Proyecto Final Inteligencia Artificial

Reconocedor de Voz a Texto y Viceversa

Juan Diego Benítez Arias

Jesús Andrés Acendra Martínez

Inteligencia Artificial

Universidad De Cartagena

2017

Reconocedor de Voz a Texto y Viceversa

Nuestro proyecto se basa en la creación de una aplicación la cual permita reconocer un archivo de audio y transcribir en texto lo que se dice en este y a su vez poder reproducir con voz un texto que se haya escrito, esta aplicación fue hecha en el lenguaje Python y las herramientas que se utilizaron se detallan posteriormente.

¿Qué es?

El reconocimiento automático del habla (RAH) o reconocimiento automático de voz es una disciplina de la inteligencia artificial que tiene como objetivo permitir la comunicación hablada entre seres humanos y computadoras. El problema que se plantea en un sistema de este tipo es el de hacer cooperar un conjunto de informaciones que provienen de diversas fuentes de conocimiento (acústica, fonética, fonológica, léxica, sintáctica, semántica y pragmática), en presencia de ambigüedades, incertidumbres y errores inevitables para llegar a obtener una interpretación aceptable del mensaje acústico recibido.

Un sistema de reconocimiento de voz es una herramienta computacional capaz de procesar la señal de voz emitida por el ser humano y reconocer la información contenida en ésta, convirtiéndola en texto o emitiendo órdenes que actúan sobre un proceso. En su desarrollo intervienen diversas disciplinas, tales como: la fisiología, la acústica, la lingüística, el procesamiento de señales, la inteligencia artificial y la ciencia de la computación.

Un aspecto crucial en el diseño de un sistema de RAH es la elección del tipo de aprendizaje que se utilice para construir las diversas fuentes de conocimiento. Básicamente, existen dos tipos:

Aprendizaje deductivo: Las técnicas de Aprendizaje Deductivo se basan en la transferencia de los conocimientos que un experto humano posee a un sistema informático. Un ejemplo paradigmático de las metodologías que utilizan tales técnicas lo constituyen los Sistemas Basados en el Conocimiento y, en particular, los Sistemas Expertos.

Aprendizaje inductivo: Las técnicas de Aprendizaje Inductivo se basan en que el sistema pueda, automáticamente, conseguir los conocimientos necesarios a partir de ejemplos reales sobre la tarea que se desea modelizar. En este segundo tipo, los ejemplos los constituyen aquellas partes de los sistemas basados en los modelos ocultos de Márkov o en las redes neuronales artificiales que son configuradas automáticamente a partir de muestras de aprendizaje.

En la práctica, no existen metodologías que estén basadas únicamente en el Aprendizaje Inductivo, de hecho, se asume un compromiso deductivo-inductivo en

el que los aspectos generales se suministran deductivamente y la caracterización de la variabilidad inductivamente.

Implementación

Librerías

Para la elaboración de esta aplicación se hizo uso del lenguaje interpretado Python en su versión 2.7.13, además de esto se hicieron uso de las siguientes librerías o módulos como son llamados en Python, estos son:

- Speech_recognition: modulo con diferentes clases y métodos encargados del reconocimiento de voz
- gTTS: (Google Text to Speech) modulo encargado de convertir el texto en audio y a su vez guardarlo en un archive .mp3
- Pocketsphinx: es una librería que viene con Speech_recognition, contiene un completo diccionario de palabras y su pronunciación, pero está en ingles
- Pydub: modulo que nos ayudara a la conversión de un archivo de audio .mp3 a uno .wav
- PyQt4: una completa librería para crear y trabajar con interfaces graficas en Python

Es necesario descargar e instalar cada uno de estos módulos para el correcto funcionamiento de la aplicación.

Código

Como ya hemos venido diciendo esta aplicación fue escrita en Python por ende su implementación fue menos complicada, para la parte de reconocimiento de voz utilizamos las APIs que nos ofrecen Google y Wit.ai, esto como no trata de ser un tutorial de como programar en Python solo explicare los puntos más importantes y como fueron usadas esta APIs para la implementación de la aplicación.

```
from PyQt4.QtCore import * #Importamos modulos necesarios para la interfaz grafica
from PyQt4.QtGui import *



class Widget(QTabWidget):

    fichero_actual = ""
    def setupUi(self, TabWidget):

        TabWidget.resize(430, 340)
        TabWidget.setMaximumSize(430, 340)
        TabWidget.setMinimumSize(430, 340)
        #Pestaña 1
        tab = QWidget()
        Aceptar1 = QPushButton(tab)
```

Importamos las librerías necesarias para la creación de la interfaz gráfica de nuestra aplicación.

```
def text_to_speech(self): # Funcion para pasar de texto a voz
    from gtts import gTTS # Importamos los modulos necesarios
    shost = self.textEdit.toPlainText()  # Obtenemos el texto que pasaremos a voz
    print shost
    tts = gTTS(text = str(shost), lang='es') # Llamamos al metodo el cual convierto el texto a voz
    tts.save("audio.mp3") # Guardamos el audio en un archivo mp3

    self.convertidor() # Llamamos a la funcion convertir para convertir el formato del anterior audio a .wav
```

Estas es la función que se llama para convertir texto a voz, como vemos obtenemos el texto que el usuario escribió y este se lo pasamos como parámetro a la función gTTS, vemos que esta recibe otro parámetro y este es el lenguaje en que se producirá la voz, por ultimo guardamos el audio en un archivo llamado “audio.mp3”

```
    def __init__(self, text, lang = 'en', slow = False, debug = False):
        self.debug = debug
        if lang.lower() not in self.LANGUAGES:
            raise Exception('Language not supported: %s' % lang)
        else:
            self.lang = lang.lower()

        if not text:
            raise Exception('No text to speak')
        else:
            self.text = text

        # Read speed
        if slow:
            self.speed = self.Speed().SLOW
        else:
            self.speed = self.Speed().NORMAL


        # Split text in parts
        if self._len(text) <= self.MAX_CHARS:
            text_parts = [text]
        else:
            text_parts = self._tokenize(text, self.MAX_CHARS)

        # Clean
        def strip(x): return x.replace('\n', '').strip()
        text_parts = [strip(x) for x in text_parts]
        text_parts = [x for x in text_parts if len(x) > 0]
        self.text_parts = text_parts

        # Google Translate token
        self.token = Token()
```

Este es el verdadero método, como vemos puede recibir otros parámetros, pero usamos los estándares, al final de todo se crea un token y este es el que se envía a los servidores del traductor de google para que nos envié la pronunciación de lo que enviamos.

```
def convertidor(self): # Funcion para convertir un archivo de audio .mp3 a .wav
    from pydub import AudioSegment # Importamos los modulos necesarios
    #AudioSegment.converter = "C:/ffmpeg/bin/ffmpeg.exe"
    sound = AudioSegment.from_mp3("audio.mp3")  # Tomamos los datos de un archivo mp3 y guardamos
    sound.export("aud.wav", format = "wav") # creamos el nuevo archivo de audio .wav
    print "Done"
```

Esta función como su nombre lo dice convierte un archivo mp3 a un archivo de audio con extensión wav, lo utilizamos para convertir el audio generado anteriormente con la función de pasar texto a voz y hacer pruebas con la función de pasar voz a texto.

```
def speech_to_text(self): # Funcion para pasar de voz a texto
    import speech_recognition as sr # Importamos los modulos necesarios
    from os import path
    AUDIO_FILE = path.join(path.dirname(path.realpath(__file__)), str(self.Ruta.text())) # Obtenemos el audio a convertir debe ser .wav

    r = sr.Recognizer() # Creamos una instacia de la clase Recognizer
    with sr.AudioFile(AUDIO_FILE) as source:
        audio = r.record(source)  # Leemos completamente el archivo de audio
```

Esta función es la que se utiliza para pasar de voz a texto, importamos las librerías necesarias, y creamos una variable llamada AUDIO_FILE que se encargara de obtener la ruta del archivo de audio wav. ¿Porque tienen que ser wav los audios? Porque wav es un formato de audio digital normalmente sin compresión de datos desarrollado y propiedad de Microsoft y de IBM que se utiliza para almacenar sonidos en el PC, admite archivos mono y estéreo a diversas resoluciones y velocidades de muestreo, su extensión es .wav.

Sabiendo esto y teniendo la ruta del archivo creamos una instancia de la clase Recognizer que pertenece al módulo de speech_recognition para trabajar con sus metodos.

```
# recognize speech usando Google Speech Recognition
try:
    # Llamamos al metodo de reconocimiento por google y le pasamos el audio
    salida = "Google dice: " + r.recognize_google(audio) # Guardamos la salida en una variable
except sr.UnknownValueError: # Definimos excepciones que se puedan presentar
    salida = ("Google Speech Recognition no pudo entender el audio")
except sr.RequestError as e:
    salida = ("no se pueden usar los servicios de Google Speech Recognition; {0}".format(e))
```

Realizamos el proceso de reconocimiento de voz con ayuda de la api de google speech recognition, llamamos al método de recognize_google y le pasamos como método el audio creado anteriormente en base al archivo de audio que se le paso a la función, este resultado lo guardamos en una variable llamada salida para ser mostrado posteriormente.

```
try:
    # Llamamos al metodo de reconocimiento por wit y le pasamos el audio, y la key
   salida2 =  ("Wit.ai dice: " + r.recognize_wit(audio, key=WIT_AI_KEY)) # Guardamos la salida en una variable
except sr.UnknownValueError: # Definimos excepciones que se puedan presentar
    salida2 = ("Wit.ai no pudo entender el audio")
except sr.RequestError as e:
    salida2 = ("no se pueden usar los servicios de Wit.ai; {0}".format(e))

self.textEdit2.setPlainText('--------- Transcripcion ---------' + '\n' + salida + '\n'
                    +  salida2)  # Imprimimos en el campo de texto las salidas
```

De igual forma hacemos para trabajar con la api de Wit.ai, solo que aca necesitamos de una Key la cual podemos obtener si nos registramos, acá también llamamos al método para reconocer con wit y le pasamos el audio y la salida la guardamos en una variable para que después sea mostrada

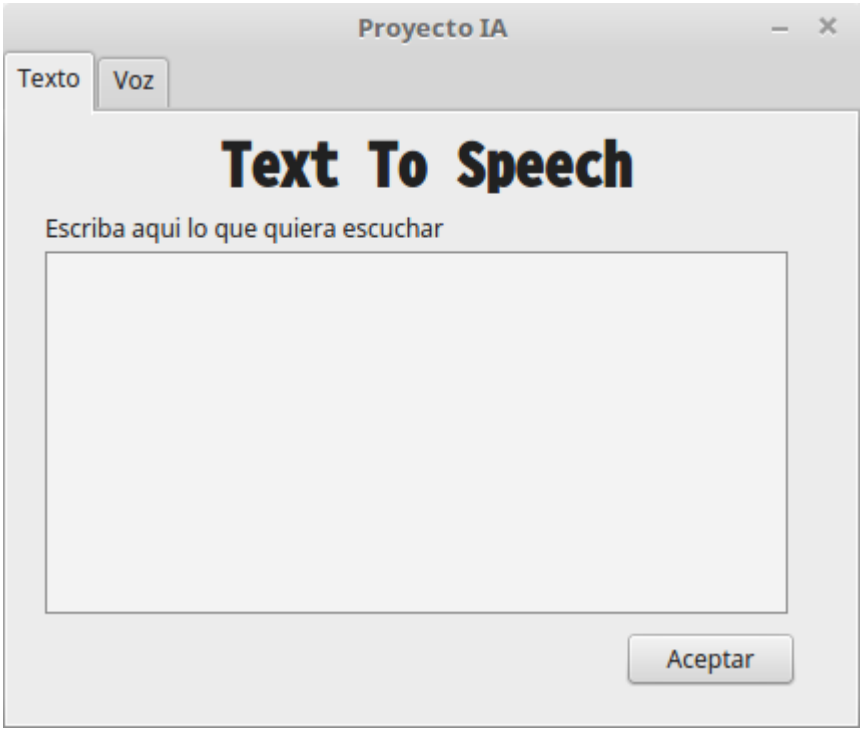
Y por último mostramos todas estas salidas en el campo de texto creado para esto.

Muchas de estas api lo que tienen en sus servidores son archivos inmensos con una cantidad de palabras y como sería su pronunciación, un ejemplo de esto es pocketsphinx es un diccionario offline con miles de palabras en inglés y nos ayuda a realizar este tipo de aplicaciones. La siguiente imagen mostrara como se ve un diccionario de estos

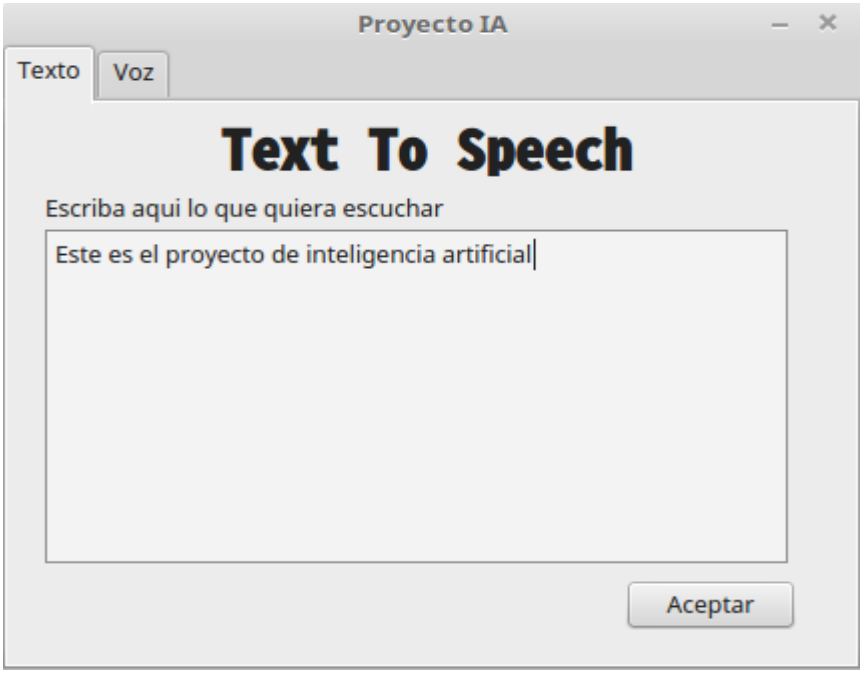
```
'bout B AW T
'cause K AH Z
'course K AO R S
'cuse K Y UW Z
'em AH M
'frisco F R IH S K OW
'gain G EH N
'kay K EY
'm AH M
'n AH N
'round R AW N D
's EH S
'til T IH L
'tis T IH Z
'twas T W AH Z
a AH
a's EY Z
a(2) EY
```

Este diccionario cuenta con más de 10000 palabras, pero todas en ingles.

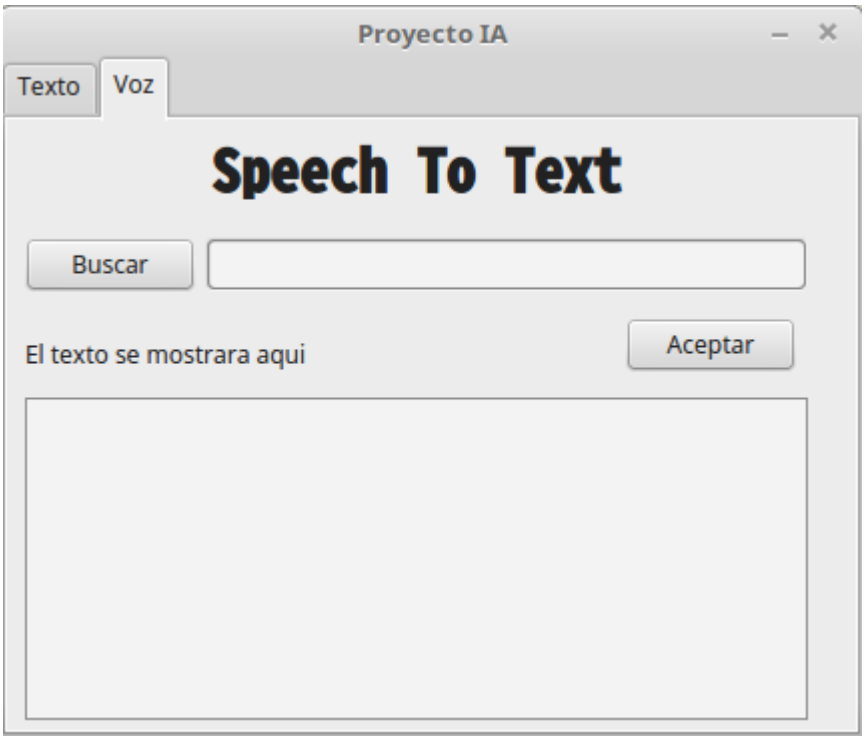
Aplicación



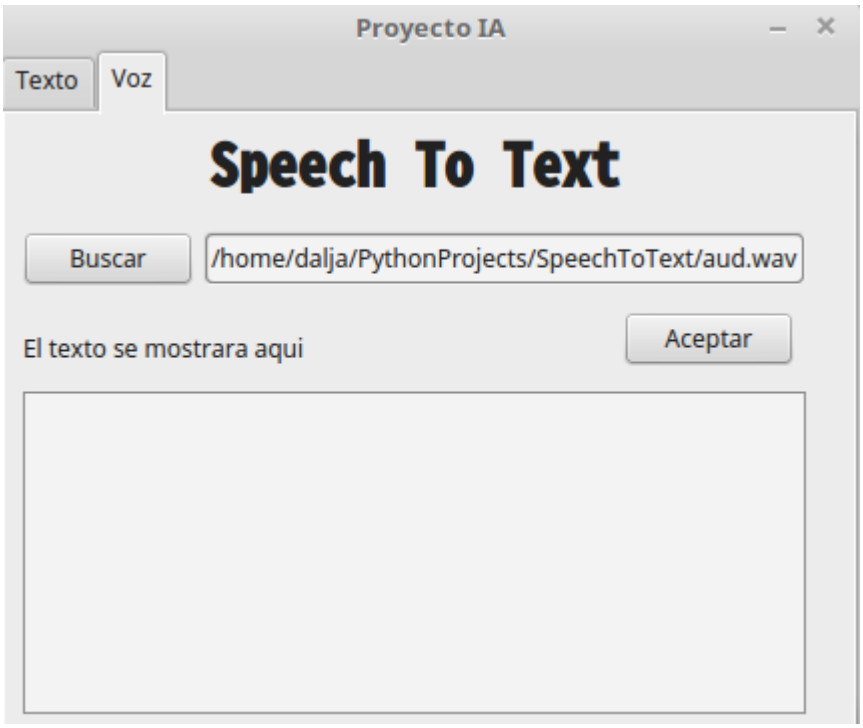
Ventana para la función de pasar de texto a voz



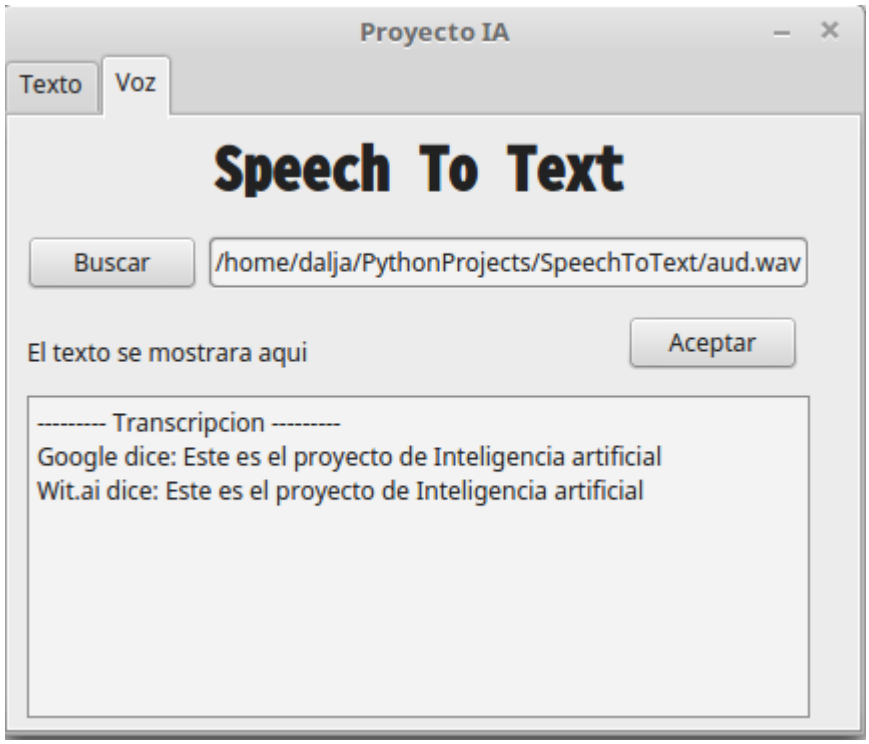
Vemos un ejemplo de cómo escribir lo que pasaremos a voz, solo resta darle al botón aceptar



Ventana para pasar de voz a texto



Ya tenemos la ruta del archivo de audio .wav a ser reconocido



Y vemos como se produce el reconocimiento de la voz y la salida que obtuvimos.

Para finalizar este informe como conclusión tenemos que es importante para nosotros como estudiantes conocer todas estas herramientas ya que nos hacen las tareas mucho más fáciles a la hora de programar o trabajar, ya que muy fácilmente podemos crear programas automatizados por voz