A generic framework for conversational agents

Group Name: Zero

Group Members	Student number	email
Fan Yang	z5127233	snowyng1994@gmail.com
Jinzhu Wu	z5102866	jinzhu.wu@student.unsw.edu.au
Zhen Liu	z5124220	liuzhen930719@sina.com

Report Submission Date：October 20, 2018

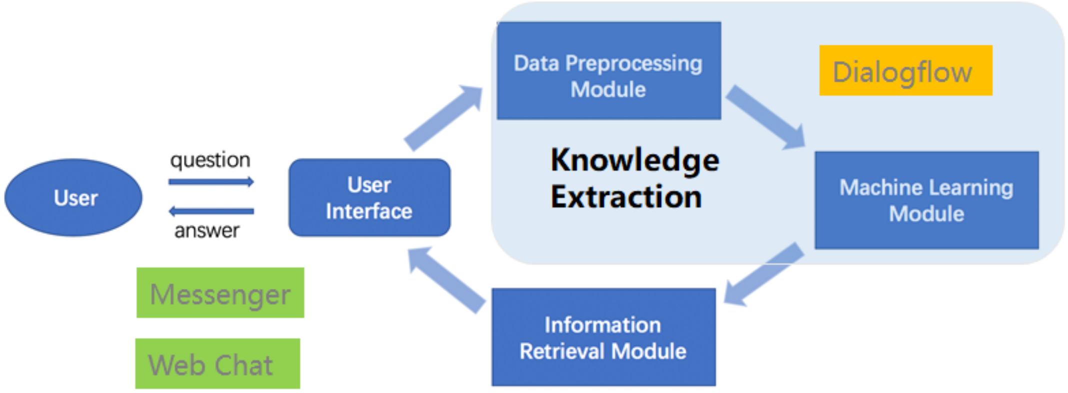
Overview

In the information age, modern libraries, as an essential element of university life for students and researchers, are forced to deal with dramatically increased available materials and user demands. How to help users access information effectively and take advantage of all the services provided becomes crucial and important. Therefore, the conversational agent Zero is built in this project this project to help free the librarian from repetitive tasks, common questions and help free the users from knowing the layout of the website, visiting the librarian in person, finding the information everywhere, possible waiting queues etc.

This conversational agent, based on the information related to the library of the University of New South Wales, can provide effective and appropriate answers and solutions to users. It mainly covers the aspects of borrow, return, print/copy/scan, and small talks as well. It can simulate natural communication on most covered topics in several ways including text, picture, and links.

Architecture

This project has a generic and modular architecture which based on three core components: a knowledge extraction/building module, a dialog builder, and an application/UI component.



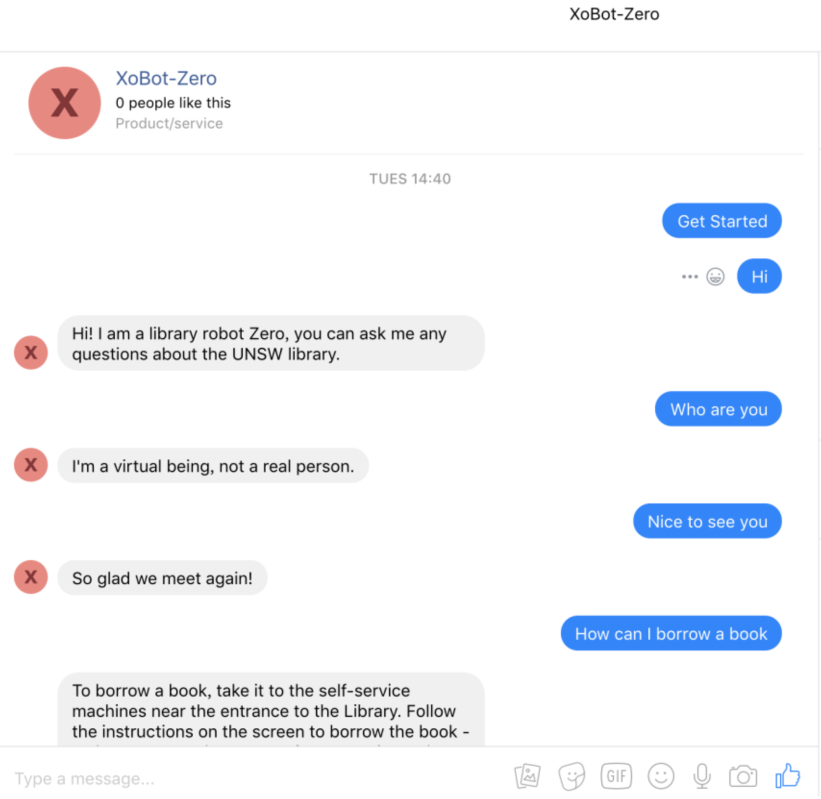
Here shows the structure of the system. As for the backend part, Dialogflow is used to extract information from inputs and do the learning. Then the intent of the input and the confidence of this intent are returned from Dialogflow. What answers are going to be sent is decided according to these intents and their confidence. To connect users with the backend, user interface plays an important role. In terms of user interface, an own web application is built. In addition to this, the agent is also applied to Messenger.

Functionalities

There are four functionalities implemented in this project: user interface, data preprocessing, machine learning, information retrieval.

User Interface

As for the User Interface, two different dialog builders are showed. One uses Facebook messenger as a communication tool and the other inserts a dialog box into the Website as a simple Web Chat. These two methods for dialog prove that our agent is allowed to plug into multiple channels.



Get Started

Hi

Hi! I am a library robot Zero, you can ask me any questions about the UNSW library.

Who are you

I'm a virtual being, not a real person.

Nice to see you

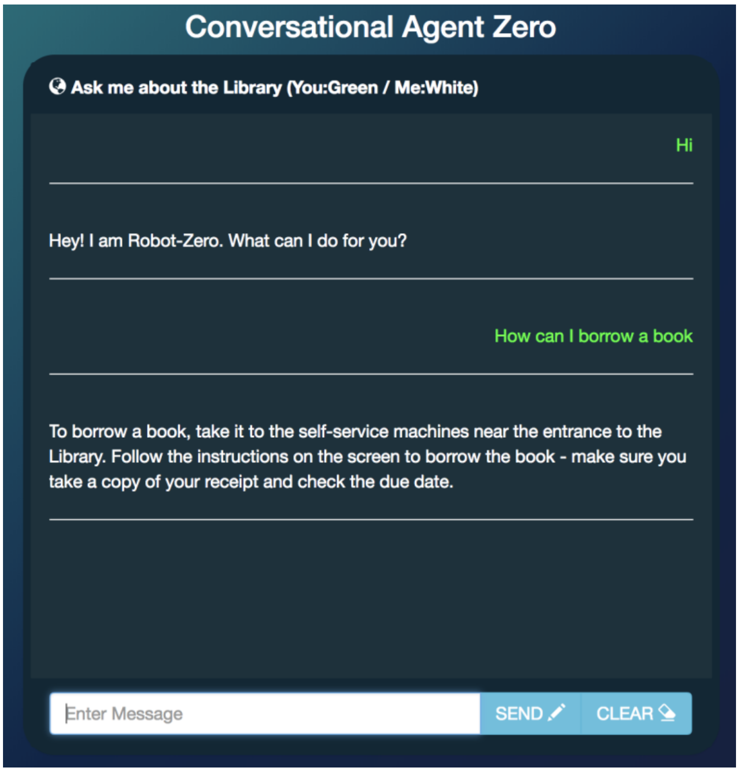
So glad we meet again!

How can I borrow a book

To borrow a book, take it to the self-service machines near the entrance to the Library. Follow the instructions on the screen to borrow the book -

The former implements receiving messages and other events sent by Messenger users by creating a webhook on the Facebook developer. The specific method of operation will be detailed in the User Documentation.

The latter implements the display of the chat dialog on the web page via HTML.



Conversational Agent Zero

Ask me about the Library (You:Green / Me:White)

Hi

Hey! I am Robot-Zero. What can I do for you?

How can I borrow a book

To borrow a book, take it to the self-service machines near the entrance to the Library. Follow the instructions on the screen to borrow the book - make sure you take a copy of your receipt and check the due date.

The agent receives the input sentences from other users by open source APIs ‘pymessenger’ and ‘Flask’. After the processing of the data, the agent selects an optimal response, then send the response back to users.

Data Preprocessing

Computers are different from humans in understanding natural language. They do not know the meaning of the word itself. Therefore, preprocessing the data received by the computer is needed. Data cleaning is a very important part. During preprocessing at the backend, tokenization and normalization of the query are first implemented, such as lowercasing words, distinguishing between different sentences, then removing stop words, stemming and tagging the words.

This format of conversational data is easy to be learned by machine learning algorithms.

Machine Learning

Machine learning is an important part for an agent to detect meaning and extract information. In this project, this function is mainly implemented on Dialogflow. Dialogflow is a Google-owned developer of human-computer interaction technologies based on natural language conversations. Dialogflow uses machine learning algorithms to understand natural language utterances, match them to intents, and extract structured data. Our conversational agent learns both from training phrases that we provide and the language models built into Dialogflow. Based on this data, it builds an algorithm for making decisions about which intent should be matched to a user utterance. Our training data is several dialog data which contains questions, answers, and intents. Intents describe the main topic of dialog and are used in classification learning algorithm.

Information Retrieval

This module is designed for the conversational agent to retrieve information efficiently in the corpus. Best Match algorithm will be used to improve the efficiency of searching for information. Whenever the conversational agent receives input statements from customers, it will analyze the topic/intention of the input statements firstly. Then it will focus on the specific topic/intention and retrieve information in the corpus. After that, the conversational agent will calculate a confidence value to evaluate the appropriation of answers. The confidence value ranges from 0.0(completely uncertain) to 1.0(completely certain). A threshold is set for the confidence value which is 0.6, and it helps the conversational agent perform intelligence.

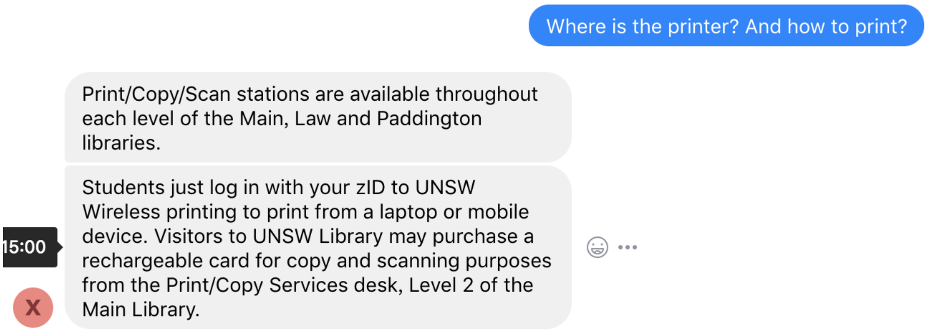
Implementation Challenges

During the implementation of this project, some challenges are met. Some tricks and special design are made in this project.

Tricks:

This section is used to illustrate some very useful small features of the agent, or we can call them tricks.

- Making a clause for user's question and then processed separately, so that when the user asks several questions at once, the agent can respond separately, and such an operation will also make the machine learning more accurate.

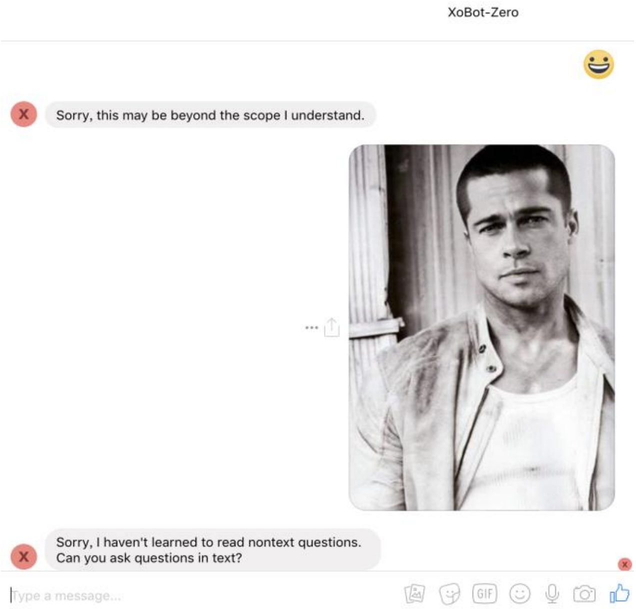


Where is the printer? And how to print?

Print/Copy/Scan stations are available throughout each level of the Main, Law and Paddington libraries.

Students just log in with your zID to UNSW Wireless printing to print from a laptop or mobile device. Visitors to UNSW Library may purchase a rechargeable card for copy and scanning purposes from the Print/Copy Services desk, Level 2 of the Main Library.

- Adding the recognition of the user's question. When the user sends a non-text question, because the agent itself cannot understand the picture content, an appropriate answer is made.

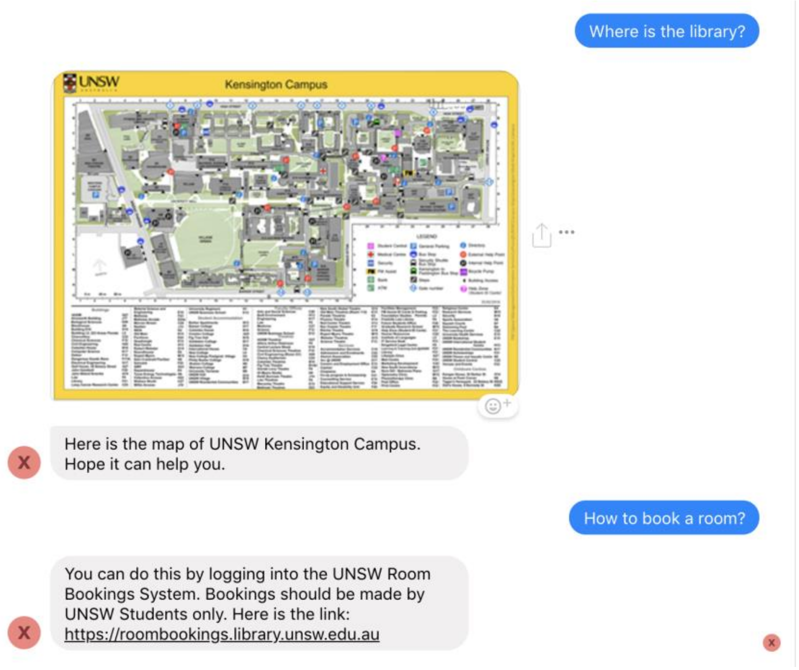


XoBot-Zero

Sorry, this may be beyond the scope I understand.

Sorry, I haven't learned to read nontext questions. Can you ask questions in text?

- Adding multiple forms of response for the agent, not limited to text responses. For example, image reply, and hyperlink reply.



Where is the library?

Here is the map of UNSW Kensington Campus. Hope it can help you.

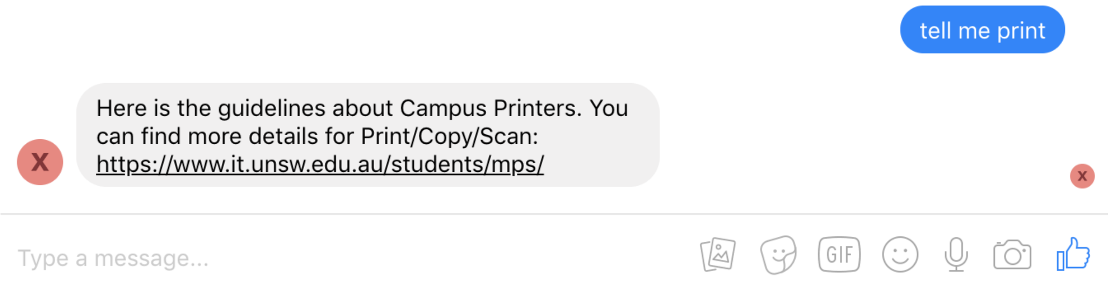
How to book a room?

You can do this by logging into the UNSW Room Bookings System. Bookings should be made by UNSW Students only. Here is the link: https://roombookings.library.unsw.edu.au

Special architecture/design

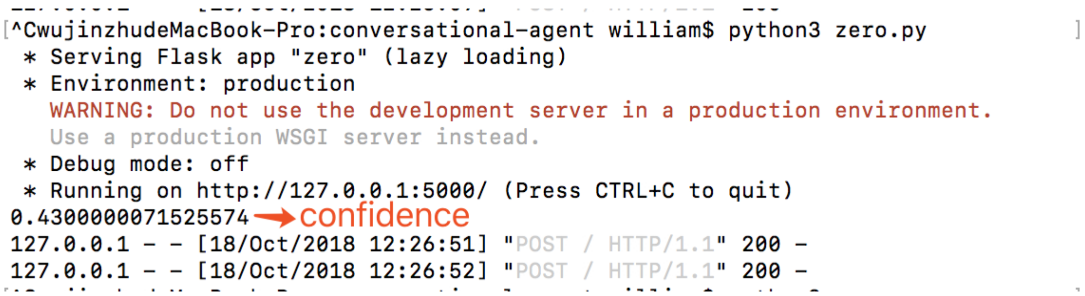
Since agent completes the understanding of natural language through machine learning, it has a certain confidence in the recognition of semantics. Therefore, when the backend obtains the intent, response, and confidence returned by Dialogflow, it will first judge the confidence, which represents the degree of matching for certain types of questions. Instead of blindly outputting the results of the Dialogflow process. For those responses with a low confidence, keyword extraction is done at the backend and then matches the answers that might be appropriate.

In this way, the agent can give useful help in the face of some very vague questions, rather than matching some wrong answers.



tell me print

Here is the guidelines about Campus Printers. You can find more details for Print/Copy/Scan: https://www.it.unsw.edu.au/students/mps/



```
[^CwujinzhudeMacBook-Pro:conversational-agent william$ python3 zero.py
 * Serving Flask app "zero" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
0.4300000071525574
127.0.0.1 - - [18/Oct/2018 12:26:51] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2018 12:26:52] "POST / HTTP/1.1" 200 -
```

As described in the above picture, when the user asks something like “tell me print”, we don't know what the user's specific problem is, but we know that it must be a question about 'print', and the confidence gained by the backend is only 0.43. Therefore, the agent will choose to reply a link to let the user get more information on his own, without responding to a specific type of question in detail.

User Documentation

To run the Facebook Messenger Chatbot, you need to make sure you have a few things installed:

- **Development Environment:** Python 3.6
- **Related library:**

✧ flask

✧ pymessenger

✧ nltk

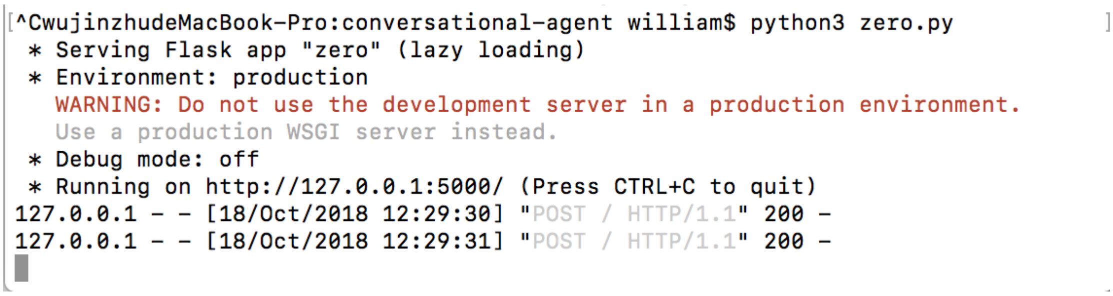
✧ json

✧ apiai

- **The other tool:** Tunneling(ngrok)

For Messenger Bot:

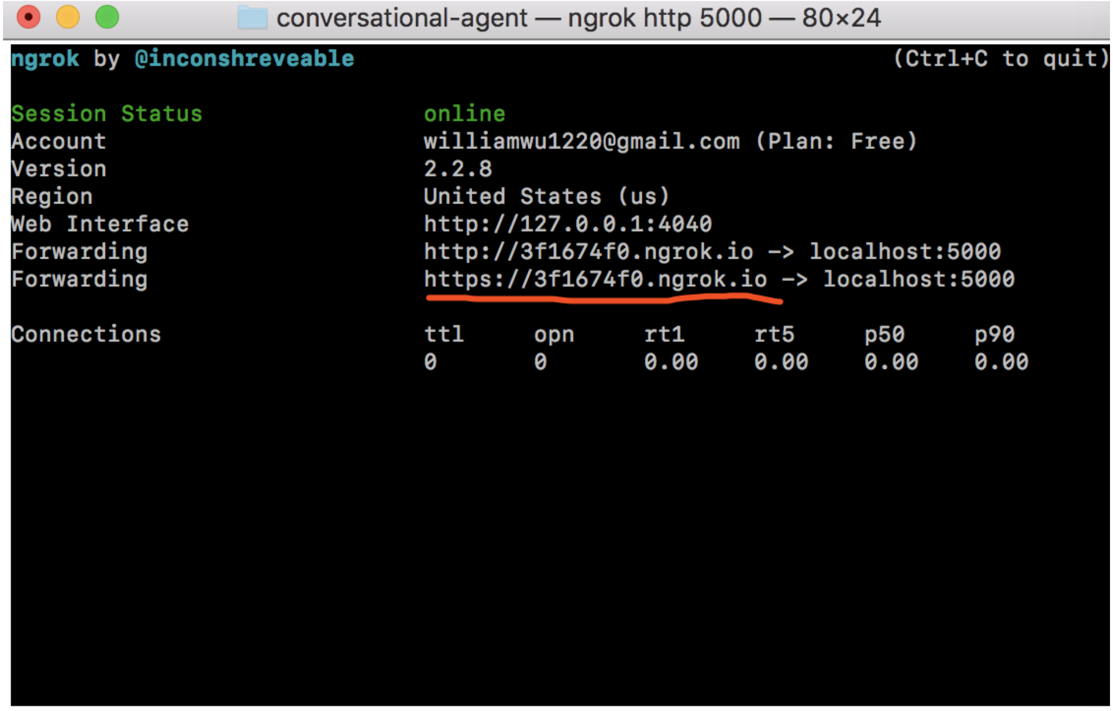
Firstly, run our code from the command line by typing `python3 zero.py` you will get a message like this:



```
[^CwujinzhudeMacBook-Pro:conversational-agent william$ python3 zero.py                ]
 * Serving Flask app "zero" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: off
 * Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
127.0.0.1 - - [18/Oct/2018 12:29:30] "POST / HTTP/1.1" 200 -
127.0.0.1 - - [18/Oct/2018 12:29:31] "POST / HTTP/1.1" 200 -
```

Then we need to use ngrok to host our code, ngrok give a chance to create a tunnel from the public internet to a port on our laptop, which allows us to run code on our laptop locally but make it accessible to users. Just run it by typing `./ngrok http 5000` and the screen will appear something like this:

(copy this Forwarding URL, and it will be used in the following step)



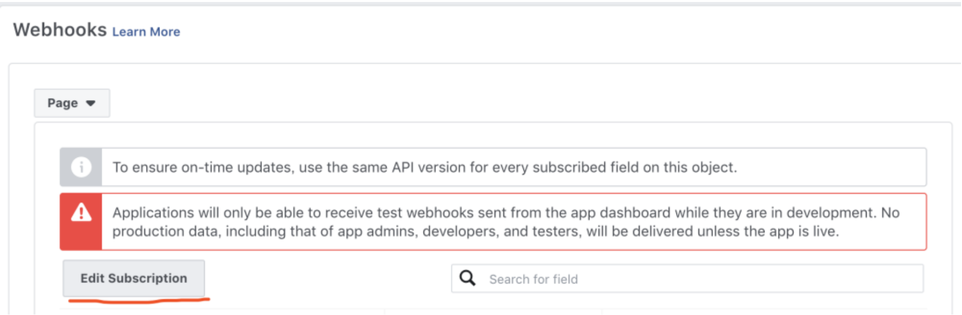
```
ngrok by @inconshreveable                                         (Ctrl+C to quit)

Session Status                online
Account                       williamwu1220@gmail.com (Plan: Free)
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://3f1674f0.ngrok.io -> localhost:5000
Forwarding                    https://3f1674f0.ngrok.io -> localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

After this step, we should connect our bot to Facebook Messenger. We have already created a Facebook Page, which named as XoBot-Zero. It is where users send messages to.

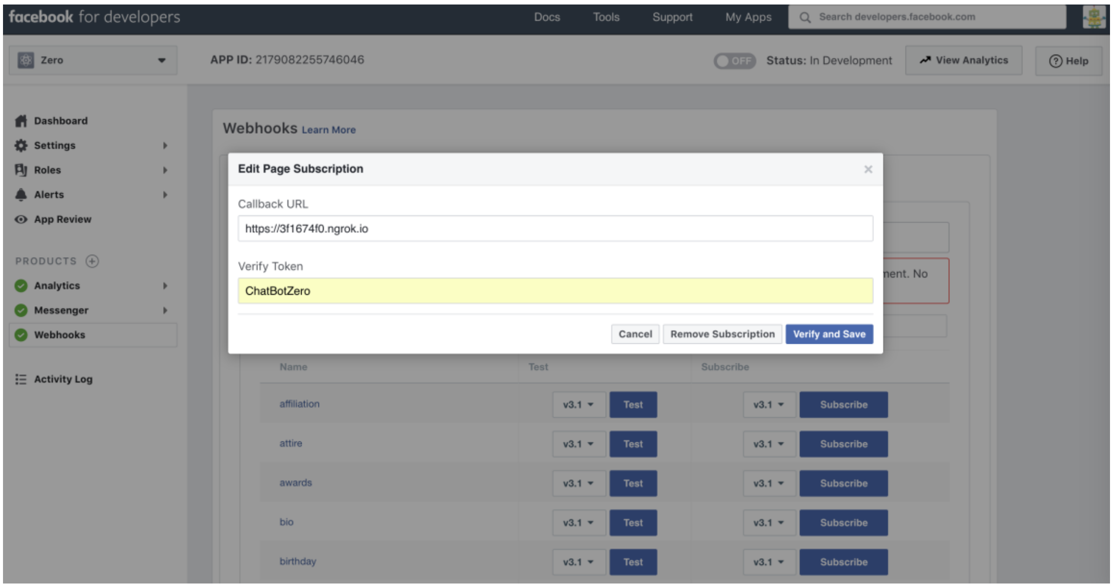
Log in the Facebook Developer, we can see a created app for Messenger Platform called Zero. Choose the tab in the left-hand corner of the page called ‘Webhooks’, and click ‘Edit Subscription’.



To ensure on-time updates, use the same API version for every subscribed field on this object.

Applications will only be able to receive test webhooks sent from the app dashboard while they are in development. No production data, including that of app admins, developers, and testers, will be delivered unless the app is live.

And paste Forwarding URL which we have copied before in Callback URL. Make sure the Verify Token are same to what in our code. In `zero.py`, we define it as ChatBotZero.



APP ID: 2179082255746046

Status: In Development

Edit Page Subscription

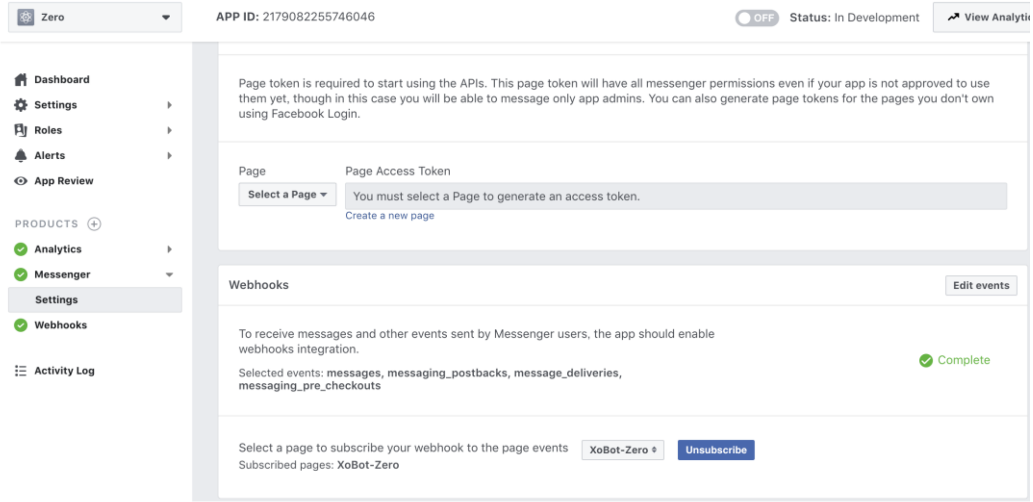
Callback URL: https://3f1674f0.ngrok.io

Verify Token: ChatBotZero

Name	Test		Subscribe	
affiliation	v3.1	Test	v3.1	Subscribe
attire	v3.1	Test	v3.1	Subscribe
awards	v3.1	Test	v3.1	Subscribe
bio	v3.1	Test	v3.1	Subscribe
birthday	v3.1	Test	v3.1	Subscribe

Click ‘Verify and Save’.

In Settings of Messenger, Just subscribe the pages: **XoBot-Zero**



APP ID: 2179082255746046

Status: In Development

Page token is required to start using the APIs. This page token will have all messenger permissions even if your app is not approved to use them yet, though in this case you will be able to message only app admins. You can also generate page tokens for the pages you don't own using Facebook Login.

Page

Page Access Token: You must select a Page to generate an access token.

Webhooks

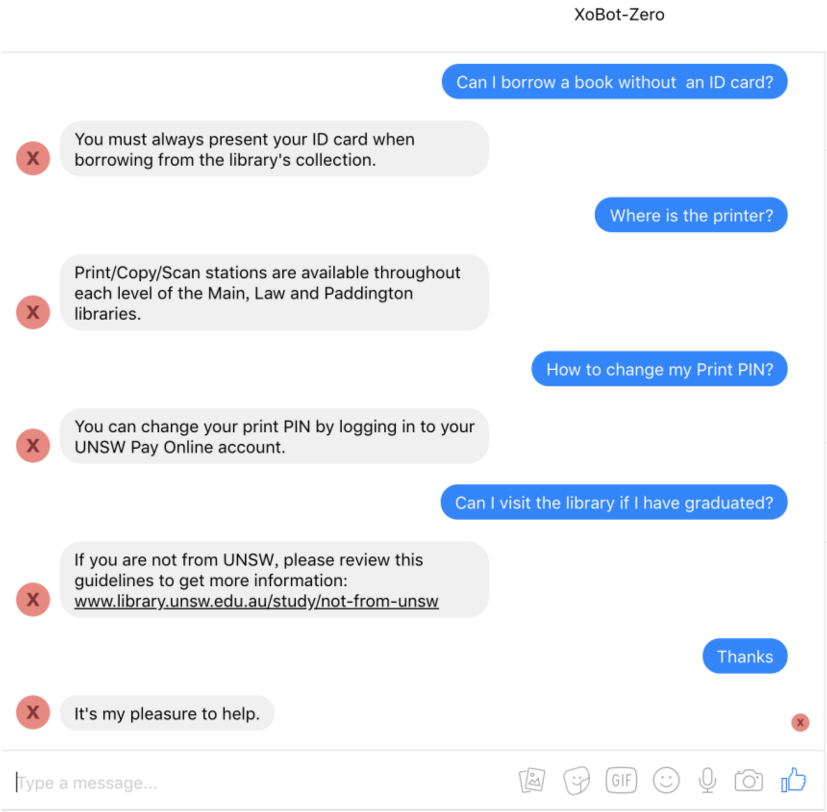
To receive messages and other events sent by Messenger users, the app should enable webhooks integration.

Selected events: **messages, messaging_postbacks, message_deliveries, messaging_pre_checkouts**

Select a page to subscribe your webhook to the page events XoBot-Zero

Subscribed pages: **XoBot-Zero**

Now we can test our chatbot, some dialogues as shown:



Can I borrow a book without an ID card?

You must always present your ID card when borrowing from the library's collection.

Where is the printer?

Print/Copy/Scan stations are available throughout each level of the Main, Law and Paddington libraries.

How to change my Print PIN?

You can change your print PIN by logging in to your UNSW Pay Online account.

Can I visit the library if I have graduated?

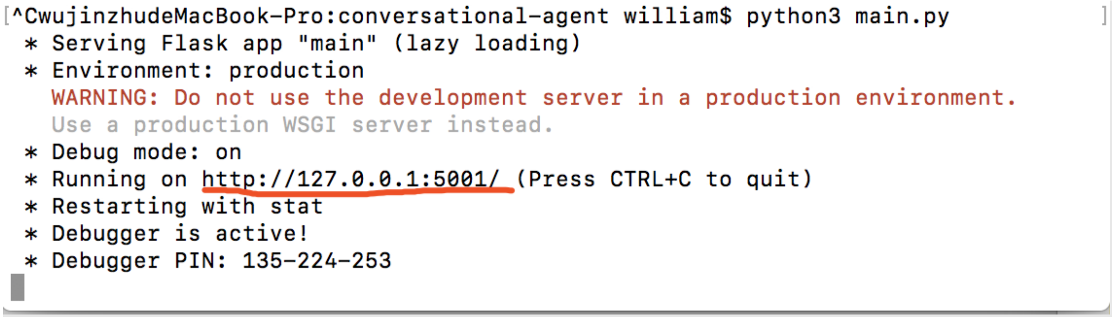
If you are not from UNSW, please review this guidelines to get more information: www.library.unsw.edu.au/study/not-from-unsw

Thanks

It's my pleasure to help.

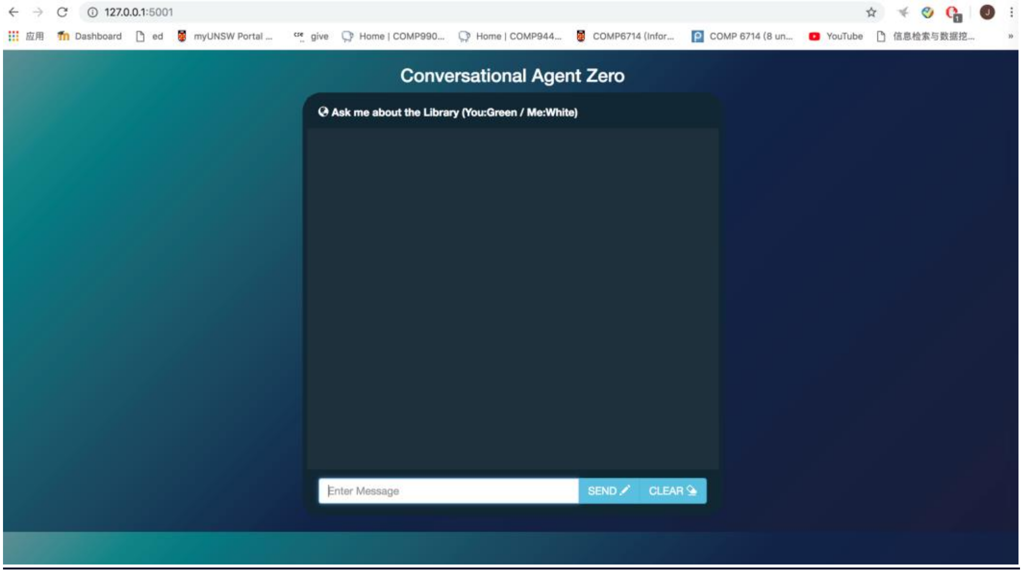
For Web Chat:

Just run our code from the command line by typing `python3 main.py,` you will see a message state like:



```
[^CwujinzhudeMacBook-Pro:conversational-agent william$ python3 main.py
 * Serving Flask app "main" (lazy loading)
 * Environment: production
   WARNING: Do not use the development server in a production environment.
   Use a production WSGI server instead.
 * Debug mode: on
 * Running on http://127.0.0.1:5001/ (Press CTRL+C to quit)
 * Restarting with stat
 * Debugger is active!
 * Debugger PIN: 135-224-253
```

And then type the URL in the browser. Now you can see the WebChat dialog.



Important explanation：

1. Since the current Facebook application is in an undisclosed stage, it is necessary to be the administrator of the application to run the robot. If a test requires, This Facebook account is used for setting webhook in facebook developer and testing bot.

   Facebook account: williamwu1220@gmail.com

   Password: project9900
2. Web chat is just to prove that our backend can be plugged into multiple channels. Because we did special processing on Messenger Bot, including some tricks and extra design, web chat will behave worse than Messenger, and some inappropriate responses may appear. Testing on Messenger, the performance will be better.