

Informe Procesamiento de lenguaje natural
Traductor de voz a texto

Presentado por:
Jose Daniel Toquica Agudelo
Yerson Arley Cenón Cabrera
Angie Alexandra Ortiz Palacios

Universidad de la Amazonia
Facultad de ingeniería
Ingeniería de sistemas
Florencia – Caquetá
2023

Informe Procesamiento de lenguaje natural
Traductor de voz a texto

Inteligencia computacional

Presentado por:
Jose Daniel Toquica Agudelo
Yerson Arley Cenón Cabrera
Angie Alexandra Ortiz Palacios

Presentado a:
Jesús Emilio Pinto Lopera

Universidad de la Amazonia
Facultad de ingeniería
Ingeniería de sistemas
Florencia – Caquetá
2023

Tabla de contenido

1	Introducción.....	5
2	Metodología.....	6
2.1	Descripción conceptual.....	6
2.1.1	Librerías.....	6
2.1.2	Lenguaje natural	9
2.1.3	.srt	9
2.2	Explicación algoritmo	9
2.2.1	Voz a texto	10
2.2.2	Traducir y transcribir video a texto	12
3	Resultados.....	18
3.1	Voz a texto.....	18
3.2	Traducción vídeo a texto.....	19
4	Referencias	22

Figura 1 Arquitectura Whisper	7
Figura 2 Librerías Python	9
Figura 3 Crear objeto para el reconocimiento de voz.....	10
Figura 4 Configurar dispositivo de entrada de audio	10
Figura 5 Crear objeto traductor	10
Figura 6 Captura de audio en tiempo real.....	11
Figura 7 Detección de voz	11
Figura 8 Convertir audio en texto	11
Figura 9 Traducción de texto a ingles.....	12
Figura 10 configuración ruta	12
Figura 11 Función mostrar menú.....	13
Figura 12 Seleccionar archivo	13
Figura 13 Realizar traducción.....	14
Figura 14 Dividir texto	15
Figura 15 Generar subtítulos	16
Figura 16 Crea vídeo con subtítulos	17
Figura 17 Configuración menú.....	18
Figura 18 Verificación script	18
Figura 19 Resultado captura de audio	19
Figura 20 opciones tipo archivo e idioma	19
Figura 21 Selección vídeo	19
Figura 22 Espera de traducción	20

1 Introducción

La tecnología avanza rápidamente y la manera en la que interactuamos con ella también, en ese sentido es importante reconocer la necesidad de mejorar procesos que involucren una relación máquina – humano. El modo en que nos comunicamos va cambiando continuamente, así mismo pasa con las plataformas de video, entretenimiento y demás, los usuarios finales necesitan eficacia e inmediatez para disfrutar del contenido de voz, texto y video.

El procesamiento de lenguaje natural (PLN) es una rama de la inteligencia artificial que se encarga de la interacción entre las máquinas y el lenguaje humano, tiene como finalidad permitir que las computadoras comprendan, interpreten y generen texto. Como expresa Echeverri y Manjarrés (2020) estudia las interacciones entre las computadoras y el lenguaje humano, por medio del análisis sintáctico, semántico, pragmático y morfológico; se escriben reglas de reconocimiento de patrones estructurales, empleando un formalismo gramatical concreto.

Los traductores de video/voz a texto son una tecnología que facilitan el reconocimiento de las palabras por medio de voz para transcribirlas a texto, este proceso tiene gran impacto para los usuarios, debido a la accesibilidad, la facilidad para buscar contenido, la automatización de procesos y para la creación de subtítulos, así buscando el beneficio para las personas con discapacidades auditivas.

En ese sentido, integrar el procesamiento de lenguaje natural con los traductores de video/voz a texto se presenta como una oportunidad para tener soluciones más eficientes en situaciones donde se requiera extraer texto de formatos de voz vídeo, en el presente trabajo abordaremos el desarrollo de un algoritmo para la transcripción de voz a texto.

2 Metodología

El presente trabajo se divide en dos partes, en la primera parte abarcaremos conceptos básicos, palabras claves, las librerías de Python y técnicas utilizadas. En la segunda parte de la metodología vamos a explicar a detalle el algoritmo implementado para el traductor haciendo uso del procesamiento de lenguaje natural.

2.1 Descripción conceptual

2.1.1 Librerías

A continuación, encontraremos las librerías de Python implementadas en el desarrollo del algoritmo para el traductor de voz a texto. (Figura 1)

2.1.1.1 *Whisper*

Es un modelo traductor, Whisper es un ASR basado en la arquitectura del Transformer, entrenado con 680000 horas de audio de 96 lenguajes y está entrenado para realizar diferentes tareas de procesamiento (Calva Galeas, 2023)

Tareas de procesamiento que puede realizar Whisper:

Transcripción: Permite obtener la transcripción de audio a texto.

Traducción: Permite traducir el audio al inglés.

Detección del idioma: Permite detectar el idioma de un audio. (Calva Galeas, 2023)

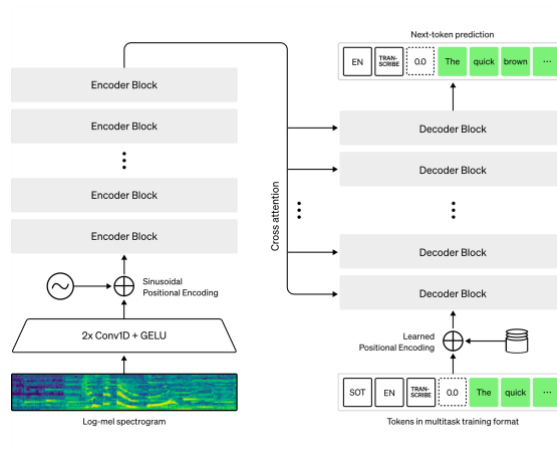


Figura 1 Arquitectura Whisper

Fuente: <https://openai.com/research/whisper>

La señal de audio se transforma en espectrograma log-Mel y este sirve como entrada al encoder del Transformer. El decoder del Transformer se entrenó para devolver la respectiva predicción sobre el audio y a través de determinados tokens especiales con los que fue entrenado, permitiendo que se puedan realizar las tareas de procesamiento. (Calva Galeas, 2023)

2.1.1.2 *speech_recognition*

Reconocimiento de voz, utilizado en inteligencia artificial, le permite a las máquinas interpretar y convertir la voz a texto, este analiza las características del habla, como el tono, ritmo y demás, con el fin de transcribir las palabras en un texto limpio y coherente. El reconocimiento de voz rudimentario tiene un vocabulario limitado y sólo puede identificar palabras y frases cuando se pronuncian con claridad. (Lutkevich & Kiwak, 2021)

2.1.1.3 *Subprocess*

El subprocessmódulo le permite generar nuevos procesos, conectarse a sus canales de entrada/salida/error y obtener sus códigos de retorno. Este módulo

pretende reemplazar varios módulos y funciones (*Subprocess — Subprocess Management*, 2023)

2.1.1.4 Tkinter

Importa todas las clases y funciones del módulo tkinter. Es una librería de Python que permite diseñar la interfaz gráfica para el usuario por medio de widgets que permiten la intercomunicación constante entre funciones, cálculos y gráficos. (Naranjo Chisaguano, 2023)

2.1.1.5 Askopenfilename

Esta función de Python es útil para función para mostrar un cuadro de diálogo de archivo abierto que permita a los usuarios seleccionar un archivo. También la función ‘askopenfilenames’ para mostrar un cuadro de diálogo para abrir archivos que permita a los usuarios seleccionar varios archivos. (*Tkinter Open File Dialog*, 2021)

2.1.1.6 MoviePy

Es una librería de Python para edición de video: corte, concatenaciones, inserciones de títulos, composición de video, procesamiento de video y creación de efectos personalizados. (*Moviepy*, 2020)

2.1.1.7 Imagemagick

Es una herramienta de software que se utiliza para convertir imágenes de un formato a otro. También es capaz de leer y escribir imágenes de diferentes formatos, además permite combinar operaciones de procesamiento de imágenes en un script. (Dhruvi Vikma, 2021)

2.1.1.8 Os

Este módulo provee una manera versátil de usar funcionalidades dependientes del sistema operativo, algunas como leer, escribir, manipular archivos y demás. (*Os —*


```
import speech_recognition as sr
from googletrans import Translator
import subprocess
import os
from tkinter import Tk
from tkinter.filedialog import askopenfilename
from moviepy.config import change_settings
from moviepy.video.tools.subtitles import file_to_subtitles, SubtitlesClip
from moviepy.editor import VideoFileClip, TextClip, CompositeVideoClip
```

Figura 2 Librerías Python

2.1.2 Lenguaje natural

El lenguaje natural permite al mundo de los dispositivos informáticos comprender, interpretar y manipular el lenguaje humano. La otra categoría es la que se centra en los chatbots, los cuales son agentes de conversación digitales que utilizan métodos de IA a través de texto y/o voz para imitar el comportamiento humano a través de un diálogo en evolución. (Sancho, Fanjul, Iglesia, Montell, Escartí, 2020)

2.1.3 .srt

Un archivo SubRip Subtitle, es uno de los formatos de archivo de subtítulos más populares para contenido de vídeo. Estos archivos de texto sin formato incluyen el texto de los subtítulos en secuencia, junto con los códigos de tiempo de inicio y finalización. (*¿Qué Es Un Archivo SRT? Explicación de Los Archivos SRT* | Mailchimp, 2023)

2.2 Explicación algoritmo

En esta segunda parte de la metodología vamos a explicar más a detalle el código haciendo uso de las herramientas implementadas.

2.2.1 Voz a texto

Objeto Voz: Se crea un objeto 'r' de la clase 'Recognizer' el cual es esencial para el reconocimiento de voz. Este hace uso de la librería 'speech_recognition'. (Figura 3)

```
# Crear un objeto de reconocimiento de voz  
r = sr.Recognizer()
```

Figura 3 Crear objeto para el reconocimiento de voz

Configurar dispositivo: Se crea un objeto denominado 'mic' de la clase 'Microphone' el cual representa el micrófono que será utilizado para la captura del audio, además proporciona funcionalidades del micrófono. (Figura 4).

```
# Configurar el dispositivo de entrada de audio  
mic = sr.Microphone()
```

Figura 4 Configurar dispositivo de entrada de audio

Objeto traductor: Se crea un objeto llamado 'translator' de la clase 'Translator' la cual es proporcionada por la librería 'googletrans' su función es realizar la traducción del texto en diferentes idiomas. (Figura 5)

```
# Crear un objeto de traductor  
translator = Translator()
```

Figura 5 Crear objeto traductor

Captura de audio: Se hace uso de 'with' para iniciar la captura de audio utilizando el micrófono configurado anteriormente. A continuación, se llama el método 'adjust_for_ambient_noise(source)' del objeto 'r' creado previamente, para la supresión de ruido. (Figura 6)

```
# Iniciar la captura de audio en tiempo real
with mic as source:
    # Ajustar el nivel de ruido de fondo para una mejor captura de audio
    r.adjust_for_ambient_noise(source)
```

Figura 6 Captura de audio en tiempo real

Leer audio de micrófono: Se utiliza un bucle ‘while true’ para leer el audio que se transmite por medio del micrófono haciendo uso del objeto ‘Recognizer’ y llamando al método ‘listen(source)’ y asignando los fragmentos de audio capturado al objeto ‘audio’. Para finalizar el bucle de ser interrumpido manualmente. (Figura 7)

```
# Leer el audio del micrófono en tiempo real
print("Comenzando la captura de audio...")
while True:
    try:
        audio = r.listen(source)
```

Figura 7 Detección de voz

Convertir audio en texto: Se hace uso de del objeto ‘recognizer’ realiza el reconocimiento de voz y convertir el audio capturado en texto utilizando la API de reconocimiento de voz de Google ‘recognize_google’ y especifica que el audio debe estar en español. (Figura 8)

```
# Utilizar el reconocimiento de voz para convertir el audio en texto
text = r.recognize_google(audio, language="es-ES")
print("Texto reconocido (español):", text)
```

Figura 8 Convertir audio en texto

Traducción: Luego de obtener el texto en español se hace uso del objeto ‘translator’ para traducir el texto obtenido a inglés, el resultado de la traducción se almacena en la variable ‘translation’, a continuación, imprime por consola el texto en inglés ‘translation.text’, para finalizar hay dos excepciones, en las cuales se muestra un mensaje por consola argumentando que no fue posible reconocer el audio o no se pudo realizar la solicitud al servicio. (Figura 9).

```
# Traducir el texto al inglés
translation = translator.translate(text, src='es', dest='en')
print("Translated text (English):", translation.text)
except sr.UnknownValueError:
    print("No se pudo reconocer el audio")
except sr.RequestError as e:
    print("Error al realizar la solicitud al servicio de reconocimiento de voz:", str(e))
```

Figura 9 Traducción de texto a inglés

2.2.2 Traducir y transcribir video a texto

Configuración de ruta: Tenemos la configuración de la ruta, en la cual se hace uso de la librería imagemagick, la cual fue explicada en la primera parte de la metodología, este se utiliza para especificar la ruta del archivo binario imagemagick (magick.exe), nos indica la ruta donde se encuentra en el equipo. (Figura 10)

```
change_settings({"IMAGEMAGICK_BINARY": r"C:/Program Files/ImageMagick-7.1.1-Q16-HDRI/magick.exe"})
```

Figura 10 configuración ruta

Mostrar menú: En la siguiente imagen desde la línea 11 a la 23 se define una función denominada ‘mostrar menu()’ la cual le va a permitir al usuario interactuar e introducir el ‘1’ o el ‘2’ para elegir el tipo de archivo (audio o video). La elección del usuario se va a almacenar en la variable ‘tipo_archivo’ definido en la línea 13. En la línea 16 se define un ciclo ‘while’ que valida la entrada del usuario hasta que la solicitud sea correcta. Una vez validada la solicitud se convierte la entrada del usuario al tipo de archivo

correspondiente, si el usuario ingreso '1' se almacena el valor audio en la variable 'tipo_archivo'. Si el usuario ingresó "2", se almacena el valor 'video' en la variable tipo_archivo. Al finalizar, la función mostrar_menu() devuelve el tipo de archivo seleccionado por el usuario. (Figura 11)

```
11 def mostrar_menu():
12     # Solicitar al usuario que ingrese "1" o "2" para seleccionar el tipo de archivo
13     tipo_archivo = input("Selecciona el tipo de archivo:\n1. Audio\n2. Video\nIngrese el número correspondiente: ")
14
15     # Validar la entrada del usuario
16     while tipo_archivo not in ['1', '2']:
17         print("Opción inválida. Por favor ingresa '1' para audio o '2' para video.")
18         tipo_archivo = input("Selecciona el tipo de archivo:\n1. Audio\n2. Video\nIngrese el número correspondiente: ")
19
20     # Convertir la entrada del usuario a tipo de archivo correspondiente
21     tipo_archivo = 'audio' if tipo_archivo == '1' else 'video'
22
23     return tipo_archivo
```

Figura 11 Función mostrar menú

Seleccionar archivo: En el siguiente bloque de código en la línea 25 a la 34 se define la función llamada 'seleccionar archivo' que permite al usuario abrir el explorador de archivos para poder seleccionar el archivo que desea convertir.

En este caso la función 'seleccionar_archivo' toma un argumento denominado 'tipo', que indica si el usuario desea seleccionar un archivo de video o audio. Según el valor de 'tipo' la función filtra los tipos de archivos que se muestran en el explorador de archivos.

La condición definida en la línea 29 a la 32 hace que cuando el usuario selecciona un archivo y hace clic en "Abrir", la función devuelve la ruta del archivo seleccionado. Si el usuario cancela la selección de archivos, la función devuelve 'none'. (Figura 12)

```
25 def seleccionar_archivo(tipo):
26     # Abrir el explorador de archivos para seleccionar el archivo
27     Tk().withdraw()
28
29     if tipo == 'audio':
30         archivo = askopenfilename(initialdir="./data", title="Seleccionar archivo de audio", filetypes=(("Archivos de audio", ".wav"), ("Todos los archivos", "*.*")))
31     else:
32         archivo = askopenfilename(initialdir="./data", title="Seleccionar archivo de video", filetypes=(("Archivos de video", ".mp4;.avi"), ("Todos los archivos", "*.*")))
33
34     return archivo
```

Figura 12 Seleccionar archivo

Realizar traducción: En este bloque de código que abarca desde la línea 36 a la 45, se define la función denominada ‘realizar_traducccion’ la cual toma 2 argumentos ‘archivo’ e ‘idioma’. La función primero imprime "Traduciendo..." en la línea 37 para indicar que el proceso de traducción ha comenzado. Luego, en la línea 39 crea una variable denominada ‘comando’ y utilizando una cadena formateada (f-string). Este comando utiliza Whisper, una biblioteca de traducción de texto, para traducir el archivo proporcionado a un idioma específico. La función crea una cadena de comando que incluye la ruta del archivo, el modelo de traducción, el idioma y el directorio de salida para guardar el archivo traducido. A continuación, en la línea 41 se utiliza una declaración ‘with’ para redirigir la salida estándar y la salida de error estándar a os.devnull. Esto evita que los mensajes de la terminal durante la ejecución del comando se muestren al usuario. Finalmente, en la línea 42 la función ejecuta el comando utilizando ‘subprocess.run()’. Este método ejecuta un comando y espera a que se complete antes de continuar. Una vez que la traducción se ha completado con éxito, la función imprime "La traducción se ha completado con éxito.". (Figura 13)

```
36 def realizar_traducccion(archivo, idioma):
37     print("Traduciendo...")
38     # Realizar la traducción utilizando Whisper
39     comando = f'whisper "{archivo}" --model small --language {idioma} --output_dir "./data/translate"'
40
41     with open(os.devnull, 'w') as null:
42         subprocess.run(comando, shell=True, stdout=null, stderr=null)
43
44     # Imprimir el mensaje de éxito
45     print("La traducción se ha completado con éxito.")
```

Figura 13 Realizar traducción

Dividir texto: En el siguiente bloque de código que va desde la línea 47 a la 60 se define la función denominada ‘dividir_texto’, que se encargara de dividir un texto en líneas para ser renderizado en el video. Manteniendo el texto dentro del ancho disponible. Esta función recibe como argumentos el texto, el ancho disponible y el ancho del video. Para ello, primero en la línea 48 se divide el texto en palabras

utilizando el método `split()`. Luego, se inicializan dos variables: en la línea 49 la variable denominada 'lineas' para almacenar las líneas del texto y en la línea 50 se define la variable denominada 'linea_actual' para almacenar la línea actual. Después, en la línea 52 se crea un ciclo 'for' para iterar sobre cada palabra en el texto. Dentro de este ciclo se crea una condición desde la línea 53 a la 57, que consiste en que, para cada palabra, se crea un objeto 'TextClip' con la línea y la palabra actuales, y se obtiene el ancho (w) de este objeto. Además, si el ancho del objeto 'TextClip' es mayor (>) que el ancho disponible, significa que la línea actual está llena y se debe comenzar una nueva línea. Por lo tanto, se agrega la línea actual al final de la lista `lineas` y se reinicia la variable `linea_actual` con la palabra actual. En caso de ser menor o igual (<=) significa que aún queda espacio en la línea actual. Por lo tanto, se agrega la palabra actual a la línea actual y se continúa con la siguiente palabra. En la línea 59, una vez que se han procesado todas las palabras en el texto, se agrega la última línea al final de la lista 'lineas' para completar el proceso. Finalmente, en la línea 60 la función devuelve el texto dividido en líneas, uniendo todas las líneas con saltos de línea (\n). (Figura 14)

```
47 def dividir_texto(texto, ancho_disponible, video_ancho):
48     palabras = texto.split()
49     lineas = []
50     linea_actual = ""
51
52     for palabra in palabras:
53         if TextClip(linea_actual + " " + palabra, fontsize=video_ancho // 20).w > ancho_disponible:
54             lineas.append(linea_actual)
55             linea_actual = palabra
56         else:
57             linea_actual += " " + palabra
58
59     lineas.append(linea_actual)
60     return "\n".join(lineas)
```

Figura 14 Dividir texto

Generar subtítulos: En el siguiente bloque de código que abarca desde la línea 62 a la 73 se define la función denominada 'generar_subtitulos'. Esta función toma dos

argumentos ‘archivo’, que es la ruta al archivo de video, y ‘ruta_subtitulos’, que es la ruta al archivo de subtítulos. En la línea 63 y 64 carga el archivo de video usando ‘VideoFileClip’. Este archivo de video se utiliza para obtener el ancho y alto del video. A continuación, en la línea 66 define una función anidada llamada ‘generator’ que toma un argumento ‘txt’. La función ‘generator’ devuelve un objeto TextClip creado a partir del texto proporcionado. Luego, en la línea 70 y 71 utiliza la función ‘file_to_subtitles’ para cargar los subtítulos desde un archivo ‘.srt’. Estos subtítulos se convierten en un objeto ‘SubtitlesClip’ utilizando la función ‘generator’ definida anteriormente. Finalmente, en la línea 73 la función ‘generar_subtitulos’ devuelve el objeto ‘SubtitlesClip’ creado a partir de los subtítulos y el archivo de video. (Figura 15)

```
62 def generar_subtitulos(archivo, ruta_subtitulos):
63     video_original = VideoFileClip(archivo)
64     video_ancho, video_alto = video_original.size
65
66     def generator(txt):
67         return TextClip(dividir_texto(txt, video_ancho, video_alto), font='Arial', fontsize=video_ancho // 20, color='white')
68
69     # Crear los subtítulos adaptados al ancho del video
70     srt_subtitles = file_to_subtitles(ruta_subtitulos)
71     subtitles = SubtitlesClip(srt_subtitles, generator)
72
73     return subtitles
```

Figura 15 Generar subtítulos

Crear video con subtítulos: En el siguiente bloque de código que va desde la línea 75 a la 82 se define la función denominada ‘crear_video_con_subtitulos’, que toma dos argumentos: ‘video_original’ y ‘subtitles’. En la línea 76 se define la impresión de un texto en consola que dice “Creando Video...” para indicarle al usuario que el proceso ha comenzado. Luego en la línea 79 se utiliza la función ‘CompositeVideoClip’ de la biblioteca ‘moviepy’ para combinar el video original con los subtítulos. Además, se utiliza la función denominada ‘set_position’ para posicionar los subtítulos en la parte inferior del video. Finalmente, en la línea 82 se escribe el video resultante en un archivo

de video denominado 'video_con_subtitulos.mp4'. además, se utiliza el codec 'libx264' para la codificación del video y el codec 'acc' para la codificación del audio. (Figura 16)

```
75 def crear_video_con_subtitulos(video_original, subtitles):
76     print("Creando Video...")
77
78     # Combinar el video original con los subtitulos
79     result = CompositeVideoClip([video_original, subtitles.set_position(('center', 'bottom'))])
80
81     # Exportar el video resultante
82     result.write_videofile("video_con_subtitulos.mp4", codec='libx264', audio_codec='aac')
```

Figura 16 Crea video con subtitulos

Menú: En el siguiente bloque de código que abarca desde la línea 84 a la 98 se define la función 'main()' la cual va a inicializar las funciones anteriores. Inicialmente en la línea 85 se llama a la función 'mostrar_menu()', que muestra un menú al usuario para seleccionar el tipo de archivo (audio o video). El resultado se almacena en la variable tipo_archivo. Luego en la línea 86 se llama la función 'seleccionar_archivo(tipo_archivo)', que utiliza el tipo de archivo seleccionado para abrir un cuadro de diálogo y permitir al usuario seleccionar un archivo específico. El resultado se almacena en la variable archivo. A continuación, en la línea 87 se solicita al usuario que ingrese el idioma al que desea traducir el archivo de audio o video. En la línea 89 se llama la función 'realizar traducción' que utiliza la herramienta Whisper para realizar la traducción del archivo al idioma especificado. Ahora en la línea 91 se obtiene el nombre base del archivo sin la extensión. Esto se utiliza para construir el nombre del archivo de subtitulos traducidos. Y en la línea 92 se construye la ruta para los subtitulos traducidos, utilizando la carpeta "./data/translate/" y el nombre del archivo base sin extensión. Luego en la línea 94 se carga el video original desde el archivo especificado utilizando la biblioteca 'moviepy', y en la línea 95 se llama a la función 'generar_subtitulos(archivo, ruta_subtitulos)', que genera los subtitulos adaptados al ancho del video. Finalmente, en la línea 97 se combina el video original con los

subtítulos. Los subtítulos se posicionan en la parte inferior central del video, y en la línea 98 se exporta el video resultante a un archivo llamado "video_con_subtitulos.mp4" utilizando el códec de video 'libx264' y el códec de audio 'aac' como se había mencionado antes. (Figura 17)

```
84 def main():
85     tipo_archivo = mostrar_menu()
86     archivo = seleccionar_archivo(tipo_archivo)
87     idioma = input("Idioma de traducción: ")
88
89     realizar_traducccion(archivo, idioma)
90
91     nombre_archivo_sin_extension = os.path.splitext(os.path.basename(archivo))[0]
92     ruta_subtitulos = f"./data/translate/{nombre_archivo_sin_extension}.srt"
93
94     video_original = VideoFileClip(archivo)
95     subtitles = generar_subtitulos(archivo, ruta_subtitulos)
96
97     result = CompositeVideoClip([video_original, subtitles.set_position(('center', 'bottom'))])
98     result.write_videofile("video_con_subtitulos.mp4", codec='libx264', audio_codec='aac')
```

Figura 17 Configuración menú

Verificación: finalmente este fragmento de código que abarca la línea 100 y 101 se encarga de verifica si el script se está ejecutando como programa principal. (Figura 18)

```
100 if __name__ == "__main__":
101     main()
```

Figura 18 Verificación script

3 Resultados

3.1 Voz a texto

Captura de audio: cómo podemos observar (Figura 19) nos muestra mediante consola los resultados del reconocimiento de voz por medio del micrófono, tenemos los parámetros 'texto reconocido' el cual hace referencia a las palabras obtenidas de manera clara, el segundo parámetro 'translated text' nos muestra el texto reconocido con su traducción al idioma inglés.

```

C:\Computacional\Lenguaje Natural\csent.py
Comenzando la captura de audio...
Texto reconocido (español): Hola Esto es una prueba
Translated text (English): Hi this is a test
Texto reconocido (español): tercera entrega
Translated text (English): third installment
Texto reconocido (español): inteligencia computacional la cual la dicta el doctor Jesús Emilio Pinto lopera y la present
an los futuros ingenieros José Daniel tópica Angie Alexander Ortiz y Jerry Palacios Jajaja Hola ahora sí ya le envié la
Imagen tiene que decir que pues debido a la a la la Pues que eso requiere máquina y que no sé qué Y pues que el PC que se v
ea con el que se hizo las pruebas no tiene suficiente o sea explicar básicamente que está que que no toma todas las imá
enes se anotó todas las palabras por tema de computación Dame tú estás viendo lo que estoy compartiendo cierto
Translated text (English): computational intelligence which is dictated by Dr. Jesús Emilio Pinto Lopera and is presente
d by future engineers José Daniel Topic Angie Alexander Ortiz and Jerry Palacios Hahaha Hello now I already sent the ima
ge you have to say that because due to it to the reason why that requires thatmachine and that I do not know what and th
en the PC that is seen with which the tests was done does not have enough or basically explain that it is that it does n
ot take all the images all the words were scored by computing theme give me you are seeingWhat I am sharing true

```

Figura 19 Resultado captura de audio

3.2 Traducción vídeo a texto

Opciones: Debemos seleccionar el tipo de archivo y el idioma al que queremos traducir, como podemos observar en la figura 20 tenemos la opción vídeo y el idioma portugués.

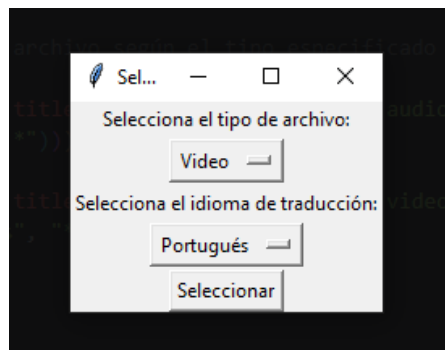


Figura 20 opciones tipo archivo e idioma

Selección vídeo: A continuación, se nos va a desplegar una ventana para seleccionar el vídeo que deseamos transcribir y traducir. (Figura 21)

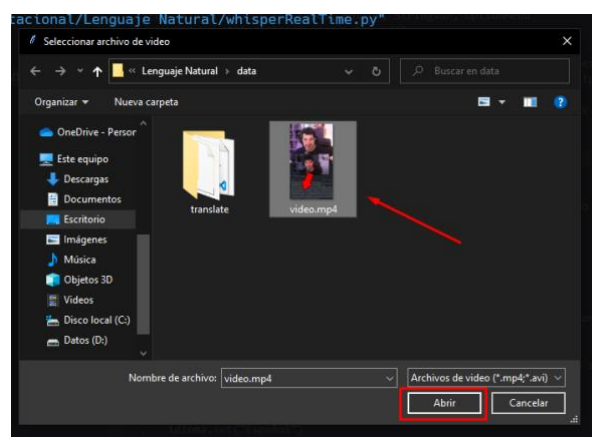


Figura 21 Selección vídeo

Espera: Ahora, esperamos unos minutos mientras el programa realiza la traducción, cabe recalcar que el tiempo estimado de espera depende de las especificaciones del equipo utilizado. (Figura 22)

A terminal window with a dark background. The text 'portugués' is displayed in a light blue font. Below it, 'Traduciendo...' is shown in a light green font. To the right, a faint, semi-transparent text 'a. Mostrar el cuadro' is visible.

Figura 22 Espera de traducción

Al finalizar, nos muestra por consola el siguiente mensaje (Figura 23)

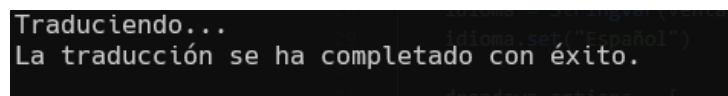
A terminal window with a dark background. The text 'Traduciendo...' is shown in light green. Below it, 'La traducción se ha completado con éxito.' is displayed in a light blue font. To the right, a faint, semi-transparent text ' idioma.set("Español")' is visible.

Figura 23 Traducción completada

Creación vídeo con subtítulos: A continuación, nos muestra el proceso de carga del nuevo vídeo con los subtítulos en el idioma seleccionado previamente. (Figura 24)

A terminal window with a dark background. The text 'Traduciendo...' is shown in light green. Below it, 'La traducción se ha completado con éxito.' is displayed in a light blue font. To the right, a faint, semi-transparent text ' idioma.set("Español")' is visible.

Figura 24 Creación vídeo con subtítulos

Cunado el vídeo ya ha terminado su proceso de carga nos muestra los siguientes mensajes (Figura 25)

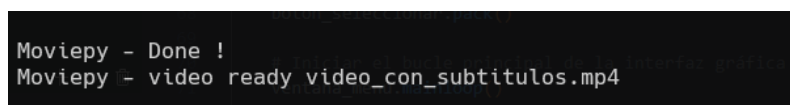
A terminal window with a dark background. The text 'Moviepy - Done !' is shown in light blue. Below it, 'Moviepy - video ready video_con_subtitulos.mp4' is displayed in a light green font. To the right, a faint, semi-transparent text 'interfaz grafica' is visible.

Figura 25 Vídeo creado

Video traducido: A continuación, podemos visualizar el vídeo original (Figura 26)

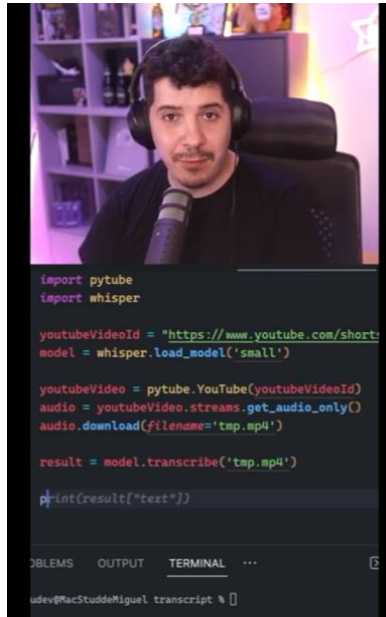


Figura 26 Video original

Al finalizar, veremos el vídeo traducido con sus respectivos subtítulos que están ubicados en la parte inferior de la pantalla. (Figura 27)

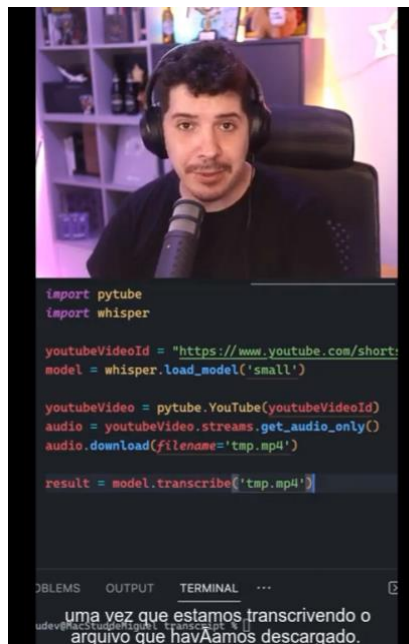


Figura 27 Video traducido

4 Referencias

Echeverri Torres, M.M., Manjarrés Betancur, R. (2020) *Asistente virtual académico utilizando tecnologías cognitivas de procesamiento de lenguaje natural.*

<https://www.redalyc.org/journal/6078/607863449007/607863449007.pdf>

Calva Galeas, C.E. (2023) *Procesamiento acústico de la voz aplicado al reconocimiento de fonemas similares en el habla inglesa comparados con sus homólogos en lengua hispánica: Evaluación de modelos basados en Transformer para la detección de errores de lectura a nivel de palabras.*

<https://bibdigital.epn.edu.ec/handle/15000/24931>

Lutkevich, B., & Kiwak, K. (2021). *speech recognition*. Customer Experience; TechTarget.

<https://www.techtarget.com/searchcustomerexperience/definition/speech-recognition#:~:text=Speech%20recognition%2C%20or%20speech%2Dto,convert%20them%20into%20readable%20text.>

Sancho Escrivá, J.V., Fanjul Peyró, C., Vayá, M., Montell, J.A., Escartí Fabra, M.J. (2020) *Aplicación de la inteligencia artificial con procesamiento del lenguaje natural para textos de investigación cualitativa en la relación médico-paciente con enfermedad mental mediante el uso de tecnologías móviles.*

<https://repositori.uji.es/xmlui/handle/10234/190883>

subprocess — *Subprocess management*. (2023). Python Documentation.

<https://docs.python.org/3/library/subprocess.html>

Naranjo Chisaguano, C.A. (2023) *Desarrollo de una herramienta informática para evaluación del comportamiento termodinámico de sistemas de refrigeración por compresión de vapor con refrigerantes naturales propano R-290* palabras.

<https://bibdigital.epn.edu.ec/bitstream/15000/24376/1/CD%2013296.pdf>

os — Interfaces misceláneas del sistema operativo — documentación de Python - 3.10.13. (2023). Python.org. <https://docs.python.org/es/3.10/library/os.html>

Tkinter Open File Dialog. (2021, January 6). Python Tutorial - Master Python Programming for Beginners from Scratch.

[https://www.pythontutorial.net/tkinter/tkinter-open-file-dialog/#:~:text=Use%20the%20askopenfilenames\(\)%20function,file%20or%20multiple%20file%20objects.](https://www.pythontutorial.net/tkinter/tkinter-open-file-dialog/#:~:text=Use%20the%20askopenfilenames()%20function,file%20or%20multiple%20file%20objects.)

moviepy. (2020, May 7). PyPI. <https://pypi.org/project/moviepy/>

Dhruvi Vikma. (2021, June 3). *The Ultimate Guide of ImageMagick in Python.* Python Pool. <https://www.pythonpool.com/imagemagick-python/>

¿Qué es un archivo SRT? Explicación de los archivos SRT / Mailchimp. (2023).

Mailchimp. <https://mailchimp.com/es/resources/what-is-an-srt-file/>

