緯育 TibaMe

Module 02
LINE Bot實作

學習目標：

- 2-1: 啟用 LINE Bot
- 2-2: 回覆 / 推播訊息
- 2-3: 整合語音辨識

緯育 TibaMe

2-1: 啟用 LINE Bot

學習目標：

- 設定 LINE Bot 聊天機器人功能
- 使用 Flask 架設 web server
- 接收用戶訊息

本介紹資訊取自於 2020/03/03

設定 Line Bot 聊天機器人功能

- 先至 Line Official Account Manager 主頁中
  - 點擊「聊天」，設定回應模式

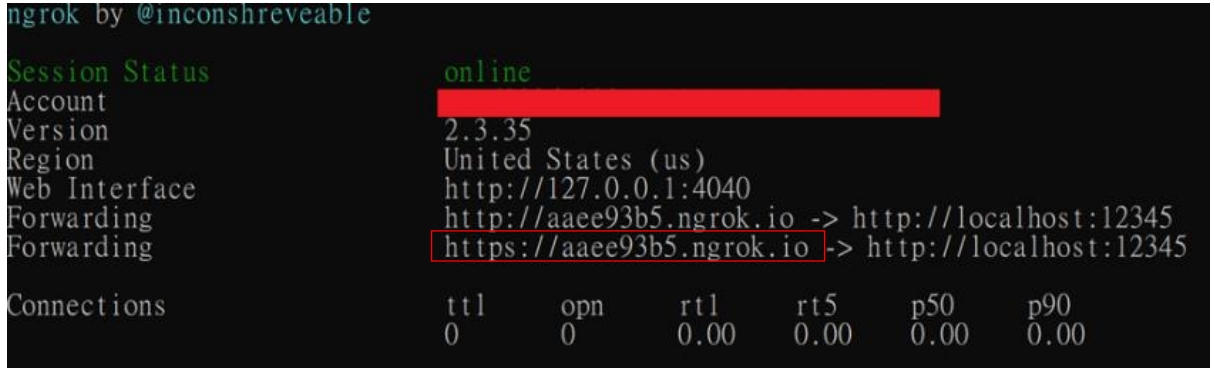
設定 Line Bot 聊天機器人功能

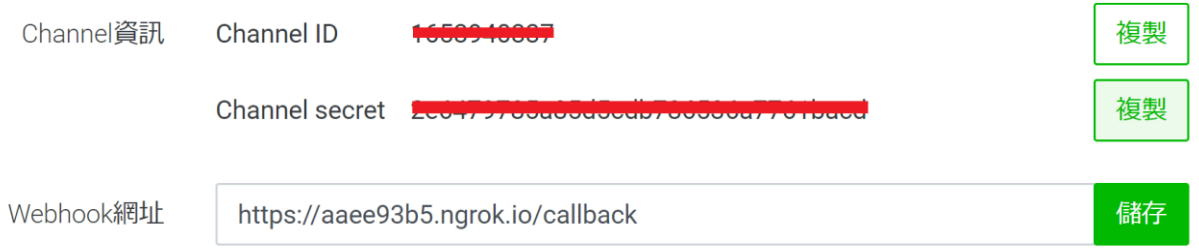
- 在「基本設定」中
  - 設定「回應模式」為：聊天機器人
- 在「進階設定」中
  - 設定「自動回應訊息」為：停用
  - 設定「Webhook」為：啟用
  - 並且點選「Message API 設定」，啟用「Message API」
    - 填寫基本資料，「服務提供者」名稱可任意填寫
    - 非必填資料可以跳過

設定 Line Bot 聊天機器人功能

- 獲得 Message API 相關資訊
  - 其中的「Channel 資訊 > Channel secret」將填入於稍後的程式中
  - 開啟 ngrok，取得 Webhook URL

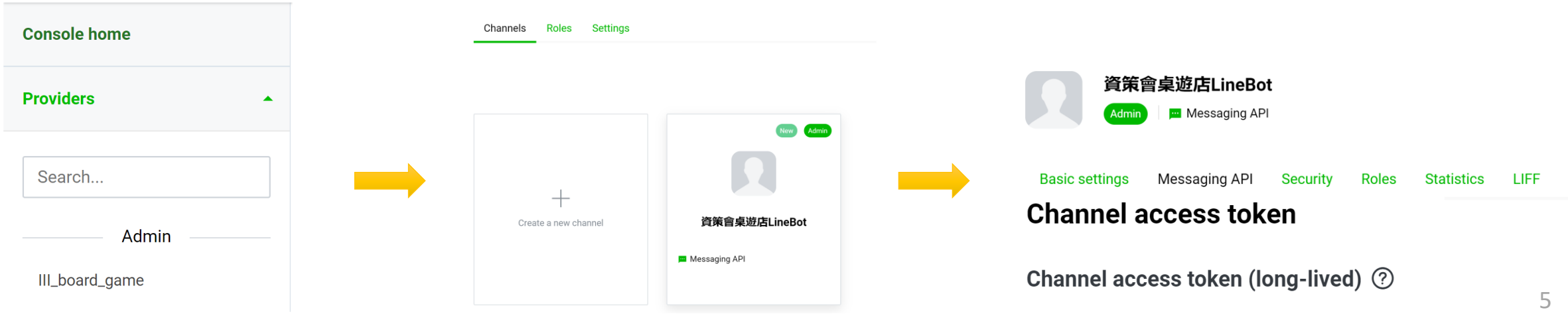
    ngrok http 12345
  - 在此「Webhook 網址」中填入：https://your_webhook_url/callback
  - 點選下方 「Line Developers」





設定 Line Bot 聊天機器人功能

- 在 Line Developers 中
  - 登入後點選個人頁面
  - 於左邊選單中點選先前輸入過的服務提供者名稱
  - 點擊右側 Channel 中的官方帳號
  - 點選 Message API，issue 頁面最下的 「Channel Access Token」



使用 Flask 架設 web server

- 什麼是 Flask？為什麼需要 Flask？
  - 在用戶傳送訊息到官方帳號後，LINE 將會發送一個 POST 請求至先前設定的 Webhook
  - 為了接收此 POST 請求，需要在 Bot server 中實作 web server
  - 將接收到的請求，利用 Message API 處理後回應用戶
  - Flask 是一個輕量級 Web 應用框架
    - 可以輕易的使用 Python 建置簡單的 web server
    - 並沒有預設使用的資料庫
  - 安裝 Flask 套件於 Python

使用 Flask 架設 web server

- 架設 web server 並且與 Line Bot 連結
  - 安裝 Line Bot API 套件於 Python

    pip install line-bot-sdk

  - 引用與設定

```python
# import flask related
from flask import Flask, request, abort
# import linebot related
from linebot import (
    LineBotApi, WebhookHandler
)
from linebot.exceptions import (
    InvalidSignatureError
)
from linebot.models import (
    MessageEvent, TextMessage, TextSendMessage,
    LocationSendMessage, ImageSendMessage, StickerSendMessage
)

# create flask server
app = Flask(__name__)
# your linebot message API - Channel access token (from LINE Developer)
line_bot_api = LineBotApi(' ')
# your linebot message API - Channel secret
handler = WebhookHandler(' ')
```

使用 Flask 架設 web server

- 架設 web server 並且與 Line Bot 連結
  - Handle 所收到的 POST 請求

```
22 @app.route("/callback", methods=['POST'])
23 def callback():
24     # get X-Line-Signature header value
25     signature = request.headers['X-Line-Signature']
26 
27     # get request body as text
28     body = request.get_data(as_text=True)
29     app.logger.info("Request body: " + body)
30 
31     # handle webhook body
32     try:
33         handler.handle(body, signature)
34     except InvalidSignatureError:
35         print("Invalid signature. Please check your channel access token/channel secret.")
36         abort(400)
37     return 'OK'
```

接收用戶訊息

- LINE Platform 將用戶訊息轉換為 JSON 後
  - 可以分類為多種 Webhook event，最常見為用戶訊息事件 (MessageEvent)
  - 使用 Python decorator 搭配 handler.add 宣告欲接收並處理的 event

- 範例：

```
39 # handle msg
40 @handler.add(MessageEvent, message=TextMessage)
41 def handle_message(event):
42     # ...
```

接收用戶訊息

- 解析 event 內容，獲得用戶資訊與訊息
  - event 由 JSON 格式轉成
  - 參考 TextMessage JSON 格式解析資料，取得用戶專屬ID與傳送之訊息
  - 利用 Line Bot API 取得用戶於 LINE 上設定之姓名
- TextMessage JSON example

```
{
  "replyToken": "nHuyWiB7yP5Zw52FIkcQobQuGDXCTA",
  "type": "message",
  "mode": "active",
  "timestamp": 1462629479859,
  "source": {
    "type": "user",
    "userId": "U4af4980629..."
  },
  "message": {
    "id": "325708",
    "type": "text",
    "text": "Hello, world!"
  }
}
```

```
42 def handle_message(event):
43     # get user info & message
44     user_id = event.source.user_id
45     msg = event.message.text
46     user_name = line_bot_api.get_profile(user_id).display_name
```

接收用戶訊息

- 將取得之資料印出

```
57 def handle_message(event):
58     # get user info & message
59     user_id = event.source.user_id
60     msg = event.message.text
61     user_name = line_bot_api.get_profile(user_id).display_name
62 
63     # get msg details
64     print('msg from [', user_name, '](', user_id, ') : ', msg)
```

- 呼叫 app.run() 啟用 web server

```
67 # run app
68 if __name__ == "__main__":
69     app.run(host='127.0.0.1', port=12345)
```

2-1 Demo

- Run Demo_2-1.py 範例程式碼
- 於 Line 中傳送訊息至此官方帳號
- 觀看 log，是否正確輸出
  - msg from [user_name] (id) : msg
  - 範例：

```
 * Running on http://127.0.0.1:12345/ (Press CTRL+C to quit)
receive msg
127.0.0.1 - - [                                    ] "POST /callback HTTP/1.1" 200 -
msg from [          ](                                        ) :  您好
```

緯育 TibaMe

學習目標：

2-2: 回覆/推播訊息

- 回覆用戶訊息
- 推播訊息

回覆用戶訊息

- 在 MessageEvent 的 JSON 中，存有一個 replyToken
  - 可使用於即時回覆訊息
  - 可經由 event.reply_token 取得
  - 屬於被動式回覆，不需要付錢
- 回覆用戶的訊息依照不同類型分為不同物件，如：
  - 文字訊息：TextSendMessage
  - 貼圖： StickerSendMessage
  - 圖片訊息：ImageSendMessage
  - 地點：LocationSendMessage

回覆用戶訊息

- 回覆訊息
  - line_bot_api.reply_message(token, message_objects)
  - 文字訊息：TextSendMessage(text = ‘欲回覆文字’)

- 範例 - 回聲機器人：
  - 回覆 用戶姓名 以及 所收到的訊息

```
51    # reply text_msg : user_name with user_msg
52    line_bot_api.reply_message(event.reply_token,
53                               TextSendMessage(text = user_name + ' : ' + msg))
```

回覆用戶訊息

- 回覆訊息
  - 貼圖：

```
56      line_bot_api.reply_message(event.reply_token,
57                                 StickerSendMessage(package_id='1',sticker_id='1'))
```

  - 圖片訊息：

```
60      line_bot_api.reply_message(event.reply_token,
61                                 ImageSendMessage(original_content_url='img_url',
62                                                  preview_image_url='img_url'))
```

  - 地點：

```
65      line_bot_api.reply_message(event.reply_token,
66                                 LocationSendMessage(
67                                      title='Store Location',
68                                      address='Taipei 101',
69                                      latitude=25.033981,
70                                      longitude=121.564506))
```

推播訊息

- 推播訊息介紹
  - 使用 user_id 指定回覆者
  - 推播訊息的物件與回覆相同，常用類型如先前所介紹
    - TextSendMessage, StickerSendMessage, ImageSendMessage…
  - 為「主動式」傳送訊息，每月有限額的訊息則數，超過要付費
  - 輕用量帳戶（免費用戶）目前每月僅可傳送 500 則推播訊息

```
73      line_bot_api.push_message(user_id,
74                                TextSendMessage(text = '您好^^'))
```

方案資訊取自於 2020/03/03

2-2 Demo

- Run Demo_2-2.py 範例程式碼
- 於 Line 中傳送訊息至此官方帳號
- 使用 list 回覆訊息
  - 包括 文字, 貼圖, 圖片, 地點
- 回覆完後主動推播一則文字訊息

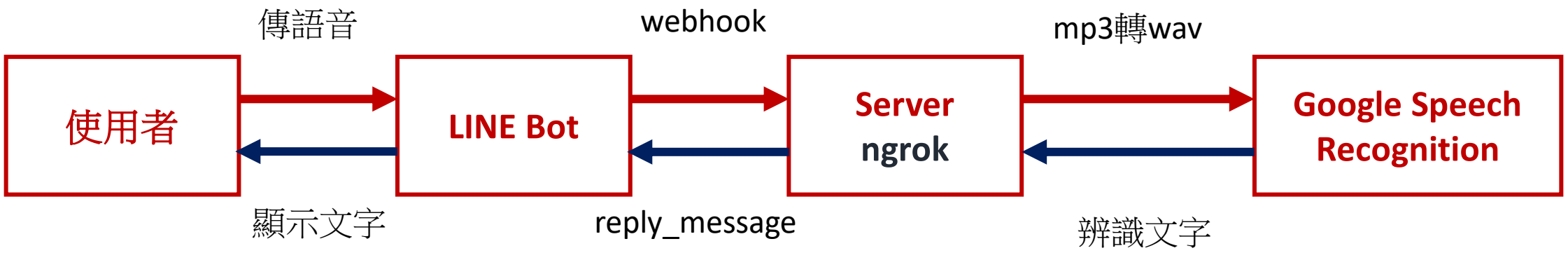
2-3: 整合語音辨識

學習目標：

- LINE Bot 語音辨識流程
- 實作LINE語音辨識
- ffmpeg 轉檔

LINE Bot 語音辨識流程

- 使用者傳語音至 LINE Bot
- LINE Bot 用 webhook 將資料傳至 ngrok 開的 Server
- Server 將音檔轉存成 mp3
- Server 將音檔轉成 wav，執行 Module 3 教的語音辨識
- 回傳文字結果到 LINE Bot



實作 LINE 語音辨識

- 要接收語音，必須使用以下程式碼
- 語音訊息是 AudioMessage
- Server 會將語音存在 recording.mp3
  - LINE會壓縮語音，取樣率僅 10000 Hz

```
@handler.add(MessageEvent,message=AudioMessage)
def handle_aud(event):

    name_mp3 = 'recording.mp3'
    name_wav = 'recording.wav'
    message_content = line_bot_api.get_message_content(event.message.id)

    with open(name_mp3, 'wb') as fd:
        for chunk in message_content.iter_content():
            fd.write(chunk)
```

ffmpeg 轉檔

- 可以透過附檔的 ffmpeg.exe 進行轉檔
  - 必須將 ffmpeg.exe 跟主程式 Demo_2-3.py 放在一起
- 轉檔的程式如下

```
import os
os.system('ffmpeg  -y  -i  from.mp3 to.wav')
```

  - -y：無條件覆蓋檔案
  - from.mp3：原始檔案路徑
  - to.wav：要轉成的檔案

執行語音辨識

- 以下為完整的 handler 程式碼
  - 執行 flask 後，記得要在本機端開ngrok指定port 12345

```
60@handler.add(MessageEvent, message=AudioMessage)
61def handle_audio(event):
62
63    name_mp3 = 'recording.mp3'
64    name_wav = 'recording.wav'
65    message_content = line_bot_api.get_message_content(event.message.id)
66
67    with open(name_mp3, 'wb') as fd:
68        for chunk in message_content.iter_content():
69            fd.write(chunk)
70
71    os.system('ffmpeg -y -i ' + name_mp3 + ' ' + name_wav + ' -loglevel quiet')
72    text = transcribe(name_wav)
73    print('Transcribe:', text)
74    line_bot_api.reply_message(event.reply_token, TextSendMessage(text = text))
```

LINE Bot 接收語音，存至 recording.mp3

← mp3轉檔wav

← 執行語音辨識

← 回傳給使用者

2-3 Demo

- 執行 Demo_2-3.py 範例程式碼
- 參考課程2-1的內容
  - 執行 flask 與 ngrok
  - 設定 LINE webhook
- 由自己的手機，對機器人傳送語音
- 說一段中文
  - 你好嗎
  - 今天會下雨嗎
  - 有什麼遊戲

線上Corelab

- 題目1：啟用聊天機器人
  - 將用戶名、用戶ID、用戶訊息顯示出來
- 題目2：回覆訊息
  - 使用邏輯判斷式 if...elif..else 來觸發不同回覆
- 題目3：整合語音辨識
  - 接收語音並回覆相對應的文字

本章重點精華回顧

- 什麼是 Flask，為什麼需要使用它
- Webhook event
- 回覆訊息與推播訊息的應用及差別
- 整合語音辨識模組
- 對 LINE Bot 說話，能顯示文字

Lab: LINE Bot 實作

- Lab01: 啟用聊天機器人並接收用戶訊息
- Lab02: 回覆與推播訊息
- Lab03: LINE Bot 建立語音辨識

Estimated time:

20 minutes


THANK YOU