

# PRÁCTICA 1

## SISTEMAS

## INTELIGENTES

MANUEL SALVADOR  
DANIEL SABBAGH  
ING. INFORMÁTICA  
SISTEMAS INTELIGENTES

# ÍNDICE

Introducción: .....	2
Resumen: .....	3
Descripción: .....	3
La ruta especificada: .....	3
Proceso de grabación: .....	4
Entrenamiento del conductor: .....	4
Gestión del audio: .....	5
La transcripción: .....	6
El procesamiento del texto: .....	6
Edición del texto: .....	7
Inserción de códigos: .....	7
Librerías y paquetes usados: .....	8
Pruebas: .....	8
Resultados: .....	9
Conclusiones: .....	9
Apéndice: .....	10
Bibliografía: .....	12

## Introducción:

Se plantea el problema de conseguir realizar un video recorriendo varias rutas donde se encontrasen stops en coche, con una duración de entre 10 y 15 minutos, tras conseguir el video, se pasará primeramente de video a audio y en tercer lugar se transcribe a texto indicando en cada secuencia especificada los códigos, se muestra un ejemplo a continuación:

**Tabla 1.** Códigos de estímulos de entrada.

Category	Action	Code	Description
Sight	Front view	FV	Look forward in the driving direction
Sight	Front view to the rearview mirror	FV-Mirror	Look through the interior rearview mirror
Sight	Left view	LV	Look left
Sight	Left mirror	LV-Mirror	Look left through the left external mirror
Sight	Front-Left view	FLV	Look diagonally, between front and left
Sight	Right view	RV	Look right
Sight	Right mirror	RV-Mirror	Look right through the right external mirror
Sight	Front-Right view	FRV	Look diagonally, between front and right
Sight	Back view	BV	Look turning head back.
Sight	Back-Left view	BLV	Look back diagonally between left and rear of the vehicle.
Sight	Back-Right view	BRV	Look back diagonally between right and rear of the vehicle.
Sound	Inside sound	IN-S	Sound from inside the vehicle
Sound	Outside sound	OUT-S	Sound coming from outside the vehicle

Por último, se le da la opción de modificar todo el texto resultante y guardarlo en un archivo “\*.txt” a elección del usuario.

Think Aloud Method (Pensamiento en voz alta): En una prueba de pensamiento en voz alta, se pide a los participantes de dicha prueba que realicen una acción y mientras la vayan describiendo, es decir, que verbalicen sus pensamientos mientras lo efectúan.

“Thinking aloud may be the single most valuable usability engineering method.”, Jakob Nielsen escribió esta bonita frase en 1993 y continúa manteniendo dicha evaluación.

El hecho de que este método se haya mantenido como el número uno durante los últimos 28 años, es una buena indicación de la longevidad de los buenos métodos de usabilidad.

La conducción naturalista es un enfoque nuevo diseñado para supervisar el comportamiento de los conductores de forma que permite registrar y recopilar información sobre las rutas que siguen dichos conductores, y con todo ello generar una gran base de datos con información variada de cada persona que recorre las mismas rutas con el objetivo de hacer las carreteras europeas más sostenibles y seguras.



Empezando en la X de color azul y siguiendo el recorrido bajando por la Avenida de Colmenar, donde llegamos a la Avenida de la Industria, aun de color azul, hasta que damos la vuelta en una rotonda, volviendo por el sentido contrario, donde pasamos a color rojo. Subimos por la Avenida de Colmenar hasta la primera salida, donde la cogemos (en el Telepizza), aun de color rojo. Bajamos esa calle, hasta el cruce donde realizamos 2 stops y damos la vuelta en una rotonda y volvemos por la misma calle en sentido contrario, se vuelve a poner en azul. Una vez cogemos esta calle de vuelta la subimos, y cuando llegamos al cruce con la Avenida de Colmenar, giramos a la izquierda, de color azul aun, seguimos por la Avenida hasta que llegamos a la primera salida la cual cogemos (giro derecha en la parte inferior de la imagen). Cuando cogemos esta salida, recorremos la calle hasta llegar al cruce con otra avenida, donde vamos a realizar dos stops. Una vez realizados, bajamos por la calle hasta llegar a una rotonda por donde ya hemos pasado anteriormente (en color rojo). Una vez llegamos a esta rotonda, pasamos a color negro, donde llegamos al cruce con la avenida y realizamos dos stops para cruzarla completamente, una vez cruzada la avenida, se termina el video, en la X de color negro.

Cambios de colores:

X azul -> Rojo -> Azul -> Negro -> X Negra.

### Proceso de grabación:

Para el proceso de grabación, en el equipo se estuvieron planteando diferentes soluciones al problema que representa grabar en el coche mientras hablas. Los principales problemas con los que nos encontramos fueron el audio y estabilizar la imagen. Con estos problemas en mente decidimos usar una "GoPro", en este caso una GoPro Silver 4 con la que grabamos todo el video en una toma. Una vez resuelto nos encontramos con otro problema a la hora de usar la GoPro y es el audio de esta ya que la GoPro se encontraba "pegada" al salpicadero y al estar cerrada en su carcasa de plástico, el audio era muy flojo, por lo que abrimos la carcasa de la GoPro para grabar el vídeo y además grabamos el audio con un móvil el cual sujetaba Daniel cerca de Manuel "conductor", para así tener dos fuentes de audio en caso de que una fallará. Una vez resueltos estos problemas, se realizó la grabación del trayecto por completo con unos resultados satisfactorios.

### Entrenamiento del conductor:

Para conseguir realizar la grabación de vídeo, consiguiendo el "Thinking out loud", Manuel el cual iba a ser el conductor, empezó el entrenamiento realizando un video simple explicando un poco lo que pensaba en ese momento, sin ser nada relacionado con la conducción o los códigos, solo pensando en alto. El segundo paso fue aprenderse bien los códigos de lo que se debía de decir, y una vez realizado esto, cogiendo un video de youtube se hizo una prueba, a partir de un video de una autoescuela donde pasan por stops todo el rato. Una vez realizada esta segunda prueba y estando más confiados en el "Thinking out loud", ya se empezó a preparar todo para la grabación del video final.

## Gestión del audio:

Parte del proceso de la práctica es conseguir extraer el audio del video que hemos grabado conduciendo. Esto lo vamos a realizar con una función, en nuestro caso Video\_to\_Audio donde recibiendo el video como parámetro vamos a extraer el audio y guardarlo como "gopro.wav". Todo esto lo realizamos usando la librería moviepy en la que importamos editor como mp. Una vez tengamos la librería importada, vemos el código donde lo primero que realizamos es guardar el video en una variable la cual vamos usaremos para extraer el audio de la siguiente manera.

```
myaudio = AudioSegment.from_file(path, "wav")
chunk_length_ms = 26000
chunks = make_chunks(myaudio, chunk_length_ms)
```

Como podemos comprobar en la imagen anterior, extraemos el audio y lo devolvemos en la función. Una vez tenemos esto nos encontramos con otro problema, siendo este el de que cuando vayamos a transformar este audio en texto, el método usado (Google en este caso), no acepta archivos mayores de 10MB para la conversión, con lo que nos encontramos con otro problema el cual hemos resuelto "partiendo" el audio en chunks los cuales tendrán un tamaño determinado para permitir al transformador de Google realizar la traducción. Este "troceo" se ha realizado con la librería de pydb.utils, donde hemos importado make\_chunks lo cual nos permite trocear el audio en trozos del tamaño que hemos determinado, en este caso es de 26 segundos.

```
myaudio = AudioSegment.from_file(path, "wav")
chunk_length_ms = 26000 # pydub calculates in millisec
chunks = make_chunks(myaudio, chunk_length_ms) # Make chunks of one sec
```

Una vez realizado esto, creamos una carpeta donde vamos a poder guardar los chunks de audio, y guardamos todos los chunks creados en esa carpeta, en nuestro caso 28. Los hemos guardado haciendo uso de la función export con la que somos capaces de especificar la carpeta en la que deseamos guardarla, en nuestro caso siendo "audio-chunks". Una vez hemos realizado todos estos pasos, acabamos con el tratamiento de audio.

```
for i, chunk in enumerate(chunks):
    chunk_name = os.path.join(folder_name, f"chunk{i}.wav")
    # print ("exporting", chunk_name)
    chunk.export(chunk_name, format="wav")
print("Chunks exportados correctamente...")
```

## La transcripción:

En esta parte del proceso vamos a realizar la transformación de audio a texto, donde vamos a partir de los chunks de audio de los que hemos hablado en el apartado anterior de gestión del audio, y vamos a ir transformándolo a texto, todo esto lo vamos a realizar dentro de un bucle, en el que vamos a ir cogiendo cada audio, y los vamos a ir usar de source, donde vamos a hacer un record sobre el mismo, lo cual guardaremos en un variables y esta se la pasaremos al recognize\_google, determinando que queremos que el idioma sea el español. Todo esto lo podemos realizar gracias a la librería de speech\_recognition la cual nos permite realizar este tipo de procesamientos. Una vez se ha realizado la llamada a la función de recognize\_google, guardamos lo que nos devuelve esta función para en el siguiente paso poder procesar el texto que se ha traducido.

```
for i, audio_chunk in enumerate(chunks, start=0): # Antes era 1
    #Exportamos el audio y lo guardamos
    #Cogemos el chunks deseado de la carpeta
    chunk_filename = os.path.join(folder_name, f"chunk{i}.wav")
    audio_chunk.export(chunk_filename, format="wav")
    #Reconocemos el chunk
    with sr.AudioFile(chunk_filename) as source:
        audio_listened = r.record(source)
        try:
            text = r.recognize_google(audio_listened, language='es-ES')
            print('chunk', i)
            text = unicode(text)
```

## El procesamiento del texto:

En este apartado vamos a ver todo el proceso realizado para poder tratar el texto de forma de que consigamos el texto tokenizado, decodificado y con los códigos expuestos en el enunciado de la práctica, siendo estos de Girar Izquierda (SW-TL-L) etc. Comenzamos donde acabamos en el anterior apartado, donde habíamos conseguido convertir todos los chunks de audio a texto con el recognize\_google. Una vez hemos realizado la transcripción a texto, empezamos por representar el texto en ASCII, lo cual lo hacemos debido a problemas con las tildes y algunos otros caracteres. Esto lo hemos realizado debido a los errores que nos encontramos a la hora de insertar códigos, los cuales no llevaron a ver que parte del problema era el “formato” del texto. Para usar el unidecode, hemos importado la librería unidecode.

- Una vez realizado esto, tokenizamos el chunk, para separarlo por palabras. Esto es un proceso muy simple ya que solo hay que llamar a función word\_tokenize, contenida en la librería nltk.

```
text = unidecode(text)
tokenizar = nltk.word_tokenize(text) # lista
```

- En nuestro caso, no hemos usado expresiones regulares ya que no es necesario, ya que con el unidecode ya somos capaces de eliminar los caracteres que no deseamos. Estos mismos son los que no encontramos en ASCII. Realizando esto, nos quitamos de hacer expresiones regulares para eliminar esos caracteres que encontremos en la transcripción.

```
text = unidecode(text)
tokenizar = nltk.word_tokenize(text) # lista
```

## Edición del texto:

Una vez hemos usado el unidecode para pasar el texto a ASCII, no necesitamos usar expresiones regulares, y pasamos al punto de edición de texto, donde en nuestro caso no se ha visto necesario realizar ningún tipo de edición de texto debido a que con el recorrido marcado, los stops están tan pegados que lo vimos necesario realizar ningún tipo de edición de video. Lo único que se editó fue el audio, reduciendo el ruido con Sony Vegas con el que conseguimos un resultado muy satisfactorio.

## Inserción de códigos:

Esta es una parte del proceso donde, aunque ya hemos hablado sobre ella, vamos a especificar el método y el razonamiento por el que se decidió realizar la inserción de esta manera. Una vez que hemos pasado el texto transcrito a ASCII, lo tokenizamos y cuando está tokenizado, se encuentra en una lista, donde lista[0] es la primera palabra y lista[1] la segunda. Con esto dicho es muy fácil recorrer la lista entera en busca de las palabras que queremos por eso realizamos un bucle que sea desde 0 hasta la longitud de la lista, y en cada fase comprobamos si la palabra en la que estamos es una de las que queremos para los códigos. Si es así, comprobamos la siguiente y si es correcta y acaba la frase ahí, introducimos el código en la siguiente posición con un lista.insert.

```
tokenizar = nltk.word_tokenize(text) # lista
y = 0
while y < len(tokenizar):
    if tokenizar[y] == 'giro' and tokenizar[y + 1] == 'derecha':
        tokenizar.insert(y + 2, '<SW-TL-R>')
    elif tokenizar[y] == 'giro' and tokenizar[y + 1] == 'izquierda':
        tokenizar.insert(y + 2, '<SW-TL-L>')
    elif tokenizar[y] == 'subo' and tokenizar[y + 1] == 'marcha':
        tokenizar.insert(y + 2, '<GU>')
    elif tokenizar[y] == 'bajo' and tokenizar[y + 1] == 'marcha':
        tokenizar.insert(y + 2, '<GD>')
    elif tokenizar[y] == 'bajo' and tokenizar[y + 1] == 'de' and tokenizar[y + 2] == 'marcha':
        tokenizar.insert(y + 3, '<GD>')
    elif tokenizar[y] == 'intermitente' and tokenizar[y + 1] == 'izquierda':
        tokenizar.insert(y + 2, '<LB-ON>')
    elif tokenizar[y] == 'intermitente' and tokenizar[y + 1] == 'derecha':
        tokenizar.insert(y + 2, '<RB-ON>')
    elif tokenizar[y] == 'piso' and tokenizar[y + 1] == 'embrague':
        tokenizar.insert(y + 2, '<G-ON>')
    elif tokenizar[y] == 'suelto' and tokenizar[y + 1] == 'embrague':
        tokenizar.insert(y + 2, '<G-OFF>')
    elif tokenizar[y] == 'piso' and tokenizar[y + 1] == 'acelerador':
        tokenizar.insert(y + 2, '<T-ON>')
    elif tokenizar[y] == 'suelto' and tokenizar[y + 1] == 'acelerador':
        tokenizar.insert(y + 2, '<T-OFF>')
    elif tokenizar[y] == 'piso' and tokenizar[y + 1] == 'freno':
        tokenizar.insert(y + 2, '<B-ON>')
    elif tokenizar[y] == 'piso' and tokenizar[y + 1] == 'freno':
```



## Librerías y paquetes usados:

Para poder realizar todo lo que hemos explicado anteriormente hemos usado las siguientes librerías:

1. **moviepy.editor:** Con lo que hemos podido realizar la extracción del audio del audio del video.
2. **De pydub importamos AudioSegment:** Es un wrapper de un AudioSegment.Object el cual nos permite manipular el audio.
3. **De pydubs.utils importamos make\_chunks:** Esta función nos permite partir en trozos el audio que hemos extraído anteriormente
4. **Speech\_recognition:** Esta librería la importamos para mantener el reconocer en una variable con la que luego cuando tengamos el audio que queramos transcribir llamaremos a la variable con recognize\_google.
5. **Importamos os:** Importamos **os** para poder acceder a funcionalidades del sistema operativo. En nuestro caso lo usamos para crear una carpeta donde almacenaremos los audios troceados.
6. **NLTK:** Esta librería la importamos porque vamos a hacer uso de ella en el proceso de tokenización
7. **Unidecode:** Importamos unidecode para poder pasar todo el texto que transcribimos a ASCII, ya que a lo largo del desarrollo hemos tenido problemas con la transcripción.
8. **SYS:** Lo importamos para usarlo a la hora de ejecutar la ventana principal o "ui".

El resto de librerías importadas, vienen todas de PyQt5 las cuales voy a listar:

- **De PyQt5.QtWidgets:** QApplication, QWidget, QPushButton, QHBoxLayout, QVBoxLayout, QLabel, QSlider, QStyle, QSizePolicy, QFileDialog y QLineEdit.
- **De PyQt5.Multimedia:** QMediaPlayer y QMediaContent
- **De PyQt5.QtMultimediaWidgets:** QVideoWidget
- **De PyQt5.QtGui:** QIcon y QPalette
- **De PyQt5.QtCore:** Qt y QUrl

El único paquete externo que se ha usado ha sido K-Lite\_Codec\_Pack\_1610\_Full el cual nos permite reproducir el video en la ventana principal de la aplicación.

## Pruebas:

En este apartado vamos a ver todas las pruebas que se han realizado para comprobar el correcto funcionamiento del programa, y las pruebas que se han realizado para comprobar que el conductor sabía lo que tenía que decir en cada momento.

La primera prueba realizada fue para comprobaciones de que las transformaciones funcionaban correctamente, usando un simple texto en el que Manuel explicaba un poco quien era y de qué iba el trabajo, esta prueba se puede ver aquí

[https://www.youtube.com/watch?v=Ijxi1Wu4Sxk&ab\\_channel=ManuSalvador](https://www.youtube.com/watch?v=Ijxi1Wu4Sxk&ab_channel=ManuSalvador).

La segunda prueba fue para entrenar al conductor a saber que decir y estar preparado para la grabación. En este video, Manuel pone voz a un video de una autoescuela donde están haciendo stops. Este video también se incluirá en la entrega. Las últimas pruebas todas fueron sobre el código, y se basaron en comprobaciones para ver si las transformaciones se realizan

correctamente en base al vídeo grabado, y si de verdad se guardaban correctamente para poder usarlas en los siguientes pasos. Estas pruebas involucran que cada vez que realicemos una acción con el texto transcrito, imprimamos el resultado para poder comprobar si esta acción se ha realizado correctamente.

Estas pruebas no están incluidas en el código ya que no son necesarias una vez se comprueban que las transformaciones y acciones realizadas son correctas.

## Resultados:

Todos los resultados obtenidos en las distintas pruebas que se han realizado y las cuales hemos comentado han sido bastante satisfactorias ya que, las transcripciones realizadas probando diferentes videos, ha sido todas exitosas dentro de un margen de error ya que el reconocer a veces confunde palabras por el ruido o por la pronunciación tan rápida. Con estas pruebas vemos que el reconocer de google es muy eficaz. Otro factor influyente es la longitud de los chunks ya que, si se prueba el primer video “brr Prueba1” con el transcriptor, puede que de error debido a la longitud de los chunks.

Para finalizar, consideramos que nuestra aplicación de transcripción es una buena opción ya que permite mucha flexibilidad al usuario para poder usar los videos que el desee con la longitud que sea necesaria ya que como nosotros “trozeamos” el audio para ir haciendo las transcripciones a texto.

## Conclusiones:

Tras haber trasteado con todas las librerías para filtrar el video en audio y posteriormente transcribirlo a texto, nos hemos quedado con moviepy.editor y con speech\_recognition.

Tuvimos varios problemas cuando estuvimos grabando el video ya que al principio nuestra cámara GoPro no capturaba correctamente el audio y estuvimos probando capturar el video por un lado y el audio con nuestros dispositivos móviles, finalmente lo solventamos y no se hizo uso del audio grabado aparte por los dispositivos móviles.


Por otro lado, al no conseguir la filtración de ruido mediante código nos hemos decantado por recoger el video a través de Sony Vegas para posteriormente omitir el ruido con Audacity y juntarlo con la imagen de nuevo.

La interfaz ha supuesto un verdadero reto ya que hemos omitido el uso del diseñador de interfaz GUI qt5 y nos hemos embarcado en una aventura eterna para codificar toda la interfaz a mano con la librería Qt5.

Por último, queremos destacar el maravilloso historial en nuestros exploradores tras estar buscando información continua sobre codificación de interfaces Python Qt5 sin IDE de diseño y conseguir unir todos los procesos y transformaciones que se deben realizar en una sola función para simplificar la lectura, ya que, si no, se empezaban usar demasiadas variables y archivos como el paso de información de una ventana a otra.

## Apéndice:

- Manual de instalación:** Para la utilización de la aplicación solo es necesario instalar la aplicación K-Lite\_Codec\_Pack\_1610\_Full, con la que vamos a ser capaces de poder reproducir el video que el usuario escoja en la interfaz.  
 A parte de esto, se van a deber tener instaladas todas las librerías expuestas en el punto de librerías y paquetes usados, los cuales son los siguientes (todos los comandos de instalación van a ser con pip):
  1. moviepy: `pip install moviepy`
  2. nltk: <https://www.nltk.org/install.html> en el siguiente enlace encontrarán todos los métodos para poder instalar NLTK.
  3. pydub: `pip install pydub`
  4. speech\_recognition: `pip install SpeechRecognition`
  5. PyQt5: `pip install PyQt5`
  6. pickle: `pip install pickle5`
- Manual de Usuario:** Para poder usar la aplicación no necesita ningún tipo de conocimiento, solo con abrir la carpeta donde se encuentre el código y tener Python instalado con las librerías y paquetes que se han expuesto anteriormente, cualquier usuario será capaz de usar la aplicación. Lo único que tienen que hacer es abrir la consola de Python en la carpeta donde se encuentre el código y usar el siguiente comando: `python3 pantallita.py`  
 Con ese comando se debería de ejecutar la aplicación y el usuario será capaz de realizar todas las funciones expuestas anteriormente. Una vez abierta la aplicación, el usuario se encontrará una pantalla tal que así:

 Thinking Out Loud - Práctica 1 - Sistemas Inteligentes

— □ ×

Open Video



Selecciona un video para iniciar el proceso, gracias.  
 Mientras no haya video, los Botones Inferiores estarán deshabilitados

GUARDAR CAMBIOS

SUGERIR CÓDIGOS

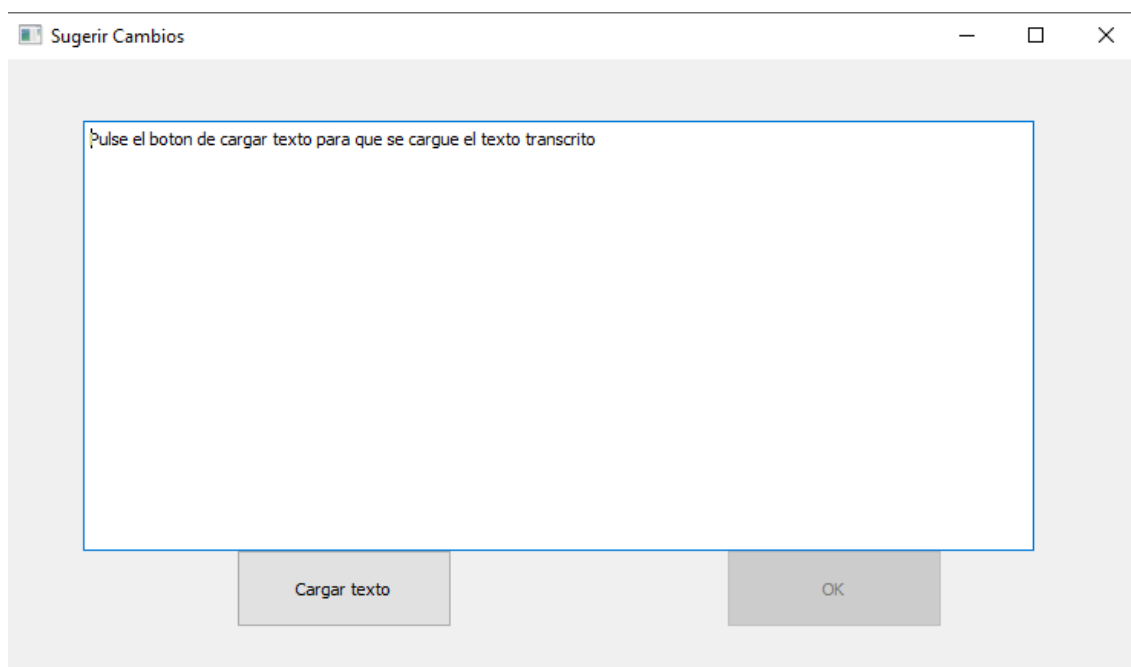
En esta pantalla el primer paso que se debe realizar para poder usar la aplicación correctamente es pulsar el botón de Open Video, donde se abrirá un explorador de archivos y donde el usuario deberá escoger el video que desea transcribir a texto.

Una vez elegido este video, la pantalla se quedará congelada durante un tiempo, más o menos 2 minutos, donde se están realizando las diferentes transformaciones del video a audio y de audio a texto, añadiendo los códigos pertinentes.

Cuando la pantalla se descongele quiere decir que la transformación se ha realizado correctamente, una vez esté descongelada, se podrá dar al botón de play video, donde se empezará a reproducir el video, a su vez imprimimos en la pantalla, todo el texto que se ha transcrito, con sus códigos pertinentes añadidos.

Una vez se haya impreso el texto transcrito, el usuario podrá hacer uso de el botón de Guardar Video y el de sugerir Códigos, con los que podrá Guardar el texto transcrito en formato txt y sugerir códigos editando el mismo.

Con el botón de Sugerir Códigos, se abrirá un pop-up tal que así:



donde aparecerá un “editor de texto” y dos botones. Para cargar el texto transcrito anteriormente tiene que pulsar el botón de Cargar Texto donde cargará el texto en el editor, donde el usuario podrá editar el texto y “proponer códigos”. Una vez realizado esto, para guardar el texto editado, el usuario tendrá que pulsar el botón de OK. Una vez pulsado el botón de OK, los dos botones se bloquearán, ya que ya se han guardado los cambios, y para salir de la pantalla el usuario ha de pulsar la X en la parte superior derecha de la ventana.

Una vez haya cerrado la pestaña de Sugerir Códigos, el usuario podrá guardar los cambios en un archivo txt el cual le permitirá escoger la ubicación de donde quiere guardar la transcripción o podrá seguir visualizando el video.

## Bibliografía:

- <https://doc.qt.io/qt-5/>
- <https://www.nltk.org/install.html>
- <https://pypi.org/project/moviepy/>
- <https://www.learnpyqt.com/tutorials/dialogs/>
- <https://stackoverflow.com/questions/49265280/get-text-from-qtextedit-and-assign-it-to-a-variable/49265583>
- <https://pypi.org/project/SpeechRecognition/>
- <https://forum.qt.io/topic/112315/pyqt-pass-data-between-windows-and-run-script-in-second-window/5>
- <https://pypi.org/project/PyQt5/>
- <https://www.youtube.com/watch?v=2aYeYdeMOLg>
- <https://medium.com/@hektorprofe/widgets-de-pyqt-5-1-ventanas-y-cuadros-de-di%C3%A1logo-937caf847789>
- <https://forum.qt.io/topic/113427/open-new-window-in-pyqt5/11>
- <https://www.learnpyqt.com/tutorials/creating-your-first-window/>
- [https://wiki.qt.io/How\\_to\\_Use\\_QTextEdit](https://wiki.qt.io/How_to_Use_QTextEdit)
- <https://www.learnpyqt.com/tutorials/dialogs/>
- <https://stackoverflow.com/questions/19929626/init-missing-1-required-positional-argument>
- <https://stackoverflow.com/questions/19929626/init-missing-1-required-positional-argument>
- <https://stackoverflow.com/questions/4838890/python-pyqt-popup-window>
- <https://stackoverflow.com/questions/47232741/python-having-multiple-qlineedit-boxes-pyqt4>
- <https://stackoverflow.com/questions/43580566/divide-several-columns-in-a-python-dataframe-where-the-both-the-numerator-and-de>
- <https://stackoverflow.com/questions/48556636/qt-get-numbers-in-specific-area-from-qlineedit>
- <https://stackoverflow.com/questions/33954886/how-to-make-push-button-immediately-disabled>
- <https://community.esri.com/t5/python-questions/python-crashing-process-finished-with-exit-code-1073741819/td-p/592084>
- <https://runestone.academy/runestone/books/published/thinkcspy/Files/AlternativeFileReadingMethods.html>
- <https://www.kite.com/python/answers/how-to-edit-a-file-in-python>
- <https://realpython.com/read-write-files-python/>
- [https://www.w3schools.com/python/python\\_file\\_open.asp](https://www.w3schools.com/python/python_file_open.asp)
- <https://pythonspot.com/pyqt5-file-dialog/>
- <https://www.techwithtim.net/tutorials/pyqt5-tutorial/messageboxes/>
- <https://stackoverflow.com/questions/4838890/python-pyqt-popup-window>
- <https://pythonprogramminglanguage.com/pyqt-textarea/>
- <https://stackoverflow.com/questions/47232741/python-having-multiple-qlineedit-boxes-pyqt4>
- <https://stackoverflow.com/questions/43580566/divide-several-columns-in-a-python-dataframe-where-the-both-the-numerator-and-de>

