

Nombre:

Ricardo Jara

Fecha:

23/05/2020

Materia:

Sistemas Expertos

Tema:

Examen IA

Chatbot usando Watson Assistant, Language Translator y Text to Speech.

Inteligencia Artificial

Se desarrolla un sistema en Python el cual se conecta a Messenger y está pendiente de los mensajes enviados a una cuenta, cuando los mensajes son recibidos estos son traducidos utilizando el servicio de , **Language Translator** de IBM.

Una vez los mensajes son traducidos estos son enviados como mensaje al Chatbot de **Watson Assistant** éste nos devuelve una respuesta de acuerdo a la configuración del Chatbot en la página de IBM.

Con la respuesta obtenida del Chatbot utilizamos el servicio de **Text to Speech** para convertir la respuesta de Chatbot en audio y así responder al mensaje de Messenger con un audio y texto a la vez.

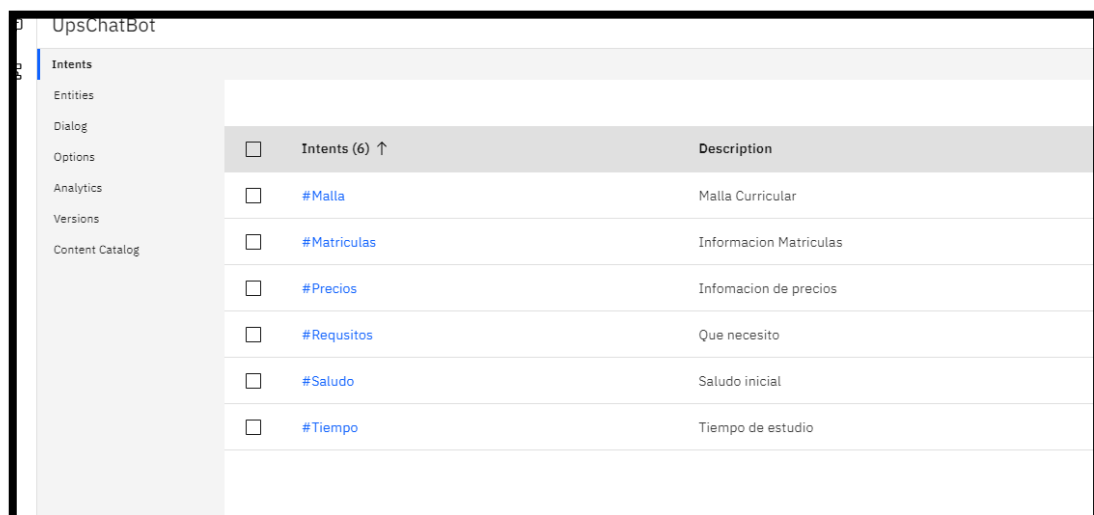
• Desarrollo.

- Creamos nuestro servicio en <https://cloud.ibm.com>.

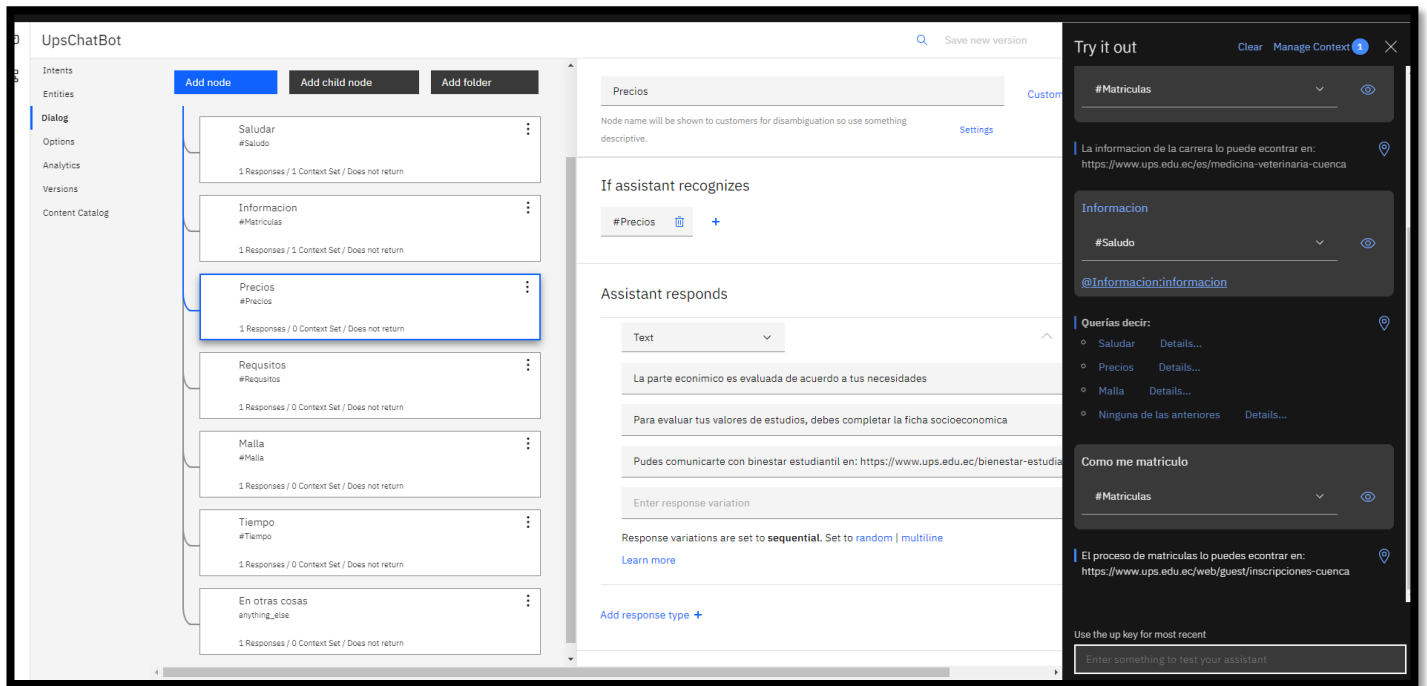


Services (3)				
ChatBotExamen	Default	Dallas	Watson Assistant	Active
UpsTrador	Default	Dallas	Language Translator	Active
VozUPS	Default	Dallas	Text to Speech	Active

- Administramos el dialogo de nuestro Chatbot en Watson Assistant.



UpsChatBot	
Intents	
Entities	
Dialog	
Options	
Analytics	
Versions	
Content Catalog	
Intents (6) ↑	Description
#Malla	Malla Curricular
#Matriculas	Informacion Matriculas
#Precios	Infomacion de precios
#Requisitos	Que necesito
#Saludo	Saludo inicial
#Tiempo	Tiempo de estudio



- En Python importamos las siguientes librerías.

```
from fbchat import log, Client, Message
from os.path import join, dirname
from ibm_watson import AssistantV2, LanguageTranslatorV3, TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
```

- Conectamos nuestro servicio de Watson Assistant

```
authenticator = IAMAuthenticator('U4IKxuhQ4XB1rskFYSLlqrB29b_A2fch9uL4gWQUZ-f4')
assistant = AssistantV2(
    version='2018-09-20',
    authenticator=authenticator)
assistant.set_service_url('https://api.us-south.assistant.watson.cloud.ibm.com/instances/20d0e70b-3f11-4c02-9137-205d38b948a9')
assistant.set_disable_ssl_verification(False)
session = assistant.create_session("633359aa-4a7e-4cfa-8ebe-78113a86ad21").get_result()
```

- Conectamos al servicio de Language Translator

```
authenticatorT = IAMAuthenticator('0Iobj63HD2qz6ChKXOCkc1kARMJ8E9-Gkq1045FQIGf7')
language_translator = LanguageTranslatorV3(
    version='2018-05-01',
    authenticator=authenticatorT)
language_translator.set_service_url('https://gateway.watsonplatform.net/language-translator/api')
```

- Conectamos al servicio de Text to Speech

```
authenticatorV = IAMAuthenticator('rQkJz0iTpTyboZqk6SymQ2hh6zfG7sfmxdZBD9V9qQIV')
service = TextToSpeechV1(authenticator=authenticatorV)
service.set_service_url('https://stream.watsonplatform.net/text-to-speech/api')
```

- Definimos un método que envía un mensaje y recibe la respuesta del Chatbot

```
def mensaje(text, session):
    message = assistant.message("633359aa-4a7e-4cfa-8ebe-78113a86ad21",
                                session["session_id"],
                                input={'message_type': 'text', 'text': text}).get_result()
    return (message['output']['generic'][0]['text'])
```

- Definimos un método que realiza la traducción de un texto que recibe como parámetro.

```
def traducir(text, language_translator):
    translation = language_translator.translate(
        text=text, model_id='en-es').get_result()
    return translation['translations'][0]['translation']
```

- Definimos un método que pasa de texto a voz guardando un .mp3 en la capeta loca.

```
def voz(text, service):
    with open(join(dirname(__file__), 'output.mp3'),
        'wb') as audio_file:
        response = service.synthesize(
            text, accept='audio/mp3',
            voice="es-LA_SofiaV3Voice").get_result()
        audio_file.write(response.content)
```

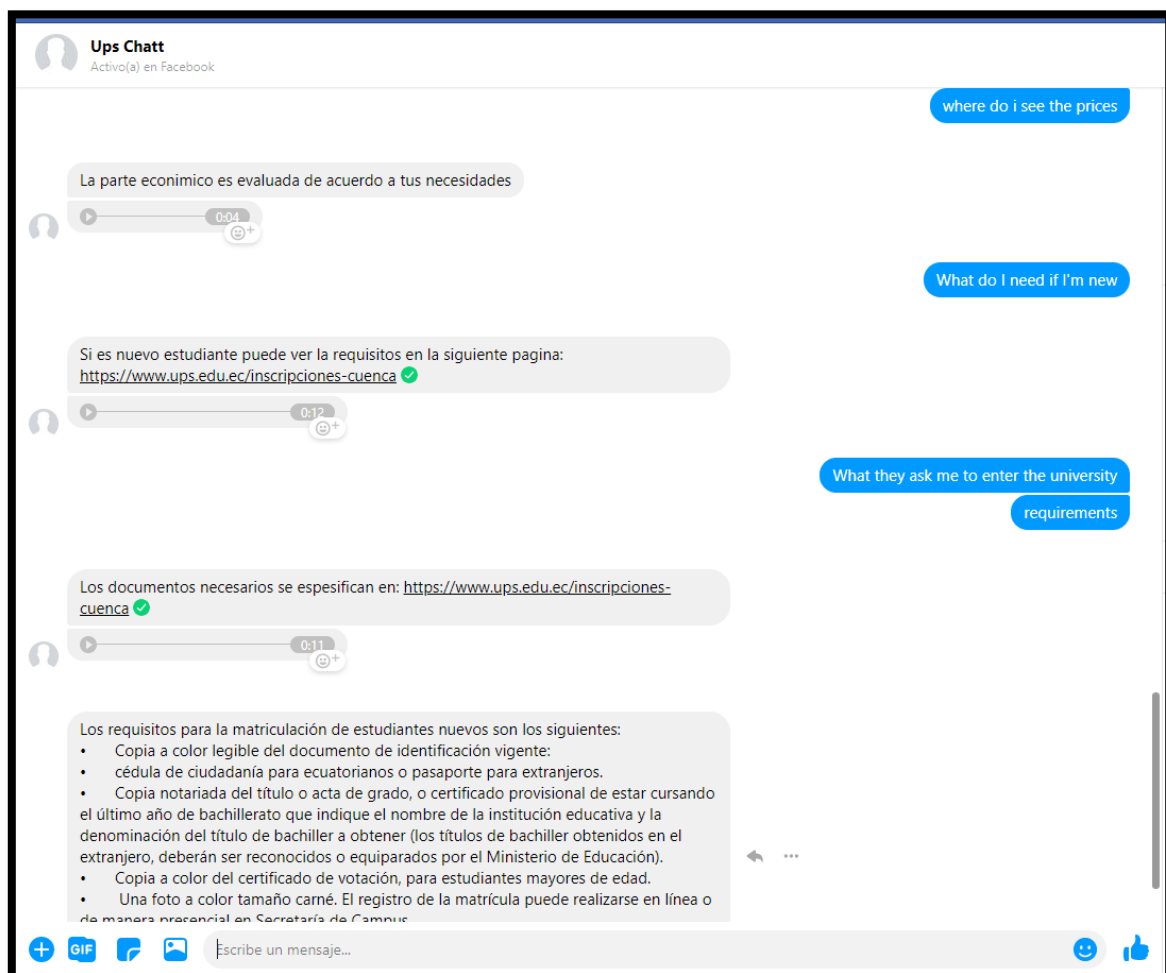
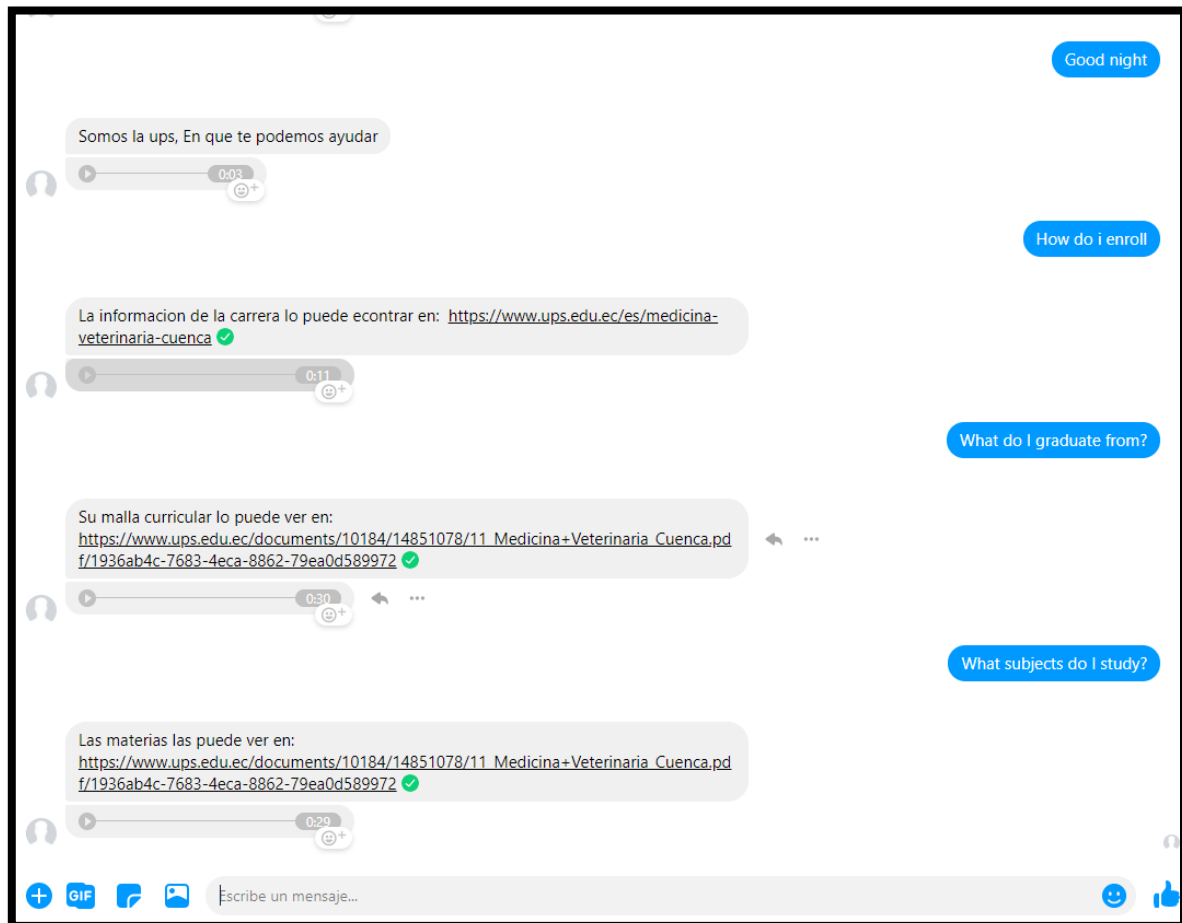
- Por último, definimos el método que estará a la escucha de los mensaje que llegue a la cuenta de Messenger.

```
class EchoBot(Client):
    def onMessage(self, author_id, message_object, thread_id, thread_type, **kwargs):
        self.markAsDelivered(thread_id, message_object.uid)
        self.markAsRead(thread_id)
        if author_id != self.uid:
            messenger = message_object.text
            print(messenger)
            traduccion = traducir(messenger, language_translator)
            print(traduccion)
            respuesta = mensaje(traduccion, session)
            print(respuesta)
            voz(respuesta, service)
            #self.send(Message(text=respuesta), thread_id=thread_id,
thread_type=thread_type)

self.sendLocalVoiceClips('output.mp3', Message(text=respuesta), thread_id=thread_id,
thread_type=thread_type)
```

Este método ejecuta a los demás métodos ya que cuando esté recibe un mensaje inmediatamente manda a llamar al método de **traducir** pasándola el mensaje y obteniendo el mensaje traducir los mensajes son recibidos en inglés y el método o el servicio los traduce al español (Este metodo usa el servicio de **Language Translator**). Una vez con el mensaje traducido llamamos al método mensaje qué es lo que hace es consumir el servicio de **Watson assistant** y me retorna un mensaje de parte del Chatbot. Ya con la respuesta del Chatbot lo que se hace es llamar al método **voz** que lo que hace es recibir la respuesta del Chatbot y la convierte en un archivo punto MP3 el cual se le guarda en la carpeta local. Por último, el método de enviar o **self.sendLocalVoiceClips** lo que hace es recibir un archivo en audio y un mensaje, le pasamos el ID del cliente que queremos que reciba el mensaje y **así emos consumido los tres servicios de IMB, haciendo que un usuario de Messenger pregunte por informacion de la carrera de Medicina veterinaria. Y nuestro Chatbot le responda con audios.**

- Ejecutamos el programa, y presentamos capturas de su funcionamiento.




```
cursor.fetchall()
con.commit()
```

- Creamos un método que nos envíe un mensaje, busqué en la base de datos cual puede ser la respuesta.

```
def buscarRespuesta(con, pregunta):
    cursor = con.cursor(pymysql.cursors.DictCursor)
    cursor.execute("SELECT p.respuesta FROM preguntas p WHERE p.pregunta =
%s", (pregunta))
    res = cursor.fetchall()
    res = res[0]
    con.commit()
    return (res['respuesta'])
```

- Llamamos al método de guardarPregunta cuando se vaya a retornar la pregunta al Messenger.

```
def onMessage(self, author_id, message_object, thread_id, thread_type, **kwargs):
    self.markAsDelivered(thread_id, message_object.uid)
    self.markAsRead(thread_id)
    if author_id != self.uid:
        messenger = message_object.text
        print(messenger)
        traduccion = traducir(messenger, language_translator)
        print(traduccion)
        respuesta = mensaje(traduccion, session)
        print(respuesta)
        voz(respuesta, service)
        guardarPregunta(con, messenger, respuesta, traduccion)
        #self.send(Message(text=respuesta), thread_id=thread_id,
thread_type=thread_type)

self.sendLocalVoiceClips('output.mp3', Message(text=respuesta), thread_id=thread_id,
thread_type=thread_type)
```

- Creamos un template que permita enviar y mostrar los mensajes de nuestra base de datos.

```
<body>

<div class="row">

    <div class="col-sm-6">
        <div class="card">
            <div class="card-header">
                CHATBOOT UPS
            </div>
            <div class="card-body">
                <h1>{{ msg }}</h1>
            </div>
            <div class="card-footer">
                <form action="/msg" method="post">
                    <input name="mensaje" class="form-control type_msg"
placeholder="mensaje..."></input>
                    <input type="submit" value="Enviar mensaje">
                </form>
            </div>
        </div>
    </div>
</div>

</body>
```

- Ya con Flask definimos los métodos de [*índex*] que renderizará el template y el método de [*mensaje*] que responderá a un POST, que recibe un mensaje y lo busca en nuestro sistema.

```
app = Flask(__name__)
@app.route('/')
def hello_world():
    return render_template('index.html')

@app.route('/msg', methods=['POST'])
def mensje():
    if request.method == 'POST':
        mensaje = request.form['mensaje']
        mensaje = buscarRespuesta(con, mensaje)
        return render_template('index.html', msg=mensaje)
```

- Por último creamos un *main*, el cual nos permitirá ejecutar cualquiera de los dos programas, siendo para activar las respuestas de Messenger o ver cuan alimentado esta mi sistema.

```
if __name__ == '__main__':
    #app.run()
    client = EchoBot(correo, contra)
    client.listen()
```

- **Código**

- <https://github.com/RicardoVinicioJara/fbchat>

- **Examen IA**

- [https://github.com/RicardoVinicioJara/FbChat/blob/master/Informe IA.pdf](https://github.com/RicardoVinicioJara/FbChat/blob/master/Informe%20IA.pdf)

- **Examen SE**

- [https://github.com/RicardoVinicioJara/FbChat/blob/master/Informe SE.pdf](https://github.com/RicardoVinicioJara/FbChat/blob/master/Informe%20SE.pdf)