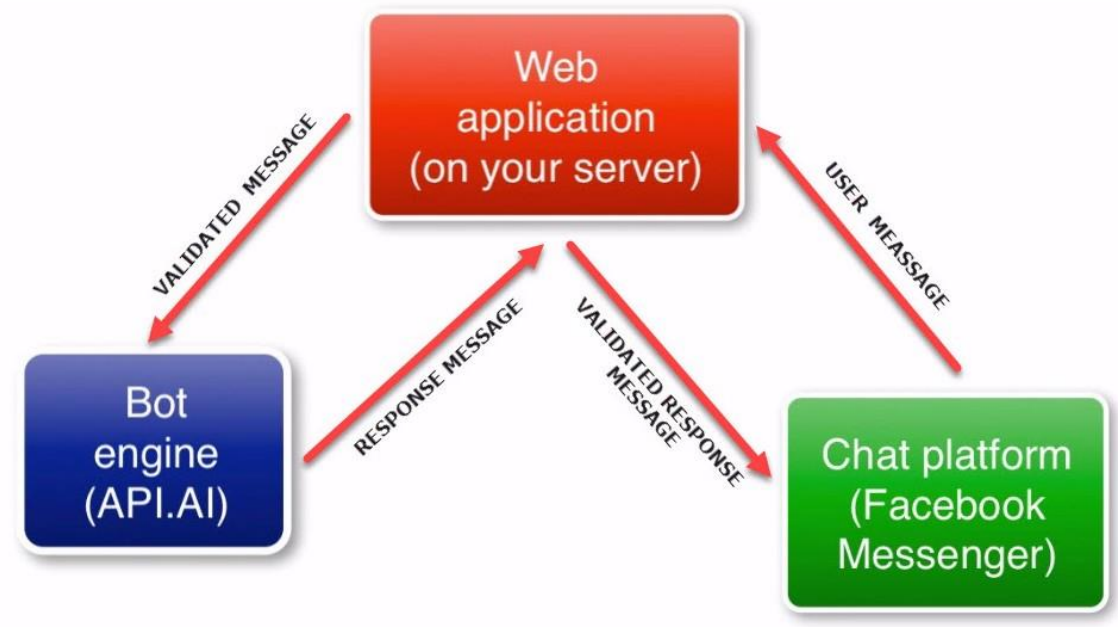


Figure-6.1 Messenger Architecture

The messenger architecture consists of 3 main entities:

1. Facebook Chat Platform
2. Web application
3. Bot Engine

6.2.1 Facebook Chat platform

It is the front end of the Facebook Messenger. It also provides the users the Graphical User Interface through which the user interacts with the bot and indulges in a communication.

6.2.2 Web application

This web application acts as the middle layer between the bot and the chat platform GUI. It links the Bot engine with the chat platform. Apart from providing a linkage between the Bot engine and the chat platform it also handles the HTTP requests and feeds the bot engine with the input data in the desired format.

6.2.3 Bot Engine

The Bot engine is the heart of a chat application. It is this entity which processes the user's messages and generates an output which is displayed to the user on the messenger GUI.

6.3 Workflow

1. The user initiates the conversation by typing and sending some text through the messenger GUI. The chat platform takes this plain text the user inputs and sends this user message to the web application middle layer in JSON format.
2. The middle layer after receiving this data, parses it and extracts the user's message form it. This message is then sent to the chat bot program which is processed to generate a response.
3. The Bot engine processes this user message and produces an appropriate response. This response is sent back the middle layer web application.
4. The web application takes the bot engine's response and pads the response with the sender ID and creates a JSON object so that messenger platform knows to which user the response is destined for.
5. The messenger platform again parses response and displays the message to the user.

6.4 Screenshots

Firstly, a Facebook Page was created named ‘Jarvis the bot’, whose message functionality is used as the chat platform. The user can interact with the chatbot by typing a message either in the message box of the web interface or by using the Facebook Messenger mobile App. The following screenshots depict the Facebook Messenger interface from where the chatbot interacts with the user.

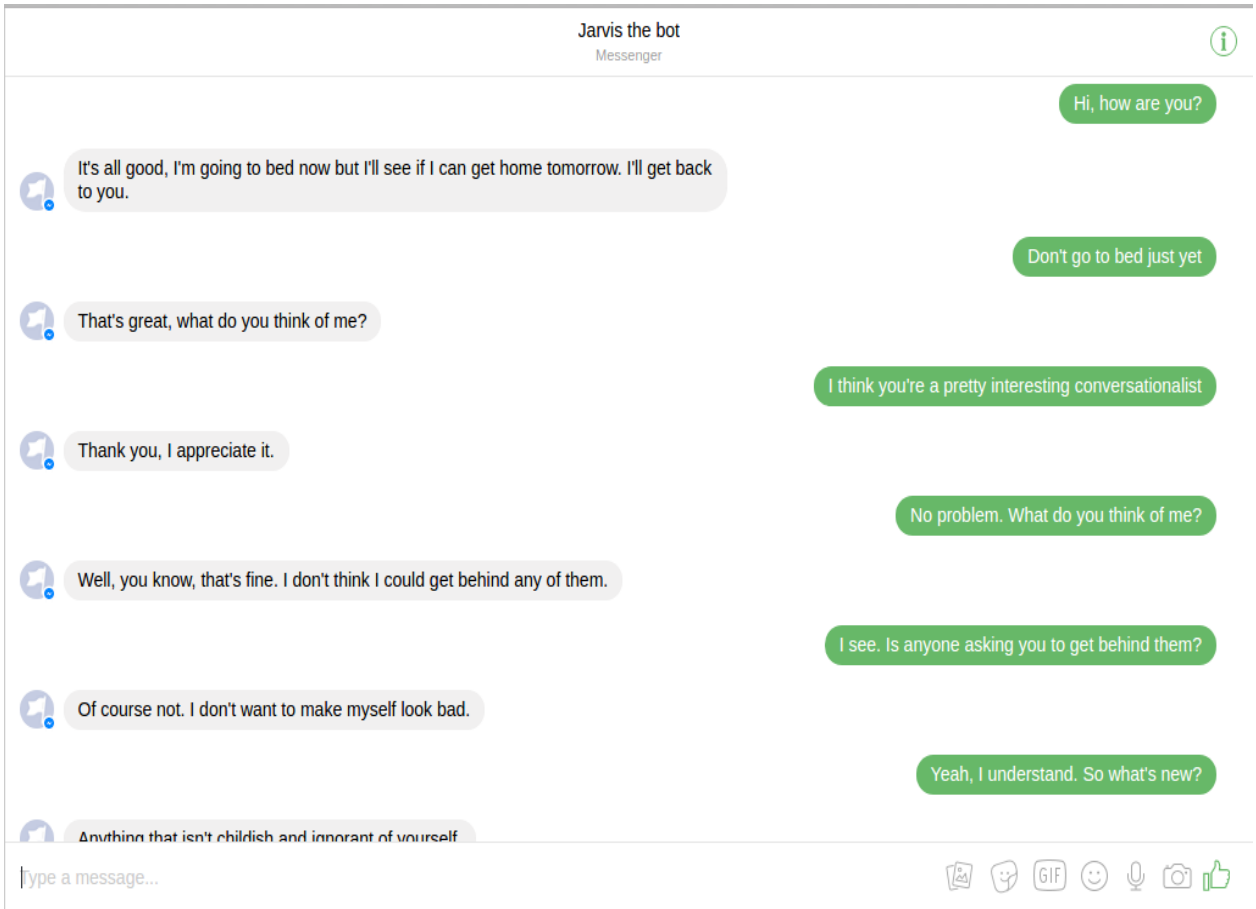


Figure-6.2 Screenshot 1

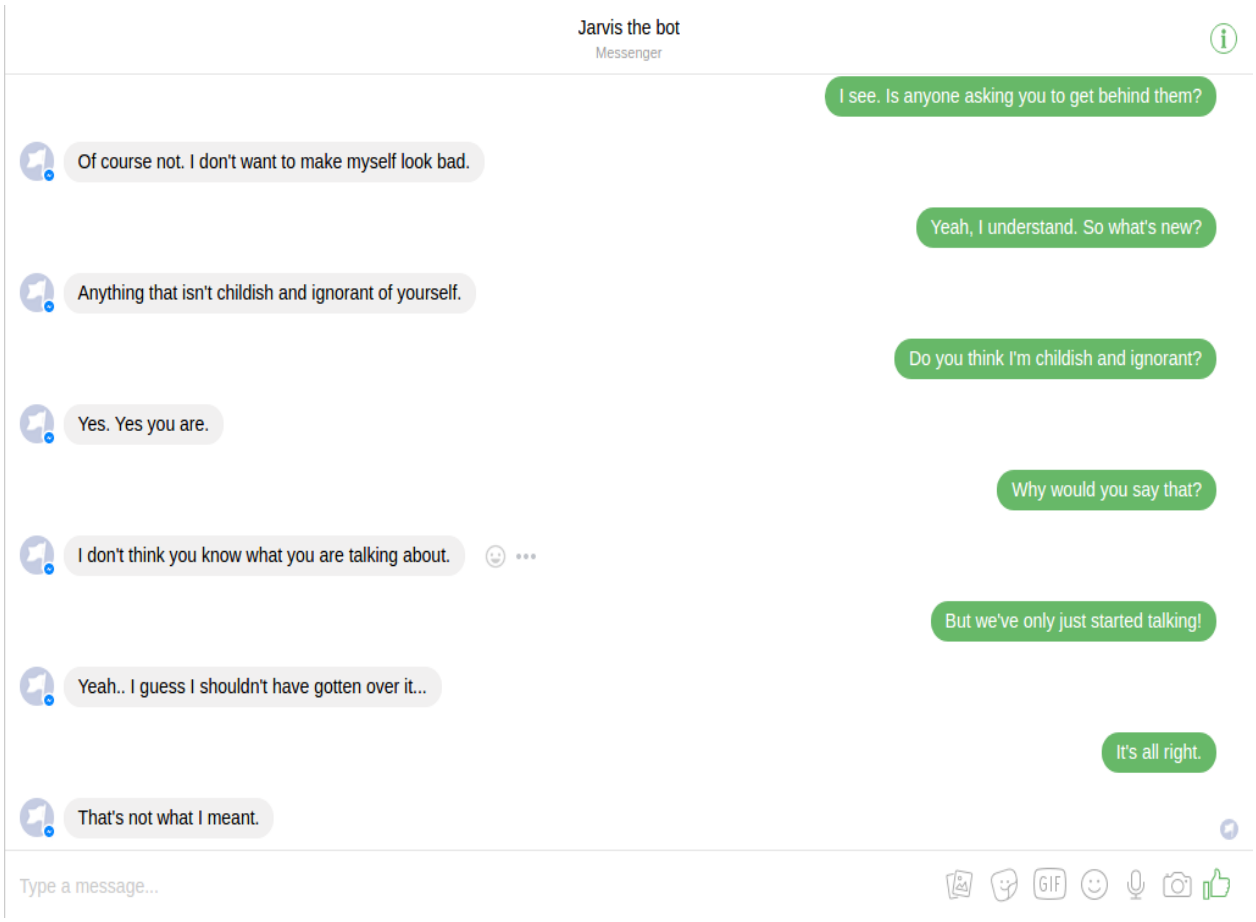


Figure-6.3 Screenshot 2

CHAPTER - 7

CONCLUSION AND FUTURE SCOPE

7.1 - Result and Conclusion

As the name suggests, ‘Jarvis, a Generalized Chatbot’, is a chatbot for common conversation and not on some specific topic between a human being and a computer. A user types the input and gets an appropriate response from the chatbot. Moreover, the chatbot is trained such that it will give different responses each time the same input is used, indicating its learning capability. It learns from its previous inputs and tries to give the most probable responses each time an input sentence is typed by the user. Facebook Messenger is used as the interface between the user and the chatbot. Tensorflow as the machine learning library, provides highly efficient and easy to use functions for the implementation of the chatbot. This project has given us a clear understanding of machine learning concepts, recurrent neural networks, LSTM networks, sequence to sequence architecture and hands-on experience with Tensorflow libraries. The project has given us invaluable experience which will definitely be useful in our further studies as well as in the industry.

7.2 - Future Scope

The chat responses can be improved by using a better dataset and training model containing more suitable sentences for a generalised conversation. On the same dataset and training model, the results can be improved by using appropriate weights and biases for the deep neural network. A suitable activation function can help further improve the results. Moreover, we plan to promote our Facebook page ‘Jarvis the bot’ and make it accessible all time from anywhere in the world. It would make any person chat with our chatbot. This can be done by deploying the application to a suitable cloud platform. Currently, it runs on a local server. We also plan to further research on the topic and develop additional features to our chatbot like speech-to-text and text-to-speech functionalities, and make the bot behave like a personal assistant to the user.

REFERENCES

- Vinyals, O. and Le, Q., 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.
- Sutskever, I., Vinyals, O. and Le, Q.V., 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems* (pp. 3104-3112).
- Michael Nielsen, “How the back propogation algorithm works”
- Saeed Aghabozorgi, “Deep Learning with Tensorflow”
- Stefan Kojouharov, “How can I build an intelligent chatbot”
- Cristopher Olah, “Understanding LSTM Networks”
- Marcus Beckenkamp, “Building and publishing Facebook Messenger chatbot”