Chatbot using Discord & Wit.ai

Soumaya SABRY

13 November 2024

The assignment must be submitted as a Python notebook with outputs of each cell visible
&
For your Discord chatbot, it is required to include a video demonstrating its functionality.

According to Fig.1, integrated Chatbots require two parts:: a conversation/canvas channel for connecting with users such as Skype, Telegram, WeChat, Slack, Discord, WhatsApp, Messenger..etc. On the other hand, artificial intelligence (AI)/natural language processing (NLP) for making the chatbot more intelligent and practical.Several NLP-as-a-Service platforms exist, such as IBM Watson Assistant, Microsoft's Azure Conversational Language Understanding, Google Cloud Natural Language API, Wit.ai, and Google's Dialogflow. Each platform proposes its service at a certain price and different package which can or can't include free trials.

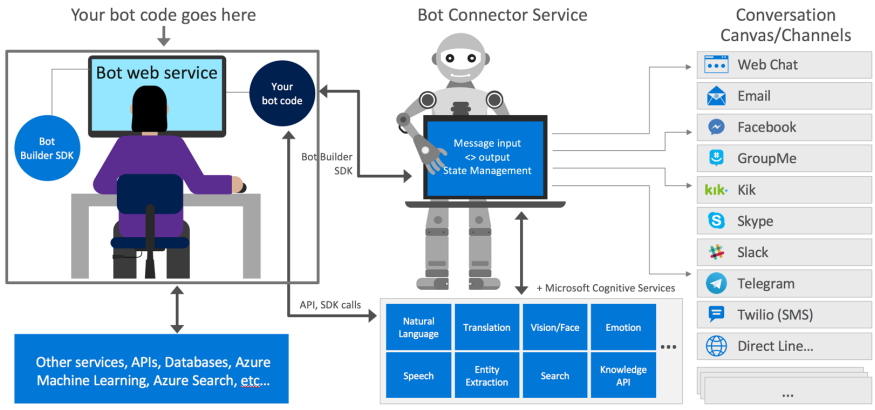


Figure 1: Integrated chatbot Framework using a Bot connector.
Taken from this blog

These services allow us to use natural language understanding (NLU), dialogue managers, and natural language generation (NLG) methods. In general, organizations choose to design or leverage existing platforms for the following reasons::

- They offer intuitive NLP tools, allowing easy definition of workflows and responses.
- Adding files, images, and emojis to enhance chatbot interactions is easy.
- Designing user-friendly buttons, quick replies, and sliders enables a seamless omnichannel experience.
- Training the chatbot, adjusting its settings, and integrating it into a website or apps is simplified.
- These tools improve conversational flow and engagement with customers.

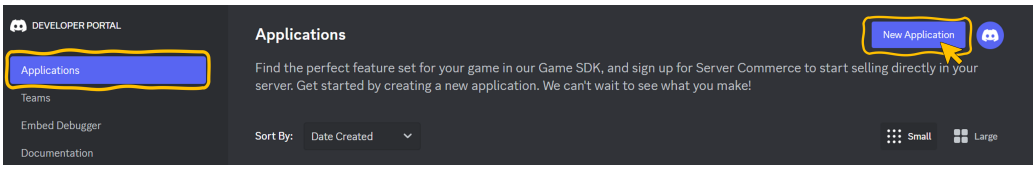
Among the existing list of conversation channels, we will use the *Discord Bot* and *Wit.Ai* as a platform for chatbot development. Wit.ai is a free chatbot software that lets you easily create text or voice-based bots on your preferred messaging platform. Wit.ai learns human language from every interaction and leverages the community: what is learned is shared across developers.
Read the following blog, if you wish to gain a better understanding of chatbot platforms: `https://cynoteck.com/fr/blog-post/best-chatbot-development-frameworks/`

1 Discord Bot

Let's start by creating a simple *Discord Bot* that repeats what the user sends.

1.1 On Discord

After logging into the Discord developer portal, you will be creating your application. An **application** allows you to interact with Discord's APIs by providing authentication tokens, designating permissions ...etc. Keep in mind that Bot-related APIs are only a subset of Discord's total interface.
Click on *New Application*, then enter a name and click *Create*.



Next, for your code to actually be manifested on Discord, you will create a bot user. Start by navigating to the *Bot* tab on the left-hand side. If you don't see a bot with the same name as your application, click on the Add Bot button to create one. Next, go to the *Installation*, and make sure to uncheck the User Install option.

Back to the bot's tab, update its name if you want to give it a name more bot-likely. Then, disable the *PUBLIC BOT* permission, so that only you can add your bot where you want, and enable the *MESSAGE CONTENT INTENT* permission which is necessary for your bot to read messages sent by users.

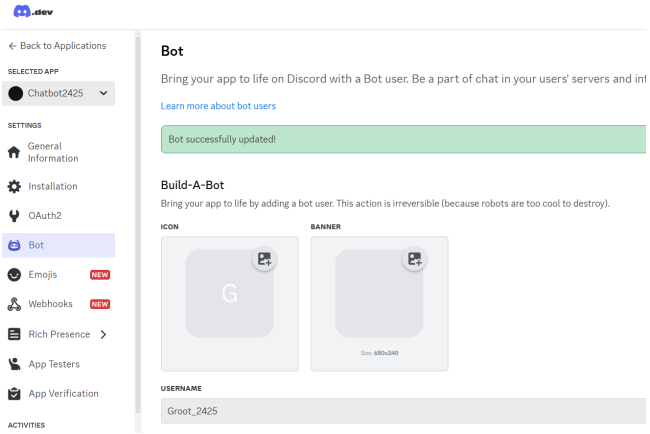


Figure 2: Bot page on the Discord developer portal

The final step is to create a new server on Discord (or use an existing one), which serves as a group of channels where users can chat and interact. Once your server is ready, you'll see the list of users on the right and the available channels on the left. Now, it's time to add your bot to this server. Unlike regular users, a bot can't accept invites directly, so you'll need to add it using the OAuth2 protocol.

Go back to the developer portal and select the *URL generator* from the *OAuth2* tab on the left-hand navigation. Then, scroll down and select bot from the SCOPES options and Administrator from BOT PERMISSIONS. Discord has generated your application's authorization URL with the selected scope and permissions. Copy the URL, then paste it into your browser, and select your server from the drop-down options. Click *Authorize*, and you're done! If you go back to your server, then you will see that your chatbot has been added.

1.2 On Python

You will be using *discord* library, which is a python library that exhaustively implements Discord's APIs in an efficient way. This includes utilizing python's implementation of Async IO. Begin by installing discord using pip:: *pip install -U discord*

Write a python code that creates the discord Connection. The first step is implementing an instance of Client discord using the following code:

```
import discord

TOKEN = "Your Token"
class MyClient(discord.Client):
    async def on_ready(self):
        print('Logged in as' + str(self.user.name))
        print('The chatbot Id :: ' + str(self.user.id))

client = MyClient(intents=discord.Intents.default())
client.run(TOKEN)
```

To run this code, you will need to replace "Your Token" with your bot's token, which you can get from the developer portal, go to your Bot page in the Discord Developer, then click Reset Token to generate a new token. This will allow you to view and copy the token needed for your bot. Finally, run the code to make sure there is an established connection.

If all is correct, the next step will be to add a function in the MyClient class with the signature *async def on_message(self, message)*, which will be responsible for receiving and sending messages from your server on discord, and make sure that the chatbot doesn't interact with its own messages.

To receive the content of the user messages, you have to change the default intents on your client variable, here is a link that could guide you: `https://discordpy.readthedocs.io/en/stable/api.html#discord.Intents`.

1.3 Practice

After completing the code needed to establish the connection, and receive & send messages on discord, as a first step, start by replying only to the user with the same string they sent. Secondly, use your weather bot from the previous session, and adapt your main to be integrated into the Discord platform.

Don't stop there, continue .... be creative, and explore other techniques used by discord to make your chatbot more robust, you can let the chatbot respond with images, GIFs, or even emojis...etc. you can make it mention the name of the user it's chatting with...etc.

2 Wit.ai

As we saw in the previous lessons, intents, and other related entities should be extracted from each input of the user.

For example in this figure, the "restaurant booking" intent should extract three more name entities: number of people, place, and date. In order to extract them, we used regular expressions before, which require manually defining several cases for the same intention that is not obvious in all contexts. For this reason, it would be interesting to introduce some machine learning (ML) techniques, especially in natural language processing (NLP), that will help you capture intent and entities much more easily and automatically.

2.1 Change Chatbot topic

First of all, let's change the chatbot topic from "*WeatherBot*" to "*MovieBot*".

Here you are free to find any other topic to build the session chatbot on it. You need to choose a topic that defines multiple intents and entities and find a good API related to it.

What is the *MovieBot*? Generally, we want to design a chatbot that knows everything about the movies. This chatbot should understand and reply back to the following questions such as::

- Tell me about Iron Man...
- Who directed Iron Man?
- When was Iron Man released?
- Who was the main actor(s) of Iron Man?

This list is not exclusive, you can think of as many questions and intents as you can think of about films, actors, directors, etc.

For the *MovieBot*, you will need a database (API) for collecting the information from. **TheMovieDB** is the one, to get your own API from it, follow these steps:

1- Start by creating an account on their website `https://www.themoviedb.org/signup`

2- Go to the setting, then choose the API tab in the left side list

3- Click on the link to generate a new API key and follow the instructions

4- Copy the API key, and save it into your python code, you will need it at the end.

For more details on the steps look up this link `https://developers.themoviedb.org/3/getting-started`

Let's have a quick look at TheMovieDB documentation *here* to have a better understanding of the API and its returned query. You can find as well several python wrappers for this API: `https://www.themoviedb.org/documentation/api/wrappers-libraries`, which are very useful instead of creating your own queries. Choose one of them and start your tests on python.

With respect to your Moviebot, you can use various endpoints. For instance, one interesting endpoint for the movie bot is *SEARCH*, search movies. You can find it in the left bar list *here*. After clicking on the search movies, you will find query string parameters. The interesting point is that you can 'try it out' manually. Check the response, and check different formats of the results as JSON or raw data.

2.2 Starting a Wit.ai App

Wit.ai is a natural language understanding (NLU) platform owned by Meta (formerly Facebook), designed to help developers create conversational bots, applications, and voice-enabled devices. Wit.ai provides a cloud-based API for natural language processing and speech recognition, allowing developers to build applications that can interpret text and spoken language. It remains a popular tool for developing conversational interfaces, especially for platforms like Facebook Messenger.

2.3 Train it to detect Intents and Entities

Start with signing to `https://wit.ai/`, create a new App for your chatbot. Then, go to your App dashboard and select the *Understanding* tab on the left. This is where you will train your bot.

Train your chatbot for understanding the following sentences and extracting their intents and entities, then click on " Train & Validated" after each sentence::

- tell me about the lord of the rings → intent: movieinfo | entity:: movieName : lord of the rings
- tell me about the spider man → intent: movieinfo | entity:: movieName : spider man
- tell me about the eternal sunshine of spotless mind → intent: movieinfo | entity:: movieName : eternal sunshine of spotless mind
- tell me about the spider man (2007) → intent: movieinfo | entity:: movieName: spider man, releaseYear: 2007
- tell me about the spider man (2012) → intent: movieinfo | entity:: movieName: spider man, releaseYear: 2012

To better train your model on Wit.ai, you should give more similar examples to the system. This is based on artificial intelligence and machine learning methods. The more you train your model, the better it predicts the considered goals.

If you click on intents, you can see the list of expressions while by clicking on free texts and keywords you can see the list of keywords. You can improve your system by defining synonyms for your keywords.

Now, train it for all the intents that you had planned such as *director* intent or *releaseYear* intent ...etc, and extract each entity accordingly. Here are some examples::

- who directed the lord of the rings? → intent: director| movie: the lord of the rings
- who directed spider man? → intent: director | movie: spider man
- when was the untouchable released? → intent: releaseYear| movie: untouchable
- when was the interstellar released? → intent: releaseYear| movie: interstellar

Your app doesn't know a lot yet, but it will start to recognize the intent of getting information about movies. When Wit starts recognizing the sentence, the intent will be pre-filled along with a level of confidence (between 0 and 1). The more examples you validate, the better Wit will understand. The confidence level should start improving.

So, keep training the system with more examples. It is required to do at least 20 for each intent, try to have as much variety in the question format as you can. Don't forget to add some basic intents as well, as you did in the last session, such as greeting, bye, help...etc.

If any of the above steps were unclear, you can read more about it in their documentation:`https://wit.ai/docs/quickstart`.

2.4 Query Wit from Python

Last step is linking Wit.ai with your chatbot via python. You can use the Wit python library: `https://github.com/wit-ai/pywit`. Before you will need a token which can be found in the setting of the management section on the left-hand of the wit.ai dashboard. Then use the following code and start testing and exploring the outputs.

```
from wit import Wit
access_token = "Your_Token"

client = Wit(access_token)
nlp_data = client.message('tell me about the Spiderman (2007)')
```

Write a function namely "extract_entity" for receiving NLP data (gathered from the wit.ai) and entity name which returns back the value of the selected entity. This function should only return back the entity value with a confidence equal or higher than 0.5. The received NLP data is an object with the following format:

```
{
   'entities': {
               " movie ":" Spiderman "
               " releasedYear ": " 2007 "
               },
   'intents': {
               'Confidence': 0.7841239431941158,
               'id': '3528338274110810',
               'name': 'movieinfo'},
   'text': 'tell me about the spider-man (2007)',
   .....
}
```

3 Bringing it all together

Now you have a trained NLP system on Wit.ai, a channel on Discord, and an API as a database. It is time to create your chatbot by integrating all these components together.
Start by writing your `main` function with all the necessary parts. In general, your main function workflow should be as follows:

- Receive the user's message input on Discord
- Send it to the Wit.ai model
- Capture intent and entities (if any)
- Query TheMovieDB API to retrieve the required data
- Rephrase the data into a full-sentence response
- Reply to the user on Discord

After setting up this basic flow, try to complete your *MovieBot* by implementing your own innovative ideas to make the chatbot more engaging. This step is optional, but here are some suggestions:

1. Add an intent *Poster* to retrieve the movie poster (as an image) from TheMovieDB and send it to the user on Discord.
2. Include traits alongside entities. Traits do not refer to specific text parts but act on the entire sentence to categorize attributes like politeness, sentiment, etc.

Once you've finished your project, upload it to Moodle under the Session 3 assignment space.

Good Luck ! 💪