

CARRERA: INGENIERIA DE SISTEMAS

ASIGNATURA: Inteligencia Artificial I & Sistemas Expertos

NRO. PRÁCTICA:

TÍTULO PRÁCTICA: EXAMEN FINAL

OBJETIVO

- Integrar varias tecnologías y los conocimientos adquiridos en el ciclo
- Realizar un chat bot inteligente para asistencia de aspirantes a estudiantes de la carrera de Educación Básica
- Consolidar los conocimientos adquiridos en clase sobre la IA en la nube (chatbot – IBM Watson).

Enunciado:

Aunque existen [muchos tipos de chatbots](#), si va a crear uno por primera vez, te recomendamos una de estas dos opciones:

- Bots informativos

Tal como su nombre sugiere, estos bots proporcionan al usuario un nuevo formato para consumir información. Por ejemplo, los bots de noticias de último momento envían historias actuales a medida que se revela la información.

- Bots de servicio

Estos bots están automatizados para completar tareas y responder preguntas. Dicho de otro modo, resuelven un problema o la inquietud de un usuario por medio de un chat. Tal vez estés pensando en bots de atención al cliente, pero cada vez hay más bots de servicio con fines como reservar citas o comprar en línea.

*En virtud de ello, algunas universidades han empleado el uso de chatbots para interactuar con los estudiantes o posibles candidatos, un ejemplo real es en la **Universidad George Washington**, después de poner a prueba su servicio de chatbot 24/7, MARTHA, el 89 % de los usuarios abogó por que la herramienta se convierta en un servicio permanente, entre otras más.*

En base a ello, se desea generar un chatbot informativo que sirva de soporte en la promoción de las carreras de grado(<https://www.ups.edu.ec/es/web/guest/carreras-grado>) de la Universidad Politécnica Salesiana, este chatbot deberá tener las siguientes características o servicios que brinde:

- Inscripciones / Ficha Socio Economica
- Perfil de egreso
- Malla curricular
- Presentación
- Poder matricularse / inscripciones
- Informacion de los grupos de investigación pertenecientes a la carreras.
- Personal Docente
- Redes sociales
- Vida Estudiantil
- Contactanos
- Varios o información adicional
- Finalmente, utilizar al menos 3 servicios distintos de IBM Watson dentro del proyecto.
- Distribucion de carreras por estudiante:

Carrera: Educación Básica

1.- IBM WATSON

Entiende la intención del usuario, comprende errores y usos del lenguaje. No requiere experiencia técnica y es adaptable a cualquier tecnología. Informes en Tiempo Real. Solución Autoescalable. Seguridad de Datos. Infraestructura Global. Multilanguage.

Servicios en IBM WATSON

Language Translator

BM Watson™ Language Translator le permite traducir mediante programación texto de un idioma a otro.(IBM WATSON)

[Resource list](#) /
Language Translator-4v Active Add tags [🔗](#)

Manage
Getting started
Service credentials
Plan
Connections

Start by viewing the tutorial

[Getting started tutorial](#) [API reference](#)

Credentials

[Download](#) [Show credentials](#)

API key:

.....

URL:

<https://api.us-south.language-translator.watson.cloud.ibm.com/instances/3f020a13-ae1f-4d67-i->

Text to Speech

El servicio de texto a voz IBM Watson™ convierte el texto escrito en voz de sonido natural para proporcionar capacidades de síntesis de voz para aplicaciones. Este tutorial basado en rizados puede ayudarlo a comenzar rápidamente con el servicio.(IBM WATSON)

[Resource list](#) /

Watson Assistant-mk

✓ Active Add tags [🔗](#)

Manage

Service credentials

Plan

Connections

Start by launching the tool

Launch Watson Assistant

Getting started tutorial [🔗](#)

[API reference](#)

Credentials

[Download](#) [↓](#) [Show credentials](#) [👁](#)

API key:

.....

URL:

<https://api.us-south.assistant.watson.cloud.ibm.com/instances/d80a5eb7-c2a1-43d2-84c0-98917>

Watson Assitant

(IBM WATSON)

Start by launching the tool

Launch Watson Assistant

Getting started tutorial [🔗](#)

[API reference](#)

Credentials

[Download](#) [↓](#) [Show credentials](#) [👁](#)

API key:

.....

URL:

<https://api.us-south.assistant.watson.cloud.ibm.com/instances/d80a5eb7-c2a1-43d2-84c0-98917>

El servicio:

Dialog

Carrera EB

Asistente para consulta de la carrera de educación básica

LANGUAGE:

Spanish

TRAINED DATA:

9 Intents | 12 Entities | 25 Dialog nodes

VERSION:

draft

DESCRIPTION:

VERSION CREATED:

Jul 28, 2020 7:24 PM -05

LINKED ASSISTANTS (1): Cosunlta Carrera Educación Básica

Entidades:

Carrera EB

Version: Development

Intents

Entities

My entities

System entities

Dialog

Options

Analytics

Versions

Content Catalog

Entity (12) ↑

Values

Modified ↑↓

@contactos

contactos

8 days ago

@costos

costo

5 days ago

@despedida

adios

5 days ago

@docentes

profesores

6 days ago

@egreso

graduado

8 days ago

@ficha

ficha socio económica

8 days ago

@grupos

investigación

8 days ago

@inscripcion

inscribir

8 days ago

@malla

materias

8 days ago

@matricula

matrícula

8 days ago

@presentación

información

8 days ago

@saludo

saludo

8 days ago

Intents:

Intents (9) ↑

Description

Modified ↑↓

Conflicts ↑↓

Examples ↑↓

#Bienvenida

saludo inicial entre el asistente y el inteeresado

8 days ago

6

#docentes

se dará a conocer a los docentes de la carrera

8 days ago

7

#ficha

facilidades para acceder a la educación

8 days ago

7

#grupos

detalle de todos los grupos de investigación de la carrera

8 days ago

7

#Inscripcion

la persona consultará sobre las inscripciones en la carrera

8 days ago

4

#matricula

la persona interesada podrá acceder a la matrícula en línea

8 days ago

4

#perfilegreso

una descripción del título que se obtendrá al culminar la ...

8 days ago

4

#redessociales

la carrera en las redes sociales

8 days ago

5

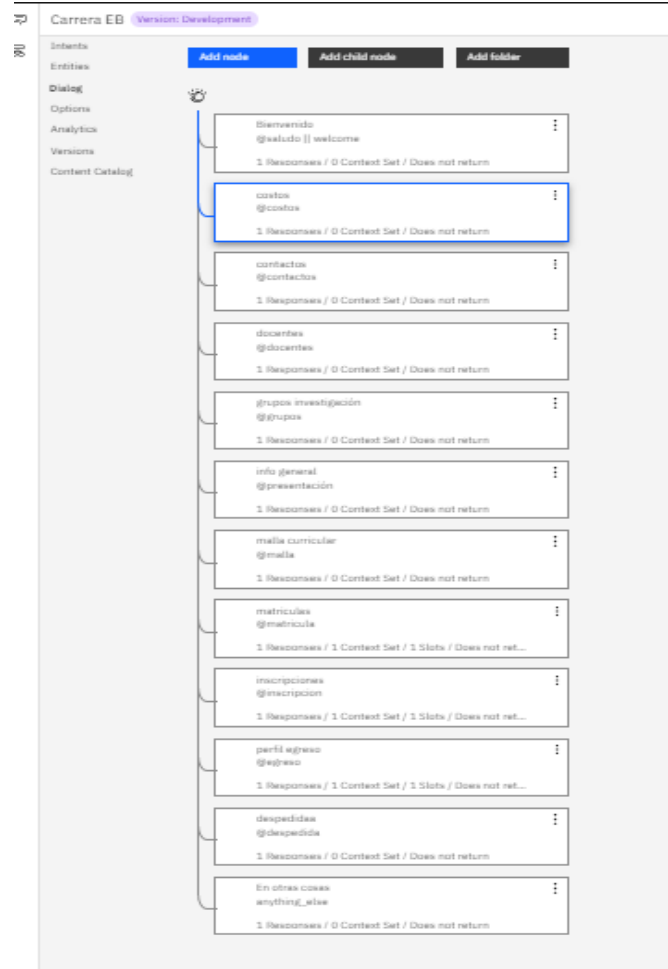
#vidaestudiantil

los estudiantes

8 days ago

0

Dianlogs:



2. Consumiendo servicios Watson desde Python

Tecnologías usadas en Python

```
: import json
from ibm_watson import LanguageTranslatorV3
from ibm_watson import AssistantV2
from ibm_watson import TextToSpeechV1
from ibm_cloud_sdk_core.authenticators import IAMAuthenticator
from ibm_watson.websocket import SynthesizeCallback
from os.path import join, dirname
import pygame
from neo4j import GraphDatabase
from pandas import DataFrame
from clips import Environment
from tkinter import ttk, messagebox
from PIL import Image, ImageTk
from tkinter import *
```

Autenticación en las diferentes tecnologías usadas

```
: #ATENTACIÓN TRANSLATE WATSON
authenticator = IAMAuthenticator('ex7fG5d2ob2SpeqnBatbjrlo0jiD3fruLRe0-zipIRG8')
language_translator = LanguageTranslatorV3(
    version='2018-05-01',
    authenticator=authenticator)
language_translator.set_service_url('https://api.us-south.language-translator.watson.cloud.ibm.com/instances/3f020a

#ATENTACIÓN TEXT TO SPEECH WATSON
authenticator = IAMAuthenticator('y6g3Klma5riCxcg7RWDp2UUst9HCW07iaT2qruSd0uTwg')
text_to_speech = TextToSpeechV1( authenticator=authenticator)
text_to_speech.set_service_url('https://api.us-south.text-to-speech.watson.cloud.ibm.com/instances/6ddfab7b-3699-48

#ATENTACIÓN ASSISTANT WATSON
authenticator = IAMAuthenticator('Vdhmwp71ttlef0bM0JCAVGM1-jqVEt-sQlHqrWFFo_H0')
assistant = AssistantV2(
    version = '2020-04-01',
    authenticator = authenticator
)
assistant.set_service_url('https://api.us-south.assistant.watson.cloud.ibm.com/instances/d80a5eb7-c2a1-43d2-84c0-98
assistant.set_disable_ssl_verification(False)

session = assistant.create_session('5d54ad00-095a-4d6f-b630-5bb1bb72fea1').get_result()

voices = text_to_speech.list_voices().get_result()
#print(json.dumps(voices, indent=2))

#AUTENTICACION NEO4J
uri = "bolt://localhost:7687"
# Connect to the neo4j database server

graphDB_Driver = GraphDatabase.driver(uri)
```

Nota: Detalle con la conexión con Neo4j

INTEGRACIÓN CON NEO4J

INTEGRACION NEO4J

DELETE

```
[ ]: comand_delete = 'MATCH (n) OPTIONAL MATCH (n)-[r]-() DELETE n,r RETURN count(n) as numero'

with graphDB_Driver.session() as graphDB_Session:

    result = graphDB_Session.run(comand_delete)
    print (result)
```

CREATE

```
13]: cqlCreate = """Create (h:Persona {gender:'Hombre',age:25 }),
    (m:Persona {gender:'Mujer', age:22}),
    (q:Motivation {description:'Mejorar la educacion'}),
    (n:Motivation {description:'Vocacion de enseniar'}),
    (h)-[:RELATION {value: 3}]->(q),
    (h)-[:RELATION {value: 1}]->(n),
    (m)-[:RELATION {value: 5}]->(n)
    RETURN COUNT(h) AS personas"""

with graphDB_Driver.session() as graphDB_Session:

    # LISTA LOS NODOS CREADOS
    nodes = graphDB_Session.run(cqlCreate)

    record = nodes.single()
    print(record)
```

UPDATE

```
[4]: pesos=[]

def merge_mujer(desc, value):
    cql_update_query = """MERGE (m:Persona {gender:'Mujer', age:22})
        MERGE (q2:Motivation{description:$desc})
        MERGE (m)-[r:RELATION]->(q2)
        SET r += {value:$value}
        RETURN *"""

    with graphDB_Driver.session() as graphDB_Session:

        # LISTA LOS NODOS CREADOS
        nodes = graphDB_Session.run(cql_update_query, desc=desc, value=value )
        result = nodes.keys()
        print(result)

def merge_hombre(desc, value):
    cql_update_query = """MERGE (h:Persona {gender:'Hombre', age:25})
        MERGE (q3:Motivation{description:$desc})
        MERGE (h)-[r:RELATION]->(q3)
        SET r += {value:$value}
        RETURN *"""

    with graphDB_Driver.session() as graphDB_Session:

        # LISTA LOS NODOS CREADOS
        nodes = graphDB_Session.run(cql_update_query, desc=desc, value=value )
        result = nodes.keys()
        print(result)
```

READ

```
: # CQL to query all the universities present in the graph
values = []
attribute=[]
pesos=[]
def list_conditional(gender):
    cqlEdgeQuery = "match (:Persona {gender: $gender})-->(m:Motivation) return (m.description)"

    with graphDB_Driver.session() as graphDB_Session:

        nodes = graphDB_Session.run(cqlEdgeQuery, gender=gender)
        for node in nodes:
            values.append(node)
        return values
    """if gender == 'Hombre':
        desc = input("Por que escogio esta carrera? ")
        value = input("ingrese valor: ")
        merge_hombre(desc, value)"""

def values(desc, gender):
    cqlEdgeQuery = "MATCH (m)-[r:RELATION]->(q2:Motivation) where q2.description = $desc RETURN (r.value)"
    with graphDB_Driver.session() as graphDB_Session:

        pesos = graphDB_Session.run(cqlEdgeQuery, desc=desc)
        for peso in pesos:
            prueba = str(peso)
            #prueba.split()
            primero=prueba.split("=", 2)
            segundo = str(primero[1])
            tercero = segundo.split(">",1)
            cuarto =str(tercero[0])
            print(cuarto[0])
            valor= int(cuarto[0])
            value = valor+1
```


INTEGRACION CLIPS

```

]: env = Environment()
def expert_system(motivacion):
    env.assert_string("(motivacion "+ motivacion +)")
    #env.assert_string("(motivacion  vocacion de ayudar)")

    rule = """
    (defrule my-rule
    (motivacion ayudar)
    =>
    (printout t "deberias ingresar a un grupo" crlf))
    """

    for fact in env.facts():
        aux = str(fact)
        lower=aux.lower()
        #print(lower)
        ccc=lower.find('ayudar')
        if ccc != -1:
            saludo = env.build(rule)
            env.run()
            return('Deberias ingresar a algun grupo')
        ccc=lower.find('vocacion')
        if ccc !=-1:
            return "Tendrias que dar tutorias a tus companieros"

        ccc=lower.find('gusta')
        if ccc !=-1:
            return "Te aseguramos que seras un gran docente"

        ccc=lower.find('educacion')
        if ccc !=-1:
            return "Sabemos que la educacion debee mejorar cada dia"

```

Consumo De servicios de IBM WATSON

```
# CONVERSACIÓN CON ASSISTANT WATSON
def assistant_bot(pregunta_persona):

    message = assistant.message(
        '5d54ad00-095a-4d6f-b630-5bb1bb72feaf',
        session['session_id'],
        input={
            'message_type':'text',
            'text':pregunta_persona
        }
    ).get_result()
    #print(json.dumps(message, indent=2))
    # RECORRO EL JSON RESULT Y OBTENGO EL CAMPO GENERIC QUE ES LA RESPUESTA
    for key,value in message.items():
        for salida in value["entities"]:
            salida

            for entityProperty in value["generic"]:
                str(entityProperty)

    condicion = str(salida)
    res_salida = condicion.split(':',2)
    respuesta = res_salida[1]
    condicion = str(respuesta)
    res_final = condicion.split(',',1)
    condicion =str(res_final[0])
    #print(condicion)

    #TEXTO
    response=str(entityProperty)
    res=response.split(':',2)
    paso_valor=str(res[2])
    return paso_valor
```

Conclusiones

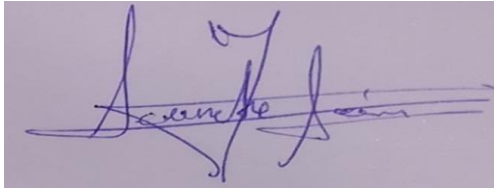
- Se tiene un mayor entendimiento de las tecnologías usadas
- La conexión a internet es fundamental para el funcionamiento de IBM WATSON
- La conexión entre Python y Neo4j es de manera sencilla si se sigue los pasos correctos
-

Recomendaciones

- Seguir usando e implementando lo aprendido en las materias de IA I y Sistemas Expertos.
- Investigar mas usos de estas tecnologías e integración entre las mismas y otras plataformas de desarrollo.
-

Estudiante: Leandro León.

Firma:

A handwritten signature in blue ink, appearing to read 'Leandro León', with a stylized flourish extending to the right.