AI Chatbot Supporting Information For Dormitory

Kien T. Ly, Faculty of Electrical and Electronics Engineering, University of Technology Ho Chi Minh City, Viet Nam

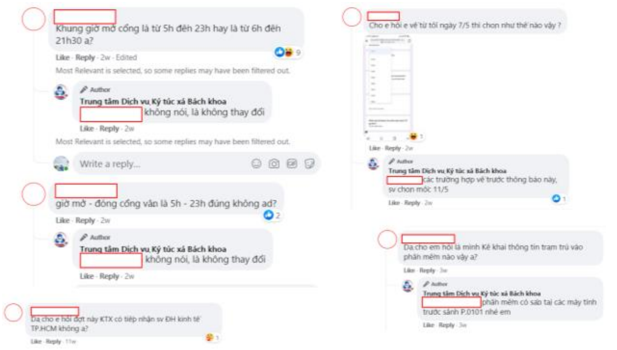
***Abstract*— Currently, living in residential environments such as apartments, dormitories, and high buildings is very common, especially in big cities. And all in those schools have common rules and regulations for everyone to follow, ensuring a comfortable life for everyone in them. However, not everyone is aware of these regulations to the difficulties for building managers, affecting the comfort of others. Although technology is increasingly developing, especially the emergence of social networks, forums that make it easy for them to connect with each other, only help building managers have a part. Realizing those difficulties, and wanting to apply some knowledge files to apply in life, I decided on the topic "AI Chatbot supporting information for dormitory". Applying artificial intelligence, especially machine learning, as well as methods of making Web Server according to other models, integrating connection with Facebook social network, making the topic flexible, fast, and more accessible.**

***Index Terms*—Chatbot, retrieval-based neural, messenger chatbot.**

I. Introduction

AFTER living for a long times in the dormitory of HCM City University of Technology, I realized that there will be many questions and concerns when there are changes and new announcements from the dormitory such as: regulations about the curfew time, the staff of the dormitory, the use of existing and newly added equipment,... Especially with new students or persons who are looking to move into the dormitory.

Understanding these difficulties, I decided to implement the topic: "AI Chatbot to support information for Dormitory" with the desire to help you access information easily, quickly and accurately. Moreover, this also helps to reduce the work of dormitory staff, even those outside of office hours.



[redacted]
Khung giờ mở cổng là từ 5h đến 23h hay là từ 6h đến 21h30 ạ?

Trung tâm Dịch vụ Ký túc xá Bách khoa
[redacted] không nói, là không thay đổi

[redacted]
giờ mở - đóng cổng vẫn là 5h - 23h đúng không ad?

Trung tâm Dịch vụ Ký túc xá Bách khoa
[redacted] không nói, là không thay đổi

[redacted]
Da cho e hỏi đợt này KTX có tiếp nhận sv ĐH kinh tế TP.HCM không ạ?

[redacted]
Cho e hỏi e về từ tối ngày 7/5 thì chọn như thế nào vậy ?

Trung tâm Dịch vụ Ký túc xá Bách khoa
[redacted] các trường hợp về trước thông báo này, sv chọn mốc 11/5

[redacted]
Dạ cho em hỏi là mình Kê khai thông tin tạm trú vào phần mềm nào vậy ạ?

Trung tâm Dịch vụ Ký túc xá Bách khoa
[redacted] phần mềm có sẵn tại các máy tính trước sảnh P.0101 nhé em

Fig. 1. Some frequently asked questions.

In Vietnam, recently, Chatbots have been widely applied in many areas of life. However, the implementation of chatbots in areas with high population density such as buildings and dormitories is still quite new. Some of the reasons it has been suppressed are due to its lack of flexibility and the lack of immediate feedback. Therefore, chatbot only partially supports consulting on basic and stereotyped issues, helping to relieve some of the pressure of consultants, not completely replacing their role.

The content of the thesis includes:

- Build basic neural network chatbot according to Retrieval-Base model.
- Develop methods to store and organize database systems.
- Build algorithms to retrieve information in accordance with user requirements.
- Build methods to communicate with users by text, audio through website, Facebook messenger.

After that, I decided to create the structure of this thesis report is as follows:

Chapter 1: General introduction of the topic.

Chapter 2: Presents the basics of artificial neural networks.

Chapter 3: Building and training artificial neural network for chatbot system.

Chapter 4: Presents how to build a database system, a web server system and an information query algorithm.

Chapter 5: Creating a method to communicate with users through 2 tools: website and Facebook messenger.

Chapter 6: Report on the results of the implementation, from which to present experience and offer the development of the topic in the future.

II. Artificial Neural Network

In this chapter, we will learn through some theories and then make a selection of models and structures suitable for the thesis.

A. *Structure of artificial neural network*

Based on the structure and function of biological neurons (BNN), an artificial neuron (ANN) model has been created. And they have certain similarities.

Neural network is a combination of layers of perceptrons, also known as multilayer perceptrons, as shown below:

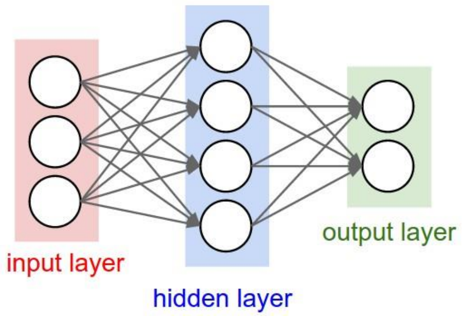


Fig. 2. Layers of perceptrons.

A neural network has three types of layers:

- Input layer: The leftmost layer of the network represents the network's inputs.
- Output layer: The rightmost layer of the network represents the output of the network.
- Hidden layer: The layer between the input layer and the output layer represents the logical inference of the network.

Note: a neural network has only one input layer and one output layer, but can have one or more hidden layers.

At each layer, the number of network nodes (neurons) can be different depending on the problem and the solution. But often when working, the hidden layers have the same number of neurons. In addition, neurons at layers are often paired together to form a full-connected network. Then we can calculate the size of the network based on the number of layers and the number of neurons.

In this thesis, I have used supervised learning model, so let's learn about this model. Much of the actual machine learning uses supervised learning. Supervised learning is where you have input variables (x) and one output variable (Y) and you use an algorithm to learn a mapping function from input to output. $Y = f(X)$. The goal is to approximate the mapping function so that when you have new input data (x), you can predict output variables (Y) for that data. It is called supervised learning because the process of learning the algorithm from the training dataset can be thought of as a "supervisor" of the learning process. The algorithm iterates over the predictions on the training data and is corrected by the "supervisor". Learning stops when the algorithm reaches an acceptable level of performance.

In this section, we learn some common activation functions from which to choose the appropriate input activation function at the layers in the neural network. An activation function in a neural network determines how the sum of the weights of the input is transformed into the output from a node or nodes in a layer of the network. A network can have three types of layers: the input layer receives raw input from the outside, the hidden layer receives input from the input layer and passes the output to the output layer, and the output layer makes predictions.

Based on the unique advantages of each function that we choose to apply in the input layer and hidden layer of the neural network, it will usually be:

- Multilayer Perceptron (MLP): ReLU activation function.
- Convolutional Neural Network (CNN): ReLU activation function.
- Regressive Neural Network (RNN): Tanh and/or Sigmoid activation function.

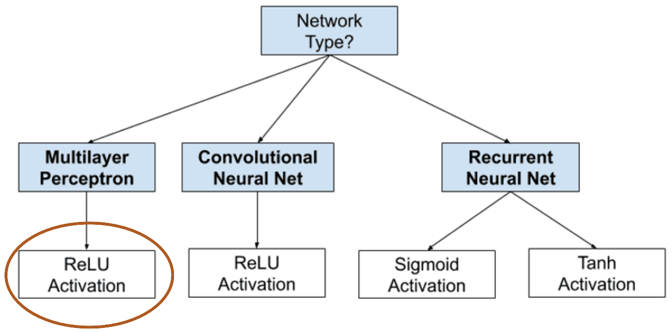


Fig. 3. The way to choose activation function for hidden layers in neural network model.

Choose an activation function for your **output layer** based on the type of prediction problem you're solving, and typically one chooses the following:

- Binary Classification: One button, sigmoid activation.
- Multi-class classification: One node for each class, softmax enabled.
- Multi-label classification: One button per class, sigmoid enabled. [1]

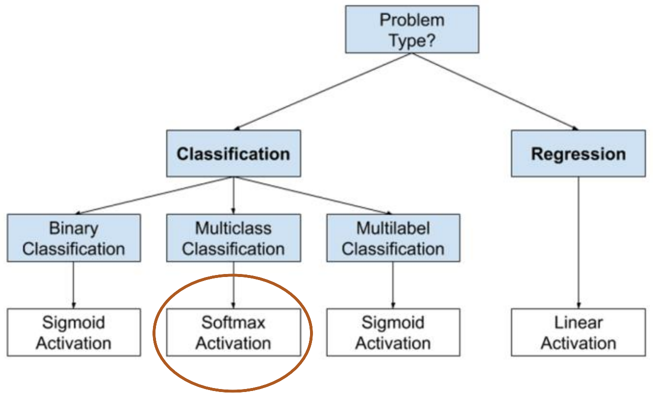


Fig. 4. The way to choose activation function for output layers in neural network model.

Based on the above analysis results, we will choose the activation function for the neural network in the input layer and the hidden layers as ReLU function and in the output layer as Softmax function.

B. *Loss function and loss function optimization*

The Loss function is a method of assessing “how well your algorithm models your data set”. If your prediction is completely off, your loss function will output a higher number. If they are pretty good, it will output a lower number. As you tweak your algorithm to try and improve your model, the loss function tells you whether you're improving. 'Loss' helps us to understand how much the predicted value differs from the actual value. In practice, the input data and the number of dimensions are too large, so the loss function becomes quite complex and it is difficult to find the optimal value by conventional formulas. Therefore, we need methods that can reach the optimal value of the loss function, which are GD, SGD, Mini-batch GD methods, … etc [2]. In this topic, we will use the SGD method. and NAG (which is an extension of GD).

Gradient Descent (GD) is an iterative optimization algorithm used in Machine Learning and Deep Learning problems (usually Convex Optimization problems) with the goal of finding a set of internal parameters for model optimization. Inside:

Gradient: is the rate of inclination or declination of a slope. Mathematically, the gradient of a function is the derivative of that function corresponding to each variable of the function. For univariate functions, we use the concept of Derivative instead of Gradient.

Gradient Descent has many different forms such as Stochastic Gradient Descent (SGD), Mini-batch SDG. But basically they are all implemented as follows:

- Initialize internal variables.
- Evaluate the model based on the intrinsic variable and the loss function (Loss function).
- Update the internal variables in the direction of finding optimal points.
- Repeat steps 2, 3 until stopping condition is satisfied.

The updated formula for GD can be written as:

$$\theta \leftarrow \theta - \eta \nabla_0 f(\theta) \qquad (1)$$

With θ is the set of variables to be updated, η is learning rate, $\nabla_0 f(\theta)$ is the gradient of the loss function f over the set θ.

The stopping condition of GD can be:

- Ends all predefined epochs.
- The value of the loss function is small enough and the model accuracy is large enough.
- The loss function remains unchanged after a finite number of epochs.

In Stochastic gradient descent algorithm, at a time, we only compute the derivative of the loss function based on only one data point x_i and then update θ based on this derivative. This is done point-by-point across the entire data, then repeat the process above. This very simple algorithm actually works very well. Each pass through all points on the entire data is called an epoch. With normal GD, each epoch corresponds to 1 update θ, with SGD, each epoch corresponds to N updates θ where N is the number of data points. On the one hand, this point-by-point update can reduce the execution speed of 1 epoch. But looking at the other side, SGD only requires a very small amount of epoch (usually 10 for the first time, then when new data comes in, just running under one epoch is a good solution). Therefore, SGD is suitable for problems with a large number of databases and problems that require a constantly changing model, online learning.

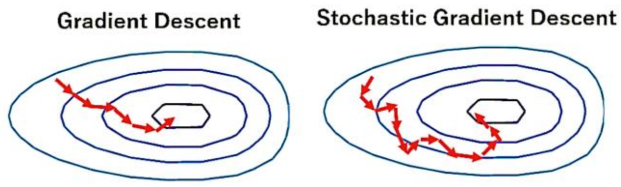


Fig. 5. The difference between SGD and GD.

One point to note is: after each epoch, we need to shuffle the order of the data to ensure randomness. This also affects the performance of SGD. Mathematically, the update rule of SGD is:

$$\theta \leftarrow \theta - \eta \nabla_0 J(\theta; x_i; y_i) \qquad (2)$$

With $J(\theta; x_i; y_i)$ is a loss function with a pair of data points (input, label) of $(x_i; y_i)$.

C. Back-propagation function

To calculate the derivative of the loss function $\nabla_0 J(W)$ in a neural network, we use a special algorithm called backpropagation. Thanks to this algorithm created in 1986, neural networks are effectively implemented and are used more and more until this day.

Basically, this method is based on the rule of series of derivatives of composite functions and inverse derivation to obtain derivatives with respect to all parameters at the same time with only 2 passes. The back-propagation algorithm is implemented as follows:

1) *Progressive propagation:*

Calculate the **a^(l)** from l=2⟶ L in turn according to the formula:

$$z^{(l)} = W^{(l)}.a^{(l-1)} \qquad (3)$$

$$a^{(l)} = f\left(z^{(l)}\right) \qquad (4)$$

With input layer $a^{(l)}$is exactly equal to the input value of the network x.

2) *Calculate the derivative with respect to z at the output layer:*

$$\frac{\partial J}{\partial z^{(l)}} = \frac{\partial J}{\partial a^{(l)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} \qquad (5)$$

With $a^{(l)}$, $z^{(l)}$ is calculated in step 1

3) *Back-propagation:*

Calculate the reverse z derivative from l=(L-1) →2 according to the formula:

$$\frac{\partial J}{\partial z^{(l)}} = \frac{\partial J}{\partial z^{(l+1)}} \frac{\partial a^{(l)}}{\partial z^{(l)}} = \left(\left(W^{(l+1)}\right)^T \frac{\partial J}{\partial z^{(l+1)}}\right) \frac{\partial a^{(l)}}{\partial z^{(l)}} \qquad (7)$$

With $z^{(l)}$ is calculated in step 1, $\frac{\partial J}{\partial z^{(l+1)}}$ is calculated in the previous loop.

4) *Derivative calculation:*

Calculate the derivative with respect to the parameter ***W*** using the formula:

$$\frac{\partial J}{\partial W^{(l)}} = \frac{\partial J}{\partial z^{(l)}} \frac{\partial z^{(l)}}{\partial W^{(l)}} = \frac{\partial J}{\partial z^{(l)}} \left(a^{(l-1)}\right)^T \qquad (8)$$

With $a^{(l-1)}$ is calculated in step 1 and $\frac{\partial J}{\partial z^{(l)}}$ is calculated in step 3. [3]

The learning coefficient and the momentum coefficient are two very important parameters that control the efficiency of the back-propagation algorithm. The learning coefficient is a positive constant that controls the rate at which new weight coefficients are adjusted based on the computed gradient-descent correction term. The momentum factor is an extra weight added to the weight factors that increases the momentum rate. The momentum coefficient helps to move the minimization process away from the local minima.

D. *Parameter estimation error*

Our model after training may not perform well when predicting with a new data. This happens because our model does not generalize to the entire data set. The reason is also quite understandable when our training set is only a small set that cannot represent the entire data and moreover it may be noisy. People divide the causes into two main types: Underfitting or Overfitting.

The model is considered underfitting if it has not yet been fitted to the training data set and to the new samples when predicting. The reason may be that the model is not complex enough to cover the data set.

The overfitting occurs at a very reasonable model, which closely matches the training set, but when making predictions with new data, it is not suitable. The reason may be that we do not have enough data to evaluate or our model is too complex. The model is overcomplicated when our model uses even large noises in the data set to learn, leading to the loss of the generality of the model.

Good Fitting is located between underfitting and overfitting, that give reasonable results for both the training data set and the new values. Ideally, it should be able to match a lot of sample data as well as new data. However in practice such models are very rare.

In machine learning, bias and variance are properties of the model where the variance of parameter estimates between samples can be reduced by increasing the bias in the estimate parameters. The bias-variance problem is that the conflict in trying to minimize these two sources of error simultaneously prevents supervised learning algorithms from generalizing beyond their training set. Bias is an error from faulty assumptions of the learning algorithm. High bias can cause the algorithm to miss relevant relationships between features and the target output (missing page). To minimize bias, we can increase the model size, increase the number of nodes, change the model architecture, add new features, … The variance is an error from the sensitivity to small fluctuations in the training set. The high variance can be caused by the algorithm modeling random noise in the training data (overfitting). To minimize the variance, we can add more data, use dropout technique, select suitable features.

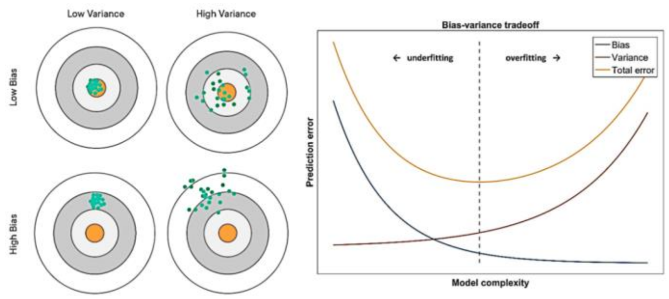


Fig. 6. The trade-off between Bias and Variance.

In this thesis, the dropout technique is used to increase the flexibility and generalization of the model and prevent overfitting. In simple terms, the droupout technique is to skip a certain proportion of the network nodes during each training of an epoch (ignoring it in both forward and reverse propagation, but not completely out of the network).

Removing some nodes makes the remaining nodes more flexible and powerful without having to depend entirely on the nodes associated with it. However, using the dropout technique will increase the number of training epochs to arrive at the sequence but will decrease the training time. In this technique, we use a dropout coefficient p representing the ratio of the number of nodes to be skipped over the total number of nodes. Usually people often choose the dropout coefficient (p) is 0.5.

III. Neural Network Model for Chatbot

A. *Chatbot Overview*

First, we are talking about the deep learning component. This component is responsible for processing the requested information and providing the appropriate response. Two models commonly used in this section are:

- Retrieval Based: uses pre-prepared sets of answers and search and request processing algorithms to select answers from available answers. Search techniques can apply machine learning to classify requests, thereby giving the most appropriate answers.
- Generative Model: self-process the request and generate the answer by itself based on machine learning techniques.

In this thesis, I have used the Retrieval Based model because the information and answers are often fixed, repetitive, requiring high accuracy. And the Generative Model is not suitable because it will complicate the build system, can make more processing time, and can also create syntax errors as well as content errors, affecting the reliability of the answers sent to the questioner. With the processing system in the chatbot in this thesis, the role of blocks will be presented in the following diagram:

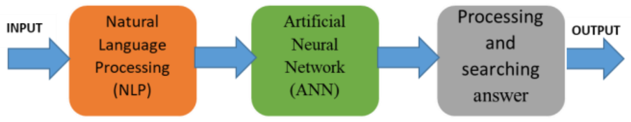

INPUT
Natural Language Processing (NLP)
Artificial Neural Network (ANN)
Processing and searching answer
OUTPUT

Fig. 7: Processing in Chatbot

- NLP block: Remove punctuation, separate words (tokenize), convert words to primitive form (lemmaztize), convert to lower case (lower) and process input sentences into appropriate data as input for the ANN block.
- ANN block: Performs prediction of input sentences in topic group and predicts which topic in that group based on probability.
- Processing and searching block: Process and collect information on the output of the ANN block, using algorithms and databases to get the most appropriate answer.

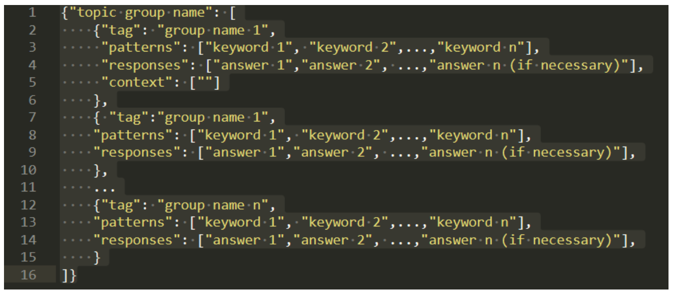
B. *Neural network construction and training*

Before going into the processing in the chatbot, we need to define the process of building the neural network in the ANN. This process consists of 3 steps: building the training set;

building structure and parameters for neural network; train, test, and save the model.

1) Building the training set:

First, we build json files (JavaScript Object Notation) to store data according to the following structure:



```
{"topic group name": [
    {"tag": "group name 1",
     "patterns": ["keyword 1", "keyword 2",...,"keyword n"],
     "responses": ["answer 1","answer 2", ...,"answer n (if necessary)"],
     "context": [""]
    },
    { "tag":"group name 1",
    "patterns": ["keyword 1", "keyword 2",...,"keyword n"],
    "responses": ["answer 1","answer 2", ...,"answer n (if necessary)"],
    },
    ...
    {"tag": "group name n",
    "patterns": ["keyword 1", "keyword 2",...,"keyword n"],
    "responses": ["answer 1","answer 2", ...,"answer n (if necessary)"],
    }
]}
```

Fig. 8. The structure of the json file.

After having data about the topic group and specific topics, we implement the word creation process using the *word_tokenize* function in the python nltk library. The job of word splitting is to identify a sentence of text and then split it into the set of words or punctuation it contains. The program will take the data in all the "pattens" (in the json file) in each topic, separate the words and store all the results into a list of "words". At the same time, the list of "documents" names will save the link data of the split results and include the topic name in each corresponding topic keyword. A list of "classes" will store all the topic names in the topic group under consideration.

Next, we filter the punctuation, convert the word to the infinitive form in the "words" list using the *lemmatizer* function in the ntlk library. The conversion to the infinitive form only applies to languages with variations such as English, French, German, ... and for Vietnamese, there is no change in word form, for example the word "**biết**", in the past is "đã **biết**", in English it is “**know”**, in the past is “**knew**” or “have **known”**. So we can skip this process for Vietnamese data.

After that, we deploy to sort the "words" list, then proceed to embed the "words" and "document" lists into the ".pkl" file (Python Pickle File - a type of Developer Files) to keep the state intact of the list as a long stream of bytes. This has the effect that when taking out the data to use, it will not change state, suitable for processing text strings on demand when making predictions with neural networks.

We proceed to create a training set for the network. We first create an empty "train" list containing the input data and its corresponding label. An "output_rempty" list has all zero values and a size equal to the number of topics in the topic pool. We create an empty list of "bag" and "patten_words" containing specific keywords in each topic, compared to the list of words in "words" and if they match, save it in "bag". Next we create an output list "output_row" equal to the size of "output_empty" and compare it with the topic under consideration. Finally put "bag" and "output_row" data into "train".

2) Neural network design and training

The process of designing an artificial neural network is as follows:

The input data consists of a number of lists with the number of elements equal to the size of the set "words" with a dimension. The input layer has 256 network nodes with the activation function "ReLU". The middle layer consists of 128 network layers with the activation function "ReLU". The output layer has the number of nodes equal to the number of topics in the topic set (5 topics are applied in this thesis) and the output activation function is "softmax".

Use a loss function of the form "categorical-rossentropy" with the SGD + NAG optimization method with a momentum factor of 0.82, a learning factor of 0.01 and a delay factor of 10^{-8}. Train 200 times with batch-size factor = 5. Network evaluation is based on the number of correct predictions on accuracy and loss function value.

Conduct training neural network, we have training results as shown below. Based on that result, we find that the models give high accuracy close to 1 and error almost equal. With such a result, threading and topic selection will be almost exactly the same as the request from the input. About the threading algorithm as well as the search algorithm will be presented in the following section.

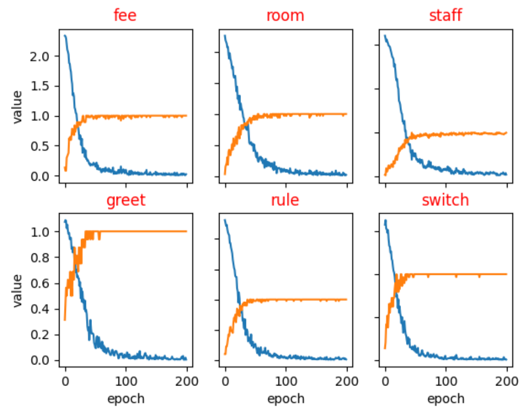


Fig. 9. Result of training model.

C. Input processing and threading

After building and training the models, we begin to process the input data to match the input of the neural network. The process of input data processing is similar to the process of creating a training set for a neural network. First, we separate the word in the sentence, filter the punctuation and convert it to lowercase. Next, we create a place to collect the words that are both in the sentence and in the list of "works" saved in pickle form. Then convert those words into binary network form and as input to the switch model, which returns the request to its subject model. Here, the system includes the following models:

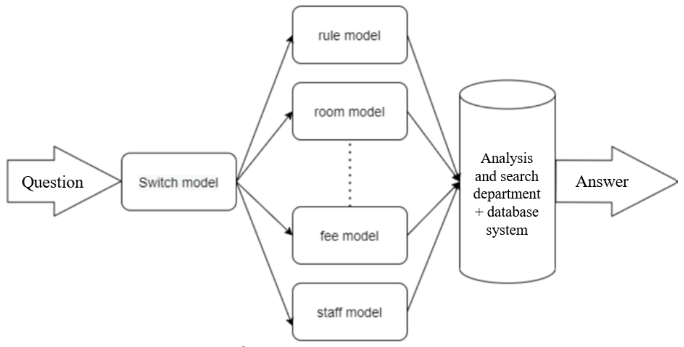


Fig. 10. The structure of the system's model.

- greet: This model simply responds to simple sentences like hello, thank you or goodbye to make it feel more friendly and natural.
- fee: This model helps us to answer questions related to service fees in the Dormitory.
- room: This model helps us to answer questions related to the type of room and its location in the Dormitory.
- staff: This model helps us answer questions related to the staff working in the Dormitory.
- rule: This model helps us to answer questions related to the rules in the Dormitory.

The process of threading or returning results according to each model is as follows: First, the data is fed into the model, and then we arrange the small topic prediction probabilities in topic groups in increasing order. We then compute the variance by topic with the greatest probability. Below is the formula for the variance (deviation of the data from the maximum value of the model's prediction).

$$\sigma^2 = \frac{1}{N}\sum_{i=1}^{N}(x_i - x_{max})^2 \qquad (9)$$

The larger the variance of the model, the higher the accuracy of the prediction probability, so the prediction probability is concentrated in only one or two specific topics, not all subjects. If the switch model input is "thank you", it is easy to know that it belongs to the topic of "greeting", so the variance *pvar* of the model is relatively large (Figure a). If the input is "it's not raining today", it will not belong to any topic in the "switch", so the variance *pvar* of the model is relatively small. After having the variance results, we determine the threshold for which we accept which model to process for the input request. In this study, we choose a threshold level of 0.5.

In the next models, we also do the same as above, but we reduce the threshold of variance to 0.25 because in the switch model we need a large policy in the selection process for threading, and in the following models, stream, because in the train set there are noisy data such as "normal 4-person room", "special 4-person room", "normal 6-person room",... even though I tried to reduce the noise, but I couldn't avoid all errors. If we choose a threshold of 0.5 or more, the variances can result below 0.5, so we temporarily accept the threshold of variance is 0.25.

After having the prediction results of the submodule, we proceed to select information from the results and extract the appropriate answer. The answer here is prepared in two forms:

- Systematic questions: The answer structure will be prepared in combination with the data stored in the database (more detailed in the next chapter) to give a systematic answer. For example, we use the system response structure: *"Service price:" + data1 + "is" + data2 + "/month"* Will give a series of answers: “Service price: Bicycle parking is 30,000 VNĐ/month”; “Service price: Refrigerator rental is 200,000 VNĐ/month”;...
- The unique question: We will prepare a few answers in the "reponse" section of the json file and then randomly choose 1 answer, reducing boredom, increasing "humanity" for chatbots.

IV. Database system

A. Database Management System (DBMS)

For creating and managing the database system, we must first understand the problem requirements for the database system. In this thesis, I only have a simple request: Design a database system to manage staff information, room system information, information about departments in the dormitory, fees and rules of the dormitory.

From the above requirements, we construct the ER diagram as follows:

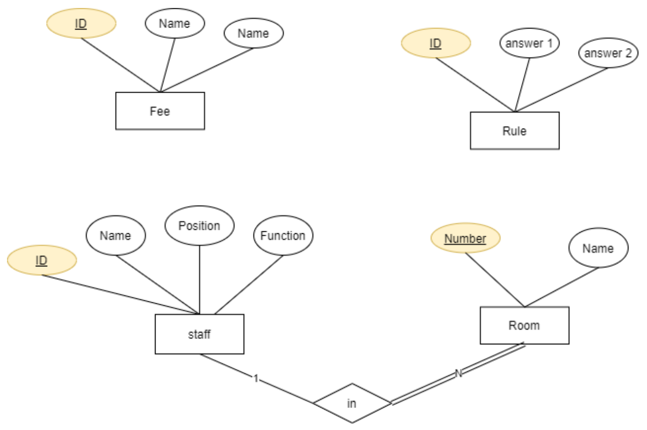
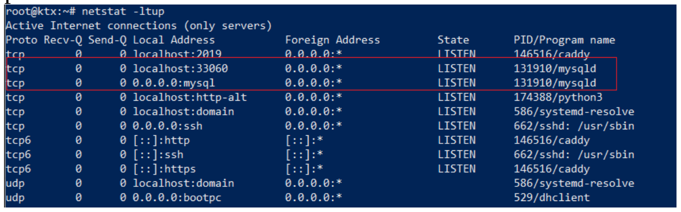


Fig. 11. ER diagram

B. Building a Database in MySQL

MySQL, the most popular Open Source SQL database management system, is developed, distributed, and supported by Oracle Corporation. In this thesis, I have set up MySQL at port 3306 on webserver



```
root@ktx:~# netstat -ltup
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:2019          0.0.0.0:*               LISTEN      146516/caddy
tcp        0      0 localhost:33060         0.0.0.0:*               LISTEN      131910/mysqld
tcp        0      0 0.0.0.0:mysql           0.0.0.0:*               LISTEN      131910/mysqld
tcp        0      0 localhost:http-alt      0.0.0.0:*               LISTEN      174388/python3
tcp        0      0 localhost:domain        0.0.0.0:*               LISTEN      586/systemd-resolve
tcp        0      0 0.0.0.0:ssh             0.0.0.0:*               LISTEN      662/sshd: /usr/sbin
tcp6       0      0 [::]:http               [::]:*                  LISTEN      146516/caddy
tcp6       0      0 [::]:ssh                [::]:*                  LISTEN      662/sshd: /usr/sbin
tcp6       0      0 [::]:https              [::]:*                  LISTEN      146516/caddy
udp        0      0 localhost:domain        0.0.0.0:*                           586/systemd-resolve
udp        0      0 0.0.0.0:bootpc          0.0.0.0:*                           529/dhclient
```

Fig. 12. Result after configuring MySQL.

Then, I have set up a database system on it.

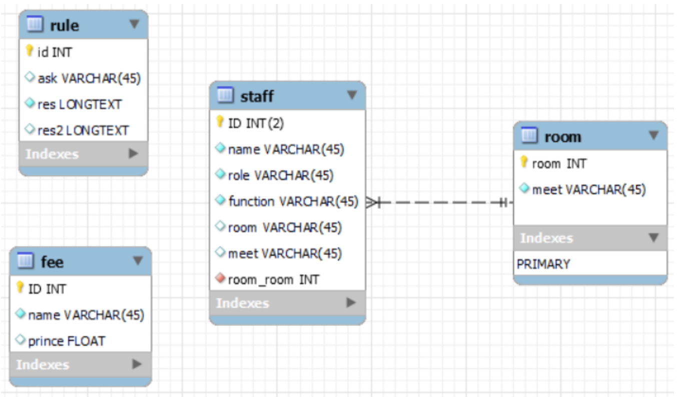
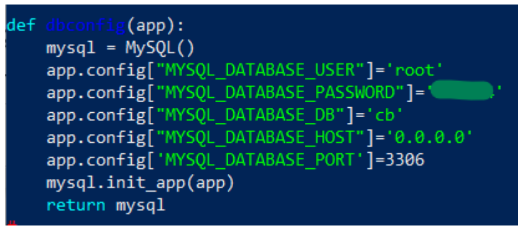


Fig. 13. The database on webserver

I have connected the database in MySQL to my application using the method:

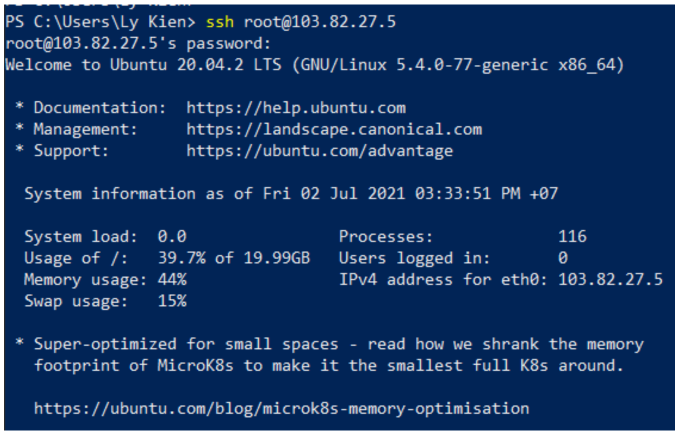


```
def dbconfig(app):
    mysql = MySQL()
    app.config["MYSQL_DATABASE_USER"]='root'
    app.config["MYSQL_DATABASE_PASSWORD"]='[redacted]'
    app.config["MYSQL_DATABASE_DB"]='cb'
    app.config["MYSQL_DATABASE_HOST"]='0.0.0.0'
    app.config['MYSQL_DATABASE_PORT']=3306
    mysql.init_app(app)
    return mysql
```

Fig. 14. The method of connecting the application to MySQL.

C. *Web server*

To do this thesis, I rented a cloud server from the website: *https://my.cloudfly.vn/* with the following machine configuration:



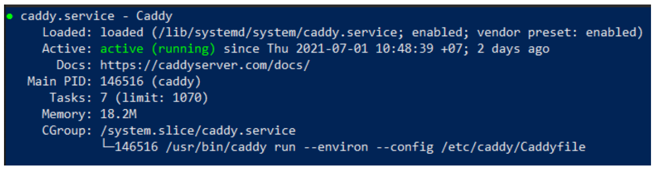
```
PS C:\Users\Ly Kien> ssh root@103.82.27.5
root@103.82.27.5's password:
Welcome to Ubuntu 20.04.2 LTS (GNU/Linux 5.4.0-77-generic x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Fri 02 Jul 2021 03:33:51 PM +07

  System load:  0.0               Processes:             116
  Usage of /:   39.7% of 19.99GB  Users logged in:       0
  Memory usage: 44%               IPv4 address for eth0: 103.82.27.5
  Swap usage:   15%

 * Super-optimized for small spaces - read how we shrank the memory
   footprint of MicroK8s to make it the smallest full K8s around.

   https://ubuntu.com/blog/microk8s-memory-optimisation
```

Fig. 15. Cloud server configuration

1) *Caddy*

In this thesis, I used Caddy server to configure my applications. Caddy is a server of servers (most people use Caddy as a web server or proxy), written in the Go language. With the necessary modules, it can take on the role of any lengthy processing.

Configurations are both dynamic and exportable with Caddy's API. Although no configuration files are required, you can still use them; The format of the configuration document comes in many forms with the configuration adapter, but Caddy's native configuration language is JSON. Most people's favorite way of configuring Caddy is to use the Caddyfile [4]. And in this thesis I also use Caddyfile to configure my server.



```
● caddy.service - Caddy
     Loaded: loaded (/lib/systemd/system/caddy.service; enabled; vendor preset: enabled)
     Active: active (running) since Thu 2021-07-01 10:48:39 +07; 2 days ago
       Docs: https://caddyserver.com/docs/
   Main PID: 146516 (caddy)
      Tasks: 7 (limit: 1070)
     Memory: 18.2M
     CGroup: /system.slice/caddy.service
             └─146516 /usr/bin/caddy run --environ --config /etc/caddy/Caddyfile
```

Fig. 16. Status of Caddy server.

2) *Gunicorn*

Gunicorn 'Green Unicorn' is a Python WSGI HTTP Server for UNIX. It's a pre-fork worker model ported from Ruby's Unicorn project. The Gunicorn server is broadly compatible with various web frameworks, simply implemented, light on server resources, and fairly speedy [5]. Gunicorn is easy to use and supports many different web frameworks like Django, Flask...

Gunicorn is implemented according to the UNIX pre-fork server model:

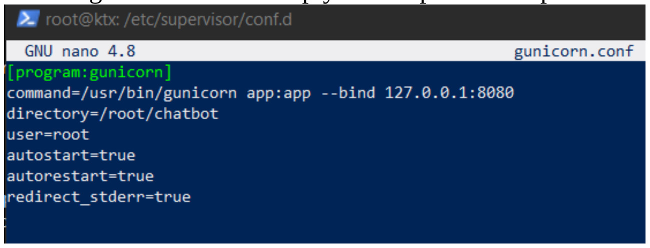
- At launch, Gunicorn opens a master process, which can be cloned (fork) into child processes, which are called workers.
- The role of the root process is to ensure that the number of workers is always the same as the number defined in the configuration files or in the command line parameters. If a worker dies due to some problem, the original process will create a new worker by forking itself again.
- The role of the workers is to receive and process HTTP requests. The word pre in the pre-fork model means that the root process creates workers before handling any HTTP requests.
- The operating system kernel assumes the role of load balancing among workers

Each worker is a UNIX process, it is an instance of the application, the workers are separate entities of the application and they do not share memory resources with each other. The reasonable number of workers when running Gunicorn on a physical machine (virtual machine) is usually (2 $*$Number_cpu) + 1.

In addition to allowing the creation of multiple workers, gunicorn also allows a worker to create multiple threads, threads within a worker sharing the same memory resources. To use threads with gunicorn, we use the threads parameter. The maximum number of concurrent tasks is *workers* $*$ *threads*.

3) *Supervisor*

Supervisor is a client/server system that allows its users to control a number of processes on UNIX-like operating systems. Supervisor is useful tool to monitor and control Gunicorn. I have configured Gunicorn simply with supervisor at port 8080.



```
root@ktx: /etc/supervisor/conf.d
  GNU nano 4.8                                              gunicorn.conf
[program:gunicorn]
command=/usr/bin/gunicorn app:app --bind 127.0.0.1:8080
directory=/root/chatbot
user=root
autostart=true
autorestart=true
redirect_stderr=true
```

Fig. 17. Configured Gunicorn simply with supervisor

V. USER COMMUNICATION METHOD

A. *Flask Server*

Flask is a micro web framework written in Python. It is classified as a microframework because it does not require specific tools or libraries. It doesn't have a database abstraction layer, forms validation, or any other components that pre-existing third-party libraries provide common functionality. However, Flask supports extensions that can add application features as if they were implemented in Flask itself. Extensions exist for object-relational mappers, forms validation, upload handlers, various open validation technologies, and several general framework-related tools. We declare the flask libraries in python as follows:

from flask import ***Flask****,* ***request****,* ***redirect****,* ***render_template***

- **Flask**: the package name used to resolve resources from within the package or directory in which the model is contained depending on whether the package parameter resolves to the actual python package (the __init__.py file directory inside) or the model standard (just a .py file)
- **request**: to access incoming request data. Flask parses the incoming request data for you and gives you access to it through that request object.
- **redirect**: returns a response object that, if called, will redirect the client to the destination location
- **render_template**: the folder that contains the templates that should be used by the application. Defaults to 'templates' folder in the root path of the application [6]

To initialize Flask Server, we use the command **app = Flask(__name__)**, and set up running the server application on the webserver, we use **app.run(host='0.0.0.0', port=8000, debug=True).**

B. *HTTPs protocol*

HTTPs stands for hypertext transfer protocol secure and is the encrypted version of HTTP. It is used for secure communication across the internet or a network. The communication protocol is encrypted using Transport Layer Security (TLS) or, formerly, Secure Sockets Layer (SSL) [7]. We need HTTPs to link the application to Facebook messenger and I bought a domain with an SSL certificate.

C. *User communication through web interface*

In this thesis, I have used HTML, CSS, Javascript and Python to design the font-end and back-end for the website. HTML (Hypertext Markup Language) is a hypertext markup language that uses opening and closing tags to mark up each object of hypertext. But using only HTML, the website will be very rudimentary and boring, so CSS (Cascading Style Sheet) is needed to animate and style objects in HTML. The combination of Javascript will make the website more dynamic and Python on the server side will help us process the information provided by the user through the chat box on the web and return the results to the user through that chat box.

This chatbot has two communication functions on text request or on voice request. Accordingly, with the text function, users only need to text the message box and then press Enter, the request will be sent to the system and processed. Same voice function when user presses the Record button. To avoid errors, when the user has pressed the Record button, the Record button will be disabled immediately until the system returns the results.

When converting speech to text, we use the Recognize Google tool, and when converting text to speech, we use the gTTS tool. These tools are all available in Python and are provided by the Google API.

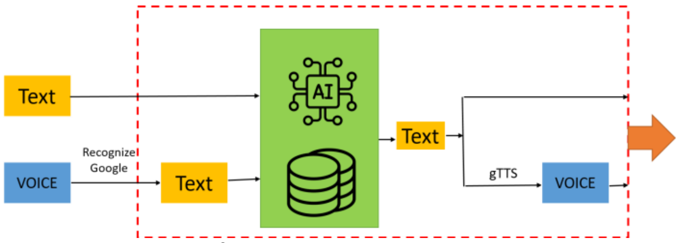
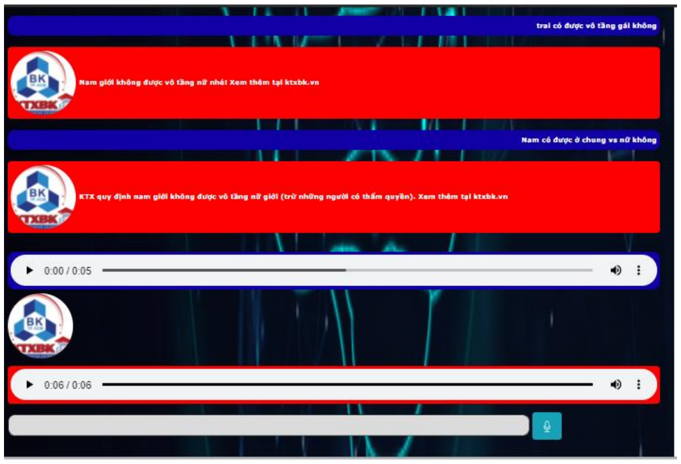


Fig. 18. The process of processing communication with text and voice.

Results are returned when we request both text and voice.


trai có được vô tầng gái không

Nam giới không được vô tầng nữ nhé! Xem thêm tại ktxbk.vn

Nam có được ở chung vs nữ không

KTX quy định nam giới không được vô tầng nữ giới (trừ những người có thẩm quyền). Xem thêm tại ktxbk.vn

0:00 / 0:05

0:06 / 0:06

Fig. 19. Communicate through chat-box.

Then, the Server side will give us the processing as shown below:

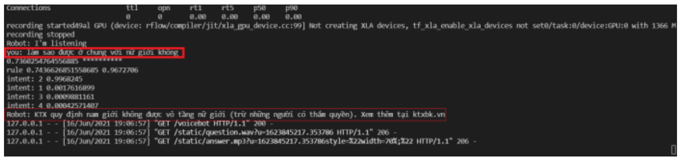

```
Connections                  ttl     opn     rt1     rt5     p50     p90
                             0       0       0.00    0.00    0.00    0.00
recording started49al GPU (device: rflow/compiler/jit/xla_gpu_device.cc:99] Not creating XLA devices, tf_xla_enable_xla_devices not set0/task:0/device:GPU:0 with 1366 M
recording stopped
Robot: I'm listening
you: làm sao được ở chung với nữ giới không
0.7360254764556885 **********
rule 0.7436626851558685 0.9672706
intent: 2 0.9968245
intent: 1 0.0017616899
intent: 3 0.0009881161
intent: 4 0.00042571407
Robot: KTX quy định nam giới không được vô tầng nữ giới (trừ những người có thẩm quyền). Xem thêm tại ktxbk.vn
127.0.0.1 - - [16/Jun/2021 19:06:57] "GET /voicebot HTTP/1.1" 200 -
127.0.0.1 - - [16/Jun/2021 19:06:57] "GET /static/question.wav?u=1623845217.353786 HTTP/1.1" 206 -
127.0.0.1 - - [16/Jun/2021 19:06:57] "GET /static/answer.mp3?u=1623845217.353786style=%22width=70%;%22 HTTP/1.1" 206 -
```

Fig. 20. Processing at the server.

D. *User communication through Facebook messenger*

Facebook is the most popular social network in the world with 2.85 billion monthly active users (as of 31 March 2021). So the application will be extremely accessible if I deploy the chatbot application on Facebook messenger. There are 3 main steps to deploy a chatbot on Facebook Messenger (only **typing** function):

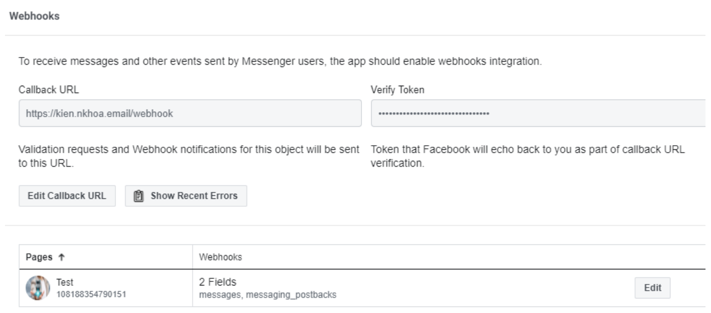
- Create a server which listens to messages from Facebook (using flask)
- Define a function for sending messages back to users (using requests)
- Forward a https connection to the webserver

The first step is to create a http server which listens to messages sent by Facebook, gets a response to that message, and eventually sends the response back to the user on Facebook. We will use flask to create this server. Create a flask app that listens for messages sent to *https://kien.nkhoa.email/webhook*. When messages are sent on Facebook they will arrive as http requests to this URL. The listen() function handles these http requests and checks that they contain a valid Facebook message. If the message is valid, the get_bot_response() function is called and the response is sent back to Facebook

Messenger. In addition, you will create a function called verify_webhook() that handles the initial authentication between Facebook and my app. The VERIFY_TOKEN and PAGE_ACCESS_TOKEN variables aren't defined yet - so you will have to set these. The VERIFY_TOKEN can be any string, it is used as a kind of password to let Facebook know that this server wants to receive messages. It is a good idea to use a long random string.

Next, I have writen a function that that sends a response back to Facebook Messenger using a python library called requests. In particular, you use the post() method, which creates a HTTP POST request. A POST request is the typical way to send information to a server.

Then, set up facebook messenger.Create a facebook page and facebook app. Add Webhook to the app. Fetch the app ID and update it in the script. Add Messenger to the app and generate token for the page which has to used for chat and select events for page subscription. Going to the settings for Messenger, scroll down to Token Generation and click on the link to create a new page for your app [8]



Webhooks

To receive messages and other events sent by Messenger users, the app should enable webhooks integration.

Callback URL: https://kien.nkhoa.email/webhook

Verify Token: ••••••••••••••••••••••••••••••••

Validation requests and Webhook notifications for this object will be sent to this URL.

Token that Facebook will echo back to you as part of callback URL verification.

Edit Callback URL | Show Recent Errors

Pages ↑	Webhooks	
Test 108188354790151	2 Fields messages, messaging_postbacks	Edit

Fig. 21. Setup Webhooks.

Once you have created a page, go back to the Token Generation settings and select this page from the drop-down menu. Copy the Page Access Token into the placeholder for PAGE_ACCESS_TOKEN.

Finally, you need to register your webhook on the Facebook developers page. Go to the Messenger tab in Products again and scroll down to Webhooks, click on Setup Webhooks, the Callback URL enter in your URL. It is important that your flask app is running at this point, because the verify_token() will be called on the next step. In the Verify Token field, you put the value to variable. In Subscription Fields make sure messages and messaging_postbacks are ticked Click Verify and Save to perform the authentication.

When a person sends a message to a business on Messenger and as long as this Page uses an app to partially or fully automate conversations, the following will happen. The Facebook server sends webhooks to the URL of the business server, where the messaging app is hosted. Using the Send API, the app can respond to the person on Messenger. In this way, developers can build guided conversations to lead people through an automated flow or build an app to serve as a bridge between your agents and your business presence on Messenger. [9]

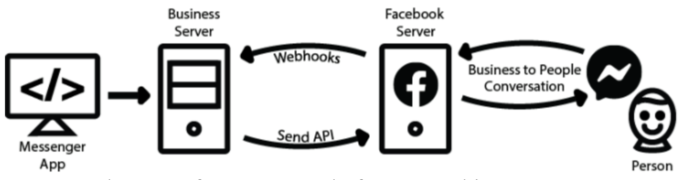


Fig. 22. The way of Messenger Platform's working.

Image of a website that integrates both box-chat and Facebook messenger:

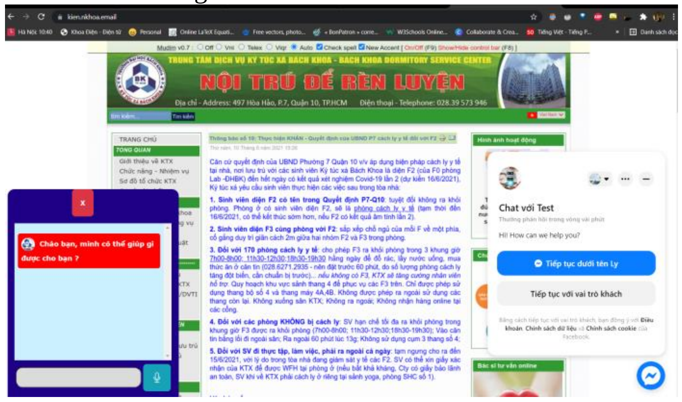


Fig. 23. The interface of the website

VI. Results and Development Direction

A. Thesis results

1) With text

With a lot of experimentation, I found that the basic correct answer to total number of answers was over 70%, and saw a variation in accuracy between classes compared to the value average is relatively small

2) With sound

For audio input, we find that the accuracy is worse than that of text input due to: Influence of noise from the environment; sometimes your voice is not loud enough or pronunciation is not clear, and due to system delay, the audio recorder is later than when the person started speaking.

B. Advantages and limitations

One of the advantages of the topic is its high applicability and simplicity in structure, easy access and use.

In terms of structural simplicity and diversity approached through system building tools. The application is built on the Python language, which is both simple and popular. In addition, the server model helps many people to access the system at the same time as well as connect to the Facebook social network. Along with using MySQL to store and process data, we can easily change the answer when there is a real change.

However, the topic still has some limitations in terms of the amount of information that has not been covered compared to the actual volume of information. Web interface is rudimentary, not eye-catching. In addition, the system has not really monitored the number of visitors at the same time to handle overload if any. Regarding the accuracy of information, the system sometimes gives some ambiguous and unclear answers that match the questioner's intention.

C. Development orientation of the thesis

The main development orientation of the project is to improve the accuracy as well as expand the system data. At the same time, we also need to increase the security of the system in terms of both the database and the computer network. In addition, we can develop some special features for the system:

- Development of real-time multi-language conversion
- Improve voice chat feature on Facebook messenger.
- Deployed on many social networking platforms, other messaging tools such as: Telegram, Viber, Skyper, Zalo,...
- Develop more integrated features with search engines like google.

REFERENCES

[1] Jason Brownlee, "How to Choose an Activation Function for Deep Learning" [Online]. Available: https://machinelearningmastery.com/choose-an-activation-function-for-deep-learning/, Accessed on: Jul. 2, 2021

[2] Jason Brownlee, "*Master Machine Learning Algorithms - Discover how they work and Implement Them From Scratch*". Machine Learning Mastery ,2016, pp. 30-45.

[3] R. Rojas: "Neural Networks", Springer-Verlag, Berlin, 1996, pp 152 - 184.

[4]"Introduction" [Online]. Available: https://caddyserver.com/docs/

[5] Gunicorn - WSGI server [Online]. Available: https://docs.gunicorn.org/en/19.3/index.html#, Accessed on: Jul. 2, 2021

[6] Application Object [Online]. Accessed on: Jul. 2, 2021. Available: https://flask.palletsprojects.com/en/0.12.x/api/#application-object

[7] Eric Rescorla, "HTTP Over TLS", The Internet Society (2000) [Online]. Available: https://datatracker.ietf.org/doc/html/rfc2818, Accessed on: Jul. 2, 2021

[8] Alan Nichol, "Deploy Your Facebook Messenger Bot with Python" February 28th, 2018. [Online]. Accessed on: Jul. 2, 2021. Available: https://www.datacamp.com/community/tutorials/facebook-chatbot-python-deploy

[9] Introduction to the Messenger Platform [Online]. Available: https://developers.facebook.com/docs/messenger-platform/introduction/, Accessed on: Jul. 2, 2021.