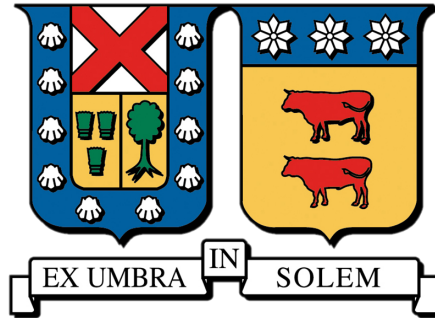


UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE ELECTRÓNICA
VALPARAÍSO - CHILE



**“REPRODUCCIÓN DE VIDEOS INICIADA POR
PALABRAS MENCIONADAS EN EL AUDIO DEL
VIDEO”**

EZEQUIEL IGNACIO FALK ALIAGA

**MEMORIA DE TITULACIÓN PARA OPTAR AL TÍTULO DE INGENIERO
CIVIL ELECTRÓNICO MENCIÓN COMPUTADORES**

PROFESOR GUÍA:

AGUSTÍN GONZÁLEZ

CORREFERENTE:

PABLO CRUZ

ENERO 2025

Agradecimientos

A Dios por darme entendimiento y paz durante toda mi instancia en la universidad, por haber estado conmigo y demostrar su gran poder en las situaciones más difíciles.

A mi papá Marco y mi mamá Maritza, por siempre apoyarme durante todos mis años en la universidad y haberme demostrado tanto amor en todo momento.

A mi hermana Ailyn, que a pesar de estar en diferentes universidades y tener horarios dispares, siempre se acordaba y se preocupaba por mí.

A mi familia que siempre estuvo preocupada y atenta a mi recorrido en la universidad y mi bienestar.

Al profesor Agustín González por permitirme tomar este tema de memoria y apoyarme en mi proyecto de titulación.

A mis amigos de la universidad Dave, David, Emilio, Camila, Darael, Natalia y Oscar, quienes me brindaron un gran apoyo y compañía durante todos mis años universitarios.

*Dedicado a
mi hermana Ailyn,
quien siempre me ha demostrado su cariño
y desde un principio valoró este trabajo.*

Reproducción de Videos Iniciado por Palabras Mencionadas en el Audio del Video

Ezequiel Ignacio Falk Aliaga

Memoria para optar al título de Ingeniero Civil Electrónico,
Mención Computadores.

Profesor Guía: Agustín González

Enero 2025

Resumen

La cantidad de contenido audiovisual disponible ha visto un aumento significativo en los últimos años, incluyendo una amplia cantidad de videos académicos. Muchos profesores han subido clases a plataformas como YouTube, facilitando el acceso a los alumnos en cualquier momento. Sin embargo, buscar un tema específico en estos videos puede ser complicado, ya que suelen carecer de descripciones claras, siendo necesario verlos en su totalidad si se quiere encontrar un tema específico. En respuesta a este problema, este trabajo desarrolla un buscador de palabras para videos en YouTube. Esta solución consiste en una aplicación de escritorio que, a partir de la URL de un video, obtiene una transcripción de este mismo. Luego, por medio de la misma aplicación, busca conceptos en la transcripción y encuentra todos los momentos donde se hizo mención de este y entrega un hipervínculo para comenzar la reproducción del video desde ese momento. El desarrollo se lleva a cabo utilizando el lenguaje de programación de Python y las bibliotecas Yt_dlp, Pydub, Speech_recognition, Custom Tkinter y Pillow. Además de hacer uso de Google Web Speech API para realizar las transcripciones. El resultado es una primera versión funcional que permite obtener transcripciones, buscar conceptos y reproducir el video desde los momentos buscados. Las transcripciones se almacenan localmente para facilitar futuras búsquedas.

Palabras claves: Buscador, Conceptos, Transcripciones, Aplicación de escritorio, URL, Hipervínculo.

Video Playback Triggered by Words Mentioned in the Video's Audio

Ezequiel Ignacio Falk Aliaga

Final Project Report towards the fulfillment of the Ingeniero Civil Electrónico,
Minor in in Computers degree (6-year program).

Advisor: Agustín González

January 2025

Abstract

The amount of available audiovisual content has significantly increased in recent years, including a vast array of academic videos. Many professors have uploaded their lectures to platforms such as YouTube, making it easier for students to access them at any time. However, searching for specific topics within these videos can be challenging, as they often lack clear descriptions, requiring users to watch them in their entirety to find a particular subject. In response to this problem, this project develops a word search tool for YouTube videos. The solution consists of a desktop application that, given a video URL, retrieves its transcription. Then, using the same application, users can search for concepts within the transcription, find all the moments when the term was mentioned, and receive a hyperlink to start playing the video from that point. The development is carried out using the Python programming language and libraries such as Yt_dlp, Pydub, Speech_recognition, Custom Tkinter, and Pillow, in addition to the Google Web Speech API for transcriptions. The result is a functional first version that enables users to generate transcriptions, search for concepts, and play the video from the desired moments. Transcriptions are stored locally to facilitate future searches.

Keywords: Search tool, Concepts, Transcriptions, Desktop application, URL, Hyperlink.

Glosario

- **Biblioteca:** Conjunto de funciones, rutinas y códigos desarrollado previamente que permite desarrollar software de manera más ágil.
- **Extensión de navegador:** Módulo de software que añade características y funciones a un explorador web.
- **Aplicación de escritorio:** Programa de software que se ejecuta en un ordenador, se instala y almacena localmente en el dispositivo del usuario.
- **Hipervínculo:** Enlace unidireccional que permite acceder a un sitio web al hacer click.
- **Pool:** Técnica en la que se crean y administran un conjunto de recursos los cuales se mantienen listos para su uso.
- **URL:** Uniform Resource Locator - Dirección web que identifica la ubicación de un recurso en Internet.
- **GUI:** Graphical User Interface - Interfaz gráfica que permite a los usuarios interactuar con una aplicación o dispositivo.
- **API:** Application Programming Interface - Software que permite que las aplicaciones se comuniquen entre sí, intercambiando datos, funcionalidades y características.
- **PATH:** Variable de entorno que especifica las rutas para buscar los ejecutables necesarios desde la línea de comandos.

Índice general

1. Introducción	1
1.1. Contexto	1
1.2. Descripción del problema	2
1.3. Motivación	2
1.4. Solución Propuesta	3
1.5. Alcance	4
1.6. Objetivos	4
2. Herramientas para el desarrollo	6
2.1. Aplicaciones semejantes	6
2.2. Python	7
2.3. Yt_dlp	7
2.4. Pydub	8
2.5. Speech_recognition	9
2.6. Google Web Speech API	9
2.7. Custom Tkinter	10
2.8. Pillow	10
3. Diseño del Reproductor de Videos Indexado por Palabras	11
3.1. Forma de Re-VIP	11
3.2. Transcriptor	13
3.3. Buscador	16

3.4. Interfaz gráfica	17
4. Desarrollo de los códigos para el Reproductor de Videos Indexado por Palabras	22
4.1. Desarrollo del código de Transcripción	22
4.1.1. Descargar el audio del video	23
4.1.2. Dividir el audio en segmentos de 5 segundos	24
4.1.3. Enviar los fragmentos de audio a la API de reconocimiento de voz y recuperar las transcripciones	27
4.1.4. Escribir las transcripciones en un archivo de texto	29
4.1.5. Eliminar el audio y los segmentos generados	31
4.2. Desarrollo del código Buscador	32
4.3. Desarrollo del código Interfaz gráfica	34
4.3.1. Ventana de transcripciones	35
4.3.2. Ventana de transcripciones guardadas	37
4.3.3. Ventana del buscador	38
5. Resultados	40
6. Conclusión y Trabajo Futuro	45

Índice de figuras

1.1. Diagrama de la Solución	4
3.1. Diagrama de relación	12
3.2. Diseño para la ventana de creación de transcripciones	19
3.3. Diseño para la ventana de transcripciones guardadas	20
3.4. Diseño para la ventana del buscador en transcripciones	20
3.5. Diseño de la navegación entre las ventanas de la aplicación	21
4.1. Archivos multimedia generados	25
4.2. Ventana para realizar transcripciones	35
4.3. Mensaje de advertencia para esperar a que termine el proceso	36
4.4. Mensaje de error por video privado	37
4.5. Ventana para seleccionar transcripciones guardadas	38
4.6. Ventana para las búsquedas	38
4.7. Ventana para las búsquedas con resultados	39
5.1. Ventana de transcripciones en marcha	41
5.2. Ventana de transcripciones una vez terminada la transcripción	41
5.3. Transcripciones guardadas desplegadas	42
5.4. Transcripción desplegada	42
5.5. Resultado de la búsqueda	43
5.6. Video comenzando su reproducción desde el instante asociado al hipervínculo	43

5.7. Fragmento de una transcripción en un archivo de texto	44
--	----

Índice de códigos

3.1. Algoritmo secuencial del transcriptor	15
3.2. Algoritmo secuencial del buscado	17
4.1. Pseudocódigo para la descarga de videos	24
4.2. Pseudocódigo para la división del audio	26
4.3. Pseudocódigo para trabajar con Google Speech Web API	28
4.4. Pseudocódigo para la escritura en el archivo de texto	31
4.5. Pseudocódigo para la eliminación de archivos	32
4.6. Pseudocódigo del buscador	33

Capítulo 1

Introducción

El consumo de contenido audiovisual ha tenido un crecimiento exponencial en la última década. Plataformas para subir y compartir videos han tenido un gran impacto en la sociedad en este sentido, siendo YouTube una de las principales plataformas que se ha consolidado como un repositorio de información educativa, cultural y recreativa. A pesar de la facilidad que disponen los usuarios para acceder a estos videos, sigue siendo un desafío intentar localizar información específica dentro de un video extenso, lo que presenta una limitante para aprovechar al máximo este recurso. Es por esto que se busca desarrollar una herramienta que facilite la búsqueda de conceptos dentro de los videos.

1.1. Contexto

YouTube es una de las principales plataformas para subir contenido multimedia. En el año 2023 se subían 500 horas de contenido por minuto y en enero de 2024 contaba con cerca de 2500 millones de usuarios activos [1]. Debido a la accesibilidad y versatilidad que presenta esta plataforma digital, es que se prefiere para subir videos de distinto tipo, incluso los de carácter académico.

El grabar clases, ayudantías e incluso conferencias para luego subirlas a platafor-

mas digitales ha sido una tendencia que se vio impulsada en los últimos años debido a la pandemia del COVID-19 en el año 2020, momento en el cual muchas instituciones educativas se vieron forzadas a pasar a una enseñanza en línea. Bajo este contexto, los docentes grabaron y subieron sus clases a estas plataformas, permitiendo a los alumnos de ese entonces y futuros acceder en cualquier momento a este material.

1.2. Descripción del problema

A pesar de que actualmente se encuentran disponibles una gran cantidad de videos en YouTube, el buscar un tema en específico en uno de estos puede convertirse en un problema, sobre todo en videos extensos. Esto se vuelve aún más relevante en un contexto académico, en donde estudiantes y profesores necesitan ubicar algún tema en específico dentro de grabaciones de clases, ayudantías o conferencias.

Tomando como referencia a la institución Universidad Técnica Federico Santa María, muchos docentes y ayudantes han subido sus clases a YouTube, teniendo estas a partir del año 2021 una duración de 70 minutos [2], por lo que buscar información específica en estos videos se vuelve un desafío, sobre todo si no se sabe en cual clase se debe buscar.

Algunos videos de clases y ayudantías son subidos a esta plataforma con un nombre que no indica los contenidos que se estudiaron. Tampoco se entrega esta información en la descripción del mismo, por lo que la única manera de saber si el contenido que se busca fue abordado en ese video es viéndolo en su totalidad.

1.3. Motivación

La motivación para encontrar una solución a este problema radica en demostrar que la Universidad Técnica Federico Santa María ha proporcionado a sus alumnos la

capacidad de crear una solución tecnológica que sea de utilidad para los estudiantes de cualquier universidad, instituto o colegio. Esto pone a prueba los conocimientos y habilidades que el autor de esta memoria obtuvo en su carrera para desarrollar un proyecto de programación, enfatizando en la capacidad y potencial que tienen estos alumnos para contribuir con herramientas que faciliten el aprendizaje.

1.4. Solución Propuesta

La solución propuesta para resolver este problema fue el desarrollar un software que permita la reproducción de videos a partir de palabras mencionadas en el audio de este mismo. Por esto, fue bautizado como Reproductor de Videos Indexado por Palabras, o Re-VIP por sus siglas. Su funcionamiento consiste en que, a través de una interfaz gráfica, el usuario ingresa la URL de un video de YouTube y obtiene una transcripción del audio del video como un archivo de texto, el cual es almacenado en el sistema para luego poder seleccionar la transcripción en la que se quiere realizar una búsqueda y escribir el concepto que se está buscando, y con esto, Re-VIP despliega hipervínculos para comenzar la reproducción del video desde cada momento en donde se hizo mención del concepto en el video, evitando así el tener que visualizar el video en su totalidad. En la Figura 1.1 se aprecia un diagrama que plasma la solución descrita.

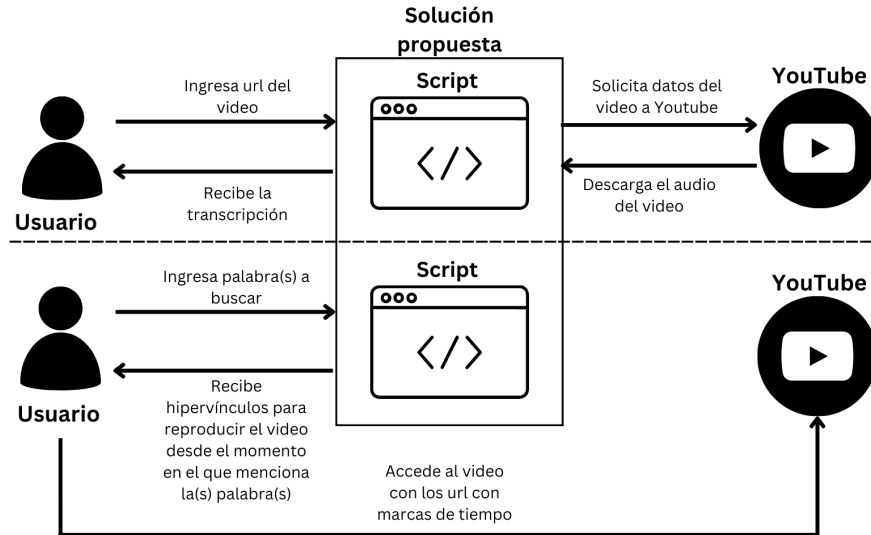


Figura 1.1: Diagrama de la Solución

1.5. Alcance

La solución considera la posibilidad de buscar palabras en el audio de un video en YouTube, por lo que no tiene contemplado el realizar búsquedas en videos los cuales no se encuentren en esta plataforma. Videos en sitios web diferentes o que se encuentren almacenados en el equipo del usuario no están considerados para esta solución. Además, estas búsquedas son realizadas desde la misma aplicación a desarrollar y no desde el dominio de YouTube.

1.6. Objetivos

Como objetivo general se tiene el desarrollar un programa que sea capaz de realizar búsquedas por palabras en el audio de un video de YouTube, para lo cual el usuario debe proporcionar la URL del video a través de una interfaz gráfica y por esta misma se despliegan los hipervínculos resultantes. Para cumplir con este objetivo general, se busca cumplir los siguientes objetivos específicos:

1. Implementar un algoritmo que realice la búsqueda de un concepto en un texto obtenido a partir de una biblioteca que transcribe el audio de un video a texto, proporcionando además las marcas de tiempo.
2. Diseñar e implementar una interfaz gráfica que facilite la entrada de la URL del video y el concepto clave al usuario, y entregue de forma clara todos los momentos donde se haya mencionado el concepto en el video.
3. Proporcionar al usuario un listado de hipervínculos de los instantes de tiempo donde se menciona el concepto buscado en el video, para de esta forma comenzar la reproducción de este en ese instante.

Capítulo 2

Herramientas para el desarrollo

2.1. Aplicaciones semejantes

Actualmente existen aplicaciones que buscan solucionar el problema descrito anteriormente. Algunas funcionan como una página web la cual genera una transcripción y es posible reproducir el video en la misma página, y otras funcionan como una extensión de navegador donde es posible buscar en el video distintos momentos escribiendo la palabra o frase que se está buscando.

No obstante, estas aplicaciones funcionan en base al análisis de los subtítulos incorporados en el video, los cuales pueden ser generados automáticamente, escritos por cualquier persona o por el autor del video [3]. Las primeras dos opciones pueden llegar a presentar inconsistencias respecto al video, dado que puede suceder que la generación automática o la persona encargada de escribir estos lo hagan de forma errónea. Sumado a esto, los videos pueden no contener subtítulos, quedando inutilizables estas aplicaciones.

2.2. Python

Python es un lenguaje de programación de alto nivel, interpretado y con semántica dinámica. Su sintaxis es clara y sencilla, lo que permite un desarrollo más expedito y reducir el tiempo que se debería dedicar para mantenimientos futuros. Sumado a esto, es un lenguaje multiplataforma, por lo que es compatible con la gran mayoría de sistemas operativos presentes en el mercado.

Según Tiobe (The software quality company), es uno de los lenguajes de programación más utilizados por los usuarios a nivel mundial a fecha de noviembre de 2024 [4]. Por otro lado, según el ranking de los lenguajes de programación más utilizados en 2024 realizado por el IEEE Spectrum, Python es el lenguaje más utilizado por el usuario típico de IEEE, también es el más reconocido en base a su tendencia y tan solo por detrás de SQL en un ámbito laboral [5].

Además, Python tiene la posibilidad de importar librerías, las cuales pueden ser estándar, que quiere decir que ya vienen en la instalación básica de Python, o externas, las cuales son desarrolladas por terceros y deben instalarse manualmente. Estas proveen funcionalidades avanzadas o específicas que no están disponibles en una librería estándar, permitiendo no tan solo ahorrar tiempo, sino que implementar funciones que de otra forma no sería posible.

2.3. Yt_dlp

Yt_dlp [6] es una herramienta de código abierto basada en Python que permite descargar tanto videos como audios de videos de distintas plataformas en línea, entre ellas de YouTube. Permite especificar ciertos parámetros de la descarga, como la calidad del audio, el formato, especificar el fragmento que se desea descargar, entre

otras cualidades. Esta herramienta es una bifurcación mejorada de la biblioteca `youtube_dl`, teniendo una mayor velocidad en las descargas, un mantenimiento más activo y la posibilidad de personalizar el formato de la descarga, tal como se mencionó antes.

No obstante, para especificar las opciones de la descarga que se busca realizar, es necesario contar con FFmpeg [7], la cual es una herramienta de código abierto diseñada para procesar audios y videos. Cuenta con múltiples funciones que permiten grabar, convertir, transmitir y manipular archivos de formato multimedia. Por esta razón, muchas herramientas y bibliotecas utilizan FFmpeg como backend para realizar operaciones más avanzadas en cuanto al procesamiento de audio y video.

2.4. Pydub

Pydub [8] es una biblioteca externa que tiene como objetivo facilitar la manipulación de audio. A pesar de que para utilizar todo su potencial es necesario contar con FFmpeg, muchas de sus funciones no requieren de este software, sino que directamente es posible trabajar tan solo con esta biblioteca.

Pydub maneja el audio como un objeto de clase `AudioSegment`, lo que permite modificarlo. Las funciones para esto, que no requieren el uso de FFmpeg, van desde opciones básicas como cortar, unir, dividir, aumentar el volumen o la duración del audio, hasta otras más avanzadas tales como modificar la frecuencia de muestreo, el canal de audio, e incluso la profundidad de bits. Esto permite exportar el audio con el que se está trabajando cambiando todos los parámetros que sean necesarios.

2.5. Speech_recognition

Speech_recognition [9] es una biblioteca de alto nivel externa de Python que permite el reconocimiento de voz, es decir, transcribir audio a texto. Para esto, cuenta con compatibilidad a distintos motores de reconocimiento de voz y APIs, tales como Microsoft Azure Speech, Tensorflow, Google Cloud Speech API, Google Web Speech API, entre otros. Por esto, funciona como un intermediario entre Python y los múltiples servicios de reconocimiento de voz.

Speech_recognition es capaz de convertir voz captada por un micrófono o archivos de audio en formato .wav o .flac conectándose a alguno de los diferentes motores de reconocimiento de voz ya mencionados. A pesar de su simplicidad para ser utilizada y que cuenta con soporte para múltiples lenguajes, presenta la desventaja de ser necesario poseer conexión a internet para ser utilizada, debido a que los servidores de estos servicios se encuentran en internet, y sobre todo que los resultados de la transcripción dependen de la calidad de audio que se quiera convertir a texto, pues si presenta una baja calidad o hay mucho ruido de fondo, el texto resultante puede no ser el esperado.

2.6. Google Web Speech API

Google Web Speech API [10] es una herramienta proporcionada por Google la cual tiene la capacidad de realizar labores de reconocimiento y síntesis de voz de manera gratuita, a diferencia de otros servicios que funcionan a partir de suscripciones. Los datos de audio se procesan en los servidores de Google utilizando sus modelos de inteligencia artificial. Además, este modelo de reconocimiento de voz admite formatos de audio como .wav o .flac.

Cabe mencionar que esta API no puede ser utilizada directamente por un usua-

rio cualquiera. Es necesario el uso de una biblioteca que simule el funcionamiento de un navegador para realizar la solicitud, debido a que esta API es utilizada por páginas como Google Traductor. Está permitido usarla a través de bibliotecas como `speech_recognition` de Python para proyectos personales, limitándose a 60 segundos por solicitud, pero no está recomendada para aplicaciones comerciales.

2.7. Custom Tkinter

Custom Tkinter [12] es una biblioteca externa basada en la biblioteca estándar Tkinter que permite crear interfaces gráficas con un acabado más moderno y estilizado para aplicaciones de escritorio desarrolladas en Python. Esta biblioteca ofrece widgets personalizados con un aspecto más estético que los que ofrece la biblioteca en que está basada. Además, se pueden utilizar en combinación con elementos normales de Tkinter.

Con esta biblioteca es posible conseguir una apariencia consistente y moderna en todas las plataformas de escritorio, pues no requiere de dependencias adicionales para su funcionamiento.

2.8. Pillow

Pillow [13] es una biblioteca de Python que tiene la función de permitir trabajar con imágenes. Ofrece herramientas para la manipulación y procesamiento de imágenes de manera eficiente. Permite abrir, guardar y convertir imágenes entre múltiples formatos, además de permitir cambiar el tamaño de estas mismas.

Gracias a sus cualidades, es posible utilizar esta biblioteca para añadir imágenes a una aplicación de escritorio.

Capítulo 3

Diseño del Reproductor de Videos

Indexado por Palabras

3.1. Forma de Re-VIP

Para el desarrollo de Re-VIP se han identificado dos posibles diseños que cumplen con los requisitos para resolver el problema planteado. Uno de estos es realizar una extensión para navegador, mientras que el otro corresponde a desarrollar el software como una aplicación de escritorio.

Entre estas opciones, la segunda alternativa resulta más atractiva para el autor de esta memoria debido a múltiples ventajas que posee. En primer lugar, permite utilizar el lenguaje de programación Python, junto con la instalación de bibliotecas externas que faciliten la implementación de las funcionalidades requeridas. Además, una aplicación de escritorio ofrece la posibilidad de crear una interfaz gráfica intuitiva que facilite la interacción con el usuario.

Sumado a esto, desarrollar Re-VIP como una aplicación de escritorio permite que el desempeño de esta misma no dependa del navegador que se vaya a utilizar, sino que

únicamente del sistema operativo del equipo.

Por lo tanto, para el desarrollo de esta aplicación de escritorio utilizando Python, se crea un programa en donde cada archivo cumpla con las funciones esenciales para alcanzar los objetivos planteados. Estos archivos se dividen según las siguientes tareas:

1. **Transcriptor:** Encargado de realizar las transcripciones.
2. **Buscador:** Responsable de realizar las búsquedas de palabras en las transcripciones.
3. **Interfaz gráfica (GUI):** Diseñada para conectar al usuario con la funcionalidad que posee la aplicación.

En la Figura 3.1, se enseña un diagrama que muestra cómo se relacionan los archivos del programa a desarrollar entre sí.

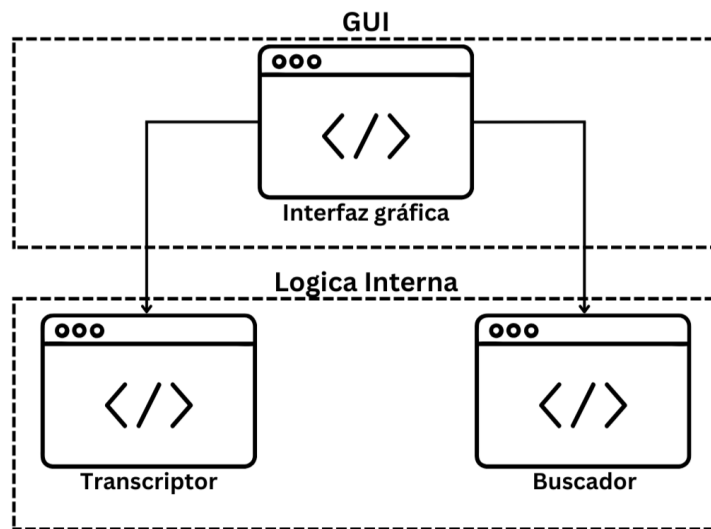


Figura 3.1: Diagrama de relación

De este modo, el usuario puede interactuar con la aplicación desde la interfaz gráfica, evitando que tenga que acceder directamente a los archivos de lógica interna,

siendo estos el buscador y el transcriptor.

Dado que se realizan búsquedas de conceptos en cualquier video en YouTube, Re-VIP no hace uso de los subtítulos de estos, sino que obtiene la transcripción por medio del análisis del audio del video, permitiendo que, aunque este no cuente con ninguna opción de subtítulos, pueda generar una transcripción.

3.2. Transcriptor

La primera etapa del desarrollo consiste en la creación de un código que permita generar una transcripción de un video a partir de una URL de YouTube y un parámetro indicando el idioma del audio del video, ambos entregados por el usuario.

Para procesar el audio del video es necesario, en primer lugar, obtener este. La opción más adecuada para esto es descargarlo, pues de esta forma se puede tener control total sobre el archivo multimedia. En este caso, se emplea el uso de la biblioteca externa `Yt_dlp`, que ofrece parámetros capaces de personalizar el formato de la descarga según los requisitos de la aplicación.

El audio descargado del video es procesado por una aplicación de reconocimiento de voz para obtener su transcripción. La biblioteca `Speech_recognition` funciona como un intermediario entre la aplicación y múltiples motores de reconocimiento de voz, de los cuales se ha determinado que la mejor alternativa es Google Web Speech API debido a sus características y gratuidad. Sin embargo, esta API presenta la limitación de transcribir únicamente un máximo 60 segundos de audio por solicitud. Por lo tanto, es necesario dividir el audio en segmentos más pequeños antes de enviarlos a Google Web Speech API, para de esta forma asegurar que todo el contenido sea procesado correctamente.

La opción más adecuada para manejar el audio en secciones de menos de 60 segundos, minimizando las alteraciones de su calidad, es segmentarlo en fragmentos de 5 segundos cada uno, pues de esta manera se asegura una mayor precisión al momento de establecer el punto inicial en los hipervínculos para comenzar la reproducción del video en donde se encuentra la palabra buscada. Para esto, se hace uso de `Pydub`, diseñada para manipular archivos multimedia. Esta permite dividir el audio original en fragmentos más pequeños de manera eficiente.

Con los fragmentos de 5 segundos ya generados, es posible realizar la conexión con Google Web Speech API para obtener la transcripción. Para garantizar que este proceso se lleve a cabo de forma eficiente, se debe evitar que el programa colapse la API con un exceso de solicitudes simultáneas, pero al mismo tiempo, no es posible enviar una única solicitud a la vez debido a lo tardado que resultaría obtener una transcripción completa.

Por esta razón, se opta por trabajar con hilos. Esto permite enviar una cantidad controlada de solicitudes en paralelo, logrando un equilibrio entre eficiencia y estabilidad en la comunicación con la API. El manejo de hilos se lleva a cabo utilizando el módulo `concurrent.futures`, que contiene la subclase `ThreadPoolExecutor`, que facilita la ejecución asincrónica de funciones a través de un grupo de hilos [14].

Otro parámetro que se debe ingresar para el uso de la herramienta de reconocimiento de voz de Google es el idioma en que se encuentra el archivo multimedia. La API no detecta de forma automática el idioma hablado en el archivo de audio, por lo que este se debe ingresar manualmente. Si esta información no es proporcionada, Google Web Speech API asume por defecto que el idioma es inglés.

La API de reconocimiento de voz retorna la transcripción de cada fragmento de audio, los cuales son almacenados en una lista para posteriormente ser compilados de

forma ordenada en un documento. Entre las posibles opciones para generar un documento, la mejor alternativa es utilizar un archivo de texto (con extensión .txt). Esta elección no solo simplifica y agiliza el proceso de escritura, sino que también permite al usuario acceder fácilmente a este archivo y realizar modificaciones según considere necesarias, ya sea agregar una nota para luego realizar búsquedas futuras más expeditas o incluso corregir errores en la transcripción generada por Google Web Speech.

Además, los archivos de texto tienen un tamaño reducido, generalmente de unos pocos kilobytes, lo que disminuye el uso de recursos del equipo para almacenar transcripciones.

Una vez que la transcripción completa se ha guardado en un archivo de texto, es necesario eliminar tanto el audio descargado como los fragmentos de 5 segundos generados a partir de este. Esto es fundamental para liberar espacio en el equipo, ya que dichos archivos ya no son útiles para la aplicación. Para esto, se implementa el uso del módulo `os` de Python, que provee de funcionalidades que dependen del sistema operativo del equipo [15].

El Código 3.1 ilustra el algoritmo secuencial que describe de manera resumida el proceso general para implementar el script encargado de realizar la transcripción.

```
Inicio
1. Ingresar URL de un video en YouTube y el idioma.
2. Descargar el audio del video.
3. Dividir el audio en fragmentos de 5 segundos.
4. Generar hilos.
5. Cada hilo:
6.     Envía un fragmento de audio a la API.
7.     Recibe la transcripción.
8.     Almacena la transcripción junto al tiempo correspondiente a ese fragmento
        del audio en una lista.
9. Ordenar la lista con transcripciones en base al tiempo de forma ascendente.
10. Crear un archivo de texto.
```

```
11. Almacenar cada transcripción en el archivo.  
12. Eliminar el audio y los segmentos creados.  
Fin
```

Código 3.1: Algoritmo secuencial del transcriptor

3.3. Buscador

Con las transcripciones generadas, el objetivo es facilitar al usuario la búsqueda de información dentro del archivo de texto, evitando que tenga que buscar manualmente. Para ello, se implementa un algoritmo de búsqueda que no solo encuentre coincidencias, sino que también permita reproducir el video desde el momento en que se menciona la frase o palabra buscada.

Esta sección de Re-VIP representa el mecanismo principal para que el usuario localice los conceptos que está buscando en cualquier transcripción generada.

En primer lugar, el archivo debe recibir como parámetro de entrada, por parte del usuario, el archivo de texto en el cual desea realizar la búsqueda y la palabra o frase (concepto) que quiere encontrar. Con esta información, el algoritmo abre el archivo utilizando el módulo `os` y, mediante el método `find` de Python [16], identifica las coincidencias entre el concepto ingresado y el contenido del archivo. Este método permite localizar rápidamente las posiciones de las palabras o frases dentro del texto al buscar línea por línea.

A partir de las coincidencias encontradas, el algoritmo genera hipervínculos para reproducir el video desde el instante en que se menciona el término buscado. Estos enlaces, junto a la frase que contiene la palabra, se retornan al usuario, optimizando así el tiempo requerido para localizar momentos específicos dentro del video.

El Código 3.2 corresponde a un algoritmo secuencial que resume de manera clara y estructurada el proceso general del código encargado de realizar la búsqueda dentro de las transcripciones.

```
Inicio
1. Ingresar el archivo de texto y el concepto a buscar.
2. Abrir el archivo de texto.
3. Por cada línea del texto:
6.     Buscar coincidencias con el concepto dado.
7.     Si existe una coincidencia:
8.         Almacena la frase.
9.         Genera un hipervínculo.
10.        Guarda el hipervínculo en una lista.
11. Retorna la lista con los hipervínculos.
Fin
```

Código 3.2: Algoritmo secuencial del buscado

3.4. Interfaz gráfica

Las secciones descritas anteriormente corresponden al backend de la aplicación, lo que implica que el usuario no necesita interactuar directamente con estas partes del código. Para facilitar el manejo de esta aplicación, es necesario contar con un frontend, que en este caso consiste en una interfaz gráfica que permita al usuario acceder de manera sencilla a todas las funcionalidades que ofrece Re-VIP.

La interfaz gráfica debe proporcionar acceso rápido y claro a todas las funciones de la aplicación, por lo que el diseño debe facilitar la navegación entre las diferentes ventanas, dejando en claro qué sección se está utilizando y cuáles son las opciones disponibles en cada vista. Con esto, se definen las siguientes cualidades para la GUI:

- **Transcribir un video**

La función principal de Re-VIP permite al usuario obtener una transcripción del

audio de un video ingresando el enlace y el idioma correspondiente. Para mantener esto simple, esta funcionalidad debe contar con una entrada de texto para el enlace del video, un selector para el idioma, y un botón que inicie el proceso de transcripción. Además, es importante indicar al usuario sobre el estado del proceso, mostrando claramente cuando la transcripción está en curso y cuando ha finalizado.

- **Transcripciones guardadas**

La aplicación permite al usuario visualizar las transcripciones que a realizado anteriormente por medio de una lista desplegable organizada por los títulos que poseen estas mismas y seleccionar alguna de estas.

- **Visualizar la transcripción seleccionada**

La transcripción seleccionada se despliega en la pantalla para que el usuario pueda revisar su contenido y confirmar si es en esta en donde quiere realizar la búsqueda.

- **Realizar la búsqueda de palabras**

Con la transcripción seleccionada, el usuario puede ingresar una palabra o frase y con la aplicación revisar si estas fueron mencionadas en el audio del video, utilizando la transcripción como referencia. Para mantener esta funcionalidad simple y fácil de usar, la GUI debe incluir una entrada de texto para escribir el concepto deseado y un botón que inicie la búsqueda.

- **Seleccionar el hipervínculo del momento buscado**

La aplicación muestra en una sección de la ventana todas las ocasiones en que el término buscado aparece en el video. Junto a cada mención, se incluye un hipervínculo que permite reproducir el video desde el instante en que se menciona ese concepto.

Con estas cinco cualidades definidas, es posible agrupar algunas funciones relacionadas en una misma ventana para optimizar la navegación dentro de la aplicación.

Esto permite que algunas ventanas cumplan con más de una función, siempre manteniendo la relación entre ellas. La organización de las ventanas queda de la siguiente manera:

■ Ventana 1: Creación de Transcripciones

Esta ventana está diseñada para la creación de nuevas transcripciones. Incluye una entrada de texto donde el usuario puede ingresar el enlace del video y otro para el idioma, además de un botón para iniciar la transcripción y otro que permite acceder a la ventana que posee las transcripciones guardadas.

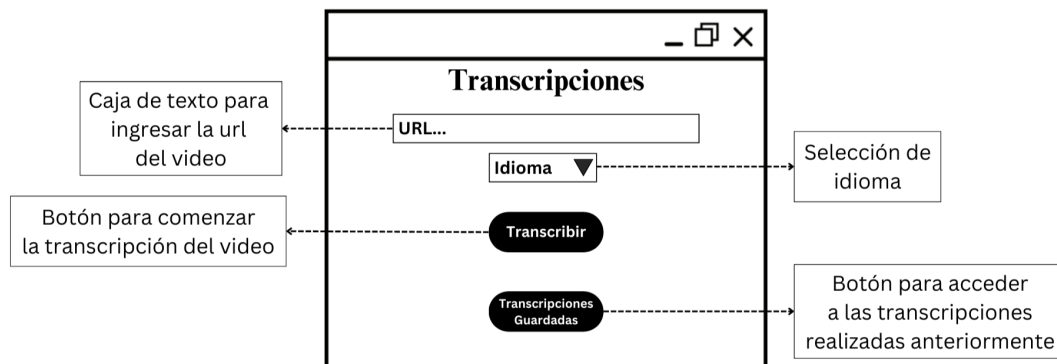


Figura 3.2: Diseño para la ventana de creación de transcripciones

■ Ventana 2: Transcripciones guardadas

Presenta un listado con las transcripciones realizadas anteriormente. El usuario puede seleccionar alguna de estas, lo que despliega su contenido para que el usuario pueda revisarla. Además, la ventana presenta dos botones, uno para volver a la ventana de transcripciones y otro para acceder a la ventana donde se realizan las búsquedas.

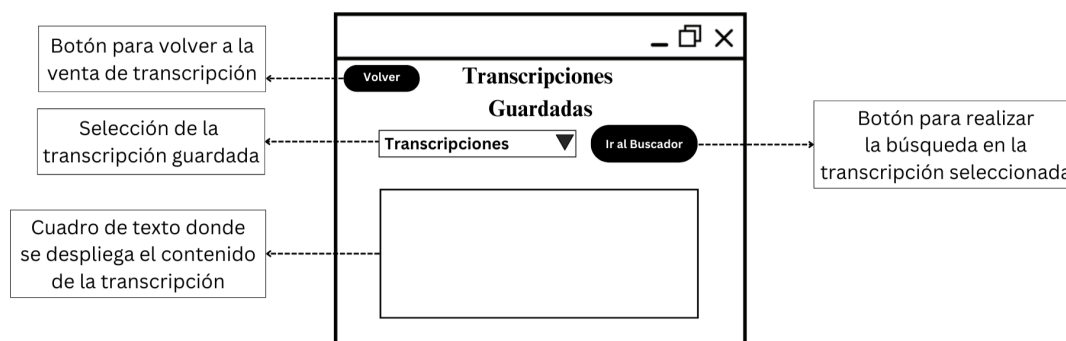


Figura 3.3: Diseño para la ventana de transcripciones guardadas

■ Ventana 3: Buscador en las transcripciones

Con la transcripción seleccionada, esta ventana contiene una entrada de texto donde el usuario puede ingresar las palabras que desea buscar y un botón para iniciar la búsqueda. Los resultados se muestran en un cuadro de texto junto a los hipervínculos. Además, contiene un botón para volver a la ventana de transcripciones guardadas.

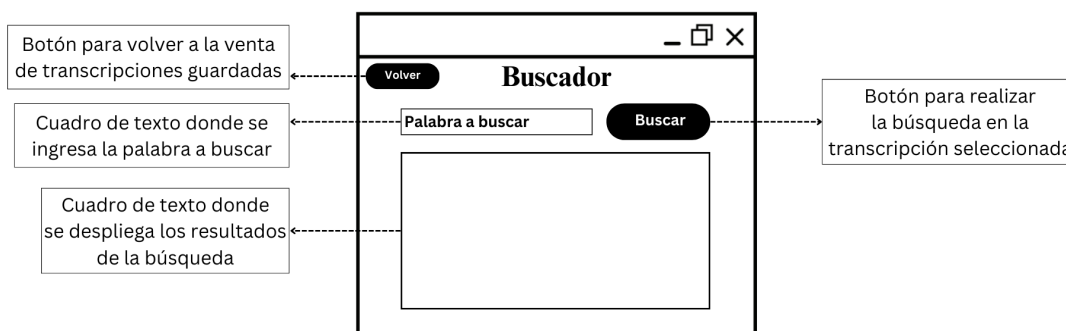


Figura 3.4: Diseño para la ventana del buscador en transcripciones

De esta forma, cada ventana queda destinada a una función específica, asegurando que el uso de cada característica de Re-VIP sea clara y precisa.

Teniendo las pantallas establecidas, la navegación completa de la aplicación junto a sus funciones queda ilustrada en la Figura 3.5.

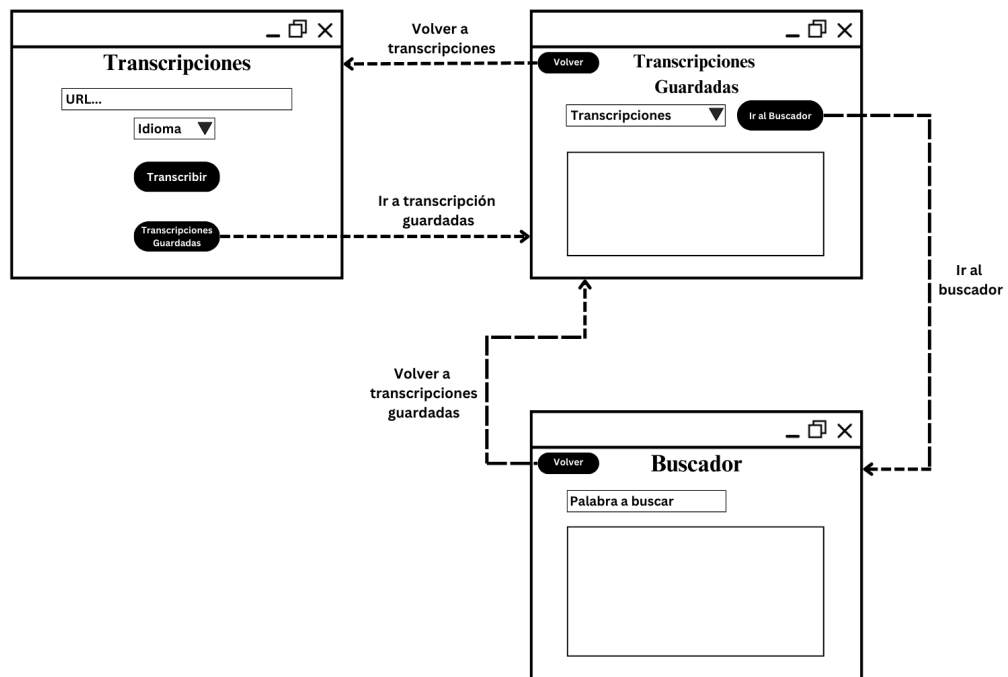


Figura 3.5: Diseño de la navegación entre las ventanas de la aplicación

Para el desarrollo de esta interfaz gráfica, se hace uso de la biblioteca externa Custom Tkinter, pues ofrece la misma facilidad para crear una GUI que Tkinter, a la par de un acabado más moderno.

Capítulo 4

Desarrollo de los códigos para el Reproductor de Videos Indexado por Palabras

El desarrollo de los códigos se divide en dos etapas, siendo la primera la del backend, que incluye la implementación del código para la transcripción de audio y el sistema de búsqueda dentro de las transcripciones. En la segunda etapa se debe desarrollar el frontend, correspondiente a la interfaz gráfica, pues con la ayuda de esta se pueden realizar las conexiones entre cada código.

4.1. Desarrollo del código de Transcripción

Para desarrollar correctamente el código encargado de realizar las transcripciones, se hace uso de la técnica de dividir y conquistar, dividiendo el trabajo en secciones más pequeñas en donde cada una busca resolver un problema específico. A partir del diseño planteado en la sección anterior, se identifican las siguientes tareas que deben resolverse durante la implementación:

- Descargar el audio del video
- Dividir el audio en fragmentos de 5 segundos
- Enviar los fragmentos a la API de reconocimiento de voz y recuperar las transcripciones
- Escribir las transcripciones en un archivo de texto
- Eliminar el audio y los segmentos generados

4.1.1. Descargar el audio del video

La descarga del audio se lleva a cabo utilizando la biblioteca externa `Yt_dlp`, para la cual se deben configurar previamente los siguientes parámetros:

- `Format`: Formato de audio a descargar.
- `Postprocessors`: Acciones que se realizan después de descargar el archivo.
- `Outtmpl`: nombre del archivo resultante.
- `ffmpeg_location`: ruta del ejecutable de FFmpeg para procesar el audio.

Para la descarga, se establece el nombre del archivo resultante como “audio”, la extensión de este `.mp3` y la tasa de bits del audio en 192, esto permite que la calidad del audio descargado sea buena y no posea un gran tamaño, aligerando el procesamiento futuro que se realiza al audio. Adicionalmente, se establece la ubicación de la carpeta bin de FFmpeg en el PATH del sistema operativo. Esto asegura que la aplicación funcione correctamente al proporcionar redundancia en la ubicación de este archivo.

Una vez definidos estos parámetros, la biblioteca comienza la descarga del audio del video según la configuración especificada. No obstante, si el video en YouTube es

privado, la descarga no se llevará a cabo y retornará un error indicando lo sucedido.

En el Código 4.1 se presenta el pseudocódigo correspondiente a este proceso.

```
Parámetros:
- título:   Almacena el título del video como un string.

Inicio
1. Establecer los parámetros de la descarga.
2. Intentar:
3.   Descargar audio del video.
4.   Recuperar título del video (por defecto "Sin título").
5. Excepción de video privado:
6.   Termina ejecución del código.
7. Eliminar los caracteres especiales del título del video.
Fin
```

Código 4.1: Pseudocódigo para la descarga de videos

El título del video debe almacenarse para después guardar el archivo de texto con la transcripción bajo este mismo nombre, de esta forma resulta más sencillo de buscar entre las transcripciones generadas. Sin embargo, el título no puede contener caracteres especiales o emojis, por ende es necesario eliminar estos del título utilizando el módulo `re` de Python, el cual permite buscar algún carácter en un string y reemplazarlo por otro, en este caso siendo un carácter de espacio, pues este permite mantener el título lo más limpio y legible posible.

4.1.2. Dividir el audio en segmentos de 5 segundos

La división del audio en fragmentos de 5 segundos se realiza utilizando la biblioteca `Pydub`, que no tan solo permite segmentar el audio, sino también configurar su formato, todo esto por medio de la función `AudioSegment`.

Entre las configuraciones aplicadas al audio, se tiene el establecer la salida del audio como monofónico. De esta manera, solo existe una pista de audio, lo que permite que la API pueda interpretar de mejor manera el archivo multimedia y entregar un mejor resultado.

Además, se establece una tasa de muestreo de 16kHz, pues esta es la mínima frecuencia que garantiza una calidad suficiente para reconocer la voz humana en aplicaciones de reconocimiento de voz. Al utilizar esta frecuencia, se reduce la cantidad de datos en el audio que se deben almacenar y procesar, optimizando así el rendimiento del sistema [17].

Dado que es necesario generar múltiples segmentos de audio (figura 4.1) y enviar cada uno a la API para obtener la transcripción, conforme al diseño planteado anteriormente, se van a generar hilos en donde cada uno trabaje con un segmento de audio y que haga uso de la API para obtener la transcripción.

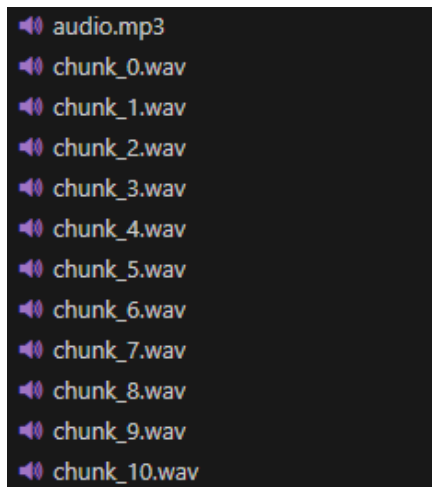


Figura 4.1: Archivos multimedia generados

Los hilos se gestionan a través de un pool, en el cual se debe indicar la cantidad máxima de hilos que se pueden llegar a generar. En la práctica, se ha determinado que

la cantidad óptima de hilos para trabajar son un total de 30. Esta cantidad evita que la API se sature por recibir tantas solicitudes y asegura que el proceso de transcripción no sea lento.

En el Código 4.2 se presenta el pseudocódigo que describe el funcionamiento del código para esta sección.

```
Parámetros:

- language:                                Idioma en el que se encuentra el audio del video.
- recognizer:                              Inicializa el reanador de voz.
- transcriptions_with_timestamps:          Lista para almacenar las transcripciones con las
    marcas de tiempo.
- transcriptions_lock:                    Parametro asociado a una función para manejar la
    sincronización entre hilos.
-chunk:                                   Nombre base del fragmento de audio de 5 segundos.
- chunk_files:                            Lista que guarda los nombres de los archivos de
    fragmentos del audio.
- duration_ms:                            Duración en milisegundos del audio.
- chunk_size_ms:                          Tamaño de cada fragmento de audio establecido en
    5000ms.
- futures:                                Lista que almacena el estado de los hilos.
- i:                                       Indice del ciclo for que además indica en que
    segmento del audio va a ser transcrito.

Inicio
1. Establecer el audio en una tasa de muestreo de 16kHz y el canal de audio en modo
    monofónico.
2. Con el pool de hilos, con un máximo de 30:
3.     Por cada i entre 0 a duration_ms con un salto de chunk_size_ms:
4.         Crear un chunk de 5 segundos bajo el nombre chunk_"i".
5.         Llamar a la función encargada de la conexión con la API de Google.
6.     Por cada estado de los hilos:
7.         Intentar:
8.             Verificar el estado.
9.         Si hay excepciones:
10.            Informar sobre el estado de la excepciones.
Fin
```

Código 4.2: Pseudocódigo para la división del audio

Todos los parámetros mencionados, a excepción de “futures”, se pasan como argumento a la función encargada de conectar con Google Web Speech API, pues son necesarios para que los fragmentos de 5 segundos transcritos se recuperen de manera ordenada y precisa, asegurando la coherencia en los resultados obtenidos.

4.1.3. Enviar los fragmentos de audio a la API de reconocimiento de voz y recuperar las transcripciones

El proceso de conexión a la API para obtener y recuperar una transcripción ha sido definido en una función propia, permitiendo que cada hilo acceda a esta de forma independiente.

Al inicio de esta función, se crea un archivo dinámico con la extensión .wav y se exporta a este archivo el fragmento de audio de 5 segundos bajo el formato de la misma extensión del archivo. Este cambio de formato es necesario porque la herramienta de reconocimiento de voz de Google solo admite ciertos tipos de archivos, entre ellos el formato .wav.

Dado que la cantidad de segmentos de audios de 5 segundos generados depende directamente de la duración del video escogido, no es posible conocerla de antemano. Por ello, los nombres de los archivos de audio creados son almacenados en una lista para facilitar su posterior eliminación.

Por medio de la biblioteca `Speech_recognition`, cada fragmento es enviado a la API de Google junto a la variable que almacena el idioma del video.

Una vez generada la transcripción, esta es almacenada en una variable la cual, junto a los segundos correspondientes al momento del audio que representa ese frag-

mento (el cual corresponde al índice `i` del ciclo “for” de los hilos), son almacenados en una lista con el formato de un diccionario, teniendo el parámetro “start_time” para los segundos, y “transcript” para la transcripción.

Para evitar conflictos entre los hilos al acceder a la lista compartida, se utiliza el módulo `threading.Lock()` asociado a una variable. Esto asegura que los hilos esperan a que la lista esté libre antes de modificarla.

No obstante, este proceso se realiza solo si la API de reconocimiento de voz logra generar alguna transcripción del fragmento de audio procesado. En caso de que la calidad en el archivo sea deficiente o no se detecten palabras audibles, la API retorna un error indicando que no pudo reconocer el audio. Para manejar esta situación y evitar que la aplicación se detenga, todo el proceso se ejecuta dentro de un bloque “try”. Si ocurre este error, es capturado y se indica en la transcripción, permitiendo que la aplicación siga transcribiendo los demás fragmentos de audio.

En el Código 4.3 se presenta el pseudocódigo para obtener la transcripción de cada fragmento de audio.

Parámetros:

- `chunk`: Segmento de 5 segundos extraídos del audio original.
- `start_time`: Indicador de a que fragmento del audio corresponde el chunk actual.
- `chunk_size_ms`: Tamaño de cada fragmento de audio establecido en 5000ms.
- `chunk_files`: Lista que guarda los nombres de los archivos de fragmentos del audio.
- `recognizer`: Inicializa el reconocedor de voz.
- `language`: Idioma en el que se encuentra el audio.
- `transcriptions_lock`: Parámetro asociado a una función para manejar la sincronización entre hilos.
- `transcriptions_with_timestamps`: Lista para almacenar las transcripciones con las marcas de tiempo.
- `result`: Parametro donde se almacenan las transcripciones resultantes de la API.

```

Inicio
1. Exportar fragmento de audio a un archivo .wav.
2. Almacenar el nombre del archivo en una lista.
3. Intentar:
4.     Obtener la transcripción.
5.     Si hay una transcripción en result:
6.         Recuperar transcripción.
7.         Con transcriptions_lock:
8.             Añadir a transcriptions_with_timestamps la transcripción y
                start_time como diccionario.
9. Si la API no puede recibir peticiones:
10.     Imprimir el error.
11. Si la API no puede comprender el audio:
12.     Reemplaza la transcripción del fragmento de audio por "Ininteligible".
Fin

```

Código 4.3: Pseudocódigo para trabajar con Google Speech Web API

4.1.4. Escribir las transcripciones en un archivo de texto

Las transcripciones, junto a los segundos correspondientes a su posición en el video, se encuentran almacenados en la lista. No obstante, debido a que los hilos con la transcripción trabajan siguiendo un patrón FIFO, el orden de los elementos en la lista no es secuencial. Por esto, es necesario ordenar la lista antes de escribir la transcripción en el archivo de texto.

Para realizar este ordenamiento, se utiliza el método `sorted` de Python, el cual por defecto ordena de forma ascendente los elementos en una lista. Posee un parámetro “key” que permite especificar una función para definir el criterio de ordenamiento [18]. En este caso, al no tener una función definida para esto, se plantea una función lambda que toma como argumento el parámetro “start_time” presente en los diccionarios dentro de la lista. Esto asegura que las transcripciones queden ordenadas siguiendo el mismo orden temporal del audio original del video.

Para mantener un orden en los archivos del programa, todas las transcripciones

se guardan en una carpeta específica llamada “Transcripts”. Si esta carpeta no existe, se crea automáticamente utilizando el módulo `os` de Python, a través del método `makedirs` [15].

Existiendo la carpeta en el equipo, se accede a ella y se crea el archivo de texto con el nombre del título del video del que se realizó la transcripción. Luego, bajo el modo de escritura, se comienza a incluir la transcripción añadiendo primeramente la URL del video, seguido de líneas que contienen el segundo del video y la transcripción de los 5 segundos correspondientes a ese fragmento.

La URL del video se agrega al comienzo del archivo de texto con el objetivo de que el programa pueda construir los hipervínculos de forma rápida y eficiente al tener siempre disponible el enlace.

Por otro lado, el formato establecido, que incluye primero los segundos correspondientes al fragmento del audio del video seguidos de la transcripción, permite una mejor visualización por parte del usuario al revisar el archivo. Además, este formato permite crear los hipervínculos de manera más eficiente, ya que toda la información necesaria para generar estos enlaces se encuentra organizada en una misma línea del archivo de texto.

De esta forma, la transcripción final posee el siguiente aspecto:

```
https://www.youtube.com/watch?v=Q8APVjn9kOo
0.0 clase 1 complemento Max 0
5.0 hoy día vamos a ver bueno el programa del curso
10.0 lo que vamos a ver el resto del semestre y vamos a partir con la parte de lógica
15.0 lógica conectivos lógicos tablas de verdad y algunas propiedades
20.0 qué vamos a ver en el semestre
25.0 fundamentos del lenguaje matemático
30.0 como lógica lo que vamos a ver hoy y conjuntos Además vamos a ver
35.0 números naturales de inducción eso un poco más adelante
```

```
40.0 con la parte de inducción y números naturales vamos a ver inducción Qué es una
    forma de demostración
45.0 notación de sumatoria y progresiones y el teorema del Binomio
50.0 Además vamos a ver trigonometría función exponencial y logaritmo geometría
```

El Código 4.4 ilustra de manera estructurada el pseudocódigo del proceso de escritura de la transcripción en un archivo de texto.

```
Parámetros:

- start_time:      Indicador de a que fragmento del audio corresponde el chunk
actual en el diccionario.
- transcript:      Elemento que contiene la transcripción en el diccionario.
- transcriptions_with_timestamps: Lista para almacenar las transcripciones con las
    marcas de tiempo.
- sorted_transcriptions_with_timestamps: Lista para almacenar las transcripciones
    con las marcas de tiempo de forma ordenada.
- file_transcripts: Nombre de la carpeta donde se almacenan las transcripciones
    (Transcripts).
- txt_transcript:  Dirección del archivo de texto con la transcripción.

Inicio
1. Ordenar la transcripción según start_time.
2. Si la carpeta Transcripts no existe:
3.     Crear carpeta Transcripts.
4. Crear archivo de texto dentro de la carpeta Transcripts.
5. Con el archivo abierto en modo escritura:
6.     Escribir la URL del video.
7.     Por cada elemento en sorted_transcriptions_with_timestamps:
8.         Escribir start_time junto a transcript en el archivo.
Fin
```

Código 4.4: Pseudocódigo para la escritura en el archivo de texto

4.1.5. Eliminar el audio y los segmentos generados

Una vez que la transcripción ha sido generada y guardada en el equipo, es necesario eliminar los archivos multimedia utilizados durante el proceso para liberar espacio en el sistema. Para esto se utiliza el método `remove` del módulo `os`.

Cada fragmento de audio posee un nombre único, los cuales fueron almacenados previamente en una lista, por lo que se debe recorrer esta lista y eliminar cada archivo utilizando el método mencionado. Finalmente, se elimina el audio original.

El Código 4.5 presenta el pseudocódigo que describe este proceso.

```
Parámetros:

- chunk_files:      Lista que guarda los nombres de los archivos de fragmentos del
                    audio.
- chunk:            Fragmento de audio de 5 segundos.

Inicio
1. Por cada elemento en chunk_files:
2.     Intentar:
3.         Eliminar chunk.
4.     Si no se tienen los permisos:
5.         Imprimir que archivo no fue posible eliminar.
6. Eliminar archivo de audio original.
Fin
```

Código 4.5: Pseudocódigo para la eliminación de archivos

4.2. Desarrollo del código Buscador

El desarrollo de este código se realiza considerando la estructura de las transcripciones, es decir, cada línea contiene el “tiempo” seguido de la “transcripción”. Este formato facilita tanto la búsqueda de palabras como la generación de hipervínculos correspondientes.

El proceso tiene contemplado que el usuario ha seleccionado una transcripción previamente, por lo que el código solamente debe abrir el archivo correspondiente para iniciar el proceso de búsqueda. Este enfoque permite que el usuario realice múltiples

búsquedas dentro de un mismo archivo sin necesidad de seleccionarlo nuevamente.

Con el archivo de texto abierto en modo de lectura, el programa procesa cada línea para buscar coincidencias con la palabra ingresada por el usuario. Para ello, se utiliza el método `find` de Python, que devuelve un entero mayor a 0 si encuentra coincidencias. Sin embargo, para garantizar que las coincidencias correspondan a palabras completas y no a fragmentos contenidos dentro de otras palabras (como “mente” en “sorprendentemente”), se valida que la palabra buscada esté rodeada por espacios. Además, se aplica el método `lower` a cada línea y a la palabra para evitar problemas de distinción entre mayúsculas y minúsculas [19], asegurando que todas las coincidencias se detecten correctamente.

Cuando se encuentra una coincidencia, se extraen los segundos al inicio de la línea y se genera un hipervínculo utilizando esta marca de tiempo junto a la URL. Cada resultado se guarda en una lista, donde se almacena tanto la transcripción completa como el enlace correspondiente.

Finalmente, se despliega al usuario la transcripción del segmento de audio donde se menciona la palabra buscada, junto al hipervínculo que permite reproducir el video desde el momento en que se menciona esta.

El pseudocódigo 4.6 presente a continuación describe este proceso:

```
Parámetros:
- palabra:      Palabra ingresada por el usuario a buscar en la transcripción.
- url_time:     Lista que almacena la transcripción y el hipervínculo.
- transcription: Transcripción donde se desea realizar la búsqueda.

Inicio
1. Con transcription abierto:
2.     Por cada linea en transcription:
3.         Si find(' ' + palabra + ' ') > 0:
4.             guardar linea en url_time
```

```

5.             guardar URL con marca de tiempo en url_time
6. retornar url_time
Fin

```

Código 4.6: Pseudocódigo del buscador

4.3. Desarrollo del código Interfaz gráfica

Para la implementación de la interfaz gráfica, se ha considerado que todas las ventanas comparten ciertos elementos comunes, como lo son los botones y las entradas de texto. Por esta razón, la construcción de estas ventanas se aborda de manera general, priorizando las funciones que conectan la GUI con los códigos desarrollados anteriormente, ya que estos elementos tienen mayor relevancia funcional.

Las ventanas se crean utilizando el método `CTkFrame` de `CustomTkinter`. Junto a estas, se crean funciones que permiten el cambio entre ellas. Cada elemento presente en las ventanas es creado con su respectivo método. En el cuadro 4.1 se expresan estas funciones de `CustomTkinter`.

Widget	Método
Botón	<code>CTkButton</code>
Etiqueta	<code>CTkLabel</code>
Entrada de texto	<code>CTkEntry</code>
Cuadro Combinado	<code>CTkComboBox</code>
Cuadro de texto	<code>CTkTextbox</code>

Cuadro 4.1: Widgets y sus métodos para ser invocados

Además, a los botones se les añade imágenes utilizando la biblioteca `Pillow` para una mayor comprensión de su funcionamiento y una mejor presentación.

En la GUI no tan solo se deben implementar funciones que conecten con el backend, sino también medidas de seguridad que eviten problemas que puedan surgir debido a interacciones incorrectas por parte del usuario. Estas medidas incluyen validaciones de entrada, gestión de errores y manejo de eventos inesperados.

4.3.1. Ventana de transcripciones

Para que la interfaz gráfica pueda invocar al código encargado de las transcripciones, se debe proporcionar los parámetros necesarios. Para esto, por medio de un `entry` se le indica al usuario que ingrese la URL del video y con un `ComboBox` que seleccione el idioma en el que se encuentra el video, tal como se muestra en la Figura 4.2.



Figura 4.2: Ventana para realizar transcripciones

La función de transcripción se asocia a un botón que, al ser presionado, inicia un hilo encargado de ejecutar el proceso en el backend. Los argumentos necesarios (la URL y el idioma) se obtienen utilizando el método `get`, lo que asegura que los datos ingresados por el usuario se pasen correctamente a la función.

Mientras se ejecuta la transcripción, la interfaz muestra una animación que indica que el proceso está en curso, proporcionando retroalimentación visual al usuario. Una vez finalizada la transcripción, se notifica al usuario mediante un mensaje en la misma ventana que el proceso se completó con éxito.

Además, es necesario implementar una medida de seguridad en caso de que el usuario intente iniciar una nueva transcripción mientras se está llevando a cabo otra. Para esto, se hace uso de una flag que cambia de estado cuando se inicia una transcripción, por lo que al presionar el botón mientras se está llevando a cabo este proceso, se despliega una ventana emergente que notifica al usuario que debe esperar a que finalice el proceso actual antes de iniciar uno nuevo (Figura 4.3). Una vez termina de ejecutarse la aplicación, la flag vuelve a cambiar de estado, permitiendo iniciar un nuevo proceso.



Figura 4.3: Mensaje de advertencia para esperar a que termine el proceso

Además, si el video es privado y por ende no es posible acceder a él, una ventana emergente indica esto y el proceso de transcripción termina (figura 4.4).

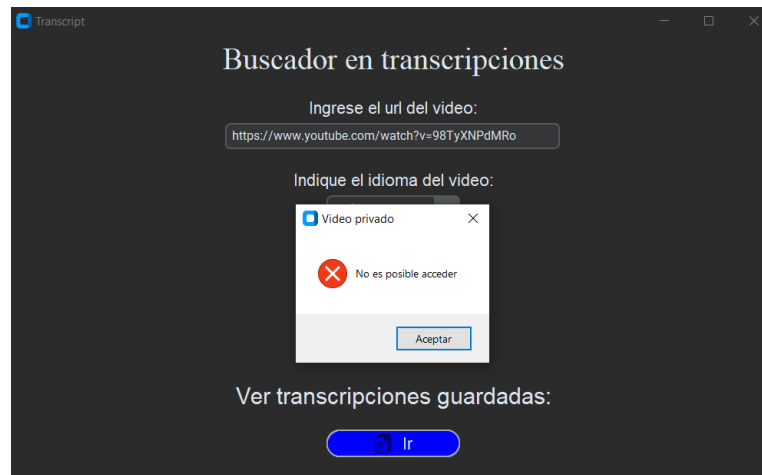


Figura 4.4: Mensaje de error por video privado

4.3.2. Ventana de transcripciones guardadas

En esta ventana, el usuario puede seleccionar una transcripción previamente generada mediante un cuadro combinado. Este `ComboBox` tiene asociada una función que, al seleccionar una opción, conecta con un cuadro de texto (`TextBox`) para desplegar el contenido de la transcripción seleccionada.

Para evitar cambios accidentales en la transcripción, el `TextBox` se establece en modo "deshabilitado", permitiendo únicamente la visualización del contenido. Esto asegura que las transcripciones originales permanezcan intactas y protegidas contra modificaciones no intencionadas.



Figura 4.5: Ventana para seleccionar transcripciones guardadas

4.3.3. Ventana del buscador

El código del backend encargado de realizar la búsqueda debe recibir como parámetros de entrada la palabra o frase a buscar y la transcripción en donde se desea realizar la búsqueda. El primer argumento es ingresado por el usuario a través de una entrada de texto, mientras que el segundo es seleccionado previamente por el usuario. Ambos valores se recuperan por medio del método `get`.



Figura 4.6: Ventana para las búsquedas

Una función asociada al botón de búsqueda en esta pantalla se encarga de llamar al método correspondiente en el código del backend, proporcionando los parámetros requeridos. Una vez que el código retorna la lista de resultados, cada elemento de la lista se inserta en el cuadro de texto, mostrando el fragmento de la transcripción que contiene la coincidencia junto con un hipervínculo que permite reproducir el video desde el instante de tiempo correspondiente.



Figura 4.7: Ventana para las búsquedas con resultados

Capítulo 5

Resultados

Para observar los resultados de Re-VIP, se simula el procedimiento que realizaría un usuario para obtener una transcripción de un video y luego llevar a cabo una búsqueda dentro de esta.

En primer lugar, se rellenan los parámetros con la información necesaria para comenzar con la transcripción, siendo estas la URL y el idioma del video. Al presionar el botón de transcribir, da inicio a este proceso en el backend, indicándole al usuario por medio de la animación establecida que este se está llevando a cabo. Esto es posible de apreciar en la Figura 5.1.



Figura 5.1: Ventana de transcripciones en marcha

Cuando el proceso de transcripción finaliza, se muestra una etiqueta en la pantalla indicando que la transcripción ha sido completada, como se ilustra en la figura 5.2.



Figura 5.2: Ventana de transcripciones una vez terminada la transcripción

Al acceder a la ventana donde se visualizan las transcripciones que fueron realizadas anteriormente, el usuario tiene la posibilidad de seleccionar cualquier transcripción en la que desee realizar una búsqueda, sin estar limitado a la que acaba de solicitar. Esta flexibilidad permite trabajar con transcripciones almacenadas de forma

eficiente, como se muestra en la figura 5.3.



Figura 5.3: Transcripciones guardadas desplegadas

Al seleccionar una transcripción, su contenido completo se despliega en el cuadro de texto, permitiendo al usuario leerla y revisarla en detalle. Esto se puede apreciar en la figura 5.4.

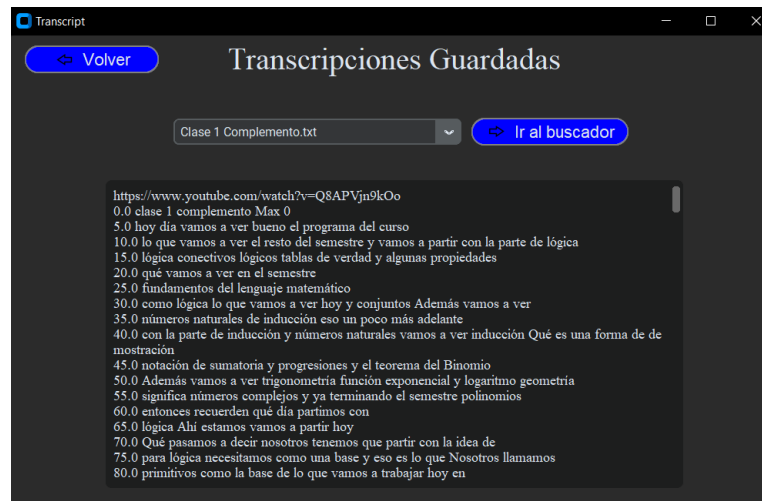


Figura 5.4: Transcripción desplegada

Una vez seleccionada la transcripción, el usuario puede comenzar a realizar búsquedas dentro de ella. Al ingresar una palabra o frase en el campo de búsqueda y presio-

nar el botón correspondiente, se despliegan en el cuadro de texto todos los resultados obtenidos (figura 5.5). Cada resultado incluye un hipervínculo que permite iniciar la reproducción del video desde el instante donde se menciona la palabra buscada (figura 5.6), abriendo una pestaña en el navegador web por cada hipervínculo abierto.



Figura 5.5: Resultado de la búsqueda

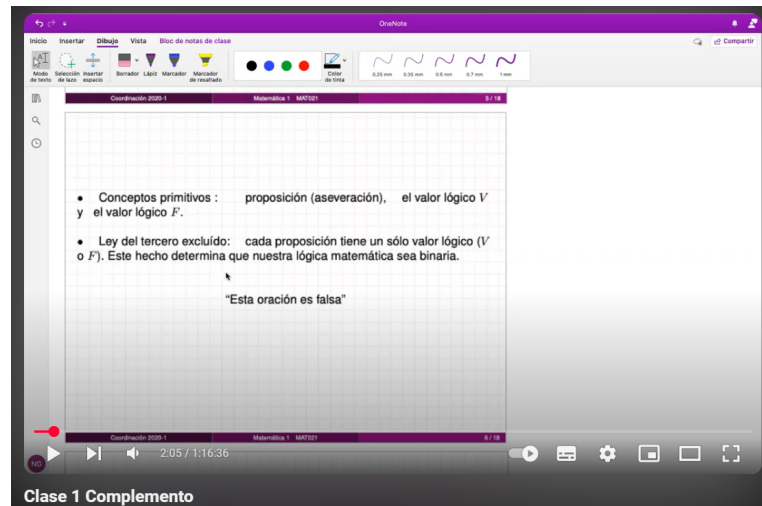
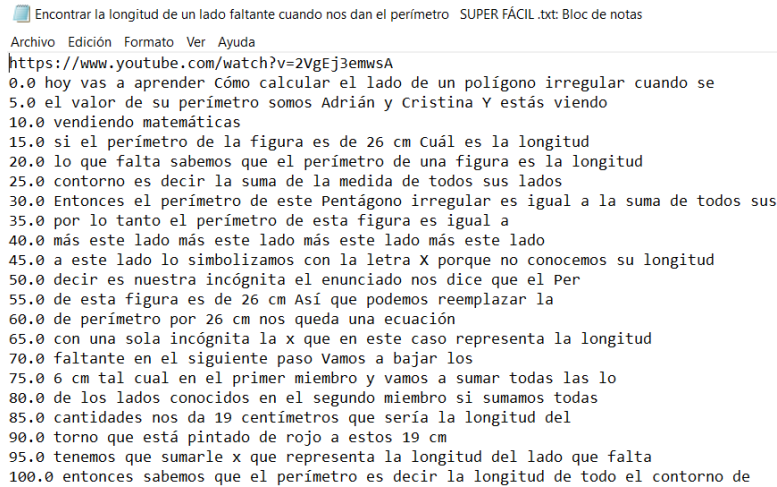


Figura 5.6: Video comenzando su reproducción desde el instante asociado al hipervínculo

En cuanto a las transcripciones generadas, se observa que son claras y fáciles de

leer, siguiendo el orden cronológico del video (figura 5.7). Esto permite que el usuario, si lo desea, pueda leer directamente el contenido de la transcripción y, gracias a que se guarda como un archivo de texto, también realizar modificaciones.



```
Encontrar la longitud de un lado faltante cuando nos dan el perímetro SUPER FÁCIL .txt: Bloc de notas
Archivo Edición Formato Ver Ayuda
https://www.youtube.com/watch?v=2VgEj3emwsA
0.0 hoy vas a aprender Cómo calcular el lado de un polígono irregular cuando se
5.0 el valor de su perímetro somos Adrián y Cristina Y estás viendo
10.0 vendiendo matemáticas
15.0 si el perímetro de la figura es de 26 cm Cuál es la longitud
20.0 lo que falta sabemos que el perímetro de una figura es la longitud
25.0 contorno es decir la suma de la medida de todos sus lados
30.0 Entonces el perímetro de este Pentágono irregular es igual a la suma de todos sus
35.0 por lo tanto el perímetro de esta figura es igual a
40.0 más este lado más este lado más este lado más este lado
45.0 a este lado lo simbolizamos con la letra X porque no conocemos su longitud
50.0 decir es nuestra incógnita el enunciado nos dice que el Per
55.0 de esta figura es de 26 cm Así que podemos reemplazar la
60.0 de perímetro por 26 cm nos queda una ecuación
65.0 con una sola incógnita la x que en este caso representa la longitud
70.0 faltante en el siguiente paso Vamos a bajar los
75.0 6 cm tal cual en el primer miembro y vamos a sumar todas las lo
80.0 de los lados conocidos en el segundo miembro si sumamos todas
85.0 cantidades nos da 19 centímetros que sería la longitud del
90.0 torno que está pintado de rojo a estos 19 cm
95.0 tenemos que sumarle x que representa la longitud del lado que falta
100.0 entonces sabemos que el perímetro es decir la longitud de todo el contorno de
```

Figura 5.7: Fragmento de una transcripción en un archivo de texto

No obstante, en algunos casos se puede observar que las palabras al final de ciertas líneas aparecen incompletas o no están presentes en la transcripción. Esto ocurre debido al método de procesamiento del audio, pues al dividirlo en segmentos de 5 segundos, es posible que algunas palabras queden divididas entre dos fragmentos, quedando incompletas. Dado que Google Speech Web API transcribe únicamente el contenido disponible en cada segmento, no completa las palabras que han sido cortadas durante este proceso.

Capítulo 6

Conclusión y Trabajo Futuro

La solución diseñada y desarrollada cumple de forma satisfactoria con los objetivos planteados, pues permite a los usuarios realizar búsquedas de manera eficiente en videos subidos a YouTube. Además, al almacenar las transcripciones en un archivo de texto, se facilita que los usuarios puedan modificarlas, ya sea para corregir un error o añadir notas personales.

Sumado a esto, como este tipo de archivo posee un tamaño bajo en cuanto a kilobytes, es posible que los usuarios puedan compartir transcripciones entre ellos, ya sea para ahorrar tiempo con el proceso de transcripción o para facilitar un archivo con anotaciones.

No obstante, aunque Re-VIP cumple con todos los objetivos establecidos, es posible mejorar su funcionalidad para hacerla más robusta y eficiente en comparación con su estado actual.

Por ello, se establece como trabajo a futuro el que los fragmentos creados a partir del audio del video no se dividan en 5 segundos, sino que se haga la segmentación cuando se detecte un silencio prolongado posterior a 5 segundos. Esta aproximación

permitiría evitar el corte de palabras pronunciadas, reduciendo la pérdida de información y logrando así tener una transcripción más confiable y precisa en relación al contenido original del video. Una forma alternativa de lograr esto sería tomar un tiempo delta posterior al fragmento, por ejemplo de 3 segundos, y comparar las sílabas de la sección media del tiempo delta con el siguiente fragmento de audio de 5 segundos hasta encontrar una coincidencia. Con esto sería posible revisar si existen palabras cortadas y reescribir los fragmentos transcritos al añadir las palabras faltantes.

Además, el usuario puede querer abortar el proceso de transcripción por distintos motivos, por lo que se establece que en un futuro Re-VIP posea una opción para detener la generación de la transcripción. Esto se puede lograr agregando un botón ligado a una función que detenga el proceso y elimine todos los archivos generados hasta ese momento para producir la transcripción.

Por otro lado, las transcripciones corresponden a un paso intermedio para que la aplicación funcione correctamente y no a un resultado final. Por esto, se propone que el leer la transcripción sea una función opcional y no un paso obligatorio; la búsqueda del archivo sería llevada a cabo por la misma aplicación utilizando la URL del video proporcionada por el usuario, aprovechando que esta misma se encuentra en la primera línea de la transcripción, por lo que solo se tiene que abrir los archivos hasta encontrar una coincidencia entre estas.

En cuanto a los nombres de los archivos con la transcripción del video, puede existir el problema que dos videos distintos tengan el mismo nombre, produciendo que un archivo sobrescriba al otro al momento de generarlo. Por lo que se propone agregar el ID del video al nombre del archivo, esto asegura unicidad entre los archivos, pues estos ID presentes en la URL son únicos para cada video.

Como complemento, al enfocar esta solución en el ámbito académico de la Uni-

versidad Técnica Federico Santa María, se abre la posibilidad de desarrollar una plataforma que almacene las transcripciones de todas las clases grabadas y subidas por los profesores a YouTube. Esto permitiría a cualquier estudiante conectarse a la plataforma y realizar búsquedas directamente en las transcripciones sin necesidad de generar estos archivos por su cuenta, dado que con el enlace del video, sería posible buscar en la plataforma si ya existe una transcripción, permitiendo el acceso directo a este material. Esta propuesta no solo optimiza el acceso al contenido académico, sino que también fomenta una experiencia de aprendizaje más eficiente y colaborativa.

Por lo anteriormente mencionado, si se busca implementar este programa a nivel institucional y no de usuario, es recomendable cambiar la API encargada de realizar las transcripciones. Como ya se mencionó, Google Web Speech API no está pensada para ser utilizada a gran escala, por lo que migrar el proyecto a otra herramienta de reconocimiento de voz más robusta, como Google Cloud Speech-to-Text API, permitiría gestionar un mayor volumen de solicitudes simultáneas de manera eficiente.

Referencias

- [1] Statista, “YouTube: Datos estadísticos” [en línea]
<https://es.statista.com/temas/10917/youtube-datos-mundiales>
[consulta: 16 noviembre 2024]

- [2] Universidad Técnica Federico Santa María, “USM establece nueva duración de hora pedagógica y equivalencia de créditos transferibles” [en línea]
<https://eli.usm.cl/2020/12/15/usm-establece-nueva-duracion-de-hora-pedagogica-y-equivalencia-de-creditos-transferibles/>
[consulta: 10 noviembre 2024]

- [3] Google Support, “YouTube Help -Traducción de videos y subtítulos” Google Support [en línea]
<https://support.google.com/youtube/topic/9257536>
[consulta: 22 Noviembre 2024]

- [4] TIOBE Software, “TIOBE Index” [en línea]
<https://www.tiobe.com/tiobe-index/>
[consulta: 25 noviembre 2024]

- [5] IEEE Spectrum, “The Top Programming Languages 2024” [en línea]
<https://spectrum.ieee.org/top-programming-languages-2024>
[consulta: 25 noviembre 2024]

- [6] yt-dlp Contributors, “yt-dlp: A youtube-dl fork with additional features and fixes” GitHub repository [en línea]
<https://github.com/yt-dlp/yt-dlp>
[consulta: 22 octubre 2024]

- [7] FFmpeg Developers, “FFmpeg Documentation” [en línea]
<https://www.ffmpeg.org/documentation.html>
[consulta: 16 Marzo 2025]

- [8] Jiaaro, “pydub: Manipulate audio with a simple and easy high-level interface”
GitHub repository [en línea]
<https://github.com/jiaaro/pydub>
[consulta: 23 octubre 2024]
- [9] Uberi, “speech_recognition: Speech recognition module for Python” GitHub repository [en línea]
https://github.com/Uberi/speech_recognition
[consulta: 23 octubre 2024]
- [10] Google, “Speech Recognition Demo,” [en línea]
<https://www.google.com/intl/es.es/chrome/demos/speech.html>
[consulta: 28 octubre 2024]
- [11] Python Software Foundation, “tk — Interface to Tcl/Tk” Python 3 documentation [en línea]
<https://docs.python.org/es/3/library/tk.html>
[consulta: 10 octubre 2024]
- [12] Tom Schimansky, “CustomTkinter: A modern and customizable Python UI-library based on Tkinter” [en línea]
<https://github.com/TomSchimansky/CustomTkinter>
[consulta: 26 noviembre 2024]
- [13] Python Pillow Contributors, “Pillow: The friendly PIL fork” [en línea]
<https://github.com/python-pillow/Pillow>
[consulta: 28 diciembre 2024]
- [14] Python Software Foundation, “concurrent.futures — Launching parallel tasks” Python 3.13 documentation
<https://docs.python.org/es/3.13/library/concurrent.futures.html>
[consulta: 25 octubre 2024]

- [15] Python Software Foundation, “os — Miscellaneous operating system interfaces” Python 3.13 documentation [en línea]
<https://docs.python.org/es/3.13/library/os.html>
[consulta: 2 noviembre 2024]
- [16] freeCodeCamp, “Métodos de string de Python explicados con ejemplo” [en línea]
<https://www.freecodecamp.org/espanol/news/metodos-de-string-de-python-explicados-con-ejemplo/>
[consulta: 19 noviembre 2024]
- [17] Hugging Face, “Audio Data - Introducción a los datos de audio” [en línea]
https://huggingface.co/learn/audio-course/es/chapter1/audio_data
[consulta: 28 noviembre 2024]
- [18] Python Software Foundation, “Sorting Techniques” Python 3.13 documentation [en línea]
<https://docs.python.org/es/3.13/howto/sorting.html>
[consulta: 28 octubre 2024]
- [19] W3Schools, “Python String lower() Method” [en línea]
https://www.w3schools.com/python/ref_string_lower.asp
[consulta: 19 noviembre 2024]