UNIVERSITY PARTNER

UNIVERSITY OF WOLVERHAMPTON

HERALD COLLEGE KATHMANDU

Project and Professionalism

(6CS020)

A2: Project Report

Online Support Chatbot

Student Id	: 1928430
Student Name	: Rukesh shrestha
Group	: C3G1
Supervisor	: Mr. Rupak Koirala
Cohort	: 3
Submitted Date	: 10th June, 2020

Declaration sheets

Award Title: BSc(Hons) Computer Science

(Presented in partial fulfillment of the assessment requirements for the above award.)

This work or any part thereof has not previously been presented in any form to the University or to any other institutional body whether for assessment or for other purposes. Save for any express acknowledgements, references and/or bibliographies cited in the work. I confirm that the intellectual content of the work is the result of my own efforts and of no other person.

It is acknowledged that the author of any project work shall own the copyright. However, by submitting such copyright work for assessment, the author grants to the University a perpetual royalty-free license to do all or any of those things referred to in section 16(I) of the Copyright Designs and Patents Act 1988. (viz. to copy work; to issue copies to the public; to perform or show or play the work in public; to broadcast the work or to make an adaptation of the work).

Student Name: Rukesh Shrestha Student Id: 1928430

Student Signature Date: 04/06/2020

Abstract

Online support chatbot is a hybrid chatbot which is developed to solve minor issues in Internet Service Provider (ISP) online support. Chatbot is a conversational agent which talks to users in natural languages. Purpose system is hybrid chatbot which is the combination of two chatbot i.e retrieval based chatbot and generative based chatbot. This chatbot is developed using deep learning libraries. They have the capabilities of understanding natural language and processing it. They have cognition to solve monotonous queries. This project is based on an iterative and incremental process model. Purpose system has web applications developed from Django. Chatbot is deployed in the web application. Users are authenticated, can register, login and chat with bot.

Table of Contents

Table of Table

Table of Figure

1 Introduction

Services provide to the user/client by understanding the natural language is known as the chatbot. It is mainly used for the customer services. It has been used in different sector for the purpose of the information gathering or provide the required information and guidance. It is one of the AI program that simulates the human activities of the conversation or speech by using key pre-calculated user phrases, auditory and text based signals (Abdul-Kader & Woods, 2015). Artificial writing beings which can communicate to the human beings in certain bases such as text based, spoken conversation or the non-verbal conversation is the chatbot. It is the software or AI application which are develop to talk to human. This application has become fascinating, stimulate and engrossing to entire world.

History of Chatbot

Chatbot has the long history back to the mid-nineties. Lots of chatbot has been develop till now. Below is the time graph development of chatbot

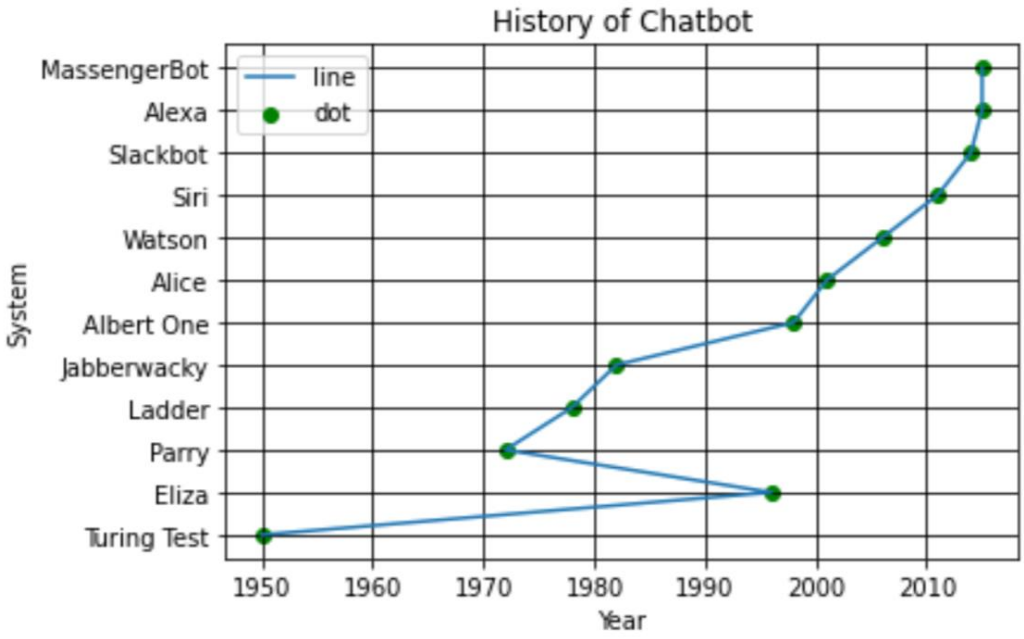


Figure 1 Graphical Representation of chatbot history

The detail of all history can be found in appendix 9.19.1.

1.1 Problem Domain and Project as Solution

Nowadays the internet has become basic needs of human beings. As it has become the basic need facility, to maintain the quality of connection all the time different ISP Companies have made different departments. Online Support is one of the important departments. Thousands of calls, e-mail messages, and chats are received due to the deterioration of the internet services provided by the ISP Company (Bergum & Iveland, 2019).All the calls and the messages sent by the user are not received. Lots of abundant calls and missed chats/messages are recorded. Manpower who is working in the concern department may suffer from different health problems as well as irritates from the chat. To maintain the quality of services chatbot can be an efficient software. It can solve the minor queries such as Wi-Fi password update, slow browsing, new connection queries, and update packages queries. Working eight hour a day by texting and talking to humans is impossible. Humans cannot be 100% efficient. While talking to clients, there may be some mistake. Same problem and question can be repeatedly asked by the client. Humans may get frustrated and lose their patience but the chatbot can answer the same question again and again without losing the patience and also encourages clients to solve the problem. In the long term view this technology can be proven as the cost effective related to human effort.

Type of Chatbot

There are mainly three types of chatbot. They are retrieval based chat-bot, generative based chat-bot and hybrid chatbot. Chatbot contains a certain collection of the Frequent Ask Question (FAQ) template which match the keyword spoken by the user and generate the response is the retrieval based chatbot. Chatbot contains a neuron network and generates the reply from input data based on natural language generation techniques is generative based chatbot (Chowanda & Chowanda, 2018). This kind of chatbot is trained in the huge amount of real data. Combination of both retrieval and generative chatbot are the hybrid chatbot (IrynaKulatska, 2019).

In the final year project hybrid chatbot is going to be developed. This chatbot uses the Deep learning neural network model. Recurrent Neural Network (RNN) is used in this project. RNN is the power algorithm to address the problem of sequence data. Long Short-Term Memory (LSTM) architecture is used to develop the model.

LSTM is used for sequence to sequence problems as it has the cognition of memories the data input by the user and can easily train the huge sequence model architecture. The model is trained with the dataset and used to generate the responses (IrynaKulatska, 2019).

There are lots of applications of the LSTM. In this project, Conversational Modeling is preferred. The domain of this project is to generate the sequence of text from the asked question (Brownlee, 2017).

1.2 Academic Question

How could chatbot solve issue of the online support and how it works?

1.3 Aims and Objectives

1.3.1 Aims

- To solve the relevant problems of online support prevailing in the ISP company by developing the chatbot.

1.3.2 Objectives

- To develop the Hybrid chatbot.
- To develop the chatbot using the neuron network.
- To develop and deploy the chatbot in web application.
- To maintain quality of services.
- To solve the minor repetitive queries.

1.4 Artifact produced and background

Artifact 1: Chatbot

It is a computer program which is developed to make the work comfortable and more accurate. Deep-learning component of AI is used to develop chatbot. Recurrent Neural Network (RNN) is going to be used for developing this chatbot. Long Short-Term Memory of RNN is going to be used for developing models. This model is mainly used in complex systems where the machine has to learn by understanding the human languages and posture. Keras sequential layers are used for retrieval based chatbot.

Artifact 2: Web Application

In this project, a web application is developed to deploy the chatbot. Django framework is used to develop the web application. This framework is free and open source. It has its own server to deploy the web pages.

Background

In the context of Nepal, there are many ISP Companies which provide chat services to clients. Thousands of queries are handled by the representative. Some of the queries are minor and some of them are complex to solve. Minor queries can be solved by the chatbot. Therefore, the representative feels less burden to work and tries to solve the complex issues.

1.5 Scope and Limitation of the project

Scope

Purpose system can guide clients to solve the minor issue such as updating the Wi-Fi password from mobile application and web application, troubleshoot the basic internet slow issue, and provide department number to client e. t. c. Solving this issue will help to benefit employees as well as clients to not wait for a long time, each representative can handle the other issue at the same time.

Limitation

- Purpose system is not able to solve the complex network issue such as:
    - Latency issue face by the client.
    - Handle the physical network issue.
    - Check the client profile.

1.6 Introduce the structure of report

Introduction

It includes the meaning of purpose system, its history, types, problem domain and project as the solution.

Literature Review

It includes the related project description, diagram and its results. Here, the literature review is divided in two part i.e. generative based review and the hybrid based review.

Supportive information

It includes the technical related information such as network definition and mathematics behind it. It includes the libraries and the framework description.

Main Body

It is divided in to two part i.e. AI model development and the software model development. Development includes who the system is developed and its phases with the development process.

Answer to academic question

It includes answers to the academic question with the proof.

Conclusion

It includes what has been discovered while writing and developing the project.

Critical evaluation of the product

It includes the finding, difficulties and opinion.

References and the Bibliography

It includes all the sources from where the project is developed and report is written.

Appendices

It includes the project planning material, project visualization code, project code and some of the theoretical definitional.

2 Literature Review

2.1 Generative chatbot review

2.1.1 Review 1:

According to the (Zalake, 2019) Generative Chatbot is developed using the RNN sequence to sequence Bidirectional LSTM architecture which is advance than the traditional RNN architecture. Different technics are found to develop the chatbot, in the past rule/retrieval based chatbot development technics were used but now saturated technic (Sequence to Sequence) models is used for chatbot development. In short, seq2seq map the sequence of text to another sequence which generate the output. Traditional technics used handcrafter/manually formed rules but seq2seq is end-to-end solution which lacks hand crafted rules and enhances the chatbot architecture.

This system can provide, end-to-end solution to the issue which results output of the given query by using the attention mechanism technics. Attention mechanism is the important technics which focuses on the important query sequence and on the basic of than sequence the result is generated. System develop using this model can learn from the queries and generate responses. For developing the AI system, data is the most predominant things. Data to train the model is one of the factor which determine the system performance. For the development of this chatbot data was clasped from the two source. Multi-domain dataset developed from the raw scripts of 600 movies was the first source and another was the single domain dataset which encompasses of the chat logs from the Ubuntu technical support channels (Zalake, 2019).

Array of the fixed length tensor or the matrix is passes as the absorption of the Neural Network (NN). This NN are not capable of taking the natural languages as the inputs. The process of converting the raw text to NN readable form is the preprocessing. In this system, the absorption and out-turn sentences are encoded by words (Zalake, 2019).

Here, let us suppose Tx: Number of maximum words in absorption sentence, Ty: Number of maximum words in out-turn sentence. For the absorption sentences, filter is done with the simple assumption of sentences which are less than or equal to Tx steps/words are taken. For the out-turn sentences, filter is done with the simple assumption of sentences which are less than or equal to Ty steps/words are taken. For the padding, words of the

sentences which has less than Tx/Ty words is padded with token '<pad>' until each words becomes Tx/Ty. One Hot Encoded sequence of word is passed of size (100000, 15, 7000) to the input layer of the model. The first parameter is the number of input sequence, second parameter is Tx and the last parameter is the size of the vocabulary. If the system sees the word which are not in vocabulary then that kind of word is handle by the '<unk>' token (Zalake, 2019).

To develop the model Bidirectional LSTM is used which has the two LSTM cells. Two LSTM cells is known as the forward LSTM cell and the backward LSTM cell. Sequence of the data is learn from the training of the model of both the LSTM. The first LSTM cell takes the absorption as the one hot encoded sequences and goes to the time steps of Tx. The second LSTM cell takes contexts absorption enumerate from the attention mechanism goes up to time steps Ty (Zalake, 2019). Below figure is the architecture of the model used to develop this system.

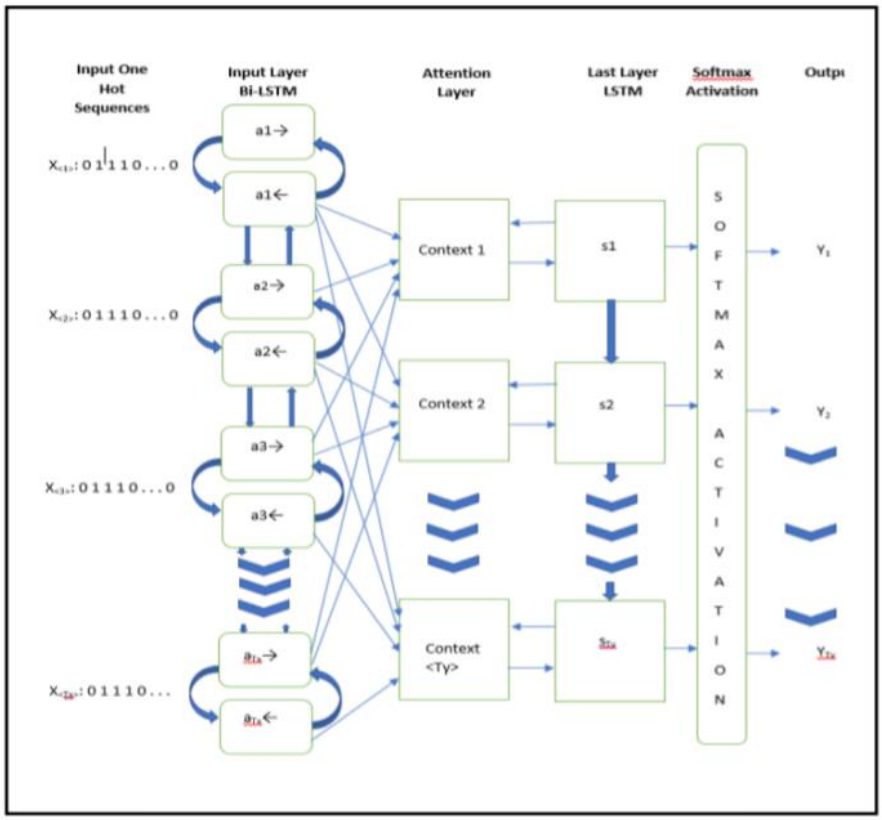


Figure 2 Model Architecture (Zalake, 2019)

While training the model, the number of epoch was set to 25000, input vocabulary size to 7002. The last two token in the vocab size is the '<unk>' and '<pad>' token. The declare vocab size is for both the NN i.e (encoder and the decoder). The padded sequence of the training sequences is of size 11 tokens. RMSprop stochastic gradient descent optimizer is used to optimize the model while training because this optimizer best suited for the RNN. To check the conversing speed Adam optimizer was also used but the result of the RMSprop was the best. Data was in the huge quantity so the batch size of 256 is used as it perform well to find the global minima point. Due to large training data and the configuration of the model architecture normal system can handle the processing of the model while training so Graphical Processer Unit (GPU) is prefer with the specification of the 50 Giga Byte (GB) of two GPUs. Although, the processing unit was powerful it took 48-72 hours to complete the training phase (Zalake, 2019).

After the training phase, evaluation of the model is done. Two dataset was taken to train the model.

First, Cornell movie corpus. Below figure shows the model performance.

Cornell Movie script corpus	
Token Number	**Accuracy**
1	88.46
2	89.58
3	90.15
4	90.84
5	91.8
6	91.93
7	92.87
8	93.18
9	94.04
10	94.73
11	95.81
12	96.37
13	97.23
14	98.37
15	99.37
Overall Accuracy/Loss	93.64/4.22

Figure 3 Cornell movie dataset result (Zalake, 2019)

Second, Ubuntu chat corpus. Below figure shows the model performance.

Ubuntu chat corpus	
Token Number	**Accuracy**
1	75.86
2	76.71
3	77.38
4	79.02
5	80.69
6	84.05
7	86.57
8	89.8
9	92.75
10	95.77
11	98.19
Overall Accuracy/Loss	85.16/7.99

Figure 4 Ubuntu chat corpus (Zalake, 2019)

Blue Score Result:

Example 1: Input source: just use xchat for irc ?
Original output: what program can be used to open .bin files ?
Predicted Output: to .bin ?can program what open used other
Score:
1-gram: 0.000000
2-gram: 0.471714
3-gram: 0.583675
4-gram: 0.645203

Example 2: Input source: can i install kde also even though i already have gnome?
Original output: be able to choose at logon or something?
Predicted output: logon or choose at able something? Be
Score:
1-gram: 0.875000
2-gram: 0.500000
3-gram: 0.632878
4-gram: 0.707107

Figure 5 Blue score (Zalake, 2019)

2.1.2 Review 2:

According to the (Bergum & Iveland, 2019), generative based chatbot can also develop from the machine translation technic, if the dataset are well enough and have the towering marvelous. Purposed system is the generative based domain specific chatbot in both English and Norwegian language. Chatbot implementing in the Norwegian language is the challenging task as this language dataset are not found and has the syntactic structure. This system is develop using the seq2seq model with the attention mechanism (Bergum & Iveland, 2019).

The goal of the purpose system is the chatbot develop in the Norwegian language can be creditably. Mainly, the deep learning models are used for the machine translation this system verify seq2seq model can also be used for creating the general purpose Norwegian chatbot (Bergum & Iveland, 2019).

Purposed system has two different chatbot with different languages. System is develop in two different languages, there will be two different dataset one for the English language and one for the Norwegian language. Dataset for the Norwegian language cannot be easily found. For the Norwegian language, data was retrieve from the personal messaging online platforms such as Facebook and Facebook messenger. The data was taken from the two Facebook user account where the corpus become more than 110000 utterance pairs. As the data has collected in the huge quantity due to the memory issue corpus is moderate to 10000 utterances. For the English language, data was taken from the Cornell Move Dialogs Corpus. This corpus consists of movie scripts data (Bergum & Iveland, 2019).

Before the data is passed to the model for the training preprocessing steps should be consider. This processing include doffing all the noise data from the dataset, new lines replace with the spaces and all the data is converted to the lowercase. Words are tokenized with the Norwegian Natural Language Processing Toolkit (NLTK). The data is split into input and the targets sets. Each utterance is limited to 20 tokens due to the limitation computational power. Model cannot process the raw data. Data should be in vector, matric or in tensor. Converting the word to the specific vector is word embedding.

Here the word is embedded and padded to make the word each of fixed length. Below figure shows the embedded and padded word (Bergum & Iveland, 2019).

Index	0	1	2	3	4	5	6
Token	<PAD>	<BOS>	how	are	you	doing	<EOS>

Figure 6 Embedded word with padding (Bergum & Iveland, 2019).

Now the data is ready for train in the model. Now model has to be created. Here the model is develop using the Keras deep learning library. Seq2seq model is implement with attention mechanism. In this model, encoder neuron network dovetails input layer with the drop out and the Gated Recurrent Unit (GRU) layer. The output of this layer is feed to the decoder GRU. The result of the decoder GRU is the input of the attention layer. Drop out is the method for overcome the issue of the overfitting. Drop out class helps to learn all the neuron equally. The encoder layer takes the one hot encoding sequence convert it to fixed length vector which holds the meaning of the query in vector form and that vector is feed to the decoder layer. Decoder is trained to response the output of the sequence one by one until the sequence token is result. Below figure is the model architecture of the purposed system (Bergum & Iveland, 2019).

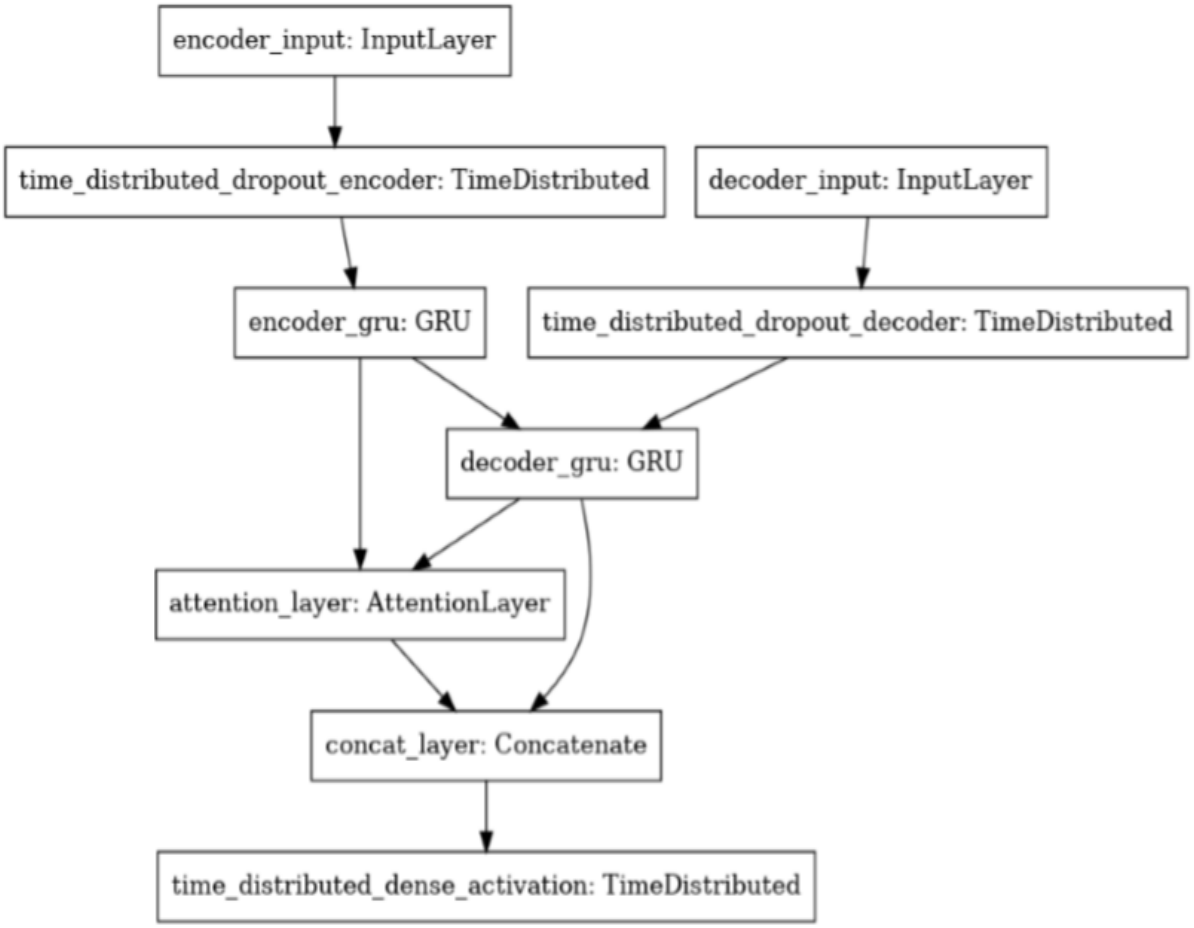


Figure 7 Keras model for chat-bot with LSTM-GRU *(Bergum & Iveland, 2019)*

While training the model, Categorical Cross entropy loss function is used to determine the loss on the model. To optimize the model rmsprop optimizer is used. Model is trained for the 300 epochs and the batch size is 64 (Bergum & Iveland, 2019).

To make our model to work inference model should be develop which takes the query check whether the input query can be converted to model input layer define size and generate the responses. The inference model does not train the obtain data again but it check the condition of the preprocessing and converted to the numpy array to feed to the model. The model work same as describe above in the model architecture. Here the purpose system using the greedy decoding scheme for the inference model (Bergum & Iveland, 2019).

After the system is complete, in training phase loss seems to be around 0.1 after the mention epoch in model architecture. Supervised human evaluation is used to check the performance of the system. Below figure shows the performance of the system in both the languages (Bergum & Iveland, 2019).

Norwegian	English
3	1
4	3
5	2
1	1
3	2
2	6
Avg: 3	**Avg: 2.5**

Figure 8 Evaluation results (Bergum & Iveland, 2019)

After the model evaluation with the help of the inference model chat is perform in both the languages. Below figure shows the result of both the languages.

Human:	hei
Bot:	går det greit
Human:	ja det går fint
Bot:	nei
Human:	jeg sa akkurat at jeg har det fint
Bot:	har ikke noen den så mye men du ikke å
Human:	er du levende
Bot:	ok
Human:	er det et ja
Bot:	e e det
Human:	du skremmer meg
Bot:	nei
Human:	jo, jeg blir skremt
Bot:	så det er jo smud
Human:	er du på gløs?
Bot:	ja
Human:	har du bord?
Bot:	ja
Human:	nice
Bot:	nei det er ikke noe særlig
Human:	er det pk på kontoret?
Bot:	kanskje
Human:	kan du sjekke for meg
Bot:	nei
Human:	da får jeg ha med selv
Bot:	hmm
Human:	hmm
Bot:	for et liv

Figure 9 Norwegian Chatbot (Bergum & Iveland, 2019)

Human:	hi
Bot:	hey there tired of
Human:	are you tired of me?
Bot:	let me see a long
Human:	a long what?
Bot:	how much
Human:	what do you want
Bot:	in the sight of god that is a sin
Human:	please forgive me
Bot:	im no i was to been to you for did its
Human:	you make no sense now
Bot:	thats not you there to the
Human:	alright
Bot:	alright
Human:	are you on campus?
Bot:	well i have to be there to that
Human:	what are you doing there
Bot:	there
Human:	right
Bot:	oh yeah
Human:	wanna go eat lunch
Bot:	what
Human:	a baguette
Bot:	whats that
Human:	it is a food
Bot:	take a round all do you
Human:	it is a french food
Bot:	he has a little

Figure 10 English chatbot (Bergum & Iveland, 2019)

2.2 Hybrid Chatbot Review

2.2.1 Review 1

According to the (IrynaKulatska, 2019), chatbot are beings popular day by day. It has become the new technology to interact between human and the machine. Although the technology now can develop the human like taking and performing activities system current chatbot system has some obstruction i.e (emotional feelings while interacting between humans, lack of appreciation, delusion of the context e.t.c) Although the chatbot has this obstruction it is suitable for many company to handle day to day work.

Purposed system is the Hybrid Chatbot for the specific domain. Here the system is implement with two technology. First, retrieval based chatbot and second Generative based chatbot. Types of the chatbot has already been discussed in the above heading 0. This system is known as the Argue Bot. The develop system debate to the user and create responses to the query. This bot consists of two technics of developing the chatbot here the generative mode replies to the query if and only if retrieval based chatbot are unable to give responses to the input queries. Developing the hybrid chatbot has proven to be prevail (over) the confines of the datasets confines (IrynaKulatska, 2019).

2.2.1.1 First implementation with Basic Functionalities.

Purposed system is first implement with the Retrieval based chatbot technics. This implementation is implement with the dialog flow and flask. The data preparation process is differ according to the model architecture. Here the data is taken from the ArguAna Counterargs Corpus. Data consists of different topic debates. It hold 1068 debates with 6779 points and 6753 counterpoints. As the data hold voluminous subset of debates heading, the data is split between training, testing and the validation folders. As looking toward the structure of data it consists of both sympathetic and unsympathetic demeanor of the debates statement. Each statement consists of sequitur, conclusion and the premises without any labels and separation. Above mention data is crawled from idebate.com. Below figure shows the data structure for the first implementation (IrynaKulatska, 2019).

Topic	Debates	Points	Counterpoints
Culture	46	278	278
Digital freedoms	48	341	341
Economy	95	590	588
Education	58	382	381
Environment	36	215	215
Free speech debate	43	274	273
Health	57	334	333
International	196	1315	1307
Law	116	732	730
Philosophy	50	320	320
Politics	155	982	978
Religion	30	179	179
Science	41	271	269
Society	75	436	431
Sport	23	130	130
Training set	644	4083	4065
Validation set	211	1290	1287
Test set	214	1406	1401
Total	1069	6779	6753

Figure 11 First implementation data structure (IrynaKulatska, 2019)

As the dataset is in huge quantity with its sub debates heading. The first implementation is done with the use of the 12 debates marked as 'Junior'. The selected debates were chosen for the young audience. Six arguments are there in each debates which include one point and one counterpoint. The length of the sentences of the each point and counterpoint is 4-8 long. Below figure shows the debates architecture (IrynaKulatska, 2019).

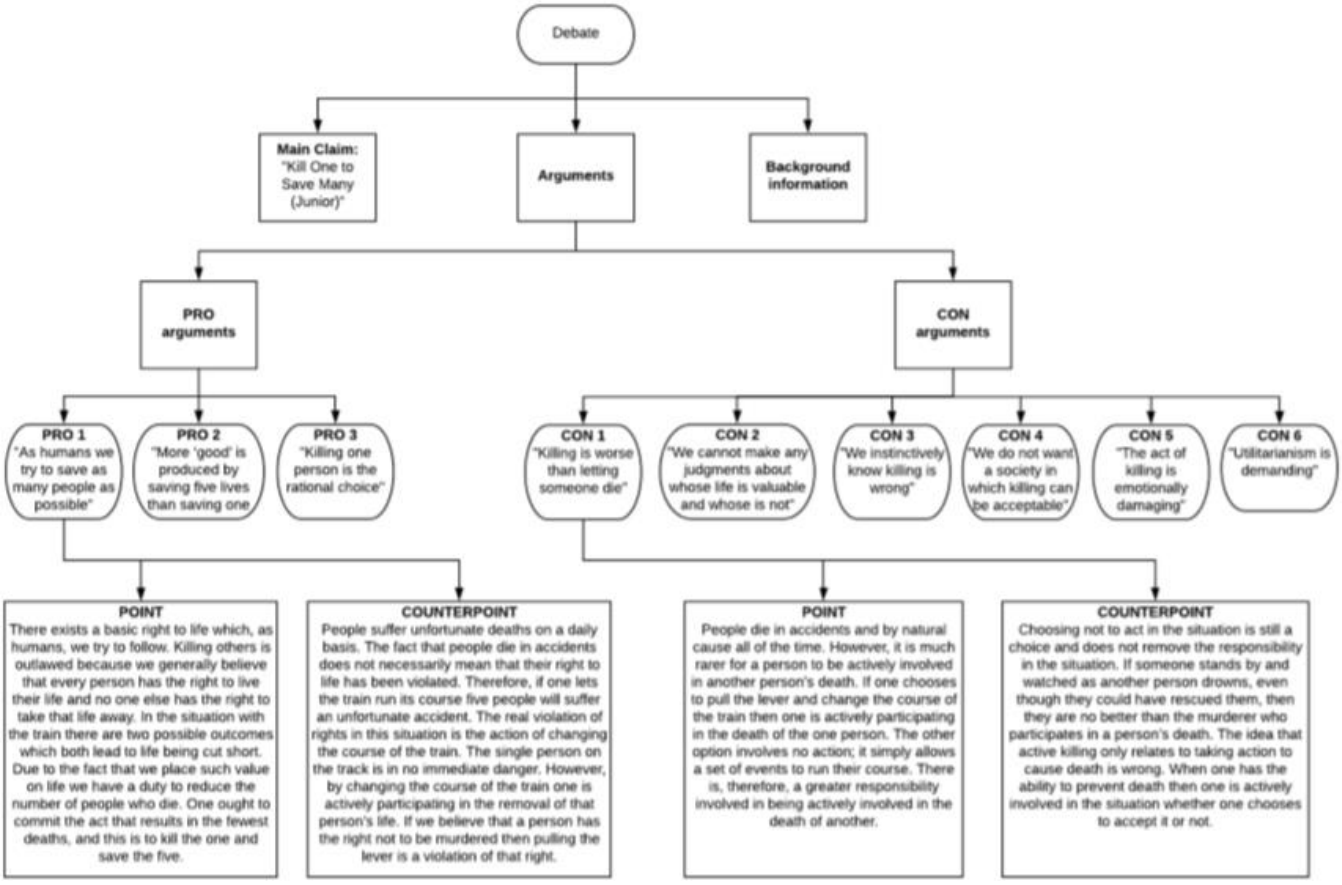


Figure 12 Debates dataset architecture (IrynaKulatska, 2019)

After the data structure is prepare for the model pre-processing is done to convert the raw data into numpy array format for the model input and store in the database. SQLite database is used to store the data. Flask and the Dialogflow is used to develop this model. Dialogflow is the cloud based google service which helps to develop the conversational agent and the interface for the web application, mobile application and can be deploy anywhere. Here the meaning of the user intake is extrapolate by the dialogflow. Flask is the voguish micro framework written in python language which has no database abstraction layer. Here the system is interact with the user from the flask. Database, dialogflow and the model is interconnect with the flask and model create the responses to the user. In this model, replies are given back to user and store in the database as the user chooses the debates topic then model discover mostly accustomed word from database and generates its object (IrynaKulatska, 2019). Below figure shows the architecture of the model.

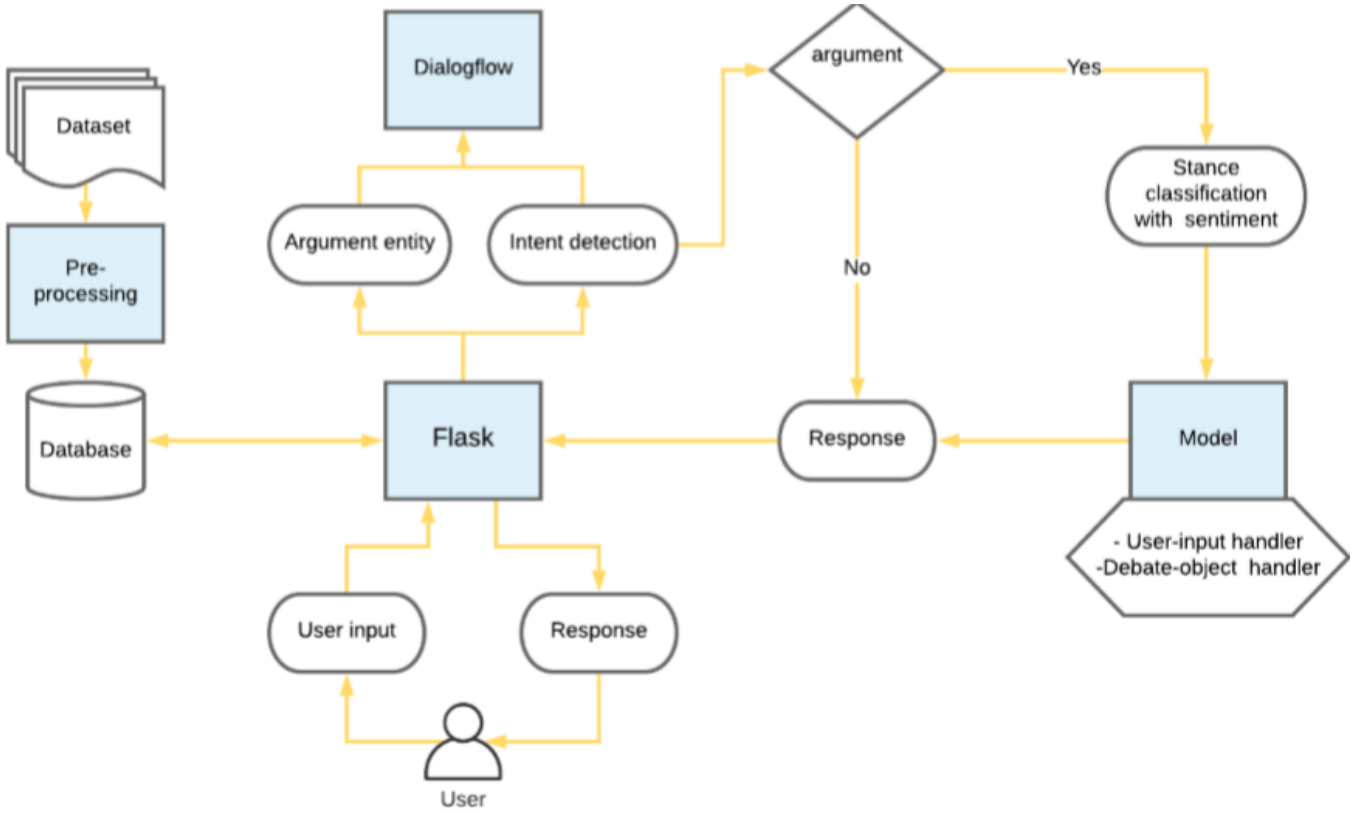


Figure 13 Architecture of the first implementation *(IrynaKulatska, 2019)*

After the model architecture is define and train the model in the prepare dataset. The evaluation and testing of the model is done. Below figure shows the evaluation of the first implementaton of the chatbot.

User Input	Intent	Response Model	Agent Output
Hello	Default Welcome Intent	response with a random argument retrieved from the database	Let's start the debate! I think that forced education achieves little. What do you think about it?
It's not so much forcing I think	Small talk Dialogflow agent	replies with customised response	Understood.
I think the brain is still too much in development early on.	argument intent	triggers argument intent, user-input analyser is then activated and determines that the polarity of the sentence is neutral and then retrieves first two sentences from the argument with the highest cosine similarity to the user input	Working at an early age can be an advantage in certain circumstances. Many families, particularly in countries with little welfare, need their children to bring income into the household.
I think the return on investment is solid. Children with more education can be more free in what they can do as an adult.	argument intent	Here the intent is classified as an argument and the polarity is computed as "positive" which is understood by the model as a "pro" stance	I agree! Working at an earlier age can help these families to survive. Furthermore, anyone who is having difficulties getting educational qualifications can gain an advantage by leaving school and gaining work experience.

Figure 14 Result of first implementation (IrynaKulatska, 2019)

2.2.1.2 Second Implementation with Machine learning

For the second Implementation, the dataset chosen is same as discussed in sub heading 2.2.1.1. The only difference is that the sub heading of debates is increase to 175 debates. Below figure shows the data structure for the second implementation (IrynaKulatska, 2019).

Topic	Debates	Points	Counterpoints
Culture	7	54	54
Digital freedoms	9	61	61
Economy	17	125	125
Education	10	76	76
Environment	5	36	36
Free speech debate	9	58	58
Health	10	77	77
International	30	233	233
Law	19	134	134
Philosophy	10	85	85
Politics	26	194	194
Religion	5	36	36
Science	8	57	57
Society	6	39	39
Sport	4	30	30
Total	175	1295	1295

Figure 15 Second Implementation data structure (IrynaKulatska, 2019)

All the process is same for the data preparation as the couple of the preprocessing steps are increase. As the dataset has the higher argument dousing of noise data is carried out. Notes, annotations, references and the footnotes were doused from the dataset. Dataset has been split to use it in different algorithm. Below figure shows the dataset separate for different algorithm (IrynaKulatska, 2019).

Method	Training	Validation	Testing
SAN	98.27	61.87	63.71
CNN	69.95	68.00	69.12
LSTM	68.65	68.20	68.81
LSTM SAM	97.64	61.48	62.26
RCNN	96.91	61.42	63.04
RNN	66.31	68.18	68.68
Data set	34686	7432	7432

Figure 16 Dataset used for the particular algorithm (IrynaKulatska, 2019)

For the model development and training it Google Collaboration (Colab) is used as it is the free Jupyter Notebook provided by google which has the facilities of using the GPU and the Tensor Processing Unit (TPU) of the google cloud. Popular Machine Learning (ML) algorithm such as Convolutional Neuron Network (CNN), Self-Attention Networks, LSTM, LSTM with self-attention mechanism, Recurrent Convolution Neuron Network (RCNN) and RNN algorithm are used to develop this chatbot for the second implementation. Stance classifier is used this in system which was develop using the ML technics. Below figure shows the model architecture (IrynaKulatska, 2019).

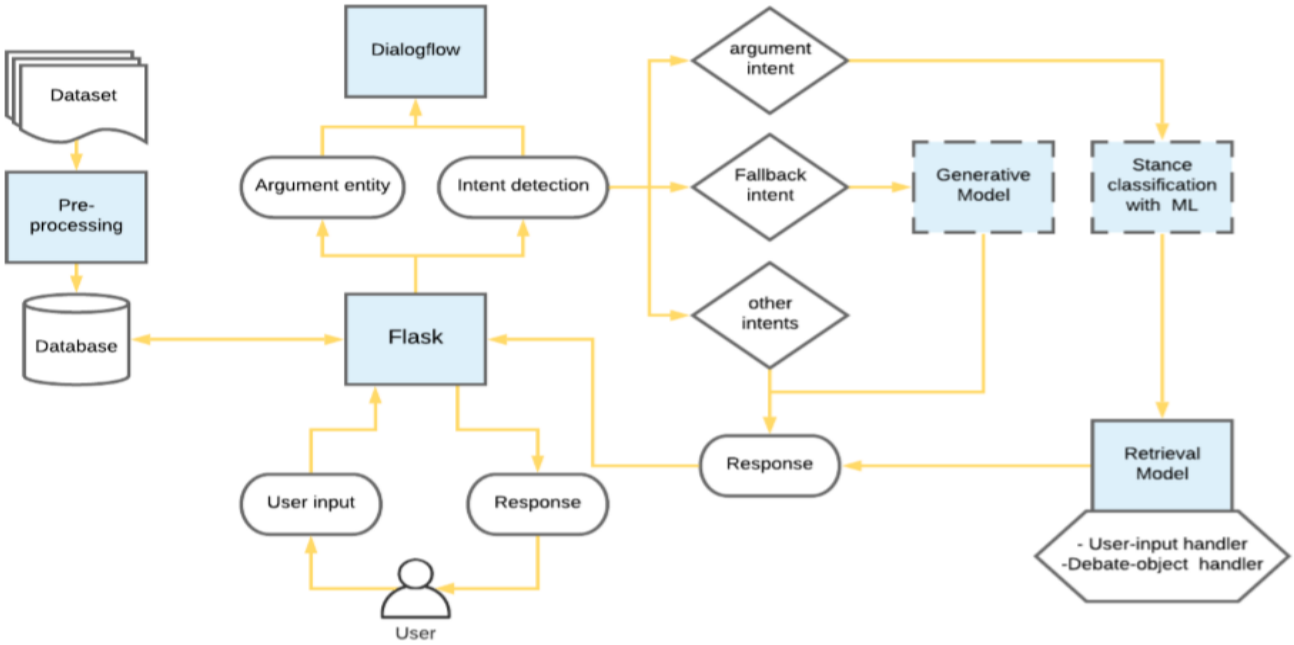


Figure 17 Architecture of second implementation (IrynaKulatska, 2019)

After the model is created and train, testing was done. Below figure shows the accuracy with the algorithm name after tuning the hyper parameter.

ML Model	Test Accuracy	F1-score	F2-score	Input (correct)
SAN	63.71	58.00	77.01	ppppcp (3 or 3)
CNN	69.12	07.12	15.39	nnnnnn (0)
LSTM	68.81	00.00	00.00	cccccc (3 or 2)
LSTM SAM	62.26	64.76	81.63	cpcpcp (3 or 3)
RCNN	63.04	55.01	74.80	ppcccc (4 or 2)
RNN	68.68	04.04	09.18	cpcppp (2 or 4)
Data set	34686	7432	7432	

Figure 18 Accuracy according to the Algorithm (IrynaKulatska, 2019)

2.2.2 Review 2:

According to the (Chisty, et al., 2018) chatbot has played the important role to communicate with the human and machine. Due to the technology, some of the chatbot can antiphon alike human beings. Purpose system is the hybrid chatbot known as (EME) which is the protracted version of the current hybrid model. Purpose system can generate the automatic reply and learned from the experiences. Lots of chatbot are implement and deploy in the internet for various purpose such as guidance, education, customer support, entertainment e.t.c Purpose system can develop conversational agent synthetically. As this chatbot is already implement by the Alibaba Group in AliMe chatbot. This chatbot implement by the Alibaba group has complicated database system and the configuration. Mention system generate responses with hook up of generative and the IR model and the seq2seq model which joins the responses. IR model is used to store the chatbot details in the database. Database is also used to store the training and testing data. This is the open domain chatbot where human can communicate in various subject. The main aim of purpose system is to remove the context problem from the root of the model. This system can also be known as the extended version of the Alime hybrid model.

Many data science library can be found to develop the chatbot but without using the some of the Application Performance Index (API), traditional coding system is used to attach the user interface and the backend coding.

EME Chatbot is implement with the Artificial Intelligence Markup Language (ATML) with the help of the conversational agent which is the retrieval based chatbot. The response is generate by collaboration with the seq2seq model, IR model, AIML using sequence re-ranking model. This system include two chatbot models.

2.2.2.1 Retrieval Based Chatbot

To prepare this model, data is collect from the giant conversational corpora from the online talking stages. This data set has the pair of the question and the answers meant as hq*,r* I. Lucene 3 fueled framework is used for the recovery usages of the dataset. Considering the pre-created dataset, the recuperation strategy can be performed using the state of-the-practice information recuperation system. When a request q is given, watchwords expelled from q and their certainty esteems are point by point as the recuperation example and feed into the recuperation system to glance through the most significant $q*$ in database. Then the r* and the q* rebounded as yield. This lead to abnormal integrate between question q and the recovered answer r*. Here the traditional method for the data recovery process is used where the recovery frameworks give exuberance of one query and score them according to linguistic integrate degree.

2.2.2.2 Generative Based Chatbot

Generative chatbot of EME is develop using the seq2seq algorithm of the RNN. Seq2seq consist of two neuron network encoder and the decoder which has already discussed in above review. Here the result of this model is the log-probability of the generate answer r+ input inquiry q. All the responses generate from this model has the large rankings due to the reasonably over probabilities value. Here GRU is selected as the RNN unit. To overcome the issue of various length sentences Tensorflow 1 API container instrument is used. This process include the (5,5),(5,10),(10,15),(20,30),(45,60) cans to pad the sentences. For example, the input query is of length 3 and the out-take is of length 4 then the first cans (5,5) is taken and the necessary padding is done in the sentences. Softmax is used to hasten the preparation form.

Seq2seq ReRank model

ReRank model is used to bidder the responses with respect to the input query. Here the mean probability is used to find the bet responses for the particular input query. Mean probability seems to be more accurate than trying to inverse the average of the cross-entropy and the harmonic mean. Below figure shows the mean probability mathematical equation used in rerank model.

$$S_{Mean\text{-}Prob} = 1 \; n \; Xn \; i=1 \; p(yi = wi|\_i)$$

Figure 19 Mean probability mathematical equation

Post Reranked

This is the important model in this system which decided the best responses by subtracting q to the r. Retrieval based chatbot response r and the generative chatbot response r+. Below figure shows the flowchart of the purpose system.

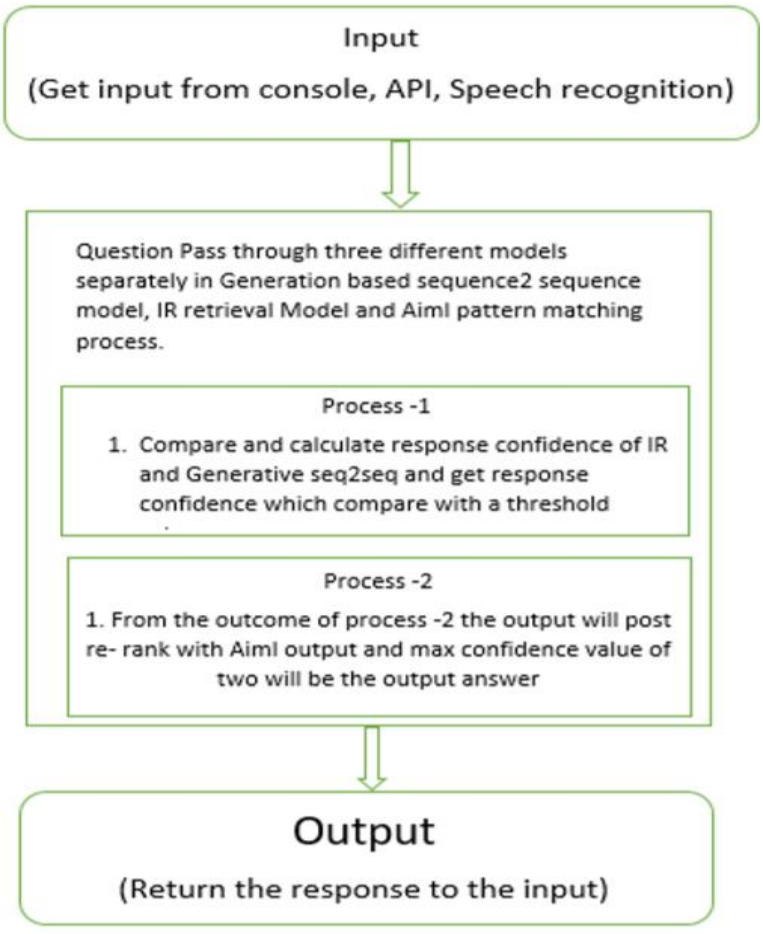


Figure 20 Flowchart of the purpose system.

After the system is develop evaluation is done to check develop system behavior the data. Here the statistical analysis is done to become scientific to take the necessary decision. This analysis collect, explore and present the large amount of data to determine elemental arrangement and trends.

The first analysis is done using the randomly sampled 500 questions dataset on two seq2seq models in three scoring benchmark i.e. mean probability, inverse of the averages cross-entropy and the harmonic mean. The analysis is done is rerank model. Below figure shoes the result of the analysis.

	IR+Reank(Hybrid Model)	IR	Extended Hybrid Model(Expected)
Basic	0.48		
Attentive	0.54	0.47	0.58

Figure 21 Comparison of rerank model

Similarly, analysis is done with another 600 questions to check the efficiency of the IR, Generation, IR +Rerank, IR + Rerank + Generation. Model has the confidence score threshold T of the 0.19 which helps to generate the response to the appropriate query. Any question greater than the T are response by rerank and other are managed by generative. Below figure shows the accuracy of the hybrid model.

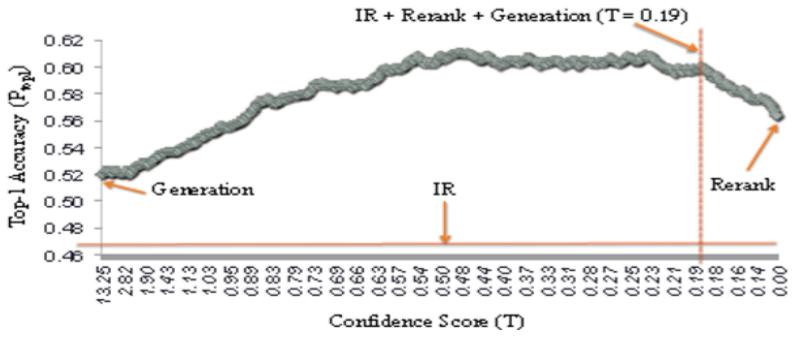


Figure 22 Hybrid model accuracy

2.3 Analysis of the Review

According to the above Initial Research, there are similar types of the system with the different domains. Pipeline to develop the chatbot is similar but steps may differ according to the model architecture. The basic pipeline to develop chatbot system are:

- Creating the dataset
- Pre-Processing
    - Cleaning
    - Tokenization
    - Vectorization
    - Dimension reduction
- Train and Test Split of the dataset
- Creating the LSTM layers (Creating the model architecture.)
- Train the model
- Save the model
- Visualizing the performance of the model
- Create the inference model
- Testing the model

In the above mentioned review, system 2, includes all the steps as mentioned above but model configuration and the architecture is different. From the above mentioned review it is clear that performance of the chatbot does not only depend on accuracy of the model but also data taken to train the model and preprocessing steps included in it. Taking the raw data and converting it to the model training dataset is a challenging task. This kind of

chatbot needs a huge amount of data and computational power to become adequate. RNN is usually used to develop the generative and hybrid chatbot as it performs well for catching the sequential information. There are different algorithms in RNN which have proof, it is superior for developing conversational agents. Mainly Seq2Seq, Seq2Seq with attention mechanism is used for the conversational agent. Seq2Seq consists of two LSTM neural nets. i.e. encoder and the decoder LSTM, encoder takes the sequential information in vector or the tensor form which is converted to fixed length vector or the tensor. Similarly, the decoder takes the input of encoder output and converts it to a fixed sequence of information. Generative based chatbots are capable of generating the new sequence of information, analyzing the input query. Attention mechanism is a popular and used algorithm in RNN which gives importance to the certain word in the sequence to generate responses.

Comparison between the reviews.

	Generative based chatbot		Hybrid based chatbot	
Basic	Review 1	Review 2	Review 1	Review 2
Data	Taken from two source : First, Multi-domain dataset developed from the raw scripts of 600 movies. Second, single domain dataset which encompasses of the chat logs from the Ubuntu technical support channels	Norwegian language, data was retrieve from the personal messaging online platforms such as Facebook and Facebook messenger. English language, data was taken from the Cornell Move Dialogs Corpus.	ArguAna Counterargs Corpus.	Randomly Question Data
Processing Unit	50 Giga Byte (GB) of two GPUs.	Central Processing Unit (CPU)	GPU & TPU of the google cloud	GPU
Optimizer	RMSprop stochastic gradient descent	RMS prop	RMs prop and Adam	RMs prop and Adam
Architecture	RNN sequence to sequence Bidirectional LSTM	Seq2seq model implemented with attention mechanism.	Hybrid Model	Hybrid Model
Accuracy	Cornell movie corpus: 93.64/4.22	Evaluation:	Training: 34686 Test: 7432	IR + Rerank : 0.54

	Ubuntu chat corpus: 85.16/7.99	Norwegian, average 3 and English, average 2.5	Validation: 7432	Extended Hybrid: 0.58

Table 1 comparison of the review

3 Supportive Information

3.1 Framework

3.1.1 Django

Django is a significant level Python web framework that empowers fast improvement of secure and viable sites. Worked by experienced designers, Django deals with a significant part of the issue of web advancement, so users can concentrate on composing applications without expecting to waste time (MDN contributors, 2019).

3.2 Neural Network

A neural system is a progression of algorithms that tries to perceive basic connections in a lot of information through a procedure that impersonates the manner in which the human mind works. It allude to frameworks of neurons either natural or counterfeit in nature (Chen, 2019).

3.2.1 RNN

Handling the sequence of data is more complex than other types of data. The problem where the input data is in the form of sequence, processing it and generating the responses is a sequence prediction problem. There are lots of sequence problems some of them are:

- Sequence Generation
- Seq2Seq Prediction

- Sequence Prediction
- Sequence Classification

Other algorithms such as CNN, ML are lacking to solve sequence problems. Therefore to handle this problem RNN is developed. It has the recurrent nature, present input depends upon the past output of the recurrent. The output is then passed to the same network over the time to get output. It has an internal memory which holds the information of the sequence data. In this algorithm all the internal state and the inputs are reliant to each other while in others network inputs are nonpartisan. Below figure shows the RNN.

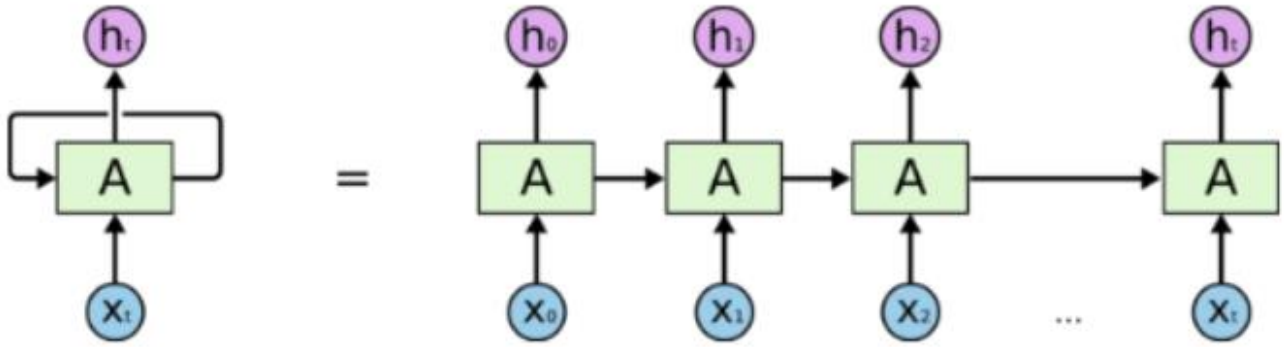


Figure 23 Recurrent Neuron Network

Mathematically,

First, it takes the X(0) from the arrangement of information and afterward it yields h(0) which together with X(1) is the contribution for the subsequent stage. Along these lines, the h(0) and X(1) is the contribution for the subsequent stage. So, h(1) from the following is the contribution with X(2) for the subsequent stage, etc. Therefore, it continues recollecting the unique situation while preparing.

The formula for calculating the internal state is:

$$h_t = f(h_{t-1}, x_t)$$

Figure 24 RNN Current State

Using the activation function:

$$h_t = \tanh (W_{hh}h_{t-1} + W_{xh}x_t)$$

Figure 25 RNN with activation function

Output

$$y_t = W_{hy}h_t$$

Figure 26 Output

Here,

X = input

h =hidden vector

X_0 ....X_t = input at time step

h_0 ….ht = output at each time step

W = weight of the network

W_{hh} = earlier hidden state at each time step

W_{xh} = current state weight

Tanh = activation function

Y_t = output

W_{hy} = weight of output state

Though, it is ancestors of all the sequence problem it has some defect,

- Exploding and the vanishing gradients.
- Suitable for short sequence.
- Preparing RNN is a troublesome errand.

3.2.2 LSTM

LSTM is developed to solve the issue of exploding and vanishing gradients. Let us understand this concept with the example, consider a language model attempting to foresee the following word dependent on past ones. In the event that they are attempting to foresee the final say regarding "I am writing the report." In this case, the next word is surely going to be reported. This kind of case can easily be solved by the RNN but if sequence become longer, the output not only depends on corresponding value but in value at the middle of the sequence where the hole between important data and the spot that required is big there RNN lack to predict this sequences and lead to vanishing gradient. Therefore to solve this issue LSTM is developed which is cognition of remembering and learning long-term dependencies. LSTM is the modified version of RNN, to remember information for significant stretches of time. In the LSTM, there are two activation functions, sigmoid and tanh. There are three different gates in the LSTM i.e. Forget Gate, Input Gate, Output Gate. It includes a cell state which is similar to a transport line. It runs straight down the whole chain, with just some minor direct communications. It is extremely simple for data to simply stream along it unaltered (Mittal, 2019).

Below figure shows the LSTM.

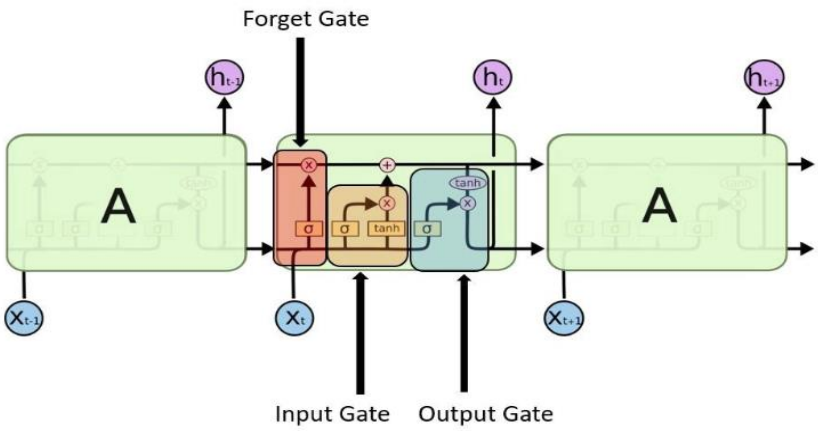


Figure 27 LSTM (Mittal, 2019)

Mathematically,

The initial phase in LSTM is to choose what data is going to discard from the cell state. This choice is made by a sigmoid layer. It takes a gander at h_{t-1} and x_t, and yields a number somewhere in the range of 0 and 1 for each number in the phone state C_{t-1}. 1 represents "totally keep this" while a 0 represents "totally dispose of this." (Mittal, 2019)

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

Figure 28 Forget gate mathematics

Now the next step is what new data are going to store in the cell state. This has two sections. Initially, a sigmoid layer called the "input door layer" chooses which esteems are updated. Below figure shows the formula of the sigmoid layer. (Mittal, 2019)

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

Figure 29 Sigmoid layer of input gate (Mittal, 2019)

Next, a tanh layer makes a vector of new competitor esteems, C~t, that could be added to the state. Below figure shows the formula of the tanh layer (Mittal, 2019).

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

Figure 30 tanh layer of the input gate (Mittal, 2019)

Above two formula work together to perform the input gate function.

Now it is time to refresh the old cell state, C_{t-1}, into the new cell state C_t. Duplication of the old state by f_t is done to overlook old value then $i_t * \tilde{C}_t$ is added which is the new competitor esteems, scaled by the amount chosen to refresh each state esteem. Below figure shows the formula of the updating cell state (Mittal, 2019).

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

Figure 31 Updating the cell sate. (Mittal, 2019)

The yield will be founded on our cell state, yet will be a sifted rendition. First, a sigmoid layer is run which chooses what parts of the cell state is going to yield. Below figure shows the formula of the sigmoid cell state (Mittal, 2019).

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

Figure 32 Sigmoid cell state (Mittal, 2019)

At that point, the cell state passes through the tanh layer and duplicates it by the yield of the sigmoid gate which provides us our output. Below figure shows the formula for tanh cell state

$$h_t = o_t * \tanh \left(C_t \right)$$

Figure 33 output of the LSTM (Mittal, 2019)

Here,

b = bias of the network

o_t = Output at time step

i_t = internal state

c_t = cell state at time step

3.2.3 Encoder-Decoder

It is the modified version of LSTM which consists of two LSTM cells. One for encoding and another for decoding the sequences. Encoder encodes the sequence to a fixed length vector and decoder decodes the fixed length vector to the sequence of information. This algorithm has proved to be good for the seq2seq predictions. This algorithm is developed to exhibit the state of art act mainly to solve the NLP problems. These algorithms seem to operate in many prediction problems. Such as Machine Transaction, Learning to Execute, Image Captioning, Conversational Modeling and movement Classification. Below figure shows the architecture of the encoder-decoder (Brownlee, 2017).

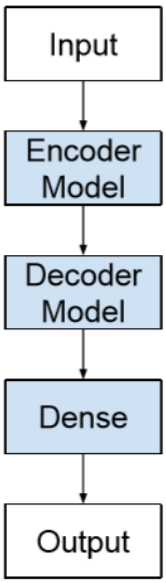


Figure 34 Encoder and Decoder architecture (Brownlee, 2017)

Purpose system is the Conversational Modelling (CM) system where the sequence of text is generated from the given sequence data.

3.2.4 Libraries

Collection of the module, packages, classes which can be used in our project to develop the fully functional system. API of this library is used to import classes of the module.

3.2.4.1 Numpy

It is the mathematical library of the python which helps to handle numeric operation as well as it furnish high performance multidimensional array object and appliance to work with them.

3.2.4.2 Pandas

It is popular library to manipulate, read and save the data. It also has visualization built in function to analyst the data.

3.2.4.3 Matplotlib

It is the image visualization library which helps to plot numpy array as well as Python Data Structure. It includes all kind of visualization technics (Soloman, 2020).

3.2.4.4 TensorFlow

It is the open source popular AI library which consists of many module for developing the model. It is the core top level library. Many of the model is based on TensorFlow backend. It also support Graphical Processing Unit (GPU).

3.2.4.5 Keras

Keras is also open source mostly used ML library run in top of TensorFlow, Microsoft Cognitive Toolkit, R, Theano and PlaidML. It is flexible, simple, modular, and friendly than the TensorFlow.

3.2.4.6 Gensim

Open source library mostly used for NLP projects, unsupervised topic modelling is Gensim. It is develop to manipulate the text data to numeric data. It include popular algorithm such as Word2Vec, Doc2Vec used for conversational, sentimental analysis projects.

3.2.4.7 Pillow

Pillow is the python library used for images related things. It is the imaging library.

4 Main Body

4.1 Process Model

Road map which aims to create a high-quality, timely, effective system is the Process Model. Set of linked enterprises which helps to evolve the software is also the process model (Pressman, 2014). Activities of the process model are:

- Communication
- Planning
- Modeling
- Construction and
- Deployment

Communication and Planning comes under the Understanding the Problem or the domain. Before starting any of the projects, communication between the domain specialist and the project manager is a vital step. Without compassionating the problem and the domain led to failure of the project. Proper communication is crucial for the development of high quality software (Pressman, 2014).

Modeling and construction comes under the plan for a solution for a specific domain. After gathering information about the system domain, designing and coding is done to implement solutions to given problems. This step carries lots of time and hard work. This step raises lots of queries such as whether the developed system can be effective for the given problem or not? Is this the potential solution? (Pressman, 2014)

Carrying out the plan is like deploying the system and testing it. This process consists of implementation of the system. These steps also get to the different queries like if the develop system can handle the solution of the plan? Is the development system legitimate for the purpose of this problem? (Pressman, 2014)

4.1.1 Types of Process Model

Several Models are there for the software development but below mention are some of the popular and frequently used models.

4.1.1.1 The Waterfall model

This is the traditional and old model which is still used for predefined results of projects. It is elementary to fathom and implement. One phase of the development process is dependent on another phase which makes this model linear-sequential life cycle. This model does not have the returning point as started project should goes on without intrusion (Pressman, 2014). This model is adequate if:

- Prerequisites are all around reported, clear and fixed.
- There are no equivocal necessities.
- Sufficient assets with required aptitude are accessible to help the project.
- Innovation is comprehended and not dynamic.

Below figure illustrate the waterfall model.

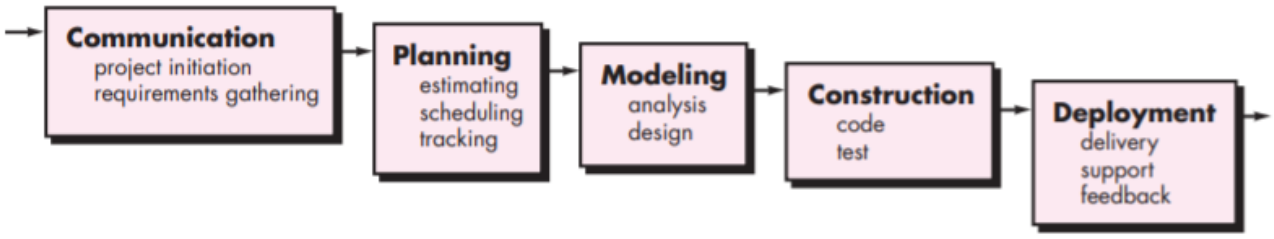


Figure 35 The waterfall model (Pressman, 2014)

4.1.1.2 Iterative and Incremental model

Process of developing the system in small chunks, and enumerating the features in each incremental phase is an Iterative and Incremental model. In an Iterative Incremental model, at first, a fractional execution of a complete framework is developed with the goal that it will be in a deliverable state. Expanded usefulness is included. Imperfections, assuming any, from the earlier conveyance are fixed and the working item is conveyed. The procedure is rehashed until the whole item advancement is finished. The reiterations of these procedures are called Iterative. Toward the finish of each iteration, an item addition is increment (Pressman, 2014). Below figure illustrate the Iterative and Incremental model.

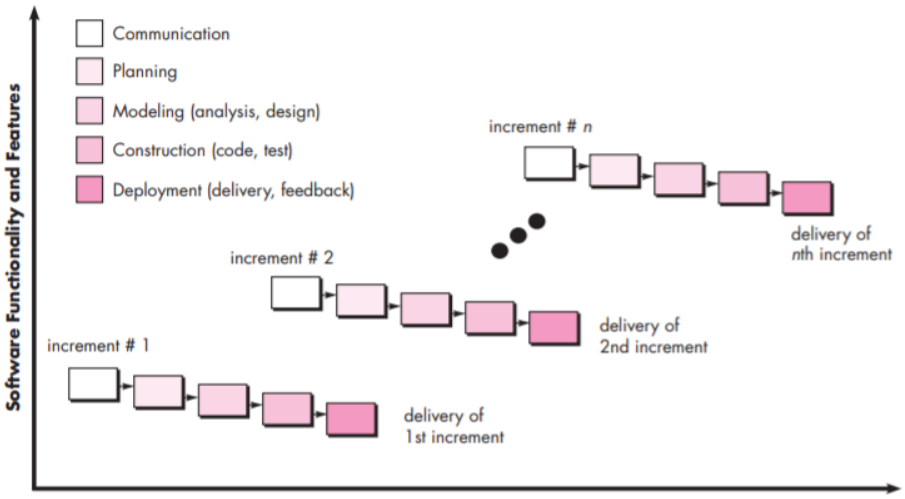


Figure 36 Iterative and Incremental model (Pressman, 2014)

In Iterative and Incremental model, there are different sub development process such as Prototyping, Agile Development Model, Rational Unified Process, Rapid Application Development (RAD) (Pressman, 2014).

This model is appropriate if:

- The vast majority of the prerequisites are known in advance however are relied upon to advance after some time.
- A project has extensive advancement plans.
- The domain is new to the group.

4.1.1.3 The Spiral Model

Combo of the Iterative and the waterfall model is the spiral model where each phase starts with a structure objective and closes with the customer exploring the advancement. It is a transformative programming process model that couples the iterative idea of prototyping with the controlled and methodical parts of the cascade model. It gives the potential for fast advancement of progressively increasingly complete renditions of programming (Pressman, 2014). Below figure illustrate the spiral model.

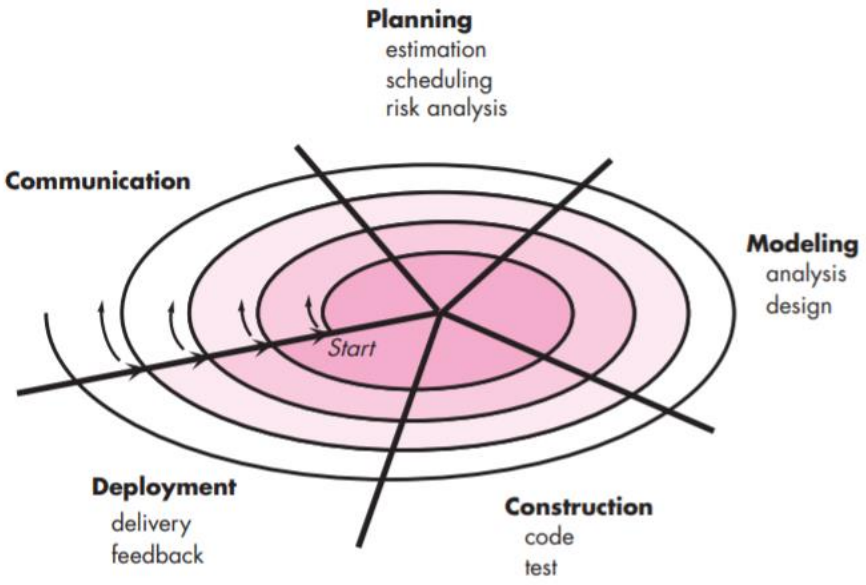


Figure 37 spiral model (Pressman, 2014)

The model seems to be effective when:

- Project is huge.
- Discharges are required to be visit.
- Making of a model is pertinent.
- Prerequisites are hazy and complex.

4.2 Software Development Process Model

Waterfall model is used to develop the web application of purpose system. Detail description of this model is listed in 4.1.1.1.

Communication

It is the initial stage for software development. Requirements are gathered by visiting the online department of a company and talking with the representatives. The requirement consists of the chatbot which can solve the minor issues and can easily deploy in the web application. For the reference WorldLink official site is taken. Software Requirement

Specification is (SRS) developed from the gathered information. SRS can be found appendix 9.2

Planning

All the requirement are gather and detail analysis is done. Official site is explore and initial Gantt chart is developed.

Modelling

Here, the wireframe, diagram is design, explore article in medium and videos in YouTube for the development.

4.2.1 Wireframe

It includes the initial design of software Graphical User Interface (GUI). On the basis of the wireframe actual design is code. Balsamiq Wireframes are used for the initial design. It is a GUI site wireframe manufacturer application. It permits the originator to orchestrate pre-assembled gadgets utilizing a simplified WYSIWYG supervisor.

Below is the home page of the application.

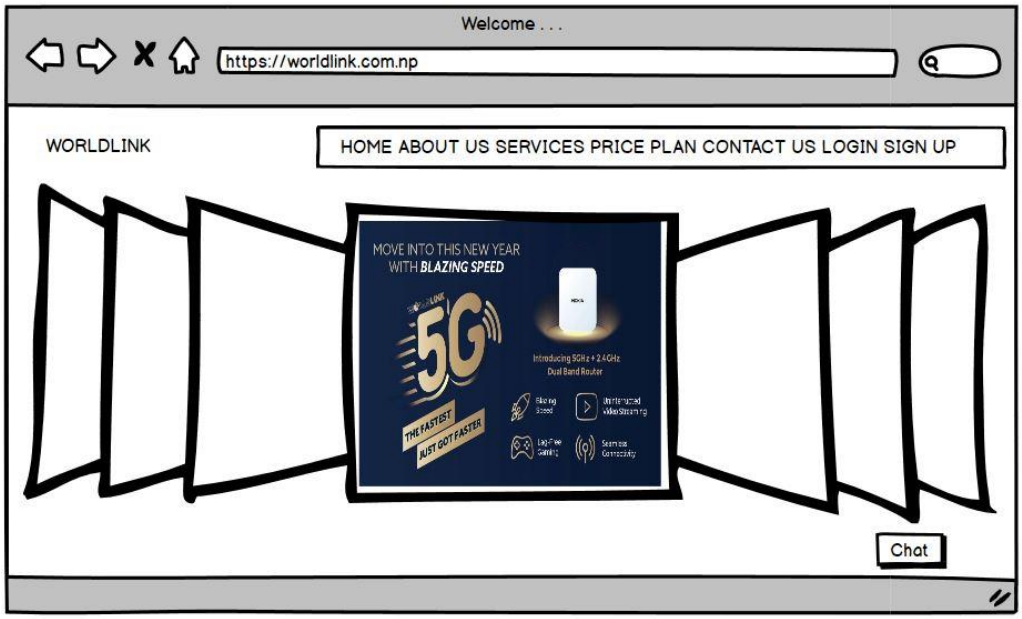


Figure 38 home page

Below is the about us page of application.

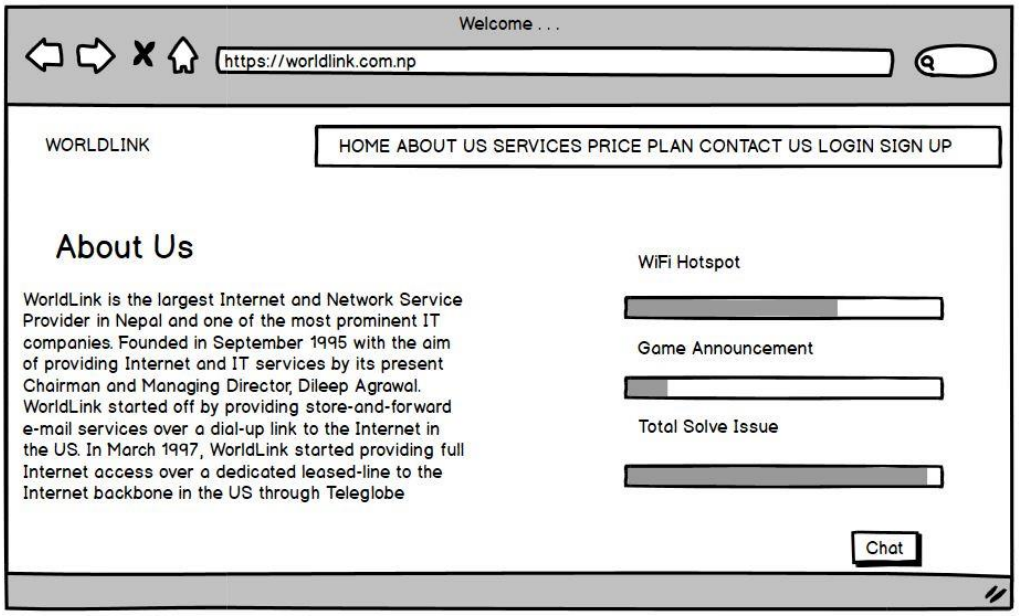


Figure 39 about us page

Below is the services page of application.

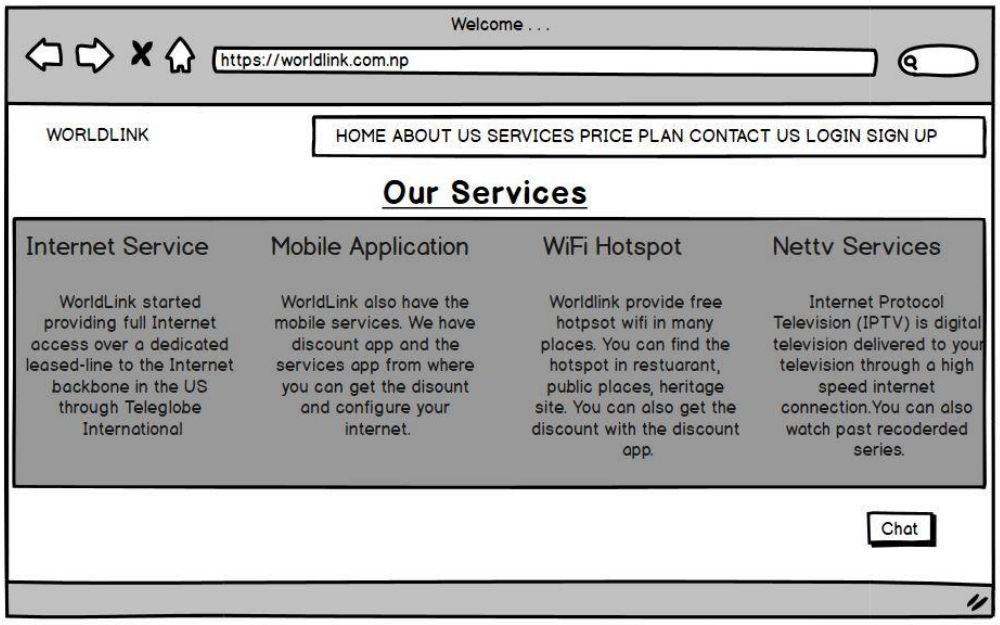


Figure 40 services page

Below is the price page of the application.

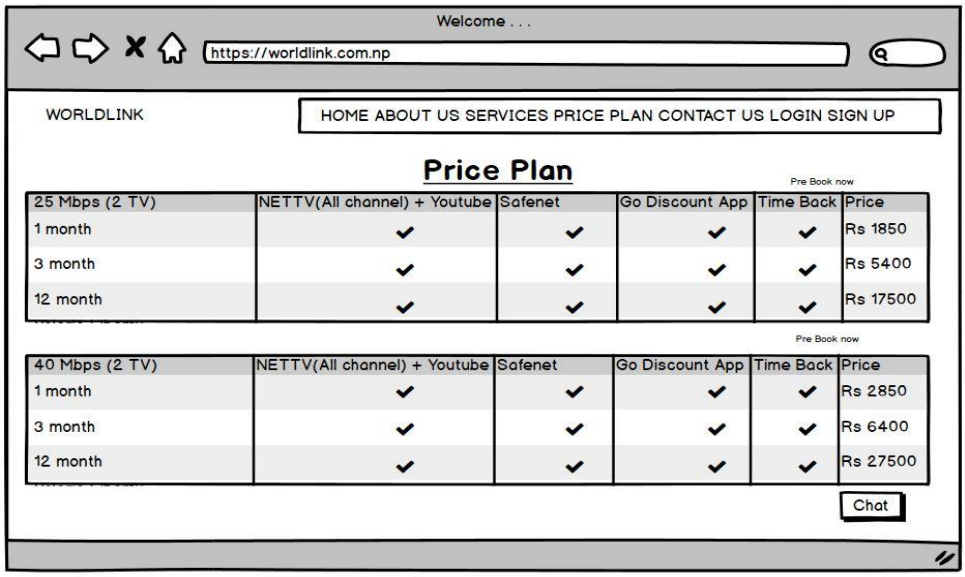


Figure 41 price page

Below is the contact page of application.

Figure 42 Contact page

Below is the register page of application.

Figure 43 register page

Below is the login page of application.

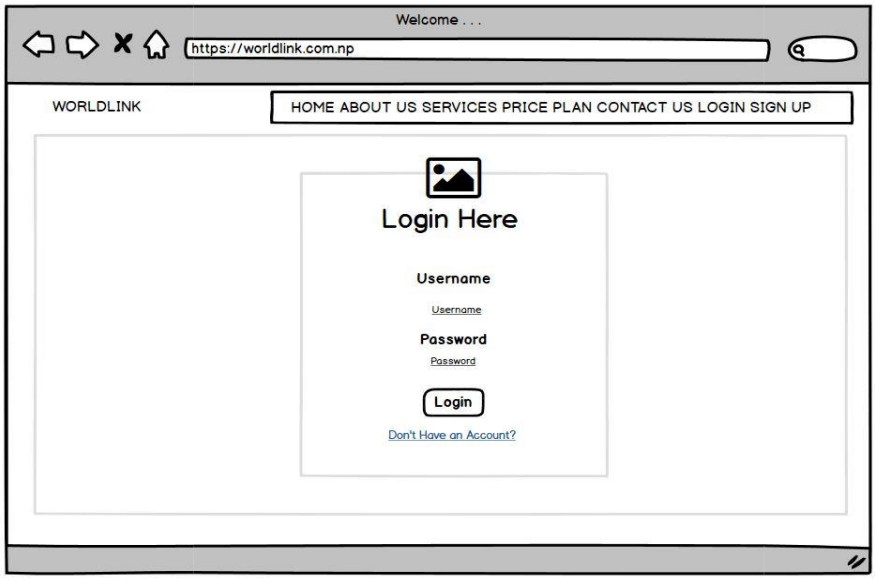


Figure 44 login page

Below is the chat authentication page.

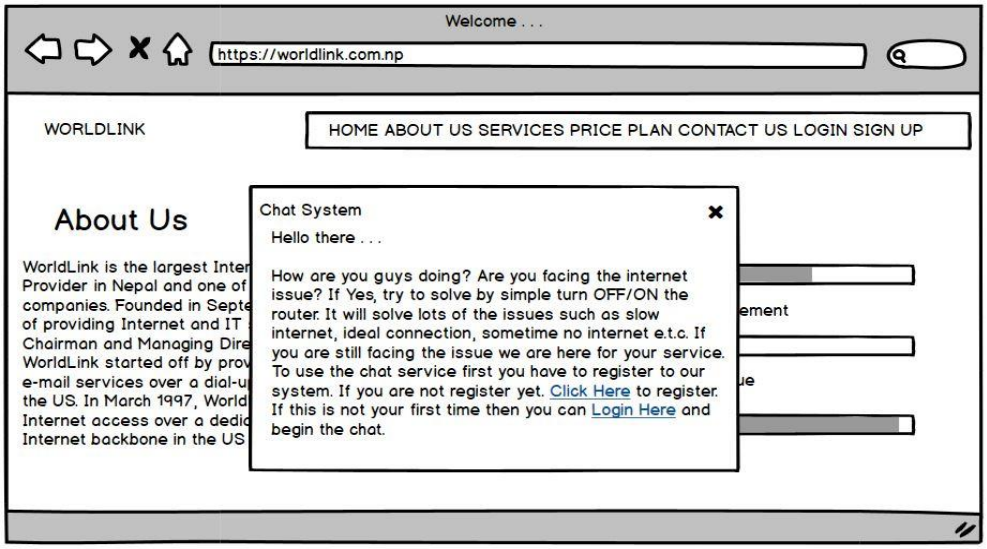


Figure 45 chat authentication page

Below is the chat GUI design.

Figure 46 chat GUI page

4.2.2 Diagram

Functional Decomposition Diagram (FDD)

The individual components of the procedure and their various leveled relationship to one another are regularly shown in a chart called FDD. The chart appears in a top-down organization showing a procedure. It includes a rectangle for the function name and arrow to connect them. In the below figure there are two components i.e. web application and hybrid chatbot.

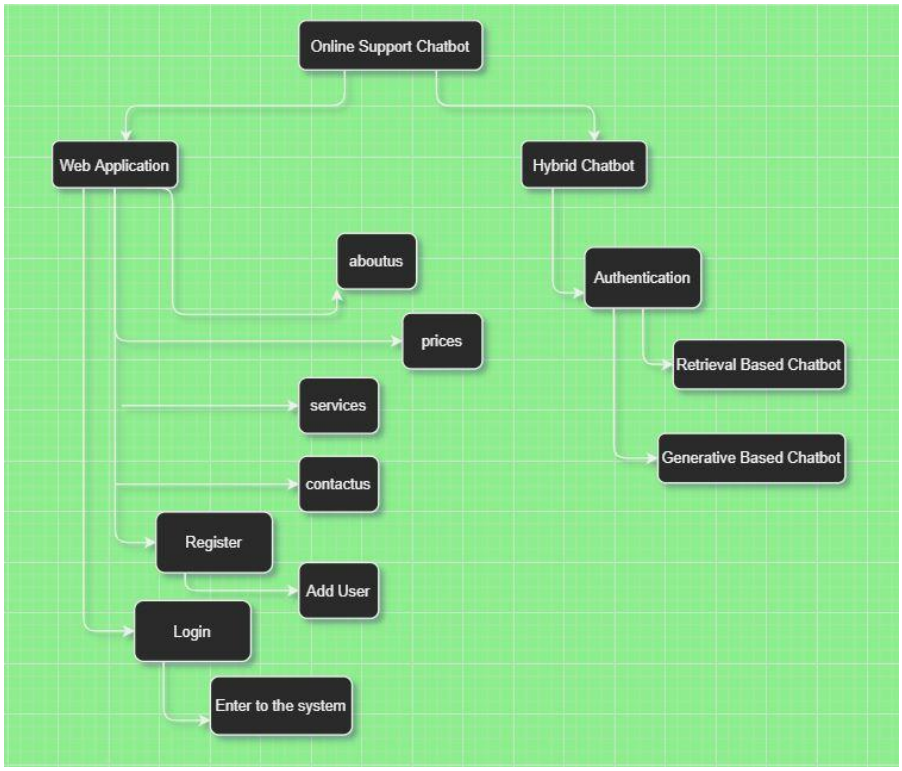


Figure 47 FDD

Use case diagram

Use case diagrams are utilized to assemble the prerequisites of a framework including interior and outside impacts. When a framework is broken down to assemble its functionalities, use cases are readied and on-screen characters are recognized. In the diagram below there are two users i.e customer and the admin. Customers can register, login, use the chatbot and browse to

different pages whereas admin can login, browse to pages, use chatbot and provide roles and permission.

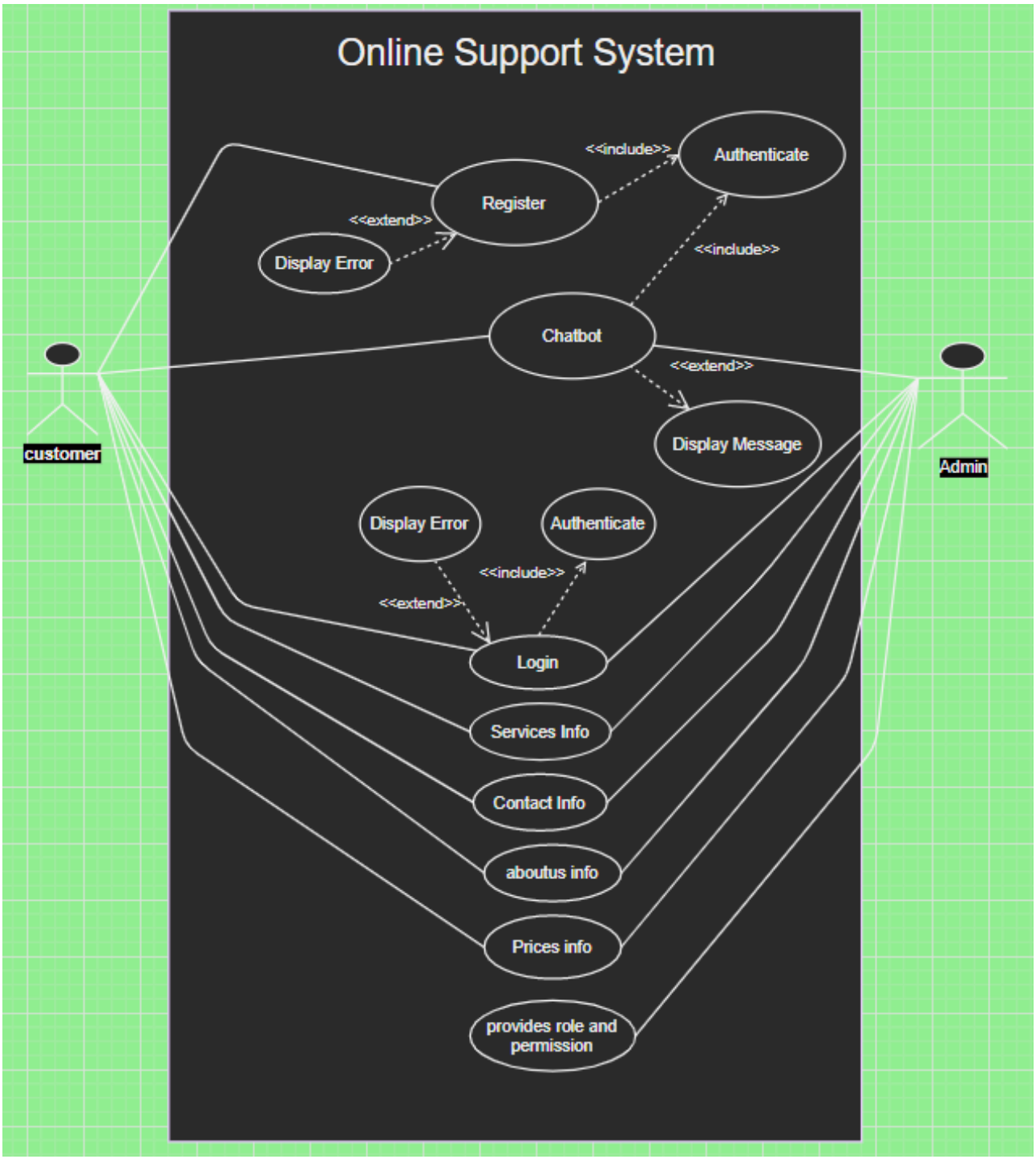


Figure 48 Usecase diagram

Activity Diagram

An activity diagram is utilized to demonstrate the work process portraying conditions, limitations, consecutive and simultaneous activities. It shows the progression of control in a framework and alludes to the means engaged with the execution of use case. It is also known as behavior diagram (geeksforgeeks.org, 2020). There are different activity diagram.

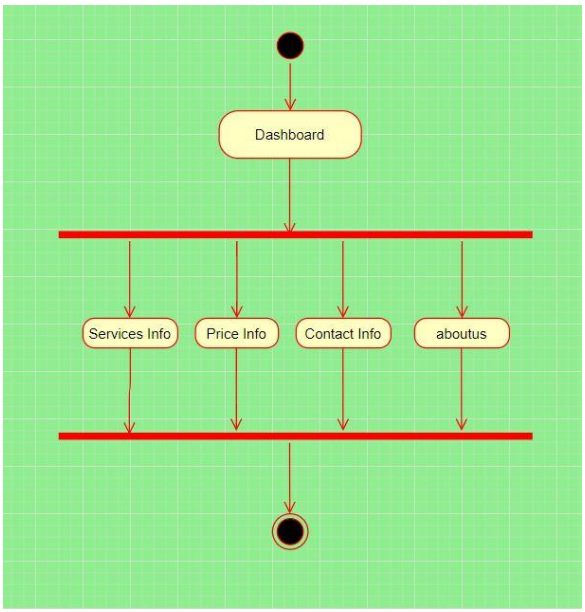


Figure 49 dashboard activity diagram

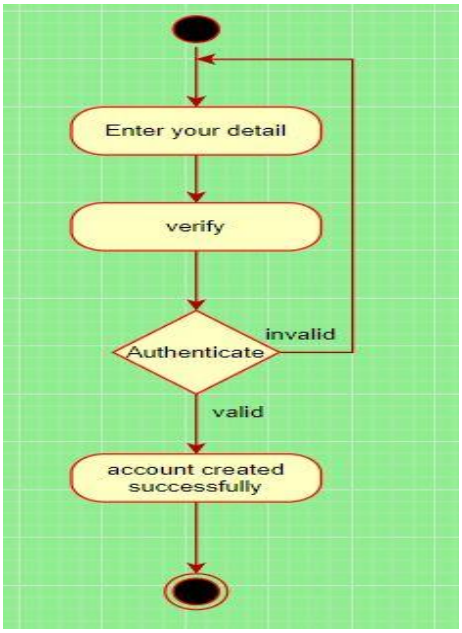


Figure 50 register activity diagram

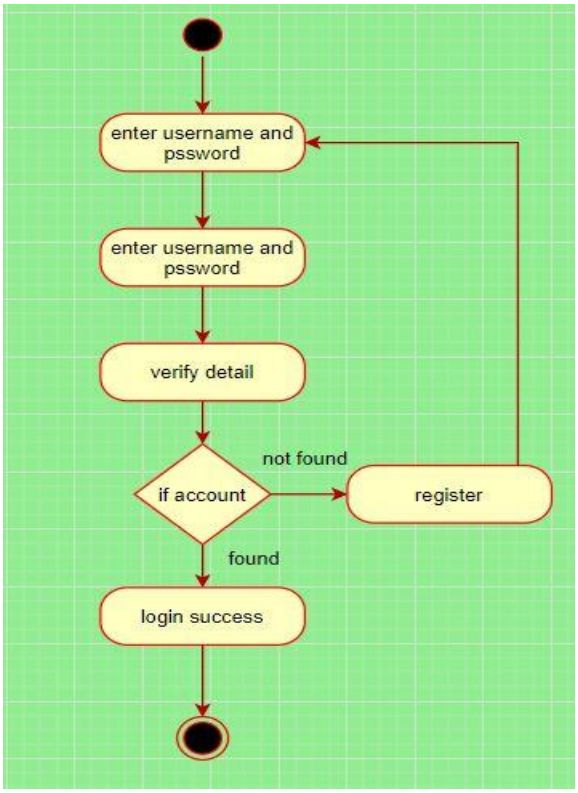


Figure 51 login activity diagram

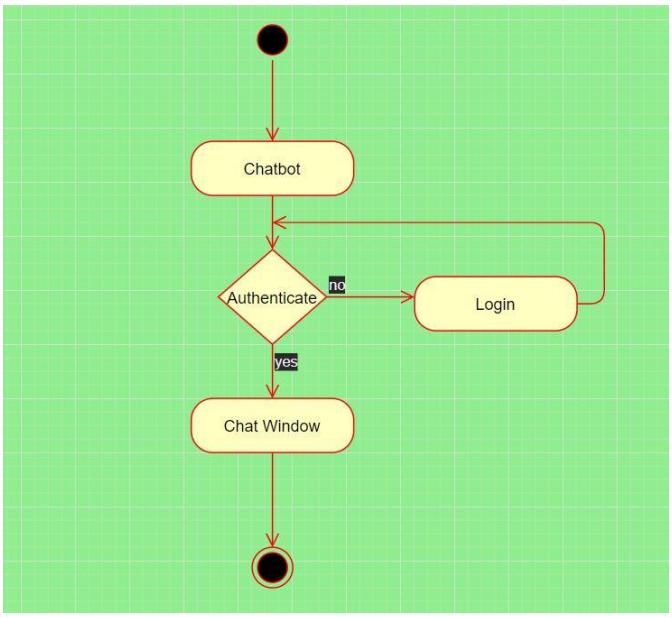


Figure 52 chatbot activity

Sequence Diagram

Sequence Diagrams are association charts that detail how tasks are done. They catch the association between objects with regards to a coordinated effort. They are time center and they show the request for the communication outwardly by utilizing the vertical hub of the outline to speak to time what messages are sent and when (geeksforgeeks.org, 2020).

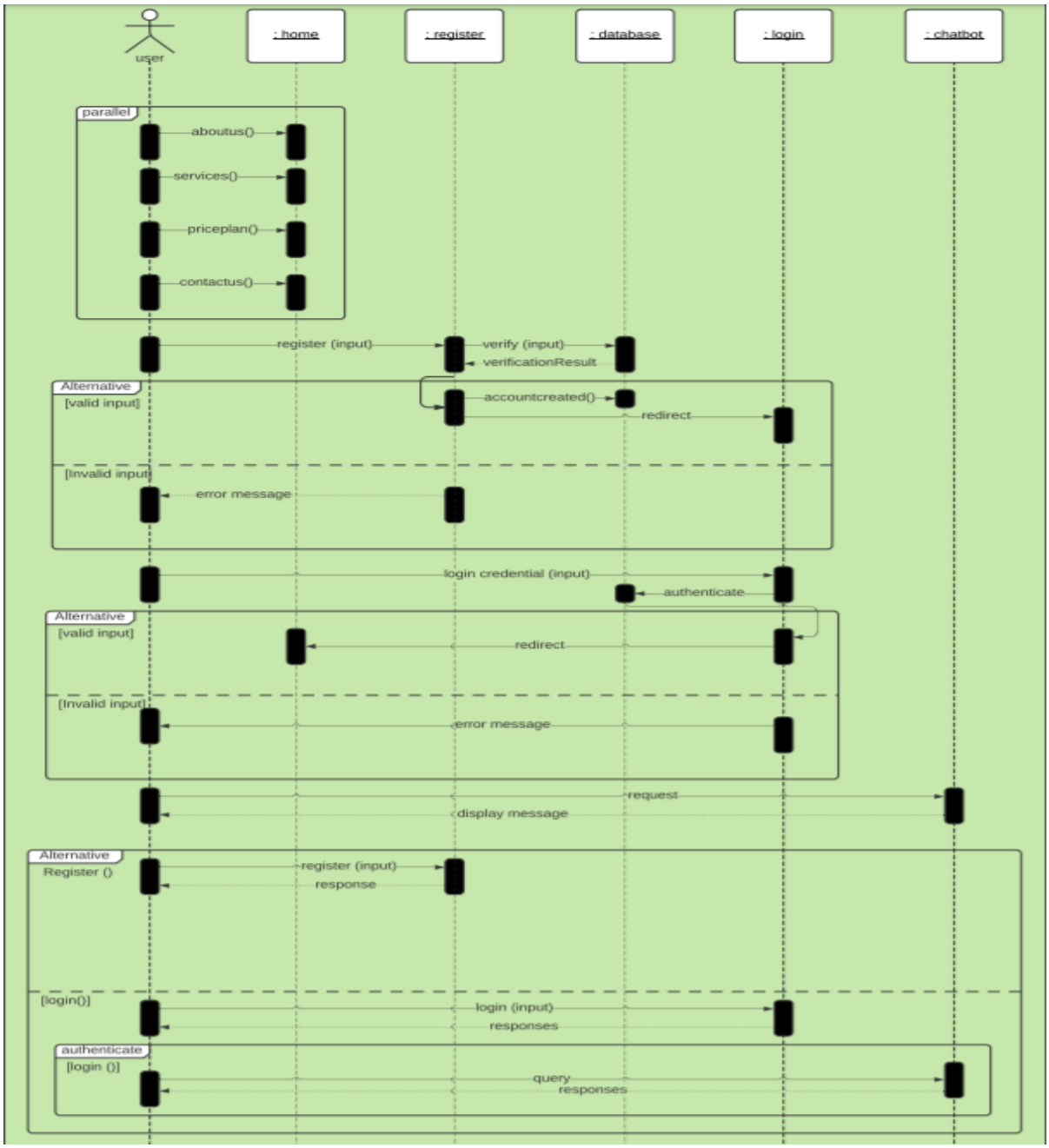


Figure 53 sequence diagram

State Transition Diagram

A State Transition Diagram is used to describe the state of activity. It describes all the objects it can have. Shows different stages through which objects passes or different state which an object can take in its entire life and what causes changes in the stage is illustrate in the diagram is state transition diagram(Copeland, 2003).

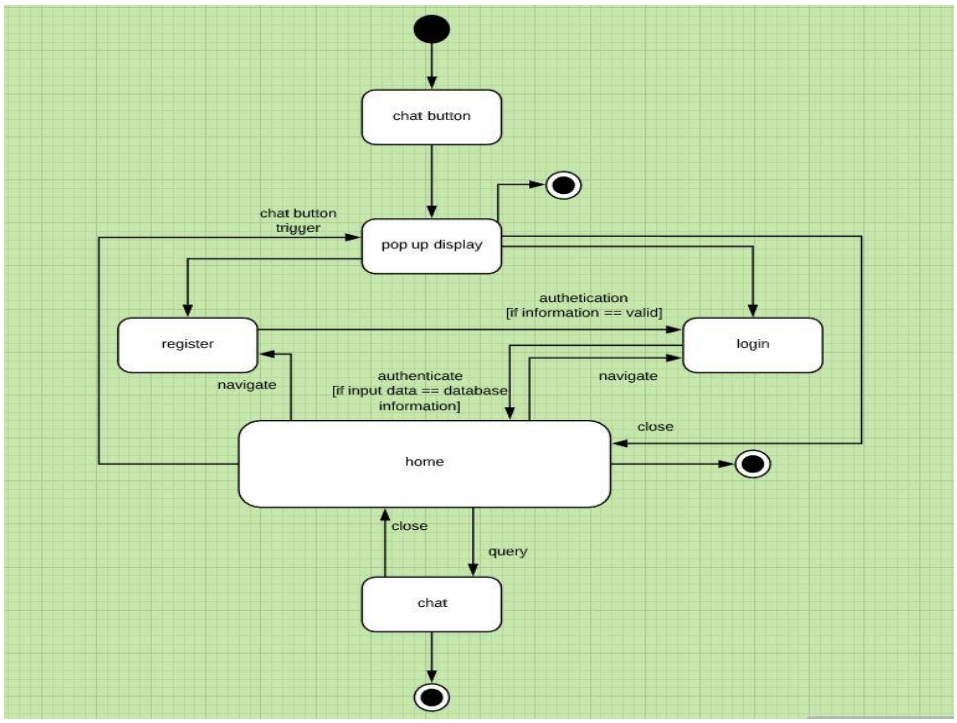


Figure 54 chatbot state

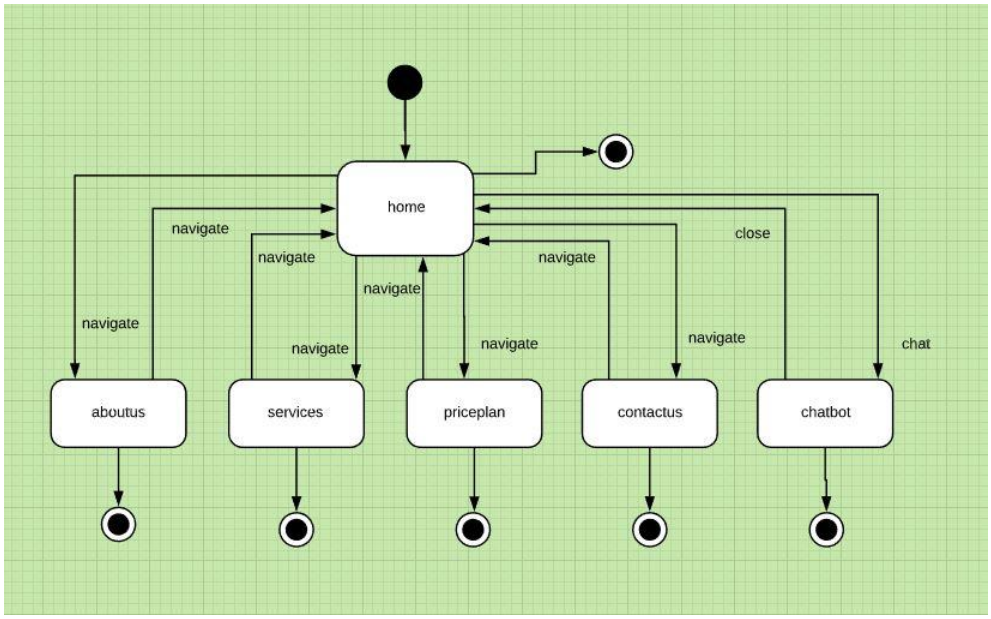


Figure 55 dashboard state

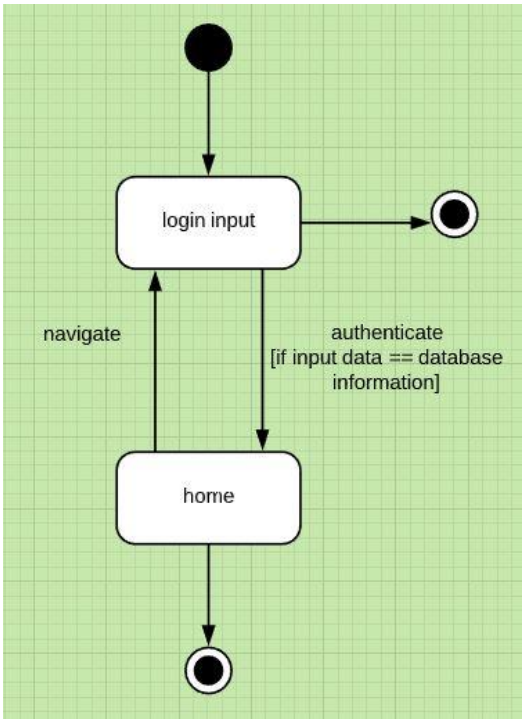


Figure 56 login state

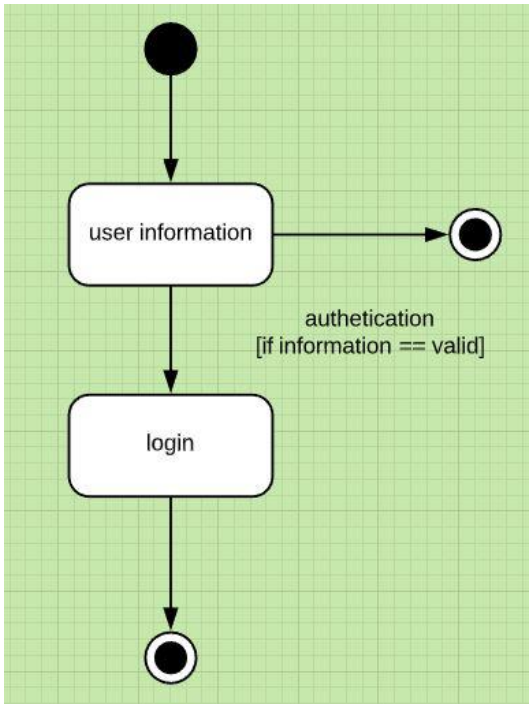


Figure 57 register state

Construction

It is the part where actual coding is done in reference of modelling diagram and design. This section include result of the coding and testing part.

Below is the actual pages of the web application.

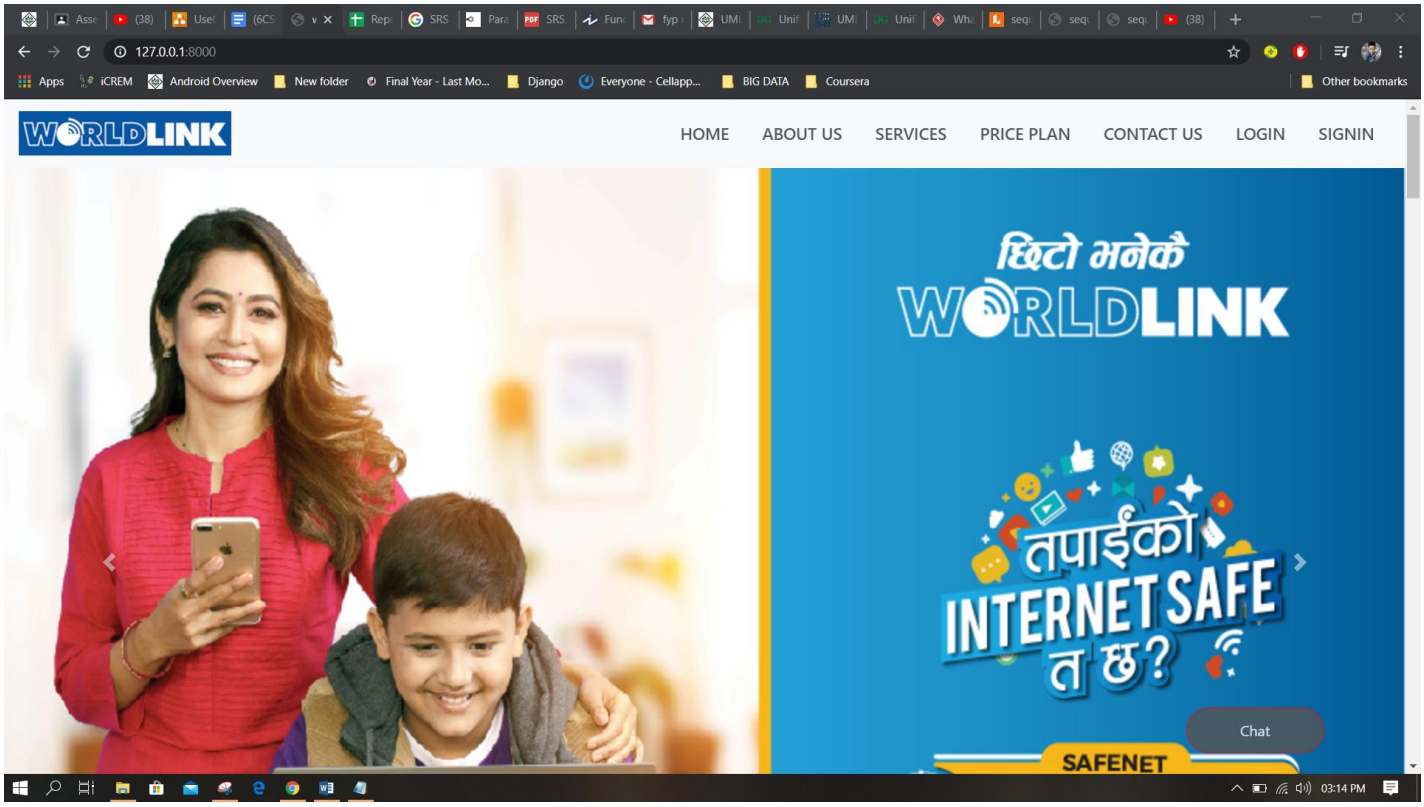


Figure 58 actual home page

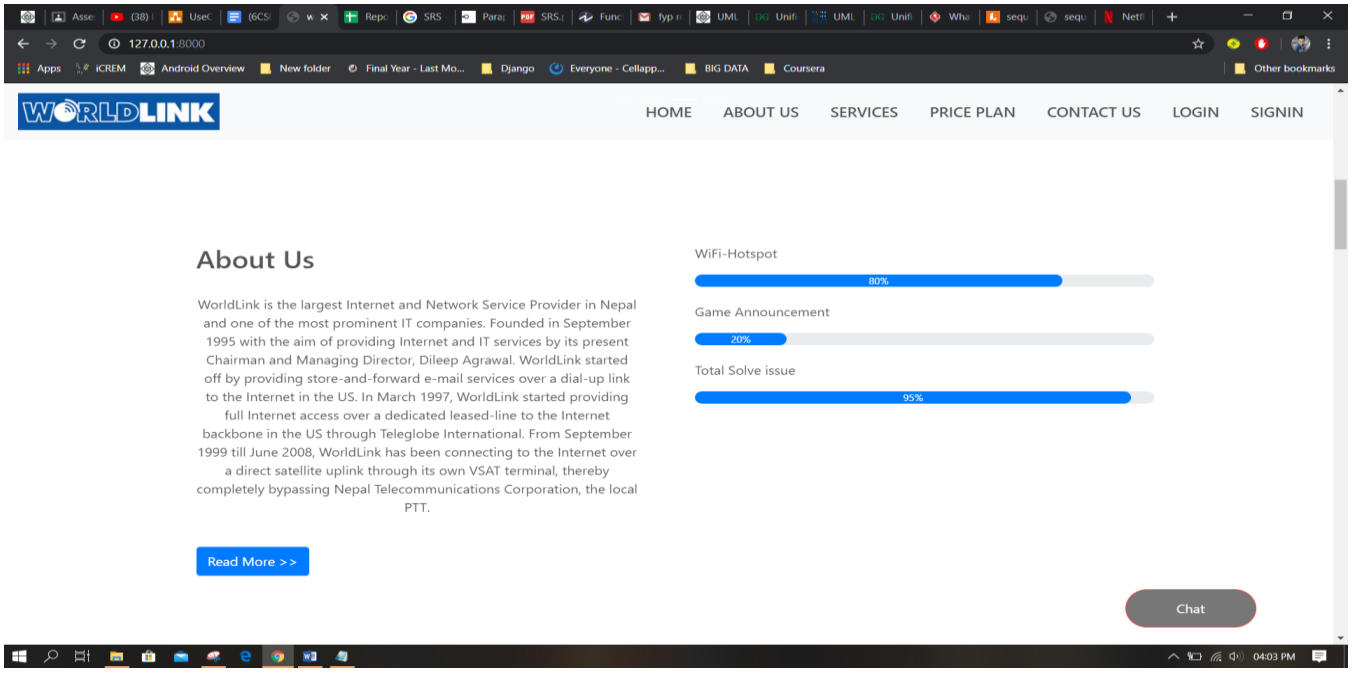


Figure 59 actual aboutus page

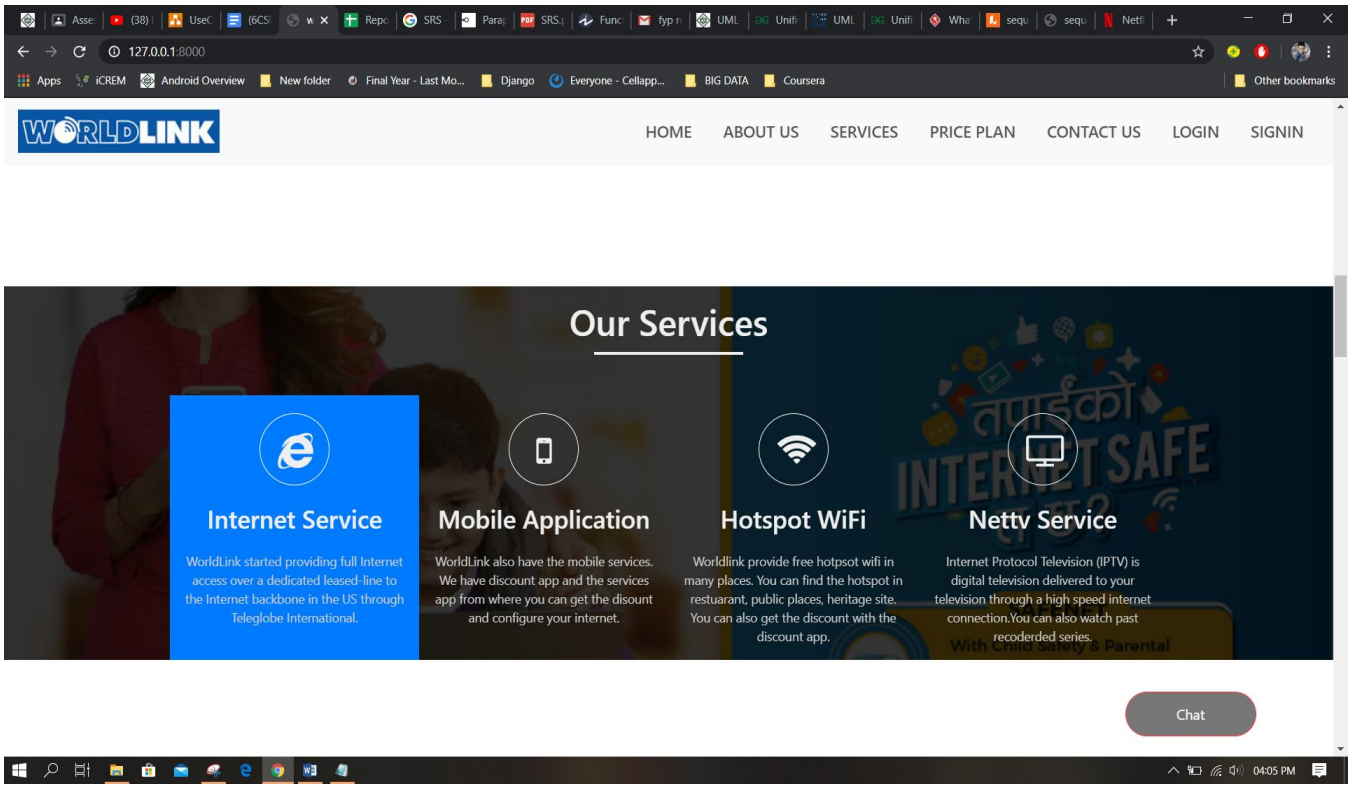


Figure 60 actual services page

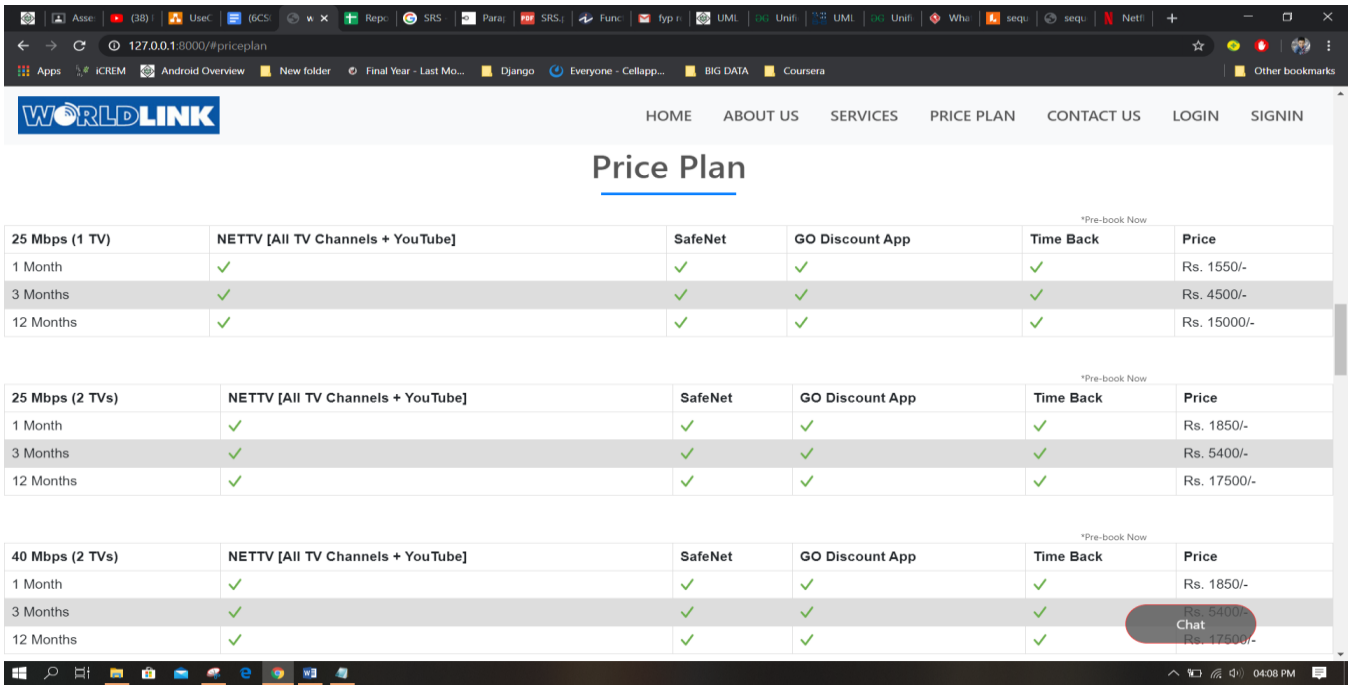


Figure 61 actual price page

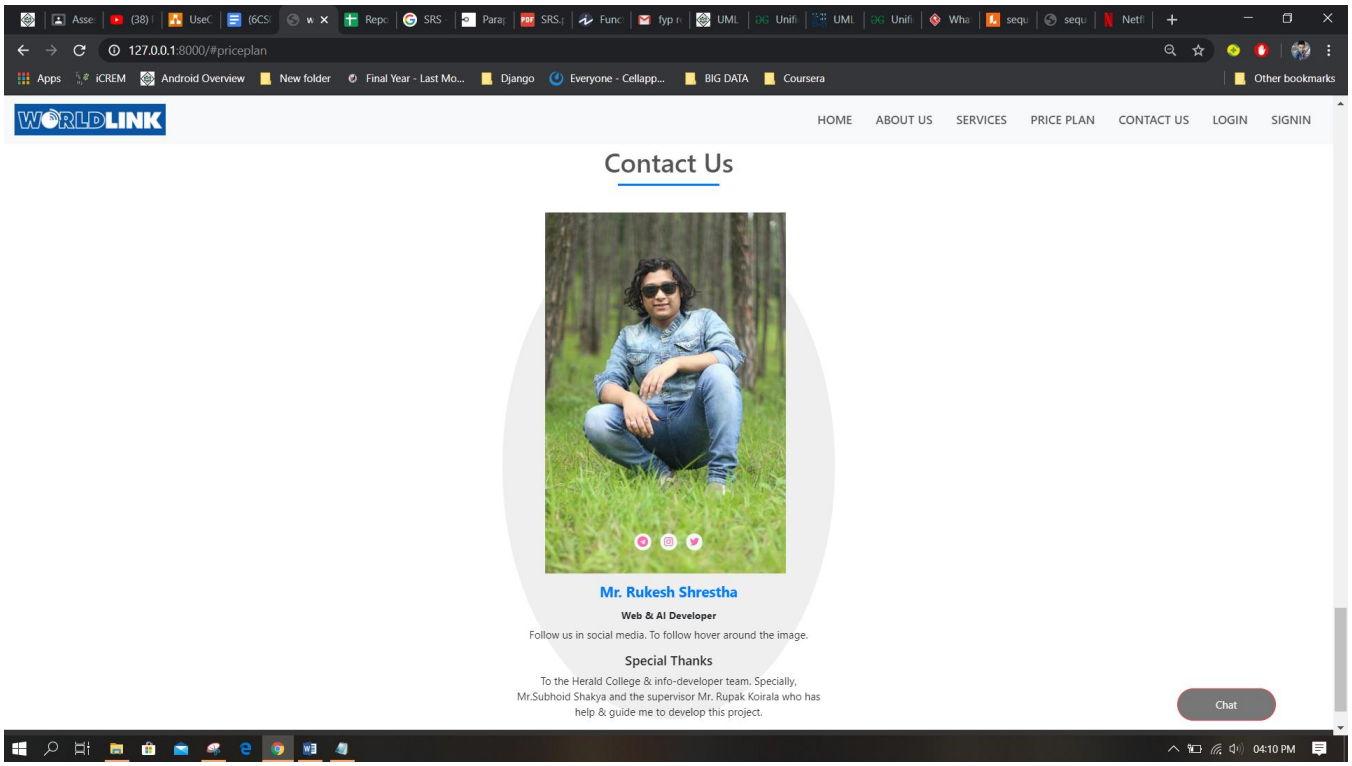


Figure 62 actual contact page

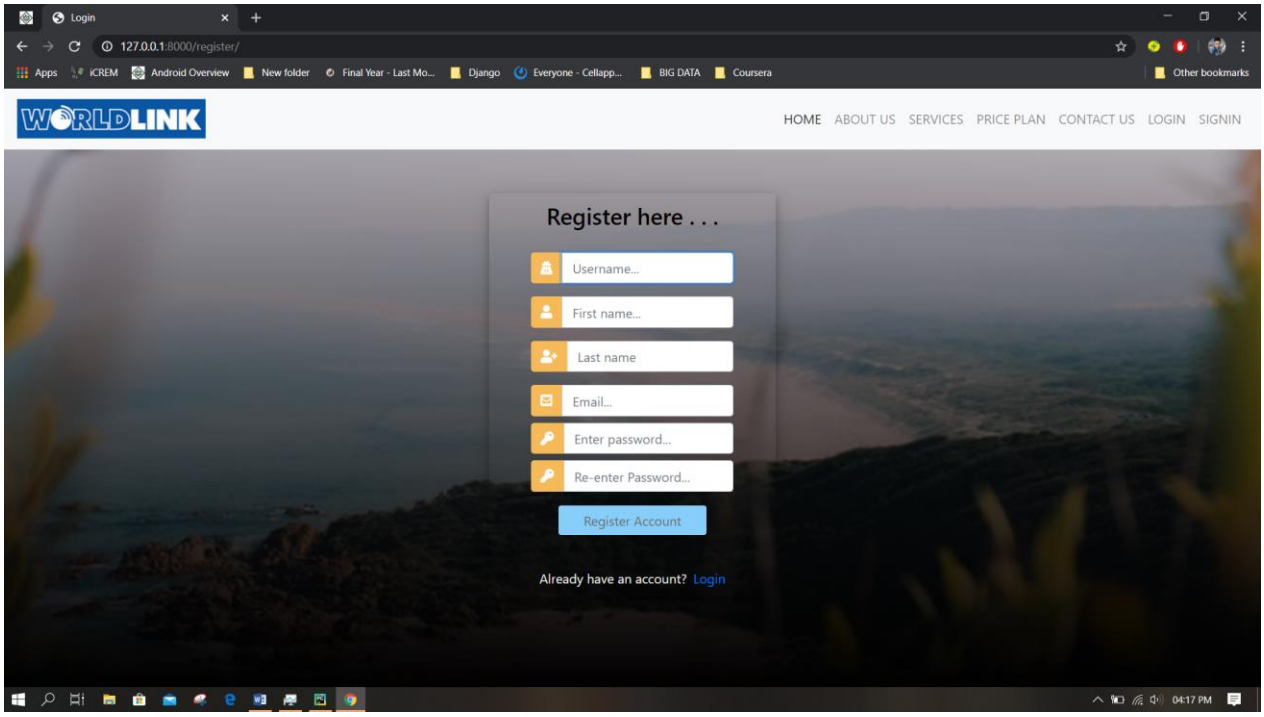


Figure 63 actual register page

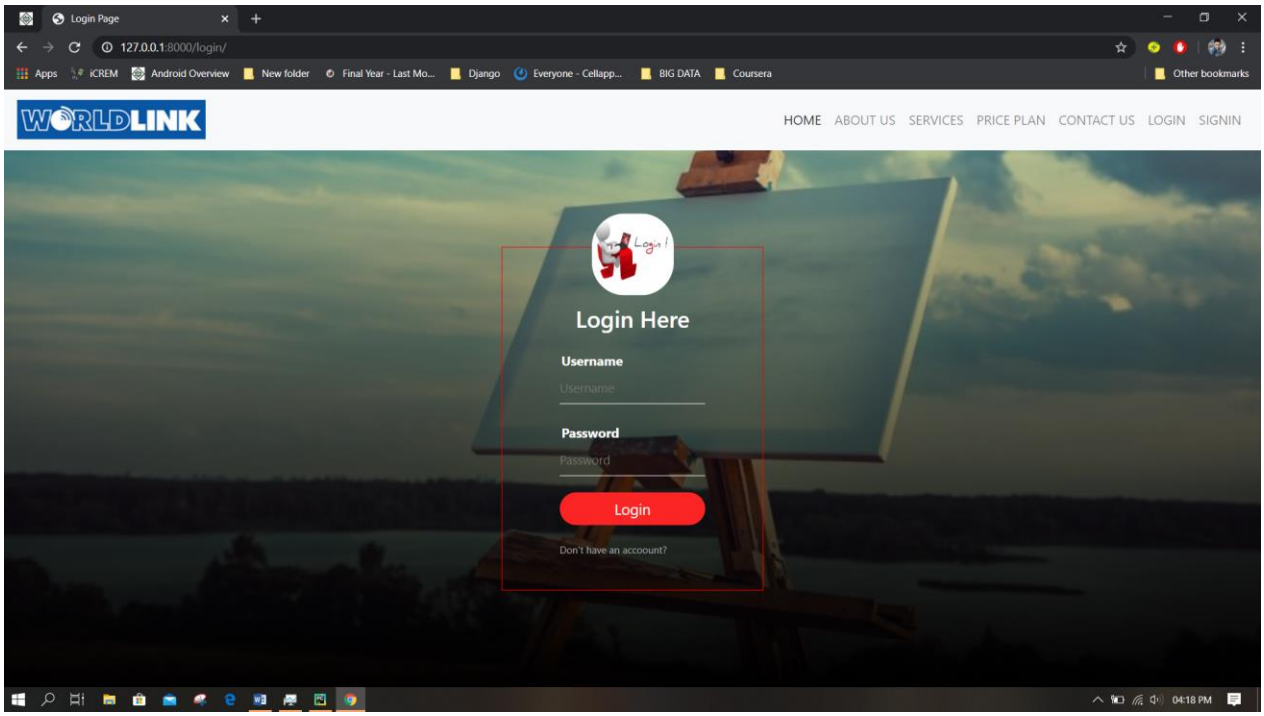


Figure 64 actual login page

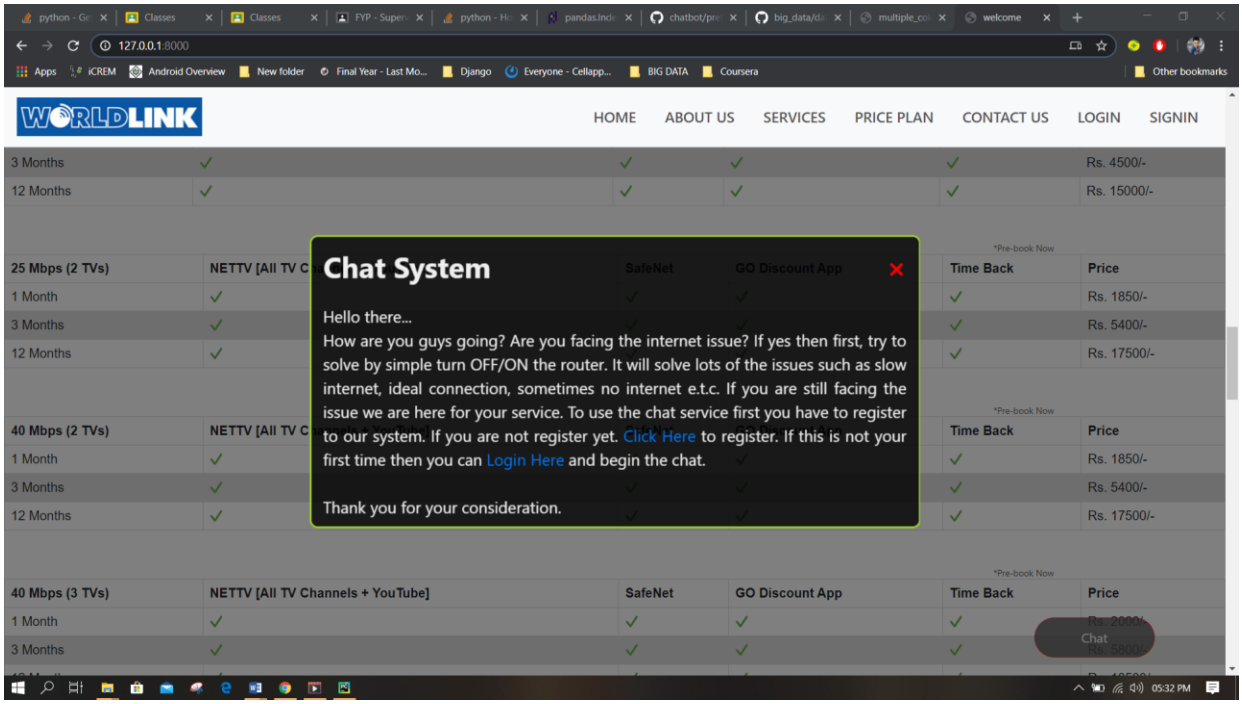


Figure 65 actual popup window

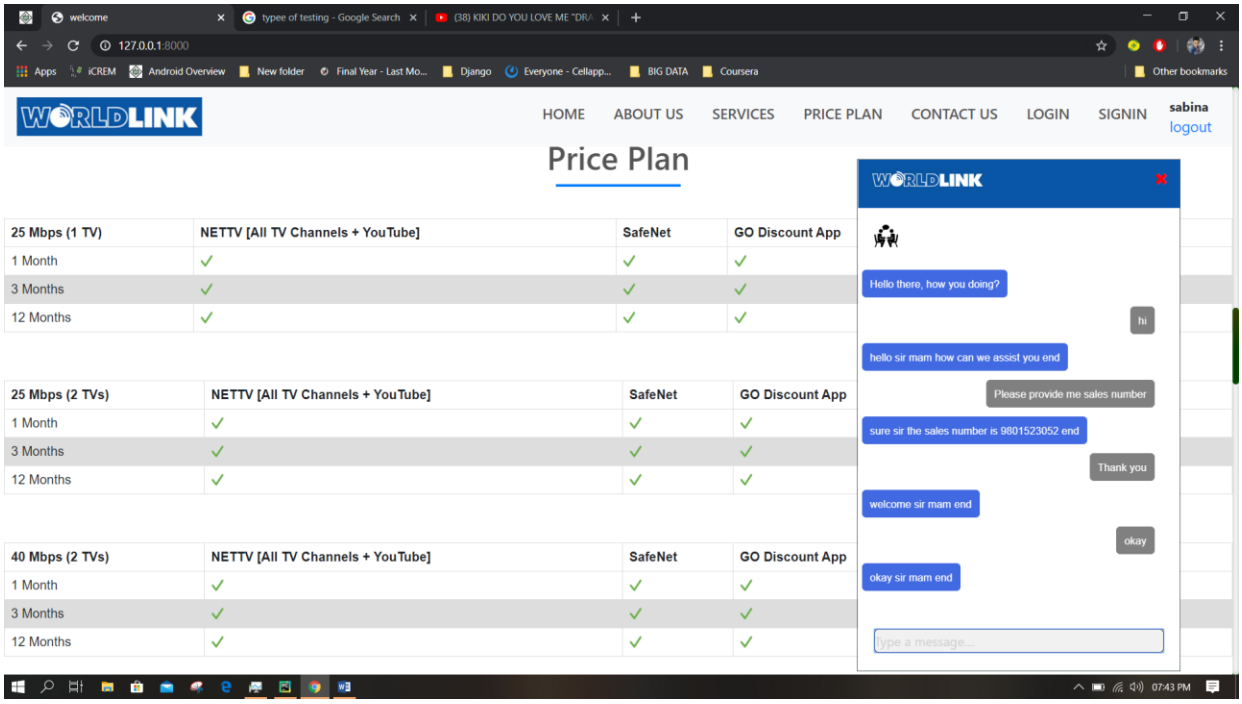


Figure 66 actual chat window

4.2.3 System Architecture

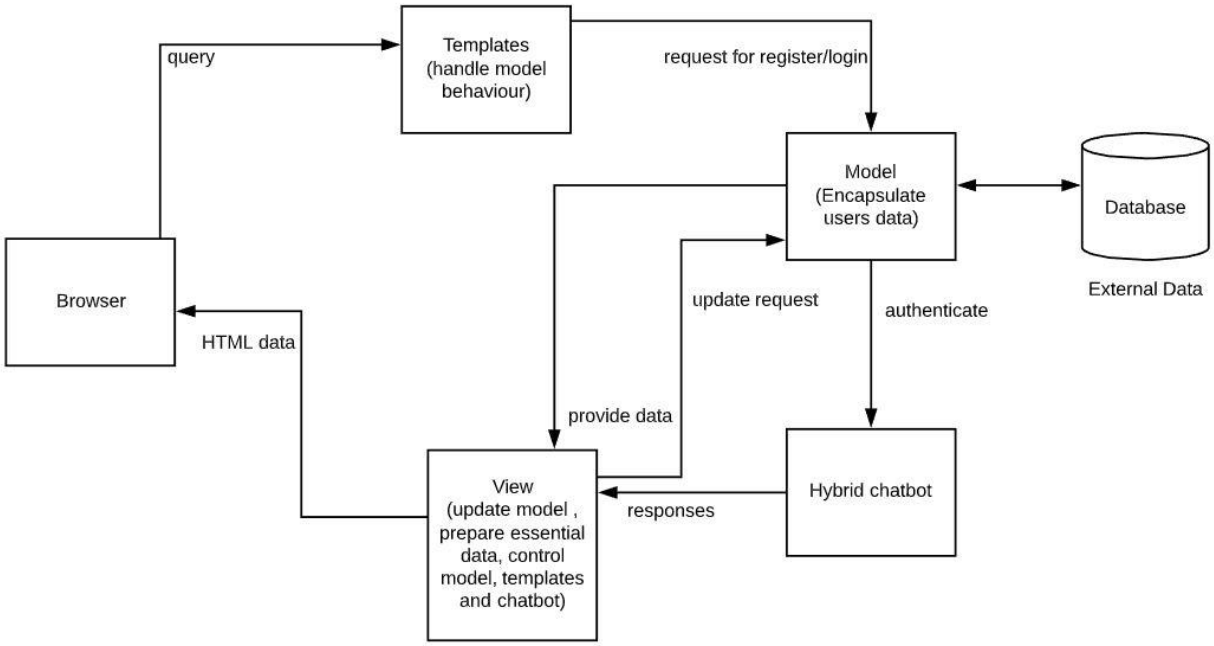


Figure 67 Proposed system block diagram

From the browser user can interact with the system. They can navigate to different pages. They can register and login to system. The form is validate with default Django form validation. If the form is valid then user data is store in database otherwise the error is display. Here, default Django database is used. Django model is used to interact with database. After registration, they can login to system and chat with chatbot. User is authenticated for chatting. It they are not authenticated, message is display in browser in the form of HTML data.

4.2.4 Testing

4.2.4.1 White Box

It is done using the coding. Django inbuilt TestCase is used for testing. Different method is develop for unit testing.

```
2020-06-10 16:57:35.042949: I tensorflow/core/common_runtime/gpu/gpu_device.c
2020-06-10 16:57:35.046604: I tensorflow/core/common_runtime/gpu/gpu_device.c
Creating test database for alias 'default'...
System check identified no issues (0 silenced).
..
----------------------------------------------------------------------
Ran 2 tests in 0.003s

OK
Destroying test database for alias 'default'...
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.iter
WARNING:tensorflow:Unresolved object in checkpoint: (root).optimizer.decay
```

Figure 68 result of unit testing

4.2.4.2 Black Box

Registration					
Username	Password Match	E-mail	Expect	Result	Screenshot
rukesh	1.adminadmin 2.adminadmin	r@gmail.com	Fail	Fail	• username ◦ A user with that username already exists.
sachin	1.admin 2.amdin	s@gmail.com	Fail	Fail	• password2 ◦ The two password fields didn't match.
susil	1.susHilsus 2.susHilsus	su@	Fail	Fail	Please enter a part following '@'. 'su@' is incomplete.
jay	2.jaynepa 2.jaynepa	su@gmail.com	Fail	Fail	This password is too short. It must contain at least 8 characters.
bijay	1.bijaybijay 2.bijaybijay	bijy@gmail.com	Pass	Fail	◦ The password is too similar to the username.
riya	1.password1 2.password1	riy@gmail.com	Pass	Fail	◦ This password is too common.

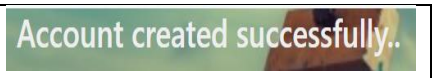
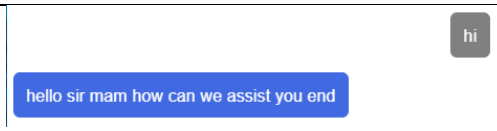
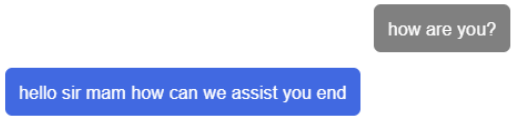
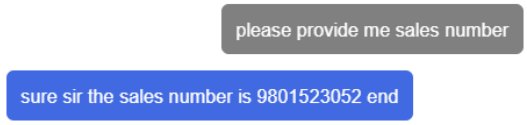
sabina	1.chatbotnepal 2.chatbotnepal	sa@gmail.com	Pass	Pass	

Table 2 register testing

Login				
Username	Password	Expect	Result	Screenshot
ravi	ravipass	Fail	Fail	
sabina	amdinamdin	Fail	Fail	
sabina	chatbotnepal	Pass	Pass	

Table 3 login testing

chatbot				
Query	Response	Expect	Result	Screenshot
hi	hello sir mam how can we assist you end	hello sir/mam how can we assist you ?	PASS	
how are you?	hello sir mam how can we assist you end	I am fine.	fail	
please provide me sales number.	sure sir the sales number is 9801523052 end	the sales number is 9801523052.	PASS	

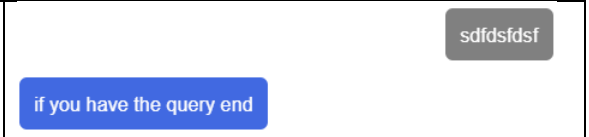
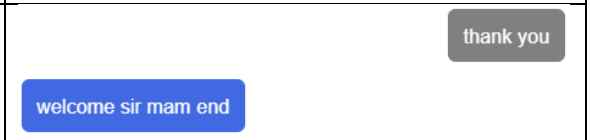
tero tauko	sorry sir end	Offensive word	fail	
sdfdsfdsf	If you have the query end	generate sequence	PASS	
thank you	Welcome sir mam end	Welcome sir/mam	PASS	

Table 4 Chatbot testing

GUI testing

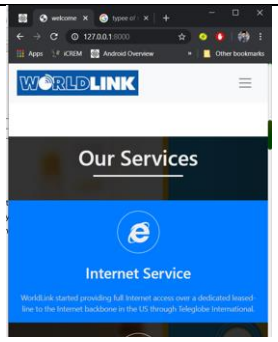
	Expect	Result	Screenshot
Responsive	PASS	PASS	

Table 5 GUI testing

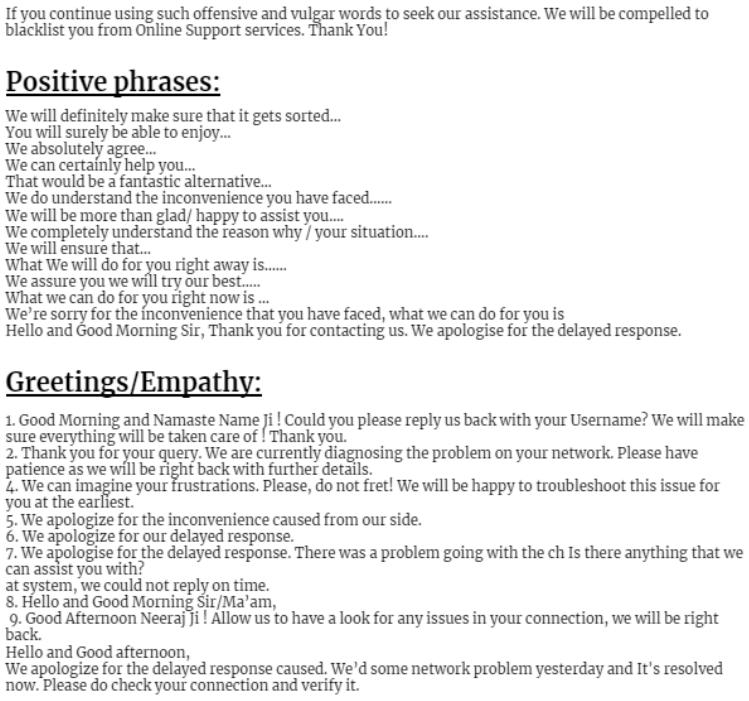
Deployment

In this section, video of the project is develop and mail to supervisor for the feedback of project.

4.3 AI Model Development Process

4.3.1 Data Understanding

Data is the important thing in every field. Without the proper data and analysis none of the systems can be effective. Data for this project is gathered by interviewing with representatives. They have some templates which include frequent ask questions (FAQ) answers. After talking with them they agree to share templates. The gathering data was rough and mainly in natural languages. To understand the templates, complete reading was essential. Therefore, after reading all the templates, research is done to convert the given data to Comma Separated Value (CSV) and JSON format.



If you continue using such offensive and vulgar words to seek our assistance. We will be compelled to blacklist you from Online Support services. Thank You!

Positive phrases:

We will definitely make sure that it gets sorted...
You will surely be able to enjoy...
We absolutely agree...
We can certainly help you...
That would be a fantastic alternative...
We do understand the inconvenience you have faced.....
We will be more than glad/ happy to assist you....
We completely understand the reason why / your situation....
We will ensure that...
What We will do for you right away is.....
We assure you we will try our best.....
What we can do for you right now is ...
We're sorry for the inconvenience that you have faced, what we can do for you is
Hello and Good Morning Sir, Thank you for contacting us. We apologise for the delayed response.

Greetings/Empathy:

1. Good Morning and Namaste Name Ji ! Could you please reply us back with your Username? We will make sure everything will be taken care of ! Thank you.
2. Thank you for your query. We are currently diagnosing the problem on your network. Please have patience as we will be right back with further details.
4. We can imagine your frustrations. Please, do not fret! We will be happy to troubleshoot this issue for you at the earliest.
5. We apologize for the inconvenience caused from our side.
6. We apologize for our delayed response.
7. We apologise for the delayed response. There was a problem going with the ch Is there anything that we can assist you with?
at system, we could not reply on time.
8. Hello and Good Morning Sir/Ma'am,
9. Good Afternoon Neeraj Ji ! Allow us to have a look for any issues in your connection, we will be right back.
Hello and Good afternoon,
We apologize for the delayed response caused. We'd some network problem yesterday and It's resolved now. Please do check your connection and verify it.

Figure 69 raw data

The complete data which is gather from the company is in appendix.

The gathered data was 21596 words. For this step, first the data is written in notepad with human and reply. Humans denote the question and reply denote its answer.

```
Human:  Namastae sir.
Reply: Namastae sir/mam. How can we help you?
Human : My internet is not working well for a past few days.
Reply: Please mention your username and the number sir/mam.
Human: My username is rukesh_home.
Reply: How far are you from your router sir/mam?
Human: I am next to my router.
Reply: How many lights are glowing in your router?
Human: I have got the black nokia router. The lights glowing are AUTH,POWER,LINK, WLAN.
Reply: Can you please turn off the router and check after five minutes.
Human: Sure sir/mam.  Thanks a lot.
Reply: welcome sir
```

Figure 70 human reply

Above mentioned is one of the conversations. Total conversation was 158, lines of text 2522. Now this is in txt format. For the model architecture, it should be converted to csv format. It was easy to convert txt format into csv format. While converting, there were lots of errors such as regular expression, line gap, case sensitive etc. Regular expression and the case sensitive error was solved by adding a couple lines of code in convert.py file while handling error of line gap was difficult. 158 conversations were there with line 2522, going through each and every line and handling it in an inefficient way. So to solve this issue with the help of Python Programming Language (PPL) a new program is written which removes all the unnecessary gaps.

human	reply
My internet is not working for the an hour. My browsing is completely slow.	Hello sir/mam. How can we assist you?
You internet is wrost. I am facing completely slow browing in the connection.	Namastae sir/mam. How can we help you?
I am facing slow internet connection.	How far are you from the router?
I am facing slow internet connection.	How far are you from the router?
I am in the office now. I certainly face slow internet connection.	If you are in office how could you know your internet in slow?
When i was at home early morning. My internet was slow.	Please update us your query when you are near to your router sir/m

Figure 71 Final CSV data

Same approach is followed for preparing data in JSON but it was more time consuming as the entire conversation cannot be converted into JSON at the time. Due to the tag needed in JSON data, similar types of conversation is separated from entire data, processed one at a time and merged it.

Raw Blame History

14 lines (14 sloc) 290 Bytes

```
{
    "intents": [
        {
            "tag": "develop",
            "patterns": [
                "can you tell me who develop you?"
            ],
            "responses": [
                "mr. rukesh shrestha develop me."
            ],
            "context_set": ""
        }
    ]
}
```

Figure 72 Single JSON data

Above figure denote the single JSON data.


```
{
  "intents":
  [
    {
      "tag": "changepasswordfrommobile",
      "patterns":
      [
        "i want to hide my wifi ssid bro.",
        "could you guide me to hide the wifi password?"
      ],
      "responses":
      [
        "okay sir/mam we have got two method to hide the network.\tsteps to hide wifi ssid\tfrom official website\t1. go to our website
      ],
      "context_set": ""
    },
{
      "tag": "thankyou",
      "patterns":
      [
        "thank you for the coorporation.",
        "okay",
        "you really suprise me.",
        "you are such an insanse.",
        "🤣🤣🤣🤣",
        "thank you for your coorporation and guidance.",
        "ummm",
```


Figure 73 Final JSON data

Figure 69 and 71 denote the final data needed for preprocessing.

4.3.1.1 Exploratory Data Analysis

It is used for analyzing datasets to epitomize fundamental peculiarity mainly using the visualization techniques. Here, the analysis is done using pandas-profiling library.

4.3.1.1.1 CSV data

In the below figure, overall data analysis is done. There is two variable in dataset which is categorical in nature. The total number of data is 629. There is 59 duplicates rows in dataset with percentage 9.4%.

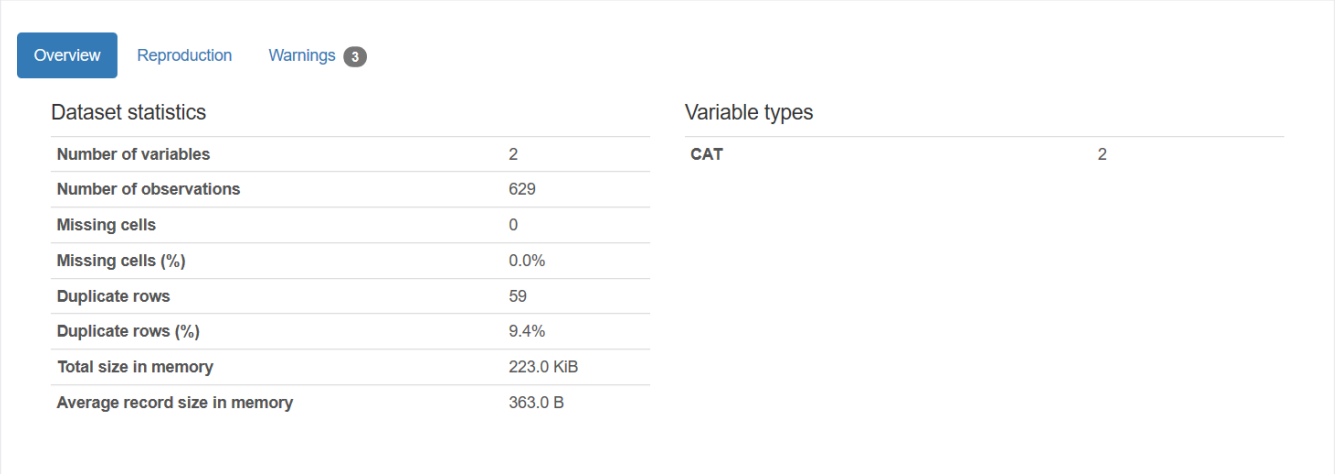


Figure 74 Overall analysis

In the below figure human and reply data are analyzed. Distinct value of human is 470 whereas reply is 349. There is no missing value in each feature. Both the feature took 5.0 kilobyte o memory.

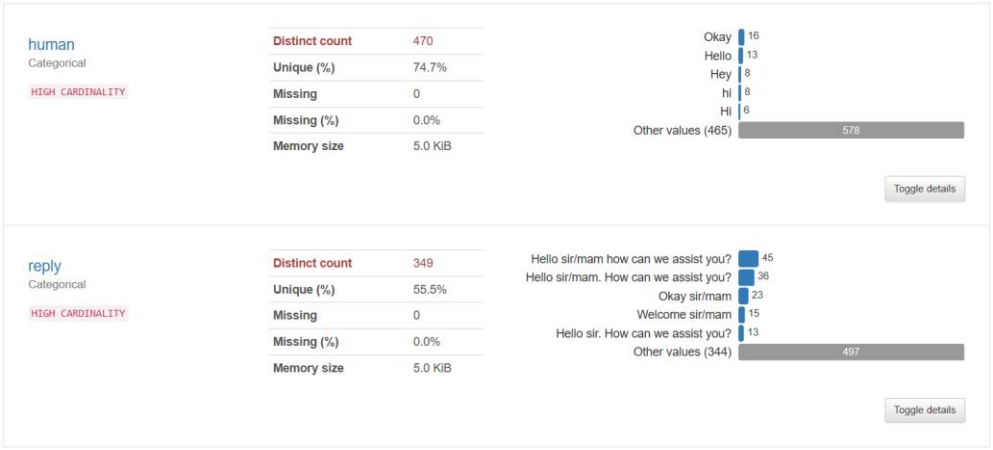


Figure 75 variables analysis

In the below figure, missing value is visualized in graph. In the total data of 629, there is no any missing value.

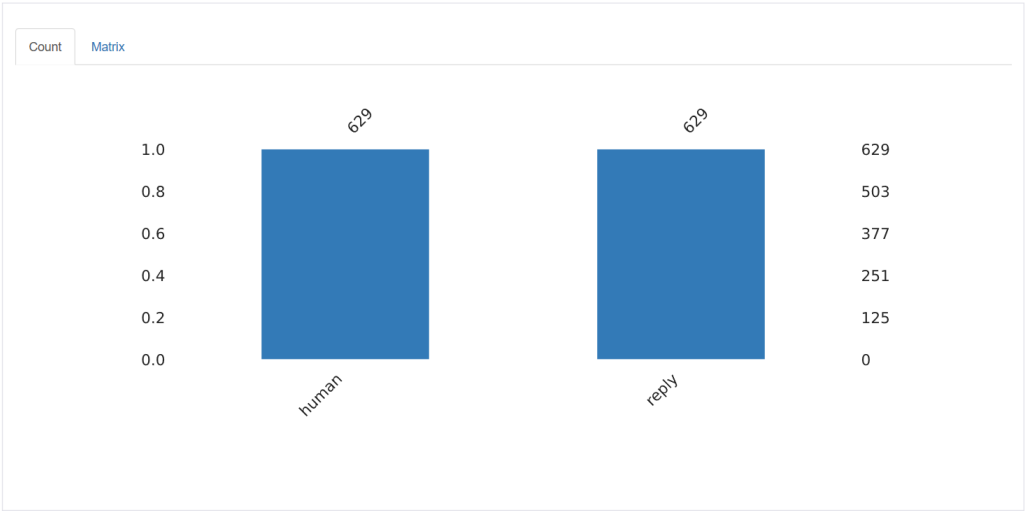


Figure 76 visualizing missing value

In the below figure, most frequent data is count and visualized.

Most frequent

	human	reply	count
10	Hey	Welcome sir/mam	4
35	Thank you so much dear. Have a beautiful day.	Welcome and same to you sir.	4
0	Are you idiot?	Hello sir/mam. How can we assist you?	3
3	Could you please mention the sales department number?	Sure sir. The sales number is 9801523052.	3
4	Could you please mention your name?	Sorry sir, We are not allowed to provide our information. Please mention your query.	3
11	Hey Internet service Provider.	Hello sir. How can we assist you?	3
12	Hey, worldlink.	Hello sir. How can we assist you?	3
14	Hi, worldlink.	Hello sir. How can we assist you?	3
20	I want a new connection. Please mention the sales number.	The sales number is 9801523052.	3
26	Okay	Okay sir/mam	3

Figure 77 visualizing duplicates

Both the below figure shows maximum, medium, mean, minimum length of the sentences in human and reply feature.

Figure 78 human length analysis

Figure 79 reply length analysis

Both the below figure, most occurrence sentences and characters type are visualized.

Most occurring blocks

Value	Count	Frequency (%)
ASCII	22015	99.5%
VS	54	0.2%
Emoticons	38	0.2%
None	16	0.1%
Punctuation	10	< 0.1%

Figure 80 Most occurrence blocks

Most occurring scripts

Value	Count	Frequency (%)
Latin	17581	79.4%
Common	4498	20.3%
Inherited	54	0.2%

Figure 81 most occurrence scripts

4.3.1.1.2 JSON data

In the below figure, overall JSON data is visualized. Data is same as the CSV but only the format is difference. There is total 35 data in JSON which use 7.8 kilobyte of memory.

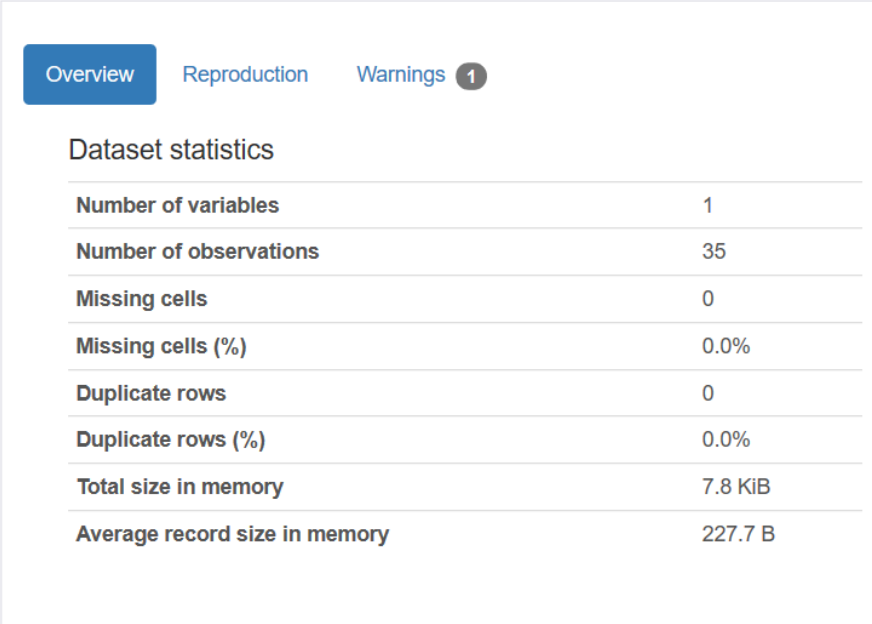


Figure 82 overview

In the below figure, missing data is visualized as there is no any missing value the bar shows the 35.

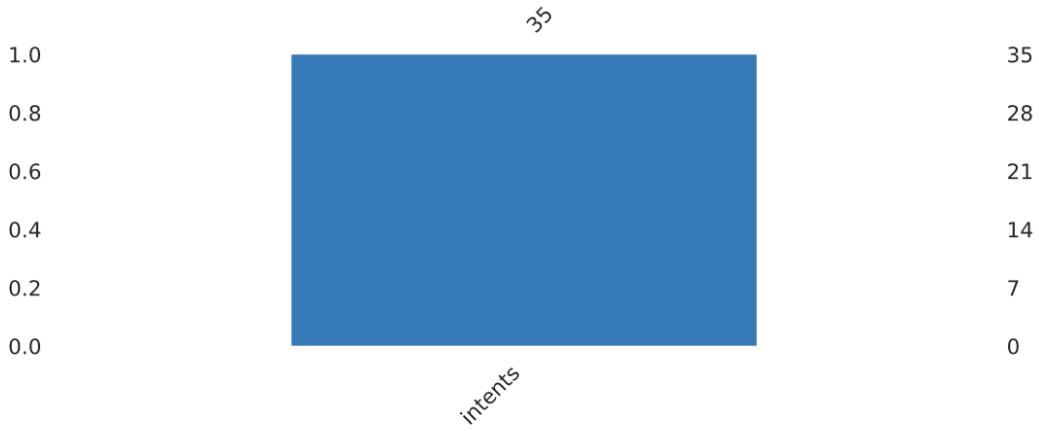


Figure 83 visualizing missing value

4.3.2 Iterate-1 (LSTM)

4.3.2.1 Data Pre Processing

Data preprocessing is a challenging task. The gathered data is raw therefore, it needs to be clean. The main goal of this step is to make data ready for developing models.

For the LSTM data preparation pre trained Word2Vec is downloaded from the AP News corpus. It consists of 669549 of vectors.

```
: model = gensim.models.Word2Vec.load('/home/rukesh/Documents/

: print(model)

Word2Vec(vocab=669549, size=300, alpha=0.025)
```

Figure 84 pretrain word2vec

Below figure shows words in vocab.

```
model.wv.vocab

{'tripolitan': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f19101cacf8>,
 'biennials': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329d30>,
 'biysk': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329d68>,
 'woodd': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329da0>,
 'woode': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329dd8>,
 'pujyapada': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329e10>,
 'verplank': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329e48>,
 'luanco': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329e80>,
 'woodi': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329eb8>,
 'maderista': <gensim.models.deprecated.keyedvectors.Vocab at 0x7f18ff329ef0>,
```

Figure 85 vocab words

After importing the vocab, similar word is search according to the project.

```
: model.most_similar('internet')
```

```
/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py:1: DeprecationWarnir
deprecated `most_similar` (Method will be removed in 4.0.0, use self.wv.most_similar() ins
  """Entry point for launching an IPython kernel.
```

```
: [('online', 0.7655734419822693),
   ('websites', 0.7572559118270874),
   ('internet-based', 0.7348347306251526),
   ('web', 0.7268290519714355),
   ('e-mail', 0.72633957862854),
   ('networking', 0.7230650186538696),
   ('email', 0.7106530070304871),
   ('blogs', 0.6928676962852478),
   ('peer-to-peer', 0.6860390901565552),
   ('file-sharing', 0.6849340200424194)]
```

Figure 86 similar word

After analyzing the similar word it seems to be accurate as output is likely to word internet.

Now the actual csv data is imported using pandas.

```
dataset
```

	human	reply
0	Namastae sir.	Namastae sir/mam. How can we help you?
1	My internet is not working well for a past few...	Please mention your username and the number si...
2	My username is rukesh_home.	How far are you from your router sir/mam?
3	I am next to my router.	How many lights are glowing in your router?
4	I have got the black nokia router. The lights ...	Can you please turn off the router and check a...
...	...	...

Figure 87 actual data

Separating the human and reply data:

```
data_human = dataset.loc[:,['human']]
data_human.head(10)
```

	human
0	Namastae sir.
1	My internet is not working well for a past few...
2	My username is rukesh_home.
3	I am next to my router.
4	I have got the black nokia router. The lights ...
5	Sure sir/mam. Thanks a lot.
6	Hello worldlink.
7	Could you please mention the Customer Accounts...
8	Thank you so much dear. Have a beautiful day.
9	Hello.

```
data_reply = dataset.loc[:,['reply']]
data_reply.head()
```

	reply
0	Namastae sir/mam. How can we help you?
1	Please mention your username and the number si...
2	How far are you from your router sir/mam?
3	How many lights are glowing in your router?
4	Can you please turn off the router and check a...

Figure 88 data separating

Below figure display the result in numpy array.

```
data_human.values
```

```
array([['Namastae sir.'],
       ['My internet is not working well for a past few days.'],
       ['My username is rukesh_home.'],
       ['I am next to my router.'],
       ['I have got the black nokia router. The lights glowing are AUTH
       ['Sure sir/mam.  Thanks a lot.'],
       ['Hello  worldlink.'],
       ['Could you please mention the Customer Accounts number?']
```

Figure 89 numpy array

Now two list is created, to store the tokenized word, one for human and another for the reply.

```
human_token = []
reply_token = []
```

```
for words in list_human:
    tokenized_words = word_tokenize(words)
    human_token.append(tokenized_words)
```

```
len(human_token)
```

```
629
```

Figure 90 tokenized words

After the tokenizing, from the pre trained word2vec, tokenized words vector is extracted and stored in vector_represent_human and vector_represent_reply

```
vector_represent_human=[]
for sent in human_token:
    vectors_human = [model[w] for w in sent if w in model.wv.vocab]
    vector_represent_human.append(vectors_human)
```

```
/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py
deprecated `__getitem__` (Method will be removed in 4.0.0, use self.w
  This is separate from the ipykernel package so we can avoid doing i
```

```
len(vector_represent_human)
```

```
629
```

```
vector_represent_reply=[]
for sent in reply_token:
    vectors_rep = [model[w] for w in sent if w in model.wv.vocab]
    vector_represent_reply.append(vectors_rep)
```

```
/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py
deprecated `__getitem__` (Method will be removed in 4.0.0, use self.w
  This is separate from the ipykernel package so we can avoid doing i
```

```
len(vector_represent_reply)
```

```
629
```

Figure 91 extracting word

As the vector is extracted, there are two variables storing it, it should be concatenated in one variable and create the zero matrix.

```
single_data = human_token + reply_token
```

```
len(single_data)
```

```
1258
```

```
item = [eachitem[0] for eachitem in single_data]
```

```
sentend=np.zeros((300,),dtype=np.float32)
```

```
for each_word in item:
    print(len(each_word))
```

```
8
2
```

Figure 92 concatenate

Now appending the zero matrix in vector and display vector.

```
for tok_sent in vector_represent_reply:
    tok_sent[14:]=[]
    tok_sent.append(sentend)
```

```
tok_sent
```

```
[array([ 0.11414972,  0.0455336 ,  0.17969868, -0.05490275,  0.08439939,
        -0.05345882, -0.04455809, -0.05177481,  0.02036242, -0.25915965,
        -0.00416537, -0.01353797,  0.04015744, -0.02325973,  0.1918035 ,
        -0.14270239, -0.02262927, -0.0318507 , -0.06595455, -0.14815669,
        -0.07393415,  0.0250927 ,  0.1285426 ,  0.07482776, -0.13591138,
        -0.07712218, -0.23563722, -0.4617028 , -0.01881071,  0.16028808,
         0.27642736, -0.18794231, -0.1429102 , -0.0851507 , -0.09648377,
        -0.00637851,  0.26491433,  0.23398484,  0.27601117, -0.11377874,
         0.16573389,  0.02424407, -0.04580966,  0.12385734,  0.05242395,
         0.05819058,  0.1259511 ,  0.20980603,  0.02003118,  0.14589104,
```

Figure 93 vector

Using the condition in vector for better result in developing model.

```
# print(word_length)
for tok_sent in vector_represent_reply:
    if len(tok_sent)<15:
        for i in range(15-len(tok_sent)):
            tok_sent.append(sentend)
```

```
tok_sent
```

```
[array([ 0.11414972,  0.0455336 ,  0.17969868, -0.05490275,  0.08439939,
        -0.05345882, -0.04455809, -0.05177481,  0.02036242, -0.25915965,
        -0.00416537, -0.01353797,  0.04015744, -0.02325973,  0.1918035 ,
        -0.14270239, -0.02262927, -0.0318507 , -0.06595455, -0.14815669,
        -0.07393415,  0.0250927 ,  0.1285426 ,  0.07482776, -0.13591138,
        -0.07712218, -0.23563722, -0.4617028 , -0.01881071,  0.16028808,
         0.27642736, -0.18794231, -0.1429102 , -0.0851507 , -0.09648377,
        -0.00637851,  0.26491433,  0.23398484,  0.27601117, -0.11377874,
```

Figure 94 condition in vector

Now the vector is ready to fit into the model. So it is dumped using pickle to use later in the model development file.

```
with open('conversation.pickle','wb') as f:
    pickle.dump([vector_represent_human,vector_represent_reply],f)
```

Figure 95 pickle dump

4.3.2.2 Predictive Modelling

In the file chatbotlstmmodel.ipynb, model is created using LSTM, Simple RNN algorithm. First the dump pickle is loaded and converted to a numpy array.

```
with open('conversationbot.pickle','rb') as f:
    vec_x,vec_y=pickle.load(f)
```

```
vec_x=np.array(vec_x,dtype=np.float64)
vec_y=np.array(vec_y,dtype=np.float64)
```

```
vec_x
```

```
array([[[-2.22353652e-01,  8.03626180e-02, -1.53709784e-01, ...,
         -2.62854546e-01,  1.13615185e-01, -1.73337255e-02],
        [-5.15206754e-02,  1.42451704e-01,  2.84696929e-02, ...,
          1.21582197e-02,  9.52853858e-02, -9.46336985e-02],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        ...,
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
          0.00000000e+00,  0.00000000e+00,  0.00000000e+00],
        [ 0.00000000e+00,  0.00000000e+00,  0.00000000e+00, ...,
```

Figure 96 loading vector

Now the vector is separated in train, test using sklearn train_test_split function. The vector is split in an 80:20 ratio. The split is done for training and validating data.

```
x_train,x_test, y_train,y_test = train_test_split(vec_x, vec_y, test_size=0.2, random_sta
te=0)
```

Figure 97 splitting vector

In the below figure, a model is created using the sequential function of Keras. Four LSTM layers are added in a sequential model. Output dimension of each layer is set to 300. The input shape is set to a training shape from 1 to bottom. return_sequences is set to true which means it returns the output unit in each time step. Weight has been initialized using he_uniform algorithm. The dense layer activation function is relu as it fires all the neurons which are greater than zero. In the output layer the activate function is sigmoid as output is needed between 0-1. While compiling the model, adam optimizer is used as it adaptive chooses the learning rate for each parameter and tries to make the model diverse to global minima.

```
model=Sequential()
model.add(LSTM(output_dim=300,input_shape=x_train.shape[1:],return_sequences=True, init
='he_uniform', inner_init='he_uniform', activation='relu'))
model.add(LSTM(output_dim=300,input_shape=x_train.shape[1:],return_sequences=True, init
='he_uniform', inner_init='he_uniform', activation='relu'))
model.add(LSTM(output_dim=300,input_shape=x_train.shape[1:],return_sequences=True, init
='he_uniform', inner_init='he_uniform', activation='relu'))
model.add(LSTM(output_dim=300,input_shape=x_train.shape[1:],return_sequences=True, init
='he_uniform', inner_init='he_uniform', activation='sigmoid'))
model.compile(loss='cosine_proximity', optimizer='adam', metrics=['accuracy'])
```

Figure 98 creating model

Now its time to fit the model, model is fit using the fit function which takes couple of parameter training data, number of epoch and validation data. Epoch is used to iterate through particular number as mention here it is set to 130. Validation data is piece of data used in training the model to evaluate the model for hyper tuning.

```
: model.fit(x_train, y_train, nb_epoch=130,validation_data=(x_test, y_test))

/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py:1: UserWarni
`nb_epoch` argument in `fit` has been renamed `epochs`.
  """Entry point for launching an IPython kernel.

Train on 503 samples, validate on 126 samples
Epoch 1/130
503/503 [==============================] - 3s 6ms/step - loss: -0.0958 - accuracy:
- val_loss: -0.1445 - val_accuracy: 0.0122
Epoch 2/130
503/503 [==============================] - 2s 4ms/step - loss: -0.1640 - accuracy:
- val_loss: -0.1620 - val_accuracy: 0.0042
Epoch 3/130
```

Figure 99 fitting model

After 130 epoch the accuracy seems to be 0.0056 with loss -635563. Train model is saved in h5 format using the save function.

```
9.1788 - val_accuracy: 0.0032
Epoch 129/130
503/503 [==============================] - 2s 5ms/step - loss: -61.4135 - accuracy: 0.0057 - val_loss: -5
8.9525 - val_accuracy: 0.0021
Epoch 130/130
503/503 [==============================] - 2s 5ms/step - loss: -63.5563 - accuracy: 0.0056 - val_loss: -5
9.7432 - val_accuracy: 0.0026
```

Figure 100 accuracy

Matplotlib library is used to plot the history of model in graph.

```
: import matplotlib.pyplot as plt
  plt.plot(history.history['accuracy'],color='yellow',label='accuracy')
  plt.plot(history.history['loss'],color='red',label='loss')
  plt.plot(history.history['val_accuracy'],color='blue',label = "val_accuracy")
  plt.plot(history.history['val_loss'],color='green',label = "val_loss")
  plt.title('Model History')
  plt.xlabel('Loss')
  plt.ylabel('Accuracy')
  plt.grid(True,color='k')
  plt.legend(['accuracy', 'loss','val_loss','val_accuracy'])
  plt.show()
```

Figure 101 history plotting

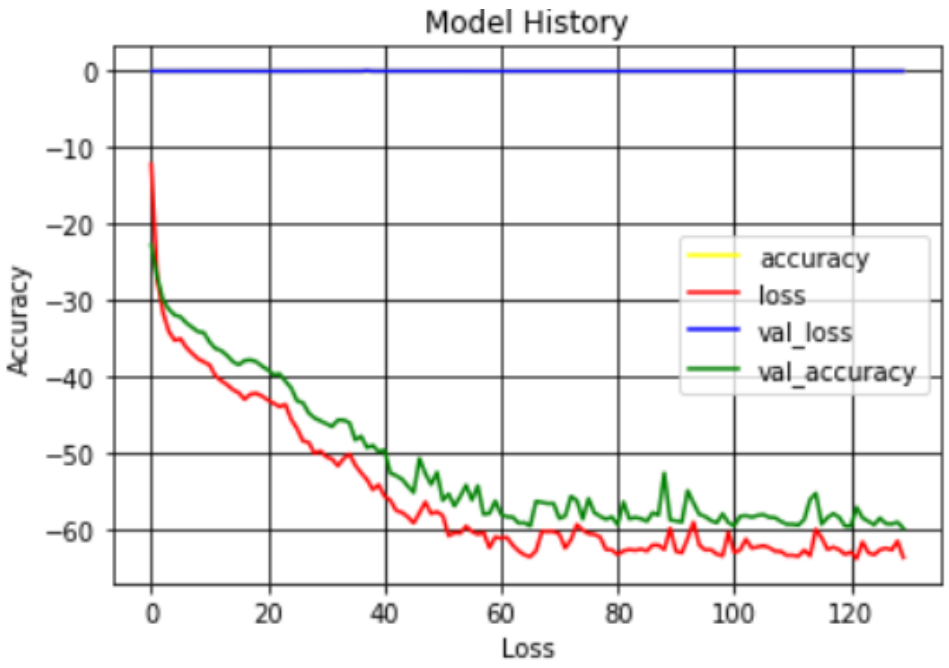


Figure 102 accuracy

In the above figure accuracy, loss, validation loss and validation accuracy is visualized. Graph shows the worst loss as it goes to negative. Accuracy is constant.

In the file chatlstm.ipynb, an inference function is created to communicate with the chatbot. Here the save model and word2vec model is loaded.

```
model=load_model('chatbotmodel.h5')
mod = gensim.models.Word2Vec.load('/home/rukesh/Documents/
```

```
WARNING:tensorflow:From /home/rukesh/.local/lib/python3.6/
rad.py:1250: add_dispatch_support.<locals>.wrapper (from te
ted and will be removed in a future version.
Instructions for updating:
```

Figure 103 loading model and vector

Here a chat function is created which takes input from the user, tokenizes it, searches for the vector in vocab, decodes it and gives the output. The output is worse from this model. Even hyper tuning cannot help to enhance its performance.

```
def chat():
    while True:
        x=input("Enter the message:")
        sentend=np.zeros((300),dtype=np.float32)

        sent=nltk.word_tokenize(x.lower())
        sentvec = [mod[w] for w in sent if w in mod.wv.vocab]

        sentvec[14:]=[]
        sentvec.append(sentend)
        if len(sentvec)<15:
            for i in range(15-len(sentvec)):
                sentvec.append(sentend)
        sentvec=np.array([sentvec])

        predictions = model.predict(sentvec)
        outputlist=[mod.most_similar([predictions[0][i]])[0][0] for i in rang
        output=' '.join(outputlist)
        print(output)
        if x == 'thank you':
            break
```

```
# import tkinter
chat()
```

```
Enter the message:hi
/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py:7: Depr
deprecated `__getitem__` (Method will be removed in 4.0.0, use self.wv.__geti
  import sys
/home/rukesh/.local/lib/python3.6/site-packages/ipykernel_launcher.py:17: Dep
o deprecated `most_similar` (Method will be removed in 4.0.0, use self.wv.mos
you that that can we we you you ? you you that that that that
Enter the message:namstae
you that that that that that that that that that that that that that that that
Enter the message:how are you?
you that that that that that that that that that that that that that that that
Enter the message:thank you
n't that that that that that that that that that that that that that that that
```

Figure 104 result of LSTM

4.3.3 Iterate-2 (Seq2Seq)

4.3.3.1 Data Pre Processing

As in the first iterative, the result was not good and so here data is preprocessed for seq2seq model architecture. First, the path of the csv data is given. The data separation steps are the same as discussed in iterative-1. Here vocab is created from the data. First, for loop is used to take all the questions and answers to the list. Second, a loop is used to add the ‘START' and ‘END’ tag to each and every question and answers. Obtain set is tokenized, unique index is given to each word and vocab is created. The vocab size is 1593.

```
dir_path = '/content/file.csv'

dataset = pd.read_csv(dir_path)
data1 = dataset.loc[:,['human']]
data2 = dataset.loc[:,['reply']]
questions = [sent_list[0] for sent_list in data1.values]
answers = [sent_list[0] for sent_list in data2.values]

answers_with_tags = list()
for i in range( len( answers ) ):
    if type( answers[i] ) == str:
        answers_with_tags.append( answers[i] )
    else:
        questions.pop( i )

answers = list()
for i in range( len( answers_with_tags ) ) :
    answers.append( '<START> ' + answers_with_tags[i] + ' <END>' )

tokenizer = preprocessing.text.Tokenizer()
tokenizer.fit_on_texts( questions + answers )
VOCAB_SIZE = len( tokenizer.word_index )+1
print( 'VOCAB SIZE : {}'.format( VOCAB_SIZE ))
```

```
VOCAB SIZE : 1593
```

Figure 105 creating vocab

The list with tag can be seen below.

```
answers
```

```
['<START> Namastae sir/mam. How can we help you? <END>',
 '<START> Please mention your username and the number sir/mam. <END>',
 '<START> How far are you from your router sir/mam? <END>',
 '<START> How many lights are glowing in your router? <END>',
 '<START> Can you please turn off the router and check after five minutes. <END>',
 '<START> welcome sir <END>',
 '<START> Hello sir/mam. How can we assist you? <END>',
 '<START> Sure sir/mam. The account number is 9801523053. <END>',
 '<START> Welcome and same to you sir/mam. <END>',
 '<START> Hello sir/mam. How can we assist you? <END>',
 '<START> Sure sir/mam. The account number is 9801523053. <END>',
 '<START> Welcome sir/mam. <END>',
 '<START> Hello sir/mam. The account number is 9801523053. <END>',
 '<START> Hello sir. How can we assist you? <END>',
```

Figure 106 list with tag

In this part custom word2vec is created. Only the word from vocab is extracted. Tokenize() function is created to tokenize each and every sentence received from the user. This function also filters all block and small letters. It returns a tokens list and the vocabulary. Questions and

Answers are concatenated, tokenized. With the gensim word2vec function all the tokenized word vectors are gathered and stored to model variables. Word2vec vocab size is 584

```
vocab = []
for word in tokenizer.word_index:
    vocab.append( word )

def tokenize( sentences ):
    tokens_list = []
    vocabulary = []
    for sentence in sentences:
        sentence = sentence.lower()
        sentence = re.sub( '[^a-zA-Z]', ' ', sentence )
        tokens = sentence.split()
        vocabulary += tokens
        tokens_list.append( tokens )
    return tokens_list , vocabulary

p = tokenize( questions + answers )
model = Word2Vec( p[ 0 ] )

embedding_matrix = np.zeros( ( VOCAB_SIZE , 100 ) )
```

Figure 107 custom word2vec

4.3.3.2 Predictive Modelling

Seq2seq is already discussed in detail at 3.2.3. Here the encoder input layer is created. Which tokenized the questions asked by the customer, get maximum length of the question, post pad the query, convert that pad sequence to numpy array and display the shape of encoder input layer and max length of query. The shape is (629,123) and max length is 123.

```
# encoder_input_data
tokenized_questions = tokenizer.texts_to_sequences( questions )
maxlen_questions = max( [ len(x) for x in tokenized_questions ] )
padded_questions = preprocessing.sequence.pad_sequences( tokenized_questions , maxlen=maxlen_questions , p
adding='post' )
encoder_input_data = np.array( padded_questions )
print( encoder_input_data.shape , maxlen_questions )
```

```
(629, 123) 123
```

Figure 108 encoder input

Decoder input data takes input as a result of the encoder input layer. It does the same things as discussed above. The core difference is it decodes the encoded input from the previous layer. It results of this layer is (629, 1393) with the max length of 1393.

```
# decoder_input_data
tokenized_answers = tokenizer.texts_to_sequences( answers )
maxlen_answers = max( [ len(x) for x in tokenized_answers ] )
padded_answers = preprocessing.sequence.pad_sequences( tokenized_answers , maxlen=maxlen_answers , padding
='post' )
decoder_input_data = np.array( padded_answers )
print( decoder_input_data.shape , maxlen_answers )
```

```
(629, 1393) 1393
```

Figure 109 decoder input

Decoder output data decode the responses. It tokenized all the responses, postpaid them, applied one hot encoding, converted one hot to numpy array and printed the shape of the results. The shape of the output is (629,1393) with a max length of 1593.

```
# decoder_output_data
tokenized_answers = tokenizer.texts_to_sequences( answers )
for i in range(len(tokenized_answers)) :
    tokenized_answers[i] = tokenized_answers[i][1:]
padded_answers = preprocessing.sequence.pad_sequences( tokenized_answers , maxlen=maxlen_answers , padding
='post' )
onehot_answers = utils.to_categorical( padded_answers , VOCAB_SIZE )
decoder_output_data = np.array( onehot_answers )
print( decoder_output_data.shape )
```

```
(629, 1393, 1593)
```

Figure 110 decoder output

All the layers are set which is necessary for developing the model. First the input layer is declared using tensorflow keras. Word embedding layer is defined with the shape of (vocab size,200). Encoder_inputs is passed as the input for this layer. Mask_zero helps to enhance the performance of a model as it returns zero where input is zero. This layer returns the vector of the words. LSTM layer is declared which is already discussed above.

For the decoder layer the same steps are followed as the encoder layer but in decoder output its output will be derived from encoder_sates of encoder layer and shape of decoder_embedding. In the last layer of the model softmax activation function is used which results in the shape of vocab_size. After describing the model architecture, it is compiled with optimizer RMSprop() as it finds suitable for the output as compared to others optimizer. Categorical_crossentropy loss function is used as the data is categorical in nature. Accuracy is passed as metrics to get the performance of the model while training.

```
encoder_inputs = tf.keras.layers.Input(shape=( None , ))
encoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero=True ) (encoder_inputs)
encoder_outputs , state_h , state_c = tf.keras.layers.LSTM( 200 , return_state=True )( encoder_embedding )
encoder_states = [ state_h , state_c ]

decoder_inputs = tf.keras.layers.Input(shape=( None ,  ))
decoder_embedding = tf.keras.layers.Embedding( VOCAB_SIZE, 200 , mask_zero=True) (decoder_inputs)
decoder_lstm = tf.keras.layers.LSTM( 200 , return_state=True , return_sequences=True )
decoder_outputs , _ , _ = decoder_lstm ( decoder_embedding , initial_state=encoder_states )
decoder_dense = tf.keras.layers.Dense( VOCAB_SIZE , activation=tf.keras.activations.softmax )
output = decoder_dense ( decoder_outputs )

model = tf.keras.models.Model([encoder_inputs, decoder_inputs], output )
model.compile(optimizer=tf.keras.optimizers.RMSprop(), loss='categorical_crossentropy',metrics=['accurac
y'])

model.summary()
```

Figure 111 creating model

Below the summary of model is illustrate.

```
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
Model: "model"
__________________________________________________________________________________________________
Layer (type)                    Output Shape         Param #     Connected to
==================================================================================================
input_1 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
input_2 (InputLayer)            [(None, None)]       0
__________________________________________________________________________________________________
embedding (Embedding)           (None, None, 200)    318600      input_1[0][0]
__________________________________________________________________________________________________
embedding_1 (Embedding)         (None, None, 200)    318600      input_2[0][0]
__________________________________________________________________________________________________
lstm (LSTM)                     [(None, 200), (None, 320800      embedding[0][0]
__________________________________________________________________________________________________
lstm_1 (LSTM)                   [(None, None, 200),  320800      embedding_1[0][0]
                                                                 lstm[0][1]
                                                                 lstm[0][2]
__________________________________________________________________________________________________
dense (Dense)                   (None, None, 1593)   320193      lstm_1[0][0]
==================================================================================================
Total params: 1,598,993
Trainable params: 1,598,993
Non-trainable params: 0
__________________________________________________________________________________________________
```

Figure 112 model summary

Google Colab is used to train the generative based model as it requires lots of memory and computational power while training. First the path of the training model is set then using the fit function model is trained to 125 epoch with batch size 128. All the model accuracy and the loss is stored in a history variable for visualization.

```
# Include the epoch in the file name (uses `str.format`)
checkpoint_path = "training_4/cp.ckpt"
# checkpoint_path = "training_3/cp-{epoch:04d}.ckpt"


checkpoint_dir = os.path.dirname(checkpoint_path)

# Create a callback that saves the model's weights every 1 epochs
cp_callback = tf.keras.callbacks.ModelCheckpoint(
    filepath=checkpoint_path,verbose=1)

# Create a new model instance
# model = create_model()

# Save the weights using the `checkpoint_path` format
model.save_weights(checkpoint_path)
```

```
history = model.fit([encoder_input_data , decoder_input_data], decoder_output_data, batch_size=128, epochs
=125,callbacks=[cp_callback] )
```

```
Train on 629 samples
Epoch 1/125
512/629 [=====================>......] - ETA: 4s - loss: 0.1761 - acc: 0.0705
Epoch 00001: saving model to training_4/cp.ckpt
629/629 [==============================] - 25s 39ms/sample - loss: 0.1721 - acc: 0.0675
Epoch 2/125
512/629 [=====================>......] - ETA: 3s - loss: 0.1456 - acc: 0.0298
Epoch 00002: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 34ms/sample - loss: 0.1481 - acc: 0.0304
Epoch 3/125
```

Figure 113 training model

After completing all the epoch the accuracy seems to be 0.7412 and loss is 0.0290 which is pretty good.

```
Epoch 124/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0294 - acc: 0.7303
Epoch 00124: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 34ms/sample - loss: 0.0293 - acc: 0.7388
Epoch 125/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0282 - acc: 0.7431
Epoch 00125: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0290 - acc: 0.7412

from keras.models import load_model model.load_weights(checkpoint_path)

model.summary()
```

Figure 114 accuracy and loss

In this iterate, to reconfirm the model, it is re-evaluate, this time the accuracy seems to be 72.55% with loss 3.04%.

```
# Re-evaluate the model
loss, acc  = model.evaluate([encoder_input_data , decoder_input_data], decoder_output_data,verbose=2)
print(" Accuracy: {:5.2f}%".format(100*acc))
print(" Loss: {:5.2f}%".format(100*loss))
```

```
629/629 - 21s - loss: 0.0304 - acc: 0.7255
 Accuracy: 72.55%
 Loss:  3.04%
```

Figure 115 evaluation

Overfit is also check by visualizing in graph. Matplotlib is used to plot the history of the graph.

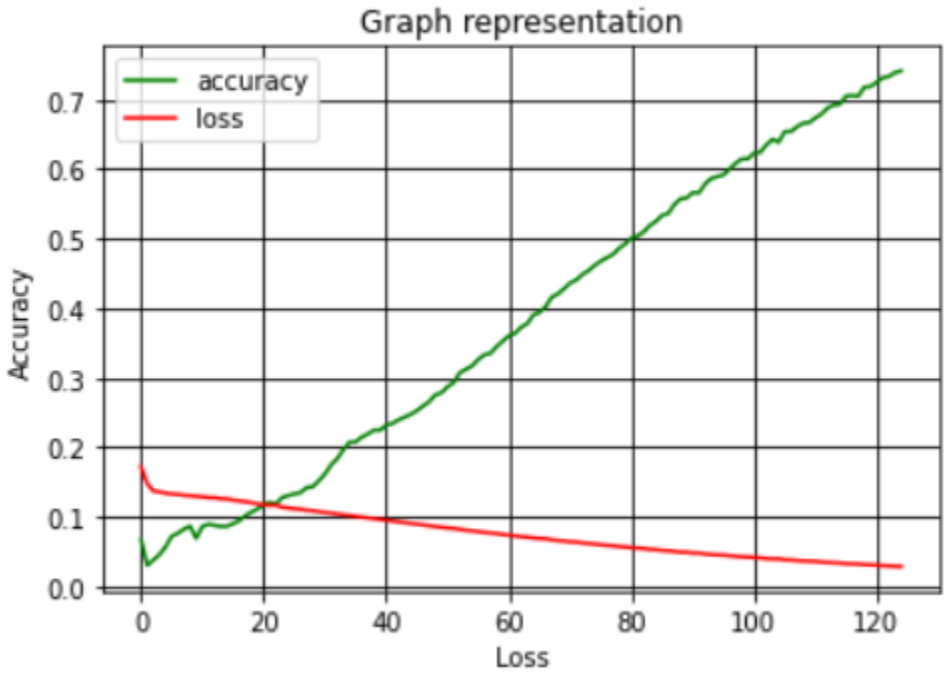


Figure 116 model history

While training the model, it does not seems to be over fit so none of the regularizations technique are used.

```
Epoch 105/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0426 - acc: 0.6396
Epoch 00105: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0399 - acc: 0.6386
Epoch 106/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0405 - acc: 0.6505
Epoch 00106: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0384 - acc: 0.6539
Epoch 107/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0392 - acc: 0.6515
Epoch 00107: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0383 - acc: 0.6540
Epoch 108/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0400 - acc: 0.6649
Epoch 00108: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 34ms/sample - loss: 0.0374 - acc: 0.6608
Epoch 109/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0375 - acc: 0.6586
Epoch 00109: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 34ms/sample - loss: 0.0369 - acc: 0.6660
Epoch 110/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0353 - acc: 0.6697
Epoch 00110: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0365 - acc: 0.6669
Epoch 111/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0396 - acc: 0.6706
Epoch 00111: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0363 - acc: 0.6735
Epoch 112/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0342 - acc: 0.6935
Epoch 00112: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 34ms/sample - loss: 0.0352 - acc: 0.6792
Epoch 113/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0324 - acc: 0.6761
Epoch 00113: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0347 - acc: 0.6879
Epoch 114/125
512/629 [=====================>......] - ETA: 3s - loss: 0.0336 - acc: 0.6846
Epoch 00114: saving model to training_4/cp.ckpt
629/629 [==============================] - 21s 33ms/sample - loss: 0.0343 - acc: 0.6921
Epoch 115/125
```

Figure 117 checking overfitting

Now the inference model is created to encode and decode queries and concerned responses. This model is developed using the model that was fit and the different layer as discussed above.

```
def make_inference_models():

    encoder_model = tf.keras.models.Model(encoder_inputs, encoder_states)

    decoder_state_input_h = tf.keras.layers.Input(shape=( 200 ,))
    decoder_state_input_c = tf.keras.layers.Input(shape=( 200 ,))

    decoder_states_inputs = [decoder_state_input_h, decoder_state_input_c]

    decoder_outputs, state_h, state_c = decoder_lstm(
        decoder_embedding , initial_state=decoder_states_inputs)
    decoder_states = [state_h, state_c]
    decoder_outputs = decoder_dense(decoder_outputs)
    decoder_model = tf.keras.models.Model(
        [decoder_inputs] + decoder_states_inputs,
        [decoder_outputs] + decoder_states)

    return encoder_model , decoder_model
```

Figure 118 inference model

Inference model return output in numpy array. It should be converted to tokens and pad them. This process is done by str_to_token function.

```
def str_to_tokens( sentence : str ):
    words = sentence.lower().split()
    tokens_list = list()

    for word in words:
        if word in vocab:
            tokens_list.append( tokenizer.word_index[ word ] )
        else:
            print(" ")
    return preprocessing.sequence.pad_sequences( [tokens_list] , maxlen=maxlen_questions , padding='post')
```

Figure 119 convert token

All the essential layer models are developed now it should be used to take query from customers and generate the responses. Chat function is created which goes to an infinite loop if and only if the word ‘quit’ is not typed by the user. This function takes the input, process through model, inference model decode the token and print responses on console.

```
def chat():
    enc_model , dec_model = make_inference_models()

    while True:
        take = input( '> ' )

        states_values = enc_model.predict( str_to_tokens( take ) )
        if take == 'quit':
            break

        empty_target_seq = np.zeros( ( 1 , 1 ) )
        empty_target_seq[0, 0] = tokenizer.word_index['start']
        stop_condition = False
        decoded_translation = ''
        while not stop_condition :
            dec_outputs , h , c = dec_model.predict([ empty_target_seq ] + states_values )
            sampled_word_index = np.argmax( dec_outputs[0, -1, :] )
            sampled_word = None
            for word , index in tokenizer.word_index.items() :
                if sampled_word_index == index :
                    decoded_translation += ' {}'.format( word )
                    sampled_word = word

            if sampled_word == 'end' or len(decoded_translation.split()) > maxlen_answers:
                stop_condition = True

            empty_target_seq = np.zeros( ( 1 , 1 ) )
            empty_target_seq[ 0 , 0 ] = sampled_word_index
            states_values = [ h , c ]

        print( decoded_translation )

chat()
```

Figure 120 taking input

Result

```
Tensor("input_9:0", shape=(?, 200), dtype=float32)
> hello
 hello sir mam how can we assist you end
> Namastae
 namastae sir mam how can we assist you end
> Please provide me sales number.
Pardon !!!
 do you have any query end
> please provide me sales number
 sure sir the sales number is 9801523052 end
> thank you
 welcome sir mam end
> okay
 yes sir mam end
> quit
Pardon !!!
```

Figure 121 responses

The result seems to be better than iterative-1 model. This result can be more accurate in hybrid approach.

4.3.4 Iterate-3 (Retrieval-tflearn)

4.3.4.1 Data Pre Processing

For the retrieval based chatbot, JSON data is used. Data is read with python and create different list.

```
import json
with open('/home/rukesh/Documents/final_year/FYP/Retrieval_based_chatbot/json_only/all_data.json') as file:
    data1 = json.load(file)
```

```
words1 = list()
labels1 = list()
docs_x1 = list()
docs_y1 = list()


words = list()
labels = list()
docs_x = list()
docs_y = list()
```

Figure 122 reading json file

From the dataset intents, patterns, tags, labels are extracted and tokenized the word.

```
for intent in data1['intents']:
    for pattern in intent['patterns']:
        wrds1 = nltk.word_tokenize(pattern)
        words1.extend(wrds1)
        docs_x1.append(wrds1)
        docs_y1.append(intent["tag"])

    if intent['tag'] not in labels1:
        labels1.append(intent['tag'])

words.extend(words1)
labels.extend(labels1)
docs_x.extend(docs_x1)
docs_y.extend(docs_y1)
```

```
print(words)
```

```
['i', 'want', 'to', 'hide', 'my', 'wifi', 'ssid', 'bro',
'.', 'could', 'you', 'guide', 'me', 'to', 'hide', 'the',
'wifi', 'password', '?', 'thank', 'you', 'for', 'the', 'c
oorporation', '.', 'okay', 'you', 'really', 'suprise', 'm
e', '.', 'you', 'are', 'such', 'an', 'insanse', '.', '🤣
🤣🤣🤣', 'thank', 'you', 'for', 'your', 'coorporation',
'and', 'guidance', '.', 'ummm', 'well', 'its', 'working',
'okay', 'let', 'me', 'check', 'thank', 'you', 'brother',
```

Figure 123 tokenize word

Here, printing the labels that have been extracted from the dataset.

```
# len(words)
len(labels)
labels
```

```
['changepasswordfrommobile',
 'thankyou',
 'slowconnection',
 'secondaryrouter',
 'safenet',
 'portforward',
 'onlineservices',
 'offensive',
 'nointernetconnection',
 'nettv',
 'goodmorning',
 'hosting',
 'greeting',
 'gamingwinprice',
 'onlinegaming',
 'fup',
 'farfromrouter',
 'goodevening',
 'compensation',
 'afternoon',
 'account']
```

Figure 124 labels

Words stemming is done with the function stemmer.stem(). It is used to extract essential information from big data. It is the process of reducing the word to its root form defined as lemma. Words are sorted with built-in function sorted.

```
words = [stemmer.stem(w.lower()) for w in words if w != "?"]
words = sorted(list(set(words)))
print(words)
```

```
['%', "'", ',', '.', '100', '11', '23', '5', '7', '8',
'```', 'a', 'abl', 'about', 'abov', 'access', 'account',
'act', 'ad', 'aft', 'afternoon', 'al', 'also', 'altern',
'am', 'amount', 'an', 'and', 'anoth', 'any', 'anyon', 'ap
olog', 'app/application', 'application/app', 'apply', 'a
r', 'as', 'at', 'avoid', 'back', 'be', 'been', 'bhast',
'bitch', 'black', 'block', 'boro', 'branch', 'bro', 'brot
h', 'brow', 'brows', 'buddy', 'but', 'button', 'ca', 'cab
l', 'cal', 'cam', 'can', "cant't", 'catch', 'certain', 'c
hang', 'channel', 'check', 'claim', 'cli', 'col', 'com',
'compens', 'complet', 'concern', 'connect', 'cont', 'cont
act', 'contain', 'context', 'coorp', 'could', 'cred', 'cs
go', 'cur', 'custom', 'cut', 'dam', 'day', 'decl', 'depar
t', 'detail', 'dev', 'develop', 'diagnos', 'did', 'diff',
'do', 'docu', 'doe', 'don', 'down', 'downloa', 'downloa
d', 'due', 'dump', 'eat', 'educ', 'elab', 'elig', 'esc',
```

Figure 125 stemming

Labels are stored in list using list comprehensive and printing it.

```
labels = sorted(labels)
out_empty = [0 for _ in range(len(labels))]
print(labels)
```

```
['account', 'afternoon', 'changepasswordfrommobile', 'com
pensation', 'farfromrouter', 'fup', 'gamingwinprice', 'go
odevening', 'goodmorning', 'greeting', 'hosting', 'nett
v', 'nointernetconnection', 'offensive', 'onlinegaming',
'onlineservices', 'portforward', 'safenet', 'secondaryrou
ter', 'slowconnection', 'thankyou']
```

Figure 126 list comprehensive

Two lists are created which store training and output numpy arrays further they are used in training the model, a unique index is given to every word and data is printed.

```
training =list()
output = list()
for x, doc in enumerate(docs_x):
    bag = list()
    wrds = [stemmer.stem(w.lower()) for w in doc]
#     print(wrds)
    for w in words:
        if w in wrds:
            bag.append(1)
        else:
            bag.append(0)
    output_row = out_empty[:]
    print(docs_y)
    output_row[labels.index(docs_y[x])] = 1
    training.append(bag)
    output.append(output_row)
training = numpy.array(training)
output = numpy.array(output)
```

```
['changepasswordfrommobile', 'changepasswordfrommobile',
'thankyou', 'thankyou', 'thankyou', 'thankyou', 'thankyo
u', 'thankyou', 'thankyou', 'thankyou', 'thankyou', 'than
kyou', 'thankyou', 'thankyou', 'thankyou', 'thankyou', 't
hankyou', 'thankyou', 'slowconnection', 'slowconnection',
'slowconnection', 'slowconnection', 'slowconnection', 'sl
owconnection', 'slowconnection', 'slowconnection', 'slowc
onnection', 'slowconnection', 'slowconnection', 'slowconn
```

Figure 127 indexing

4.3.4.2 Predictive Modelling

Here, the model is created using tflearn from tensorflow. First the graph is reset to default to apply session reset in the current thread. The input layer takes the size of training data. The first and second are the fully connected layers with 8 neurons. Softmax is used as the activation function with adam optimization.

```
tensorflow.reset_default_graph()
net = tflearn.input_data(shape=[None, len(training[0])])
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, 8)
net = tflearn.fully_connected(net, len(output[0]), activa
tion="softmax")
net = tflearn.regression(net)
model = tflearn.DNN(net)
```

```
WARNING:tensorflow:From /home/rukesh/.local/lib/python3.
```

Figure 128 tflearn model

The model is fit using fit function with epoch 2000, batch_size 16 and show_matric = True to get the accuracy of the model. The model is saved in h5 format.

```
history = model.fit(training, output, n_epoch=2000, batch
_size=16, show_metric=True)
model.save("model.h5")
```

```
Training Step: 25999  | total loss: 0.19475 | time: 0.022
s
| Adam | epoch: 2000 | loss: 0.19475 - acc: 0.9872 -- ite
r: 192/198
Training Step: 26000  | total loss: 0.18040 | time: 0.024
s
| Adam | epoch: 2000 | loss: 0.18040 - acc: 0.9885 -- ite
r: 198/198
--
```

Figure 129 training tflearn

While training the accuracy seems to 0.9885 with loss 0.18040.

Bag-of-words (BOW) is the important technique used in NLP. It is an approach of feature extraction from text data. Here, BOW function is defined which extracts all the words from the sentences, tokenizes it, provides unique indexes and returns a numpy array of features.

```
def bag_of_words(s, words):
    bag = [0 for _ in range(len(words))]

    s_words = nltk.word_tokenize(s)
    s_words = [stemmer.stem(word.lower()) for word in s_w
ords]

    for se in s_words:
        for i, w in enumerate(words):
            if w == se:
                bag[i] = 1

    return numpy.array(bag)
```

Figure 130 BOW

Now all the functions and essential techniques are implemented for chatting with chatbot. The main function to establish chat is to be developed. This function takes input from the user, predicted with BOW, numpy argmax() function is used to denote the classes from huge predicted probability.

```
def chat():
    print("Start talking with the bot (type quit to sto
p)!")
    while True:
        inp = input("Questions:  ")
        if inp.lower() == "quit":
            break

        results = model.predict([bag_of_words(inp, word
s)])
        results_index = numpy.argmax(results)
        tag = labels[results_index]

        for tg in data1["intents"]:
            if tg['tag'] == tag:
                responses = tg['responses']

        print(random.choice(responses))
```

```
: chat()
```

Figure 131 chat

In the below, result of retrieval chatbot is illustrate.

```
Start talking with the bot (type quit to stop)!
Questions:  hi
hello sir/mam. how can we assist you?
Questions:  tero tauko
if you continue using such offensive and vulgar words to
seek our assistance. we will be compelled to blacklist yo
u from online support services. thank you!
Questions:  FUP
users who abuse the internet for immense download/upload
uncontrollably, massive sharing & inappropriate package s
ubscription.
Questions:  FUP
in worldlink we have “smart fup” system which monitors ou
r user’s behaviour, when a user exceeds daily data thresh
old, flags will be tagged automatically and on the basis
of flags tagged our system automatically restricts those
abusers and keeps overall experience balanced for other n
ormal users.
Questions:  i want new connection
well it can be done but only from the web application. it
can be block from the mobile application.
```

Figure 132 result

4.3.5 Iterate-4 (Retrieval-keras)

4.3.5.1 Data Pre Processing

The steps are the same as mentioned in iterate-3 but some of the techniques are different. Here, lemmatize is used instead of stemmer. It is similar to stemmer but lemmatize performs morphological analysis and it hookup similar words. Pickle is used to dump the classes and the words.

```
# lemmaztize and lower each word and remove duplicates
words = [lemmatizer.lemmatize(w.lower()) for w in words i
f w not in ignore_words]
words = sorted(list(set(words)))
# sort classes
classes = sorted(list(set(classes)))
# documents = combination between patterns and intents
print(len(documents), "documents")
# classes = intents
print(len(classes), "classes", classes)
# words = all words, vocabulary
print(len(words), "unique lemmatized words", words)

pickle.dump(words, open('words.pkl', 'wb'))
pickle.dump(classes, open('classes.pkl', 'wb'))
```

```
199 documents
21 classes ['account', 'afternoon', 'changepasswordfrommo
bile', 'compensation', 'farfromrouter', 'fup', 'gamingwin
price', 'goodevening', 'goodmorning', 'greeting', 'hostin
```

Figure 133 lemmatiz

4.3.5.2 Predictive Modelling

For developing the model, first, a sequential instance is created, first layer is added with neuron 128, activation function relu. Dropout regularization is used to overcome the issue of overfitting. Second layer has 64 neurons, the third layer consists of training numbers of neurons with softmax activation function.

```
# Create model - 3 layers. First layer 128 neurons, secon
d layer 64 neurons and 3rd output layer contains number o
f neurons
# equal to number of intents to predict output intent wit
h softmax
model = Sequential()
model.add(Dense(128, input_shape=(len(train_x[0]),), acti
vation='relu'))
model.add(Dropout(0.5))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(train_y[0]), activation='softmax'))
```

Figure 134 keras model

For compiling the model Stochastic Gradient Descent (SGD) is used with learning rate 0.01 and momentum 0.9 and categorical_crossentropy loss function is used. The model is trained with epoch 700, batch_size 8.

```
try:
    model = load_model('model.h5')
    print("model load successfully")
except:
# Compile model. Stochastic gradient descent with Nesterov accelerated gradient gives good results for thi
s model

    sgd = SGD(lr=0.01, decay=1e-6, momentum=0.9, nesterov=True)
    model.compile(loss='categorical_crossentropy', optimizer=sgd, metrics=['accuracy'])

    # fitting and saving the model
    history = model.fit(np.array(train_x), np.array(train_y), epochs=700, batch_size=8, verbose=1)
    model.save('model.h5', history)

    print("model created")
```

Figure 135 model compile

Below figure shows the summary of the model.

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 128)               31616
_________________________________________________________________
dropout_1 (Dropout)          (None, 128)               0
_________________________________________________________________
dense_2 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_2 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_3 (Dense)              (None, 34)                2210
=================================================================
Total params: 42,082
Trainable params: 42,082
Non-trainable params: 0
```

Figure 136 keras model summary

Here the history of a model is plotted in graph with the help of matplotlib library. First, the history of a model is printed and plotted in graphs.

```
print(history.history.keys())
```

```
dict_keys(['loss', 'accuracy'])
```

```
import matplotlib.pyplot as plt
plt.plot(history.history['accuracy'],color='blue',label='accuracy')
plt.plot(history.history['loss'],color='red',label='loss')

plt.title('Model History')
plt.xlabel('Loss')
plt.ylabel('Accuracy')
plt.grid(True,color='k')
plt.legend(['accuracy', 'loss'])
plt.show()
```

Figure 137 matplotlib code

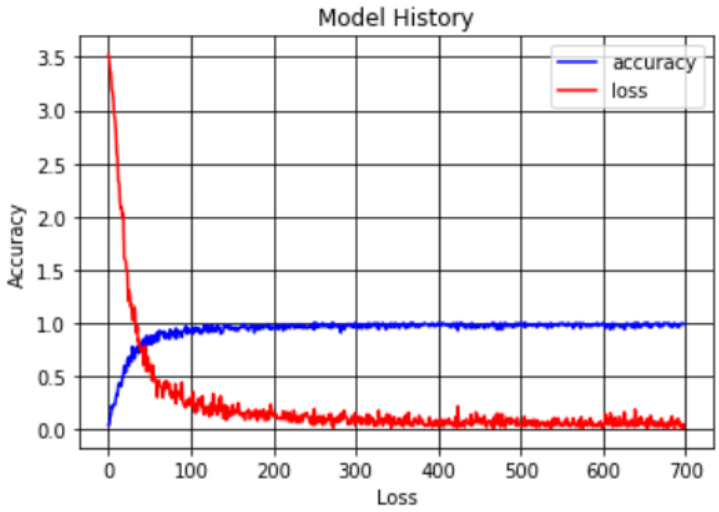


Figure 138 model history

In the above figure, accuracy is increasing slowly and loss is decreasing.

The process of taking a query and providing responses is similar to iterate-3 but here one clean_up_sentences() function is added to tokenize and lemmatize the query which returns the BOW array.

```
def clean_up_sentence(sentence):
    # tokenize the pattern - split words into array
    sentence_words = nltk.word_tokenize(sentence)
    # stem each word - create short form for word
    sentence_words = [lemmatizer.lemmatize(word.lower())
for word in sentence_words]
    return sentence_words

# return bag of words array: 0 or 1 for each word in the
bag that exists in the sentence
```

Figure 139 new function

Below there is the result of keras retrieval chatbot, it seems to be accurate than tflearn model.

```
> - hello
hello sir/mam. how can we assist you?
> - FUP
sir, fup is right that every isp reserve. we are sure soo
ner or later it is going to be implemented by any isps ac
cording to their needs and requirements. so on with the i
ncreasing number of clients in worldlink we intend to pro
vide the best possible internet experience to all our cus
tomers. for this, we rely on customers being fair in the
way they use the internet. this "fair usage policy" is he
re to give us an option that if usage of any unlimited cu
stomer is excessive over a period of time to the point, i
t impacts on other users experience then the bandwidth wi
ll be capped to fallback speed and is effective only duri
ng peak hours (8pm to 11pm) whereas the speed remain unch
anged beside mentioned timing.all of our valued clients m
ust be able to use equal service. here, fup is an option
```

Figure 140 keras result

4.3.6 Iterate-5 (Hybrid)

4.3.6.1 Data Pre Processing

All the preprocessing is already done in the above section.

4.3.6.2 Predictive Modelling

For the hybrid purpose, both the systems are combined. If else statement is used with probability. Retrieval based gives responses with the probability. Certain probability threshold is used to pass the sequences from retrieval to generative.

```
def chat():
    while True:
        inp = input("> - - - ")
        if inp.lower() == "quit":
            break
        results = predict_class(sentence=inp)

        results_index = np.array(results)
        confidence = results_index[1]
        # print(confidence)
        # print(type(results_index))
        co = (confidence.astype('float64'))
        print('co = ', co)
        print(type(co))
        val = np.float32(co)
        pyval = val.item()
        print(pyval)

        if pyval > 0.6:


            tag = results_index[0]

            list_of_intents = intents['intents']
            for i in list_of_intents:
                if (i['tag'] == tag):
                    result = random.choice(i['responses'])
                    break
            print (result)
        else:
            gene(inp)
```

Figure 141 hybrid chatbot

In the above figure confidence holds the probability of words predicted, using the assignment operator probability greater than 0.6 is handled by retrieval based chatbot and less than 0.6 is handled by generative based chatbot.

```
> - - - FUP
co =  [0.99998856]
<class 'numpy.ndarray'>
0.9999885559082031
please elaborte your query also mind your language.
> - - - Please tell me about FUP
co =  [1.]
<class 'numpy.ndarray'>
1.0
users who abuse the internet for immense download/upload uncontrollably, massive sharing & inappropriate package subscription.
> - - - torrent
co =  [0.7892914]
<class 'numpy.ndarray'>
0.7892913818359375
Tensor("input_5:0", shape=(None, 200), dtype=float32)
 if you have the issue as soon as possible sir mam end
> - - - online services password
co =  [0.99814296]
<class 'numpy.ndarray'>
0.9981429576873779
to reset your online services password to reset your services password go to www.worldlink.com.np click on “online services” (eservice.worldlink.com.np/login/) c
 (eservice.worldlink.com.np/recover/forgot_password) enter your username and email address/mobile number registered to the username.
> - - - how can i reset my online services password.
```

Figure 142 keras hybrid result

Above figure is the result of hybrid chatbot development using keras retrieval bot. Probability of word occurrence can easily be seen in the figure. The sentences without end at the bottom are responses by retrieval chatbot and sentences with end tag at bottom are responses from generative chatbot.

```
how can i assist you?
You: please provide me account number
(1, 21)
confidence :  0.8683416
sure sir/mam. the account number is 9801523053.
You: please provide me sales number
(1, 21)
confidence :  0.36032486
Tensor("input_3:0", shape=(?, 200), dtype=float32)
 sure sir the sales number is 9801523052 end
You: jjyg
(1, 21)
confidence :  0.4170699
Tensor("input_5:0", shape=(?, 200), dtype=float32)
Pardon !!!
 if you continue using such offensive and vulgar words to seek our assistance we will be compelled to blacklist you from online support services thank you end
You: fuk
(1, 21)
confidence :  0.4170699
Tensor("input_7:0", shape=(?, 200), dtype=float32)
Pardon !!!
 if you continue using such offensive and vulgar words to seek our assistance we will be compelled to blacklist you from online support services thank you end
```

Figure 143 tflearn hybrid result

From the above analysis it seems that tflearn hybrid chatbot is able to provide better responses to query than keras hybrid. The accuracy of keras hybrid is good but in overall tflearn has proof to be effective.

4.3.7 Testing

4.3.7.1 Integration Testing

This testing combine all the small chunks and test the system.

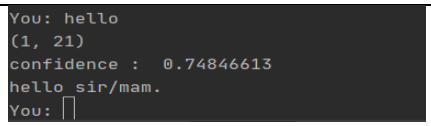
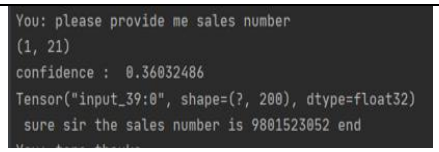
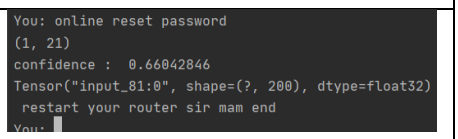
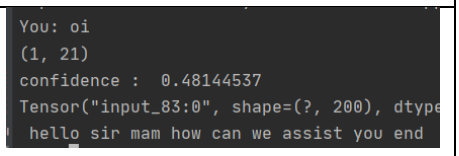
Integration Testing				
Query	Responses	Expect	Result	Screen shot
hello	hello sir/mam	Hello sir/mam	pass	
Please provide me sales number	Sure sir the number is 9801523052 end	Sure sir/mam the number is 9801523052	pass	
Online reset password	Restart your router sir mam	Sorry sir/mam	fail	
Oi	Hello sir mam how can we assist you end	Generate responses	pass	

Table 6 integration testing

4.3.7.2 System Testing

This testing check if the system is properly working or not after deployment.

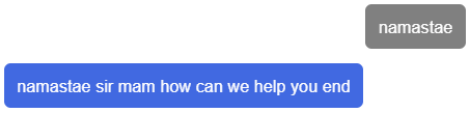
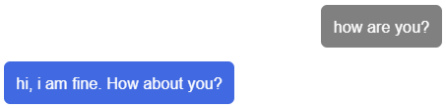
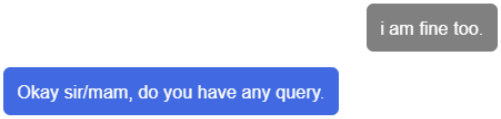
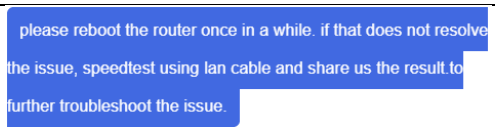
System Testing				
Query	Responses	Expect	Result	Screen shot
Hi	Hi	Hi	pass	
Namastae	Namastae sir mam how can we assist you end	Namastae sir mam (generate sequence)	pass	
How are you?	Hi, I am fine. How about you?	I am fine how about you?	pass	
I am fine too.	Okay sir/mam, do you have any query.	Okay sir/mam, do you have any query?	pass	
I am facing slow connection.	Please reboot the router one in a while……	Please reboot the router.	pass	

Table 7 System testing

5 Academic Questions

The motive of this purpose system is to solve minor issues of online support. This system is capable of responding to vulgar words, greetings, different department numbers, guide to reset online services password, new connection details, renew connection etc.

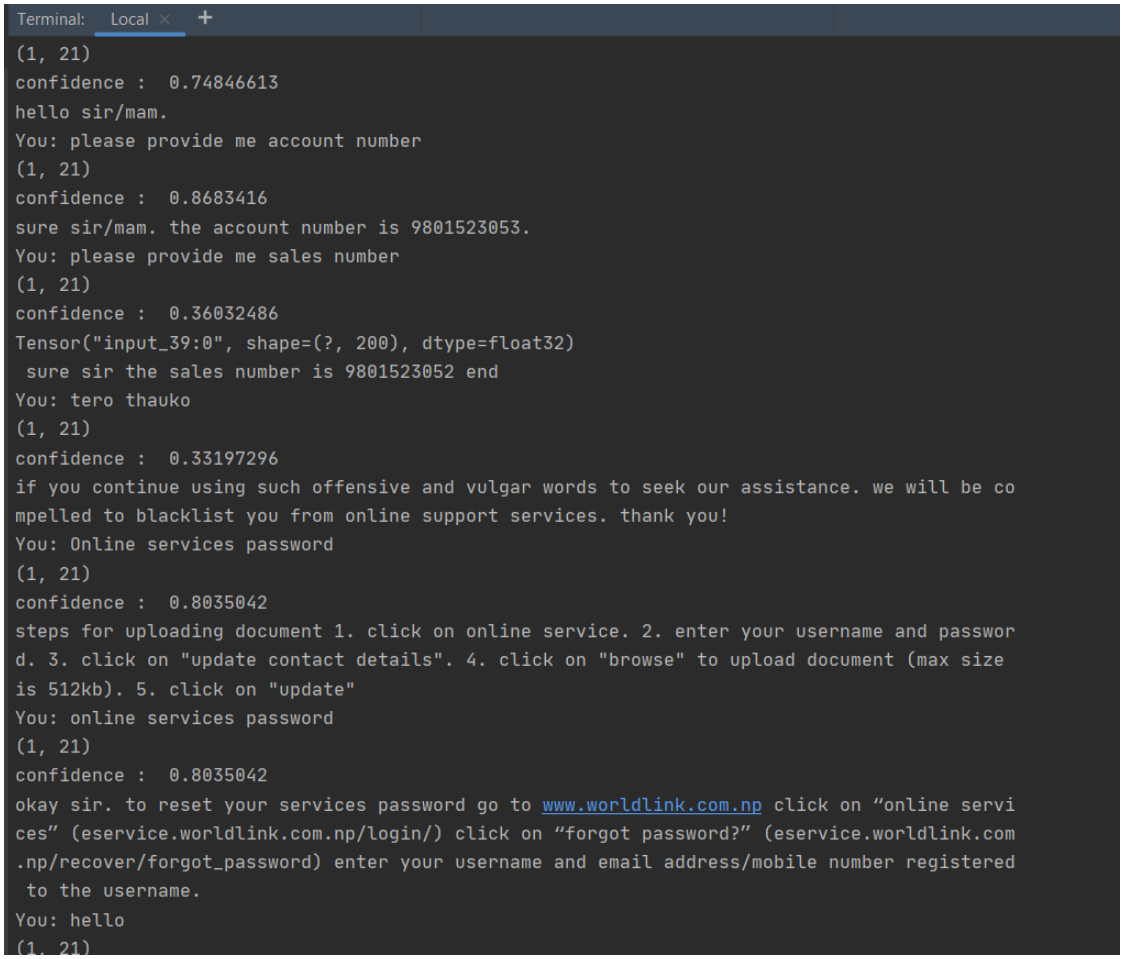


Figure 144 results

This is the NLP project. First query is obtained from the user, preprocess it, passed to model, find the tag, classes or understand the query, predict the word, decode the array, find the probability and render the output to users.

Note:

All the code of project is in appendix 10.1.

6 Conclusion

Online support chatbot is mainly developed for ISP online support teams. Various queries are asked by customers, some of them become recast, simple and major. With the ambition to solve minor issues in online support, this chatbot has been developed. This system seems impressive to handle queries. Hence, the developing system is capable of solving minor issues.

The object was to develop generative chatbot but due to accuracy of responses and to maintain the statute of chosen process model, hybrid chatbot is developed. Hybrid chatbot seems to be more accurate and efficient while handling queries. Results of the system are directly dependent on data and preprocessing steps. Windows maintains the memory while Linux fails to handle it. For developing the AI model, it requires lots of computational power. Using Graphical Processing Unit (GPU) can be an efficient method.

While developing this project, the meaning of chatbot, its types, how different chatbot perform activities and how it develops is clearly inferred. Many errors and issues are faced, supervisor, reader and info dev teacher guide to solve particular issues. Second, stack overflow, different online articles, YouTube videos helps to solve particular bugs. Guidance of the supervisor is at top while writing the report

7 Critical Evaluation

7.1 Final report

Due to word limitation, lots of technical and elaboration is summaries. If the word limit was 8000 to 16000. The report would be fabulous.

7.2 Artifact

Developing the web application does not lead to major issues but while deploying chatbot in web application, versioning issues were found such as tflearn is supported by TF V1.14.0 only but in Django TF V2.1.0 was only supported. Hence to solve this issue Keras retrieval based chatbot is developed.

7.3 Finding

For developing the chatbot lots of data are needed. Generative based chatbot is mainly developed for open domain whereas for the close domain retrieval based chatbot seems to be effective. Data seems to be important for developing the model. Accuracy and efficiency depends upon 80 % of data, 20 % of model architecture and problem solving technique. While deploying the system in web application, windows seems to be more efficiency in managing memory than linux distribution.

The window maintains its memory after leaving the device for an ideal, after 30 minutes whereas linux memory does not maintain.

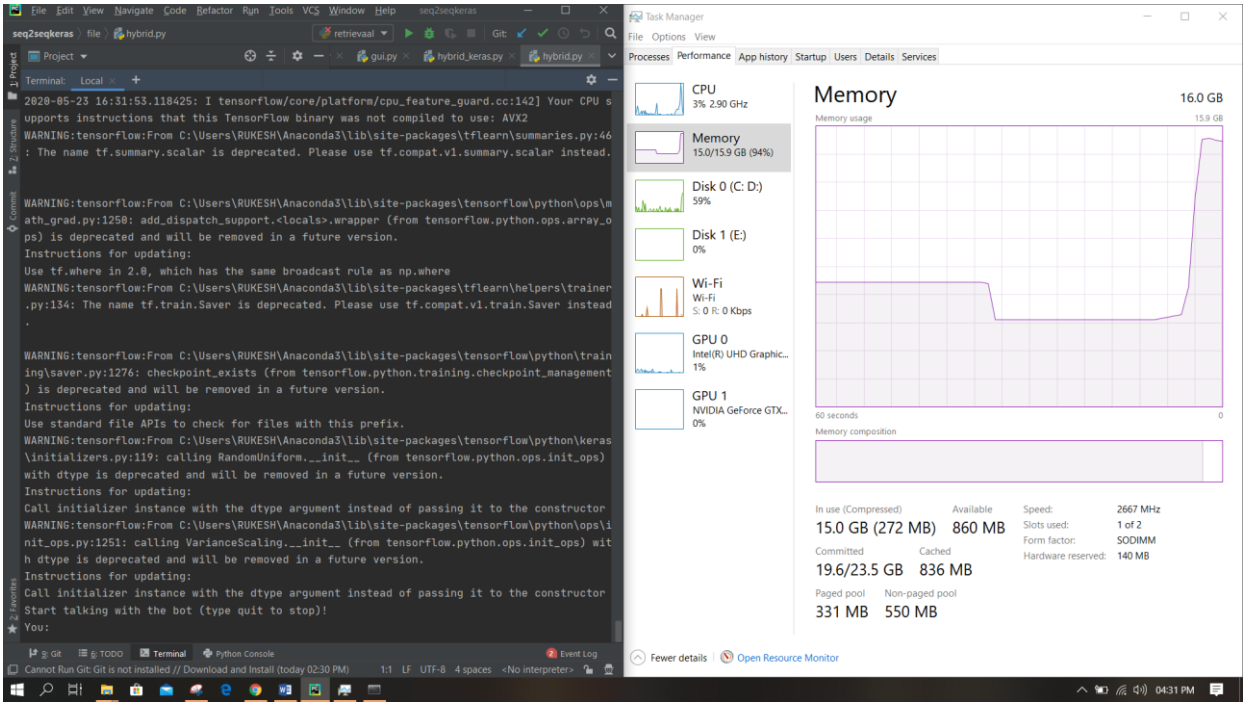


Figure 145 high memory

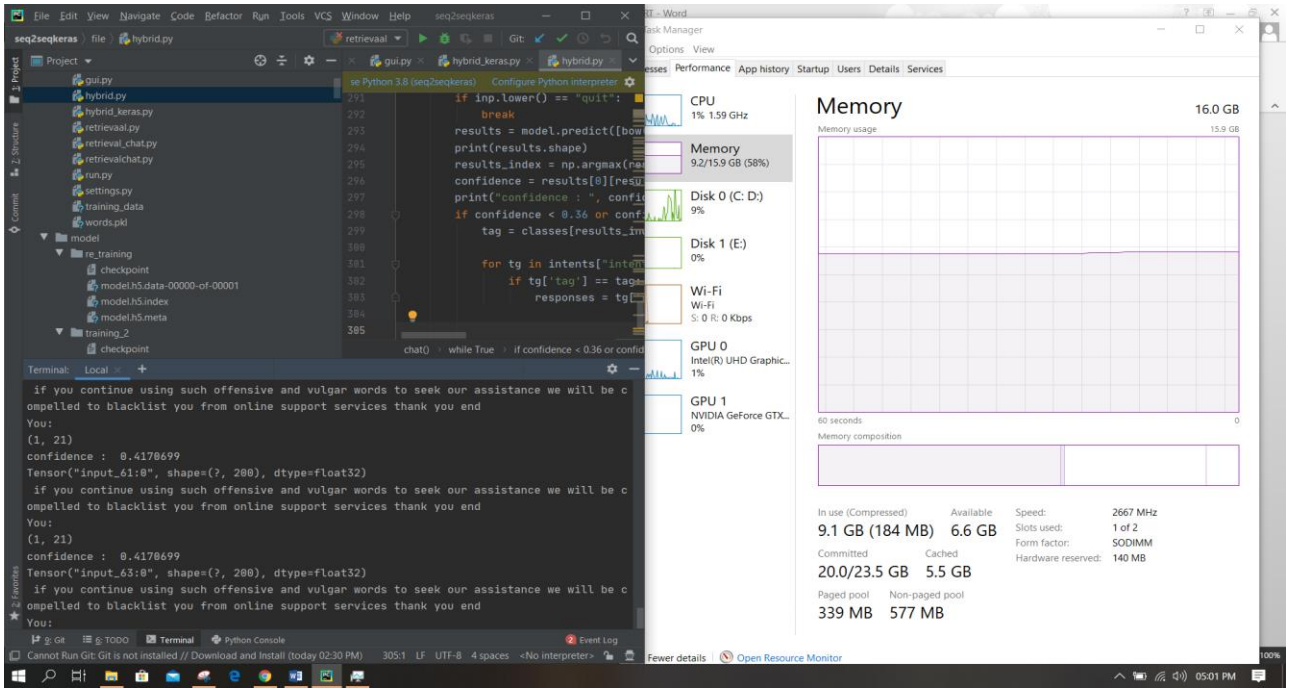


Figure 146 window after 30 minutes

7.4 Process itself

Due to the iterative and evolutionary model, this system is able to develop. As compare to other process model, this model try to improve the system in each and every iterative. Due to key features of repetitive development, new features and techniques are able to be implemented in this project at each iterative. To maintain statute of process model, more research and effort is implement in the project.

8 References

Abdul-Kader, S. A. & Woods, D. J., 2015. Survey on Chatbot Design Techniques in Speech Conversation Systems. *(IJACSA) International Journal of Advanced Computer Science and Applications.,* 6(7), pp. 72-80.

Bergum, V. A. & Iveland, T., 2019. *Norwegian Generative Chatbot using Personal Chat Log Data,* Trondheim: TDT4310 - Intelligent Text Analytics and Language Understanding Norwegian University of Science and Technology .

Bitnami, 2019. *GitLab CE Certified by Bitnami.* [Online] Available at: https://console.cloud.google.com/marketplace/details/bitnami-launchpad/gitlab?utm_source=google&utm_medium=cpc&utm_campaign=japac-APAC-all-en-dr-skws-all-all-trial-b-dr-1003987&utm_content=text-ad-none-none-DEV_c-CRE_256078342681-ADGP_Hybrid%20%7C%20AW%20
[Accessed 3 12 2019].

Brownlee, J., 2017. *Long Short-Term Memory Networks With Python.* v1.0 ed. s.l.:s.n.

chatbots.org, 2020. *Chatbot.* [Online] Available at: https://www.chatbots.org/chatbot/
[Accessed 21 04 2020].

Chen, J., 2019. *Neural Network.* [Online] Available at: https://www.investopedia.com/terms/n/neuralnetwork.asp
[Accessed 08 05 2020].

Chisty, J., Halder, R. & Rabbi, M. K. P., 2018. *AN ARTIFICIAL CHATBOT,* Dhaka: Department of Computer Science and Engineering.

Cho, K. et al., 2014. *Learning Phrase Representations using RNN Encoder–Decoder for Statistical Machine Translaction,* s.l.: s.n.

Chowanda, A. & Chowanda, A. D., 2018. *Generative Indonesian Conversation Model using Recurrent Neural.* Syahdan No. 9, Jakarta 11480, 3rd International Conference on Computer Science and Computational Intelligence 2018.

Copeland, L., 2003. *State-Transition Diagrams.* [Online] Available at: https://www.stickyminds.com/article/state-transition-diagrams [Accessed 16 05 2020].

geeksforgeeks.org , 2020. *GeeksforGeeks.* [Online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-sequence-diagrams/ [Accessed 15 05 2020].

geeksforgeeks.org, 2020. *GeeksforGeeks.* [Online] Available at: https://www.geeksforgeeks.org/unified-modeling-language-uml-activity-diagrams/ [Accessed 15 05 2020].

Ghahrai, A., 2018. *Testing Excellence.* [Online] Available at: https://www.testingexcellence.com/iterative-model/ [Accessed 05 12 2019].

Greff, K., Koutn´ık, R. K. S. J. & Schmidhuber, B. R. S. J., 2017. *LSTM: A Search Space Odyssey ,* s.l.: TRANSACTIONS ON NEURAL NETWORKS AND LEARNING SYSTEMS.

Heller, M., 2018. *What is CUDA? Parallel programming for GPUs.* [Online] Available at: https://www.infoworld.com/article/3299703/what-is-cuda-parallel-programming-for-gpus.html [Accessed 07 05 2020].

IrynaKulatska, 2019. *ArgueBot: Enablingdebatesthroughahybrid retrieval-generation-basedchatbot.* AEEnschede, UniversityofTwente.

Jain, S., 2017. *Chat-bot using RNN.* [Sound Recording] (Bennett University (Youtube: "https://www.youtube.com/watch?v=McKQ0He69PI")).

Jha, A. K., 2019. Journey to the Realm of Chatbots. *International Journal of Research in Engineering, Science and Management,* 2(3), p. 45.

Ketchum, D., 2019. *What Is the Purpose of Photoshop?.* [Online] Available at: https://www.techwalla.com/articles/what-is-the-purpose-of-photoshop [Accessed 3 12 2019].

Makai, M., 2020. *Full Stack Python.* [Online] Available at: https://www.fullstackpython.com/flask.html [Accessed 07 05 2020].

MDN contributors, 2019. *Django introduction.* [Online] Available at: https://developer.mozilla.org/en-US/docs/Learn/Server-side/Django/Introduction [Accessed 07 05 2020].

Mittal, A., 2019. *Understanding RNN and LSTM.* [Online] Available at: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e [Accessed 10 05 2020].

Pressman, R. S., 2014. *Software Engineering A practioner's Approach.* Seventh ed. s.l.: Raghothaman Srinivasan.

Rouse, M., 2011. *TECHTARGET NETWORK.* [Online] Available at: https://searchsoftwarequality.techtarget.com/definition/iterative-development [Accessed 05 12 2019].

Singh, N., 2017. *The concept of Iterative Model in a Software Development Life Cycle (SDLC).* [Online]
Available at: https://www.pixelstech.net/article/1505127276-The-concept-of-Iterative-Model-in-a-Software-Development-Life-Cycle-%28SDLC%29
[Accessed 05 12 2019].

Singh, N., 2017. *The concept of Iterative Model in a Software Development Life Cycle (SDLC).* [Online]
Available at: https://www.pixelstech.net/article/1505127276-The-concept-of-Iterative-Model-in-a-Software-Development-Life-Cycle-%28SDLC%29
[Accessed 11 5 2019].

Soloman, B., 2020. *Python Plotting With Matplotlib (Guide).* [Online] Available at: https://realpython.com/python-matplotlib-guide/ [Accessed 07 05 2020].

Taylor, F. M., 2019. [Online] Available at: https://www.quora.com/Is-CSS-or-HTML-a-programming-language/answer/Frank-M-Taylor-1 [Accessed 3 12 2019].

web.njit.edu, 2016. *ELIZA: a very basic Rogerian psychotherapist chatbot.* [Online] Available at: https://web.njit.edu/~ronkowit/eliza.html [Accessed 21 04 2020].

Zalake, N., 2019. *GENERATIVE CHAT BOT IMPLEMENTATION USING DEEP RECURRENT NEURAL NETWORKS AND NATURAL LANGUAGE UNDERSTANDING.* s.l., Mahatma Education Society's Transactions and Journals.

9 Appendix

9.1 History of Chatbot

Turing Test (1950)

In 1950 turing test was developed by Alan Turing to test the ability of the bot to behave like the human beings. The machine was tested for the five minute and it should not be distinguish. The machine should behave like the human beings to pass this test. These idea enhance the revolution for the chatbot development (Jha, 2019).

Eliza (1966)

By an English computer scientist in 1966 the first conversation agent was develop by J.Weizenbaum at MIT lab. Eliza name was given to that conversation agent. It is the ancestor of the chatbot world. Eliza was written in MAD-SLIP script. It has the cognition to process the Natural Language Processing (NLP). It provide responses to the user text by matching the keyword in the text (Jha, 2019).

Parry (1972)

Parry is the conversational agent develop by Stanford scientist Kenneth Colby in 1972. It was the first conversational agent to implement in the health sector. Disease name schizophrenia affected person behavior was implement with the crude model by this chatbot (Jha, 2019).

Ladder (1978)

It is also one of the conversational agent develop in 1978 which has the cognition of learning the new concepts and able to implement the learning concept in problem solving situation. It was develop with the deductive algorithm and the semantics rules (Jha, 2019).

Jabberwacky (1982)

This is the fascinating, delightful chatbot develop by British Programmer Rollo Carpenter in 1982 to simulate natural conversation with the human beings. The word chatterbot is used to denote this conversational agent. This chatbot was proficient to take the input as sound and used the contextual pattern matching technique to result the answer (Jha, 2019).

Albert One (1998)

It is also the conversational agent developed to mimic the human conversation supplied with multi-faceted approach in NLP by Robby Garner in 1998. It has the built in database to stimulus the responses to the human. It uses the multiple response method for the responses (Jha, 2019).

Alice (2001)

It is the award winning AI program inspired by the Eliza which has the ability to conversation with the human beings by using the heuristically pattern matching rules to the human input. This chatbot is develop by Richard Wallace in 2001. Alice uses Artificial Intelligence Markup Language (AIML) to state the heuristics rules (Jha, 2019).

Watson (2006)

It is the powerful system to answer the question which is post in the quiz show Jeopardy in NLP develop by IBM. It has the ability of hypothesis generation and the dynamic learning. This system feature is increasing with the time. Now it can generate the question with the given answers (Jha, 2019).

Siri (2011)

It is the advanced and popular chatbot develop by apple in 2011. It can generate responses in both textual bases and in sound. It uses the NLP algorithm and train in huge

amount of data. It has the cognition of the speech recognition to departure the input speech into digital data then uses Dynamic Time Warping (DTW) to grasp the input.

Slackbot (2014)

It is similar to the chatbot. Slackbot is used to solve the variety of tasks in the slack application. It can send the notification, reminder to the team members. It can gather the information and analyze it. It is also develop from (AI). It can also assist the user to buy necessary stuff, help account to gather the data of the sales (web.njit.edu, 2016).

Alexa (2015)

It is the popular chatbot develop by the Amazon hosted in cloud. It has cognition of understanding the NLP. It has the capabilities to generate the voice. The sound is generated by Long-short term Memory (LSTM) algorithm. It utilize shortlisting-re ranking approach for grasp the input and uses skill-specific classification technique for the results (Jha, 2019).

Messenger Bot (2015)

It is the software which uses the AI, NLP to communicate with the user in the messenger app. One of the popular example is the Facebook messenger bot which generates the responses with the understanding of the input. This kind of bot has the cognition to auto reply the input queries (chatbots.org, 2020).

9.2 SRS

Project Title: Online Support Chatbot

Category

Web Application

Purpose

Online Support Chatbot is developed to overcome the work pressure felt by employees of the Internet Service Provider (ISP) online support team. Oodles of responses are recorded per hour. There are a limited number of the employees working in the support team as per the observation during data collection. Due to tons of messages, employees feel stress in their work. While talking with one of the employees, working eight hour a day ahead of the computer leads to weakness in eyesight and endure back, hip pain. Therefore, to reduce this issue chatbot can be effective.

Scope

Purpose system can solve the minor issue such as:

- Updating the WiFi password from mobile application and web application.
- Troubleshoot the basic internet slow issue.
- Provide department number.
- Provide knowledge of Fair Usages Policy.
- Can know about the internet prices.
- Handle the offensive word.

Introduction

Administrations give to the client/customer by understanding the natural language is known as the chatbot. It is for the most part utilized for client care. It has been utilized in various parts with the end goal of data assembling or giving necessary data direction. It is one of an AI programs that reproduces the human exercises of the discussion or discourse by utilizing key pre-determined client phrases and sound-related and message based signs. Fake composing creatures which can impart to the people in specific bases, for example, content based, expressed discussion or the non-verbal discussion is the chatbot. It is the product or the AI application which is created to converse with the human. Chatbot has been entrancing, animated and charming to the whole world.

- Existing System

  Present system is the manual system where the chat system is handled by Tawk.to. The online representative replied to queries. Below figure shows the recent system is used by Worldlink Communication Private Limited.

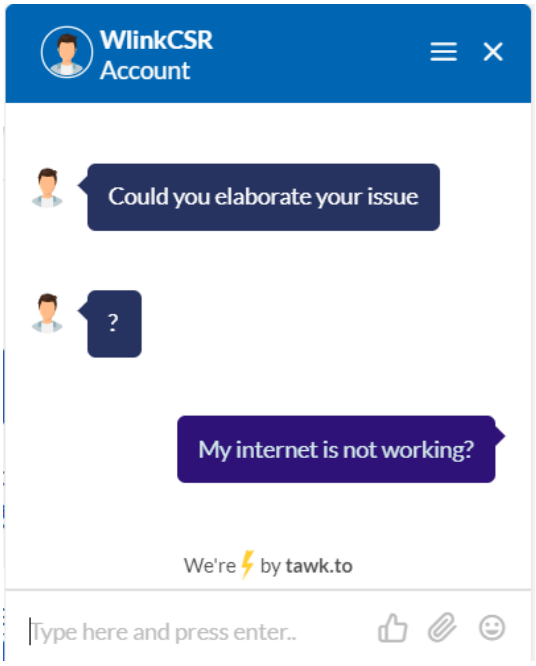


Figure 147 Existing system

Above mention system has following disadvantages:

- The response to the query is not professional and it is time consuming. People have to wait for the reply.
- It is not user friendly.
- Manual system cannot handle lots of chat at the same time.
- There is no history of the client or data to analyze the query handled by the representative.

- Purposed system

  Proposed system is the AI system which handles the minor queries. This system kept log of the client. This system has capabilities of generating the new responses. This system is the combination of two AI component of conversational agent i.e. Retrieval Based Chatbot and the Generative Based Chatbot.

  This system has the following features:

  - **Registration:** New user are allowed to register to the system to use the chatbot feature.
  - **Login:** After registration, user are log to the system for track the user to use propose system.
  - **Authentication:** Without authenticating the user they are not allowed to use the chatbot feature.
  - **Prices:** In the web application, prices of the internet connection can be gathered and made the decision.
  - **Chatbot:** It is the main feature of a system. Purposed system is the hybrid system. It has the cognition of generating the sequences of responses.

Advantages

- Advantages to company
  - The quality of services and problem solving skills will be enhanced as time can be spent to solve complex issues such as online gaming issue, latency issue in the devices.

- Less manpower can handle the department as minor issues are handled by this system.

- Company will be renowned as prompt services can provide to the customer.

- Advantages to employee
  - It provides a means to interact with the colleague and representative can take small breaks to refresh themselves.

  - This system will provide flexibility to the employee as above mentioned points are frequent repetitive issues. If this issue can be handled by this system there will be comparatively less amount of query handled by the support representative.

Functional Requirements

- Users are able to create the account for interacting with the system.
- Administration can add and delete the users.
- Administration can analyze the user using the chatbot feature.
- User authentication is needed to interact with the system.
- System has cognition of generating the responses.
- Web Application can provide detailed prices of the internet, thus users can explore other services of ISP Company.

Non- Functional Requirements

- Maximum time availability.
- Prompt Responses.
- Professional and user friendly.
- Minimized stress of the representatives.

Software Tools

- Programming Language: Python, JavaScript
- Frame work: Django, Flask
- Libraries needed for chatbot: Data Science library i.e. numpy, pandas, genism, keras, tensorflow. Matplotlib, CUDA toolkit.
- Libraries needed for web application: Pillow
- Frontend Languages: Hyper Text Markup Language (HTML), Cascading Style Sheets (CSS)
- Development Tools: Pycharm, Visual Studio
- Client: Any Web Browser
- Database: Default SQLite

Deployment

- Any Operating System (OS) with the internet and browser.

Hardware Specification

- Processor: Intel core i7 8th generation
- Random Access Memory (RAM): 16 Giga Byte(GB)
- Storage Capacity: Solid State Disk (SSD) 256 GB and hard disk 1 Tera Byte (TB)

10 WorldLink Research Paper

WORLDLINK COMMUNICATIONS PVT. LTD.
Jawalakhel, Lalitpur, PO. Box: 8207, Kathmandu, Nepal. Tel.: +977-1-5523050, Fax: +977-1-5529403
E-mail: info@wlink.com.np Website: https://www.worldlink.com.np

19th May 2019

TO WHOM IT MAY CONCERN

This is to certify that Mr. Rukesh Shrestha, a student of BSc. (HONS) in Computer Science at Herald College Kathmandu affiliated to University of Wolverhampton, UK has successfully compieted his research program for his project for the module Research and Development Skills on 14th May 2019 at WorldLink Communications Pvt. Ltd., Jawalakhel Branch, Jawalakhel, Lalitpur.

During his project, he demonstrated good research designing skills with hardworking ambitious and self-motivated attitude to learn new things. His performance meets our expectations and was able to complete the project successfully on time.

We wish him all the best for his future endeavors.

Yours sincerely,
For, Worldlink Communications Pvt. Ltd

Human Resource Department

10.1 Code

10.1.1 Data Understanding

Raw Data

The file can be found here.

CSV converting

The code can be found here.

JSON converting

The code can be found here.

Concatenating line

The code can be found here.

Merged JSON

The code can be found here.

10.1.2 Exploratory Data Analysis

The code can be found here.

10.1.3 Chatbot History Visualization

The code can be found here.

10.1.4 Iterative -1

The code can be found here.

10.1.5 Iterative – 2

The code can be found here.

10.1.6 Iterative - 3

The code can be found here.

10.1.7 Iterative - 4

The code can be found here.

10.1.8 Iterative - 5

The code can be found here.

10.1.9 Whole System

The final system code can be found here.

10.1.10 Unit Testing

The code can be found here.