Proyecto - NPL

(MÓDULO-10)

Squad ETL:

Hernani, Marco

Padilla Nieto, Alberto

Tenreiro, Ruben

López Casas, Jessica

Marco General del Proyecto

La empresa Sky2travel requiere de nuestro servicios, y no contrata para que realicemos un bot transaccional enfocado a facilitar la búsqueda de vuelos y viaje de los clientes que lo soliciten para poder integrarlo en dispositivos móviles y diferentes apps.

El objetivo, es que los clientes con un mensaje de texto,realicen una solicitud y el bot sea capaz de devolver la información necesaria.

Además, las solicitudes de reserva deben guardarse en formato Json.

Markov General del Proyecto

Idea de crear un ChatGPT_Assembler

1ºer Intento: tag y prueba. **FAIL**

2ºer Intento +tag y +prueba **FAIL**

3ºer Intento: ++tag y ++pruebas **+FAIL**

Assembler Institute of Technology

Índice

1. Librerías
2. Funciones Auxiliares.
3. Modelo Markov + Train.
4. Reglas.
5. Creación JSON.
6. Chat-Bot.
7. Conclusiones

1- Librerías

- nltk, speech_recognition, pandas, pyttsx_3

2- Funciones Auxiliares

```python
def talk_to_user(s, engine):
    engine.say(s)
    engine.runAndWait()
```

```python
def get_voice():
    # Crear instancia del objeto reconocedor
    r = sr.Recognizer()
    # Utilizar el micrófono del sistema como fuente de audio
    with sr.Microphone() as source:
        # Ajustar nivel de ruido ambiental
        r.adjust_for_ambient_noise(source)
        print("Di algo...")
        # Escuchar audio del usuario
        audio = r.listen(source)

        # Convertir audio a texto utilizando el reconocimiento de
        frase_user = r.recognize_google(audio, language='es-ES')
        frase_user = unidecode.unidecode(frase_user)

        # Imprimir la frase reconocida
        #print(f"Has dicho: {frase_user}")

        return frase_user
```

2- Funciones auxiliares

```python
def get_json(_sentence):
    tree_sentence_tag = parser.parse(hmm.tag(tokenizar(_sentence)))
    semi_query = []
    for x in tree_sentence_tag[:]:
        if (type(x) == Tree):
            semi_query.append((x.label(),x[:]))

    json_query = {'numeroBilletes': '', 'Origen': '', 'Destino': '',

    for x in semi_query:
        if(x[0] == 'fecha'):
            fecha_vuelo = list(map(lambda t: t[0], x[1]))
            json_query['fecha'] = " ".join(fecha_vuelo)
        else:
            json_query[x[0]] = x[1][0][0]

    return(json_query)
```

```python
ciudades = pd.read_csv("cities.csv", names = ["ciudad"])
ciudades = list(ciudades['ciudad'].str.lower())
ciudades
```

```
['new york',
 'los angeles',
 'chicago',
 'dallas',
 'philadelphia',
 'houston',
 'toronto',
 'washington',
 'miami',
 'atlanta',
 'boston',
 'san francisco',
 'detroit',
 'riverside',
 'phoenix',
```

2- Funciones auxiliares

```python
def query_correction(_query, engine):

    while(_query["aerolinea"] not in list(map(lambda s: s.lower(),
        talk_to_user('Por favor, introduce el nombre de la aerolin
        _query["aerolinea"] = get_voice().lower()

    while(_query["Destino"] not in list(map(lambda s: s.lower(), c
        talk_to_user('Por favor, introduce la ciudad de destino: '
        _query["Destino"] = get_voice().lower()

    while(_query["Origen"] not in list(map(lambda s: s.lower(), ci
        talk_to_user('Por favor, introduce la ciudad de origen: ',
        _query["Origen"] = get_voice().lower()

    while(len(_query["fecha"]) == 0):
        talk_to_user('Por favor, introduce la fecha: ', engine)
        _query["fecha"] = get_voice().lower()

    else:
        print('Gracias, procesando tu peticion...')
```

3- Modelo Markov + Train.

```
hmm = HiddenMarkovModelTagger.train(train_corpus)
```

```
[('quiero', 'vmip3p0'),
  ('reservar', 'vmic000'),
  ('un', 'sps00'),
  ('vuelo', 'ncms000'),
  ('desde', 'sps00'),
  ('madrid', 'np00000'),
  ('a', 'sps00'),
  ('new york', 'np00000'),
  ('york', 'np00000'),
  ('para', 'sps00'),
  ("el", "da0ms0"),
  ("8", "Z"),
  ("de", "sps00"),
  ("diciembre", "mes"),
  ('con', 'sps00'),
  ('air europa', 'np00000')],
```

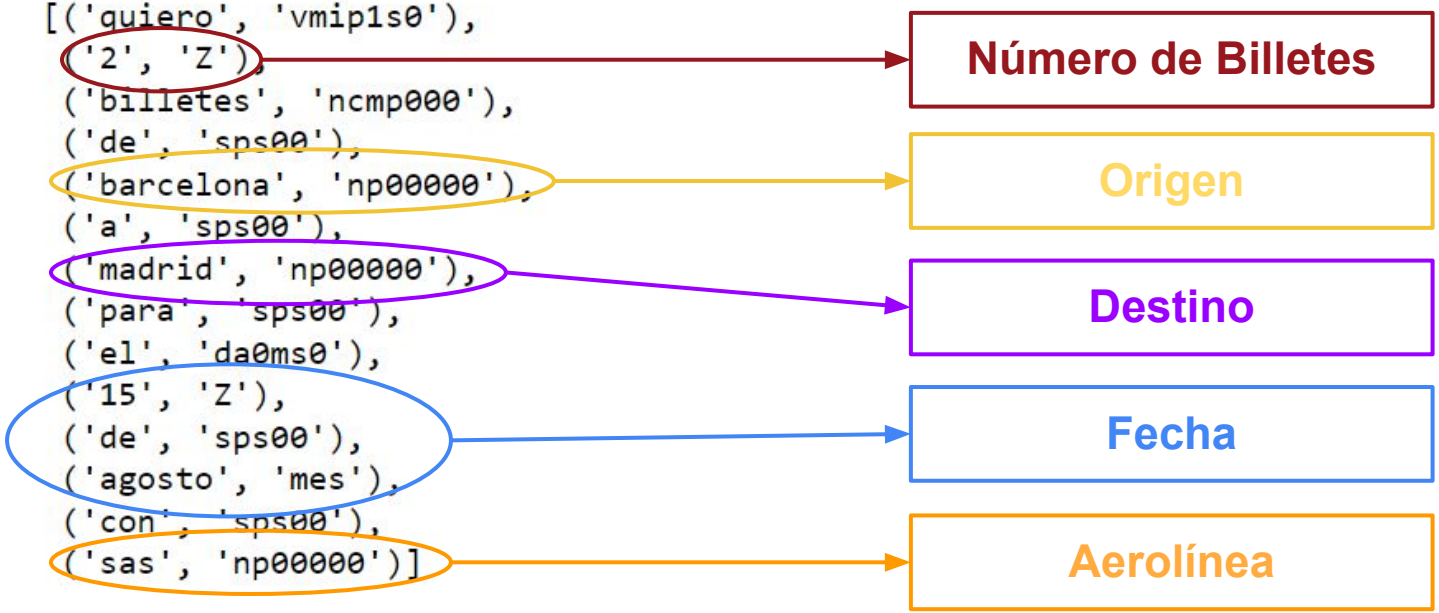
x 20

Assembler
Institute of Technology

3- Probando HiddenMarkov

```
hmm.tag(get_tokens("Quiero 2 billetes de Barcelona a Madrid para el 15 de agosto con SAS"))
```

```
[('quiero', 'vmip1s0'),
 ('2', 'Z'),
 ('billetes', 'ncmp000'),
 ('de', 'sps00'),
 ('barcelona', 'np00000'),
 ('a', 'sps00'),
 ('madrid', 'np00000'),
 ('para', 'sps00'),
 ('el', 'da0ms0'),
 ('15', 'Z'),
 ('de', 'sps00'),
 ('agosto', 'mes'),
 ('con', 'sps00'),
 ('sas', 'np00000')]
```



Assembler
Institute of Technology

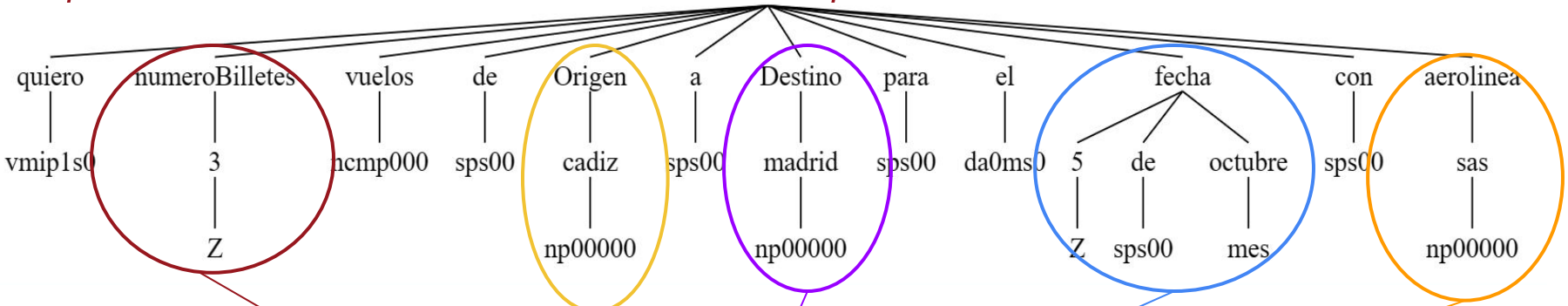
4- Reglas

```
reglas = r"""

Origen: <sp.*> {<np.*> <np.*> <np.*> | <.*> <np.*> | <np.*>} <sp.*>

Destino: <Origen> <sp.*> {<np.*> <np.*> <np.*> | <.*> <np.*> | <np.*>}

numeroBilletes: {<Z>} <nc.*>

fecha: {<Z> <sp.*><mes>}

aerolinea: <fecha> <sp.*> {<np.*>}

"""

parser = nltk.RegexpParser(reglas)
```

Assembler
Institute of Technology

5- Creación Json.

"quiero 3 vuelos de Cadiz a Madrid para el 5 de Octubre con SAS"




```
10      for x in semi_query:
11          if(x[0] == 'fecha'):
12              fecha_vuelo = list(map(lambda t: t[0], x[1]))
13              json_query['fecha'] = " ".join(fecha_vuelo)
14          else:
15              json_query[x[0]] = x[1][0][0]
```

x=Tree('fecha', [('5', 'Z'), ('de', 'sps00'), ('octubre', 'mes')]),

x=Tree('Origen', [('cadiz', 'np00000')])

6- ChatBot

```python
def ultron_chat(engine):

    quit_words = ['n', 'no']
    q = ''

    print("Hola, bienvenido a sky2travel, como te puedo ayudar?")

    while(q not in quit_words):
        user_sentence = get_voice()
        json_sentence = get_json(user_sentence)

        query_correction(json_sentence, engine)

        print("Tu petición es:")
        print(json_sentence)
        print(f"""Perfecto. Voy a realizar la búsqueda de
        {json_sentence["numeroBilletes"]} billetes para tu viaje desde
        {json_sentence["Origen"]} a {json_sentence["Destino"]} para el
        {json_sentence["fecha"]} con {json_sentence["aerolinea"]}.""")

        q = input("Quieres continuar s/n : \n")
```

6- Prueba Chat

```
In [20]:  1 ultron_chat(engine)
                                                  'Phoenix para el 3 de noviembre con '
                                                  'American Airlines'},
                                 {   'transcript': 'quiero dos billetes de de Detroit a
'
                                                  'Phoenix para el 3 de noviembre con '
                                                  'American Airlines'},
                                 {   'transcript': 'quiero dos billetes de Detroit a '
                                                  'Phoenix para el 3 de noviembre con '
                                                  'American Air Lines'}],
     'final': True}
Gracias, procesando tu peticion...
Tu petición es:
{'numeroBilletes': 'dos', 'Origen': 'detroit', 'Destino': 'phoenix', 'fecha':
'3 de noviembre', 'aerolinea': 'american airlines'}
Perfecto. Voy a realizar la búsqueda de dos billetes para tu viaje desde
        detroit a phoenix para el 3 de noviembre
        con american airlines.
Quieres continuar s/n :
n
```

7- Conclusiones

1. No es práctico crear un chatbot de cero. Es mejor ayudarnos de modelos ya creados de NLP y adaptarlos.

2. Se necesitan gran cantidad de datos, ya que la **estructura** de las frases puede ser muy diversa. Hemos comprendido que 20 frases no se adaptan a la realidad,y hemos tenido que adecuar las frases en un contexto bastante definido y preestablecido.

3. Necesidad de conocer el idioma con el que se va a tratar es fundamental, y ha entendido la dificultad de tratar expresiones coloquiales..

Assembler
Institute of Technology

Muchas Gracias por su Atención

Assembler
Institute of Technology