# Semillero de investigación en robótica ASIMOV

**Equipo de inteligencia artificial**
**reconocimiento de voz**
**Mayo, 2021**

# 1. Objetivo

**1.1 Objetivo General:** Desarrollar un programa basado en reconocimiento de voz y procesamiento del lenguaje natural, que permita a Pascal responder a comandos de voz y ejecutar tareas.

## 1.2 Objetivos Específicos

**1.** Utilizar la librería "SpeechRecognition" para desarrollar un programa capaz de entender comandos de voz.

**2.** Desarrollo del Chatbot capaz de responder de forma coherente al usuario.

- **2.1** Crear el Dataset con las posibles preguntas que podrá recibir Pascal.
- **2.2** Técnicas de procesamiento del lenguaje natural que se implementaron.
- **2.3** Construir la red neuronal y definir los parámetros adecuados para entrenar el modelo.

**3.** Dar voz a Pascal.

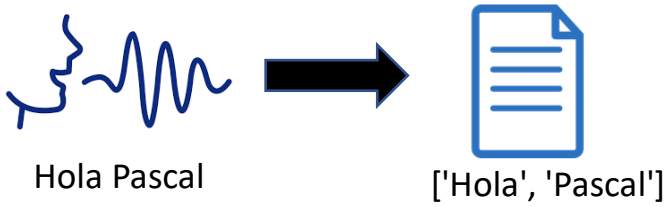
- **3.1** Crear las pistas de audio de las palabras que se le enseñaron al Chatbot.
- **3.2** Crear una función para expresar Strings en arrays por medio de audios pregrabados.

**4.** Unir el motor de reconocimiento de voz con el ChatBot, expresar la respuesta usando los audios pregrabados y ajustar los parámetros para garantizar un correcto procesamiento de las respuestas de Pascal.

## 2. Diseño metodológico

### Momento 1: Reconocimiento de voz

Es necesario utilizar un motor de reconocimiento de voz para poder identificar las frases que dice el usuario y entregar un formato de texto legible para el programa.




Librerías utilizadas:

-SpeechRecognition: Contiene varias API, se utiliza la API Google Web Speech

-Unidecode: Se uso para codificar el Output del motor de reconocimiento.

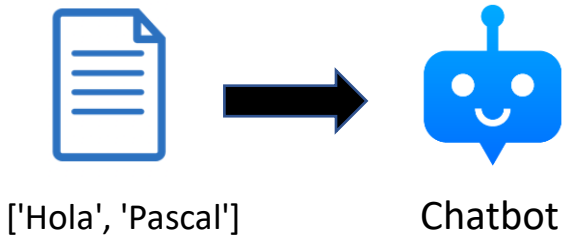
```
def recognize(t):
    with SRG.Microphone() as s:
        r.adjust_for_ambient_noise(s)
        audio_input = r.record(s,  duration=t)
        try:
            text_output = r.recognize_google(audio_input, language="es-ES").lower()
            text_output = unidecode.unidecode(text_output)
            print("tu:"+ text_output)
            return text_output
        except:
            text_output = ("")
            return text_output
```

Se definió como “recognize” a la función que activa el reconocimiento de voz con “ t ”, como parámetro de entrada que define el tiempo que estará activa la función; dado que el ruido exterior puede interferir en la grabación se uso “r.adjust_for_ambient_noise(s)” para ajustar el micrófono al nivel del ruido.

El texto extraído de la grabación se guarda en la variable “text_output”, se aplica “.lower()” junto a “unidecode.unidecode()” para eliminar las mayúsculas y los caracteres especiales; esto es necesario para que el texto sea compatible con el dataset, el cual se explicara más adelante.

## Momento 2: Desarrollo del Chatbot capaz de responder de forma coherente al usuario.

Para poder procesar el texto extraído de la grabación de audio se hace necesario el uso de un Chatbot.




### 2.1 Crear el Dataset con las posibles preguntas que podrá recibir Pascal.

Para el desarrollo del Chatbot es necesario enseñarle las posibles preguntas que recibirá y como debe responder a estas, por tanto, se debe crear un dataset con dicha información.

El dataset se creo en un archivo .json y se acordó que no se usaran letras mayúsculas, tildes o caracteres especiales; esto para evitar inconvenientes en el reconocimiento de Strings.

Es necesario clasificar los tipos de preguntas que recibe Pascal por categorías, estas se definen como “Tags” y cada uno contiene una lista de “Patterns” y “Responses”.

| | | |
|---|---|---|
| Tag: Estado | **Patterns:**<br>como estas<br>como va todo<br>como has estado<br>como te ha ido<br>que tal tu dia | **Responses:**<br>muy bien gracias<br>super bien<br>bien gracias<br>estoy bien gracias |

- Patterns: contiene diferentes formas de hacer la pregunta, pero en general todas llevan a la misma respuesta.
- Responses: contiene las formas adecuadas en que se puede responder a la pregunta.

Es importante clasificar las preguntas y entender cuál es el objetivo realmente, el usuario podría decir:
- Donde queda la papelería?
- Donde puedo imprimir un documento?
- Por favor, llévame a la papelería.

Desde luego, que las frases anteriores están relacionadas con la clase "papelería", sin embargo, cada una tiene una intención diferente, por esta razón se debe ampliar la cantidad de "tags" y de esta manera hacer más especificas las preguntas del usuario.

## Tags de ubicación:

| | | |
|---|---|---|
| Tag: Ubicacion_Papeleria | **Patterns:**<br>-donde encuentro la papeleria<br>-donde hay una papeleria<br>-En que piso queda la papeleria<br>….. | **Responses:**<br>-la papeleria queda en…<br>-puedes encontrar la papeleria en…<br>….<br>….. |

Se usan para clasificar las preguntas relacionadas con la ubicación de algún lugar.

## Tags de información:

| | | |
|---|---|---|
| Tag: Informacion_Papeleria | **Patterns:**<br>-donde puedo imprimir un documento<br>-donde puedo comprar un cuaderno<br>…. | **Responses:**<br>-en la papeleria te pueden ayudar<br>-puedes preguntar en la papeleria<br>….. |

Se usan para clasificar las preguntas de acuerdo a la necesidad del usuario.

## Tags de navegación:

| | | |
|---|---|---|
| Tag: Navegacion_Papeleria | **Patterns:**<br>-pascal vamos a la papeleria<br>-llevame a la papeleria<br>-acompañame a la papeleria<br>-ayudame a llegar a la papeleria | **Responses:**<br>-vamos<br>-de acuerdo<br>….<br>…. |

Se usan para clasificar las ordenes de navegar a un lugar específico, cuando se utiliza un tag de navegación se debe activar la función establecida para navegar al lugar.

## 2.2 Técnicas de procesamiento del lenguaje natural que se implementaron.

Para el procesamiento del lenguaje se utilizaron dos técnicas; “Tokenize” y “Bag of Words”

- **Tokenize:**

```
def tokenize(sentence):
  return nltk.word_tokenize(sentence)
```

Se definió la función “Tokenize” con parámetro “sentence”, su función es separar el texto entregado por el motor de reconocimiento de voz por palabras y eliminar caracteres especiales de ser necesario.

- **Bag of words:**

```
def bag_of_words(tokenized_sentence, words):
  sentence_words = [stem(word) for word in
tokenized_sentence]
  bag = np.zeros(len(words), dtype=np.float32)
  for idx, w in enumerate(words):
    if w in sentence_words:
      bag[idx] = 1
  return bag
```

Se utilizo esta técnica, ya que es necesario el uso de una red neuronal para entrenar el modelo, por esa razón se debe hacer una “traducción” del texto a un formato que la red neuronal pueda entender. Se define como “All_words” a un array que almacene todas las palabras que conoce Pascal.

```
All_words =["Hola", “Pascal", "Robot", "Pascal", "Buen”, "dia“]
```

Se procede a “traducir” las preguntas y respuestas que se almacenaron en el dataset de la siguiente manera:

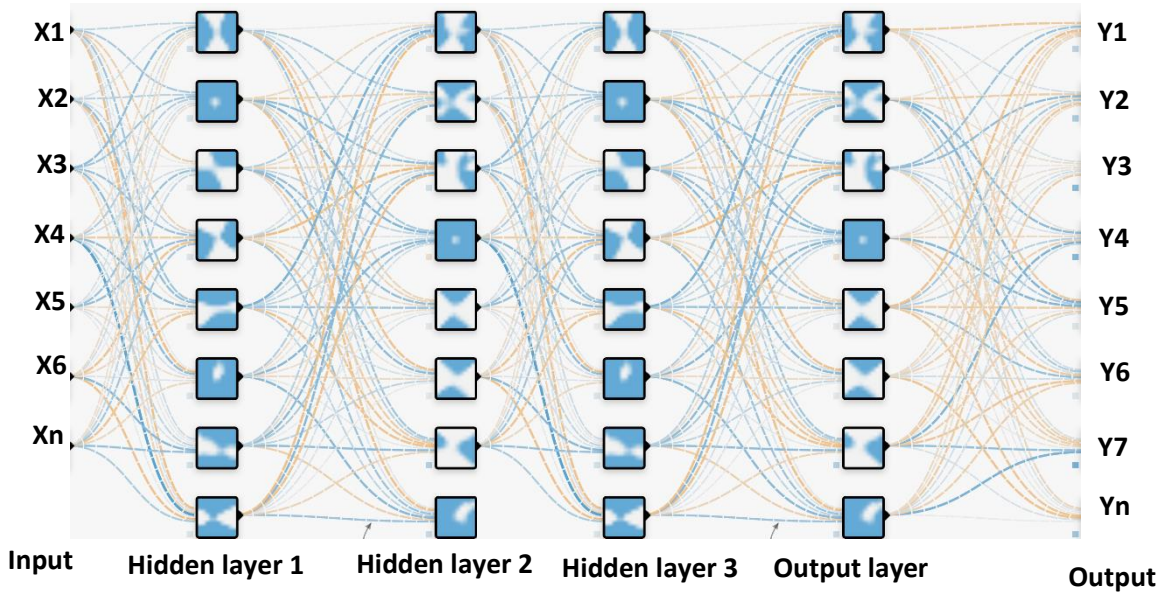
Comparamos el array “All_words” con el array “Pregunta” y únicamente activaremos las posiciones donde se encuentren palabras repetidas.

```
                    All_words =["Hola", “Pascal", "Robot", "Pascal", "Buen”, "dia“]
Pregunta =["Hola", “Pascal”] → X1 =[ 1 , 1 , 0 , 1 , 0 , 0 ]
Pregunta =[“Buen", "dia“]    → Y1 =[ 0 , 0 , 0 , 0 , 1 , 1 ]
```

Nota: Se define como preguntas al Texto entregado por el usuario y Respuestas al texto entregado por Pascal

**2.3 Construir la red neuronal y definir los parámetros adecuados para entrenar el modelo.**

Para entrenar el modelo se utilizó una red neuronal profunda (DNN) con 3 capas ocultas, cada una con 8 neuronas, y se definió “Relu” como la función de activación en la capa de salida.




Como parámetros de entrenamiento:

- Iteraciones: 1000
- Tamaño de lotes: 1 (por la cantidad de datos no fue necesario agrupar por lotes)
- Coeficiente de aprendizaje: 0.01
- Neuronas por capa oculta: 8

## Momento 3: Dar voz a Pascal.

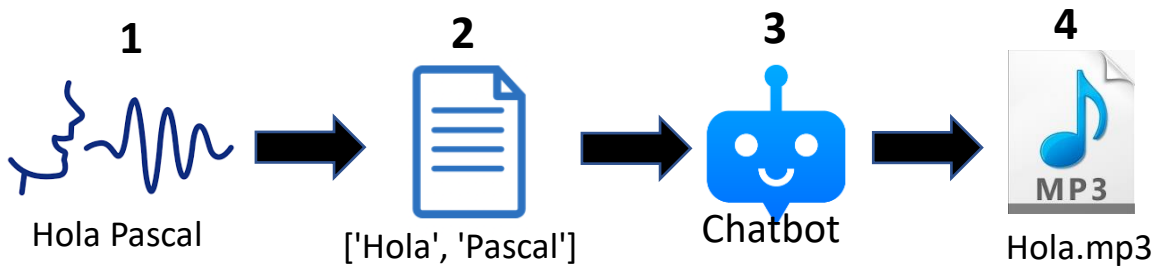
### 3.1 Crear las pistas de audio de las palabras que se le enseñaron al Chatbot.

No fue posible empezar con las grabaciones para la voz de Pascal, sin embargo, con ayuda del equipo de mercadeo se trabajo en definir la personalidad y el tono de voz adecuado para Pascal; queda como objetivo trabajar en este punto.

### 3.2 Crear una función para expresar Strings en arrays por medio de audios pregrabados.

A pesar de que no se construyó esta función, se pudo concluir que se utilizara la técnica “Tokenize” para separar las palabras y posteriormente reproducir los audios de acuerdo a la respuesta de Pascal; queda como objetivo trabajar en este punto.

**Momento 4. Unir el motor de reconocimiento de voz con el ChatBot, expresar la respuesta usando los audios pregrabados y ajustar los parámetros para garantizar un correcto procesamiento de las respuestas de Pascal.**




1. Reconocimiento de voz.
2. Procesamiento del texto.
3. Procesar respuesta.
4. Reproducir en audio la respuesta.

Se pudo cumplir con las 3 primeras funciones; Pascal es capaz de entender la orden del usuario, procesar el texto y responder por medio de Strings en consola.

**Parámetros:**

- Tiempo que estará activo el micrófono esperando el comando de activación: 3 segundos
- Tiempo de espera entre preguntas: 5 segundos.
- Tiempo de espera para desactivar a Pascal por inactividad: 50 segundos.

## 3. Objetivos por cumplir

- Dar voz a Pascal.

- Crear una base de datos para guardar las preguntas y respuestas que procesa Pascal.

- Ampliar el Dataset.

- Probar a Pascal en condiciones de ruido.