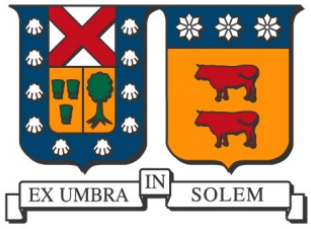
UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
SANTIAGO - CHILE



“OBTENCIÓN DE INFORMACIÓN A PARTIR DE LA COMPONENTE VISUAL EN PRESENTACIONES EN VIDEO”

FRANCO PALMA MUÑOZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Lioubov Dombrovskaia
Profesor Correferente: Roberto León

Agosto - 2023

DEDICATORIA

A mis papás Paola y Juan por haberme dado esta tremenda oportunidad de formarme en una de las mejores universidades, a mi pareja Renata por apoyarme en los momentos más difíciles, a mi hermana Samya, cuñado Braulio y sobrinos Martín y Vicente por animarme y motivarme a seguir adelante, a toda mi familia y amigos por la compañía y apoyo en los momentos que me permitieron llegar hasta aquí.

AGRADECIMIENTOS

A todos los profesores y ayudantes que se dieron el tiempo, y tuvieron la paciencia para explicar más de una vez lo que fuera necesario y a aquellos que transmitieron sus ganas e interés por lo que enseñaban tanto a mí como a mis compañeros.

Gracias a los compañeros con que pude compartir y aprender, sobre todo debo agradecer a Bastián quien me acompaño y fue un excelente apoyo en la gran mayoría de trabajos y hasta en las ultimas de este largo camino. También quiero expresar mi gratitud a los compañeros de proyectos, laboratorios y trabajos tan exigentes como la Feria, además mencionar y agradecer a la profesora Lioubov que nos apoyó durante nuestro proyecto y que para esta etapa final es mi profesora guía.

RESUMEN

Resumen— Un constante incremento en la creación de empresas pone presión para la obtención de financiamiento para desarrollar sus ideas y productos. La empresa Digevo, se encarga de incubar emprendimientos y evaluar su viabilidad a través de un extenso formulario llenado de manera manual. Por ello, se busca extraer automáticamente la información relevante de los *video pitches* incluso desde el contenido visual de las presentaciones, por esto se desea encontrar un método para obtenerla directo del video y aprovechar esta información que de otra manera se perdería.

Se creó una librería que permita la manipulación de los *frames* de un video y la obtención de una transcripción del texto presentado: A partir de los videos en varios formatos, extrae los *frames* y realiza una limpieza para eliminar información redundante, para luego, identificar diapositivas distintas y transcribirlas a texto usando sistemas OCR. Aplica un proceso de estructuración, que incluye la agrupación de textos según la división espacial en las diapositivas y también lematización o *stemming* sobre el texto.

De los resultados obtenidos se observó un buen y similar desempeño entre los sistemas OCR utilizados y para el caso de algunos parámetros dejados a libre elección en el proceso, se obtuvieron valores optimizados para la muestra de entrenamiento. En la fase de pruebas y validación, se obtuvo una baja pérdida en las estimaciones de diapositivas y se logró una precisión aproximada del 60 % para las transcripciones finales.

Palabras Clave— Transcripción, *video pitch*, procesamiento de texto, video OCR, reconocimiento inteligente de texto

ABSTRACT

***Abstract*—** A constant increase in the creation of companies puts pressure on obtaining financing to develop their ideas and products. Digevo, as a company, takes on the task of incubating startups and evaluating their feasibility through an extensive form filled out manually. Therefore, the aim is to automatically extract relevant information from video pitches, even from the visual content of the presentations, to find a method that directly obtains this information from the video and utilizes it, avoiding any potential loss.

A library was created to allow the manipulation of video frames and obtain a transcription of the presented text: starting from videos in various formats, it extracts frames and cleans them to eliminate redundant information. Then, it identifies different slides and transcribes them to text using OCR systems. The library applies a structuring process, including the grouping of texts based on spatial division in the slides and also lemmatization or stemming of the text.

The results showed good and similar performance between the used OCR systems, and for some parameters left to be freely chosen in the process, optimized values were obtained for the training sample. In the testing and validation phase, there was minimal loss in slide estimations, and an approximate accuracy of 60 % was achieved for the final transcriptions.

Keywords— *Transcription, pitch video. text processing, OCR video, intelligent text recognition*

GLOSARIO

API: *Application Programming Interfaces* (interfaz de programación de aplicaciones).
DI: Departamento de Informática.
EDSR: *Enhanced Deep Super-Resolution* (Súper resolución profunda mejorada).
ESPCN: *Efficient Sub-Pixel Convolutional Neural Network* (Sub pixel eficiente red neuronal convolucional).
FSRCNN: *Fast Super-Resolution Convolutional Neural Network* (super resolución rápida red neuronal convolucional).
JSON: *JavaScript Object Notation*.
LapSRN: *Laplacian Pyramid Super-Resolution Network* (Red de súper resolución con pirámide Laplaciana).
NSNL: *No Software No Life*, nombre de equipo de Feria del software.
OCR: *Optical character recognition* (reconocimiento óptico de caracteres).
POS: *part-of-speech* (parte de la oración).
PRPV: Porcentaje de palabras reconocidas por video.
PSNR: *peak-signal-to-noise ratio* (Proporción Máxima de Señal a Ruido).
PYPI: *Python Package Index*.
RGB: *Red-green-blue*. Corresponden a la representación de colores usadas para la presentación de una imagen.
SCTF-idf: Similitud coseno usando vectores de tipo Tf-idf.
SSIM: *structural similarity index measure* (Medida del índice de similitud estructural).
TF-IDF: *term-frequency times inverse document-frequency*.
UTFSM: Universidad Técnica Federico Santa María.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Es evidente que en los últimos años ha habido un auge en la idea de emprender. Sin embargo, crear un emprendimiento no es una tarea sencilla, ya que implica enfrentar diversos desafíos; como definir una idea sólida, identificar un público objetivo claro y generar valor en el producto o servicio. Entre todos estos desafíos, uno de los más significativos para dar vida al proyecto y convertir la idea en realidad es asegurar la financiación necesaria.

Existen varias formas de obtener financiamiento, pero en su mayoría requieren someterse a una evaluación exhaustiva. Estas evaluaciones suelen abarcar aspectos financieros, como explicar el método de generación de ingresos y realizar estimaciones presupuestarias. Asimismo, también pueden incluir evaluaciones técnicas, que consideran factores como el nivel educativo del equipo, la experiencia previa en emprendimientos, el sector en el que se desarrolla el emprendimiento, entre otros.

En el caso de Digevo, una incubadora de *startups* especializada en el apoyo de emprendimientos que hacen uso de herramientas de inteligencia artificial, se emplea un riguroso proceso de evaluación. Este proceso se inicia con un extenso cuestionario que consta de más de 80 preguntas relacionadas con el emprendimiento y sus integrantes. Dichas preguntas abarcan diversos aspectos, tanto financieros como técnicos, y se centran específicamente en el manejo de herramientas y el conocimiento sobre inteligencia artificial. Estos cuestionarios deben ser completados manualmente por los emprendedores y a su vez revisados por evaluadores, quienes dan la calificación según las respuestas.

Con el objetivo de simplificar este proceso tanto para los solicitantes como para los evaluadores, un equipo de desarrolladores de software ha considerado automatizarlo para mejorar la eficiencia en la evaluación de startups. Para lograrlo, se propuso utilizar una fuente de información que reemplace los cuestionarios tradicionales, que son la principal carta de presentación de las startups. Esta fuente de información es el *video pitch*, también conocido como elevator pitch, que consiste en un video de aproximadamente 2 a 5 minutos en el cual se resume todo el proyecto en una presentación concisa que incluye el modelo de negocios, los clientes objetivo y otros aspectos relevantes.

Solo centrándonos en el texto presente en las diapositivas presentadas, que entre las opciones disponibles es la fuente menos accesible, nos enfrentamos a un desafío para extraer la máxima cantidad y la más fiel información del video. El uso del video como fuente de información presenta algunas complicaciones para el análisis automatizado, ya que el contenido escrito no se puede extraer directamente debido a que las diapositivas se extienden a lo largo del video sin una estructura agrupada u ordenada.

Con el fin de mejorar los datos obtenidos de la presentación, se busca implementar mejoras en la extracción de información de las diapositivas presentes en el video. Esto implica utilizar herramientas OCR (Reconocimiento Óptico de Caracteres), llevar a cabo procesos de depuración de datos y formatear la información obtenida. Estas acciones permitirán obtener una

versión más estructurada y procesable de los datos presentes en las diapositivas del video.

El presente documento se desarrollará en el siguiente orden: en el primer capítulo la definición del problema, se explica el porqué y de donde nace el problema original del cual se desprende este trabajo junto a un resumen de la solución y los objetivos. En el segundo capítulo, el marco conceptual, se desarrollan los conceptos previos utilizados para la elaboración de la solución. En el tercer capítulo se explica la propuesta de solución, incluyendo el esquema de procesamiento de la información, explicaciones de cada etapa y dificultades encontradas durante el desarrollo. En el cuarto capítulo, validación de la solución, se describe la arquitectura de la solución, la estructura del proceso implementado, descripción de las pruebas (las muestras usadas, métricas usadas para evaluar, etc.). En quinto y último capítulo se dan las conclusiones generales, desarrollos posibles y algunas palabras finales.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Contexto u origen problema

En la actualidad existe un número elevado de emprendimientos y este ha ido en aumento en los últimos años, incluso durante la pandemia, se puede ver en el aumento de un 14.4 % en creación de empresas en el 2020 en comparación con el año anterior, fechas donde fue el apogeo mismo de las restricciones sanitarias y algunas otras limitantes por el coronavirus [Sustentable, 2021].

Otro factor importante a considerar en el crecimiento de los emprendimientos es la explotación de herramientas de inteligencia artificial (IA), que de por sí la inversión mundial en IA tuvo un aumento del 14 % en el 2021 y se espera que siga creciendo. En Chile se puede visualizar en el Índice de Nivel de Innovación y Crecimiento de IA (INICIA) tuvo un valor de 32 % en 2018 y que en el 2020 fue de 48 %, lo que indica un aumento importante en trabajos sobre herramientas con mayor innovación [Estrategia y line, 2021].

Ahora, considerando el número de empresas creadas, emprendimientos en creación y *startups* por venir, estas deberán conseguir financiamiento para materializar sus ideas, crear los productos y comercializarlos. Aquí es donde pueden recurrir a servicios financieros como créditos bancarios o algo más especializado como fondos públicos, capitales semilla y/o empresas incubadoras. En cualquiera de estos casos los proyectos serán evaluados para obtener su posibilidad de renta/éxito en el mercado.

1.2. Desarrollo problema y solución

La empresa Digevo cuenta con una rama llamada Digevo Ventures, la cual tiene como objetivo incubar emprendimientos, por tanto, recibe continuamente ideas de *startups* buscando obtener financiamiento para implementarla. En esta tarea, Digevo determina la viabilidad de una startup, analizando un formulario extenso con toda la información propia del proyecto y de la agrupación quien creó o propuso la idea. De ese proceso, tanto el llenado como el revisado del formulario requieren gran labor de parte de los postulantes como de los evaluadores, siendo esta una tarea realizada manualmente y toma muchas horas para lograrlo.

En este contexto, Digevo Ventures tiene el interés de obtener esta información de una forma más práctica y eficiente, la cual implica procesar los *videos pitch* de las *startups* y buscar en estos la mayor cantidad de información que aporte a la obtención de un factor de viabilidad, aquí es donde entra el equipo NSNL para implementar una solución (proyecto del que se desprende este trabajo de título).

La solución, una página Web en donde cada *startup* que quiera postular a un fondo de financiamiento o mejorar su presentación, pueda enviar su *video pitch*. Este siendo una explicación resumida y directa de su proyecto, el cual mantiene la estructura de *video pitch* con variaciones dadas por la originalidad del equipo de la *startup*. Teniendo el video se procede a la extracción del contenido expuesto, las fuentes utilizadas para obtener estos recursos son tres: un formulario, un documento PDF y el video con la exposición.

Para el caso del formulario, este tiene como fin asegurar información básica del emprendimiento, por lo que solo se obtienen datos de identificación de la *startup* y datos de contacto sobre uno de sus fundadores, por lo cual se limita lo obtenido por este medio.

El documento PDF son las diapositivas usadas para la presentación del emprendimiento, el cual idealmente se ingresará en la plataforma, pero solo como un campo opcional dado su especificidad. Al tener menor disponibilidad en este formato, se recurre a la última alternativa.

La última fuente de información y la más importante para el proyecto es el video, por el cual se pueden obtener dos tipos de recursos, el auditivo, que considera solo la voz del presentador y todo lo mencionado por él, de donde se extrae una transcripción directa utilizando inteligencia artificial.

Mientras que el otro recurso utilizado para extraer información es la presentación o contenido visual expuesto en el video, siendo este en el que se enfocará esta memoria, por lo que a continuación se detallará más sobre este; como obtenerlo, su procesamiento, estructuración y algunas problemáticas en el transcurso de esto.

Este recurso visual contempla una exposición, ya sea tipo PowerPoint, Canva, o creada con otras herramientas de diseño de exposiciones o simplemente texto, imágenes, animaciones, etc. La importancia de esta fuente es que muchos de los *videos pitch* presentan ciertos datos solo en este formato, ya que el presentador tiende a mencionar solo los que aportan mayor conocimiento sobre la idea central y el resto es dejado a la vista o lectura del público, o se podría llegar a hacer alusión verbal, pero no siempre se explicita todo lo escrito en la presentación, por esto recae gran importancia en obtener información desde esta fuente, con una calidad suficiente y un formato idóneo para su posterior análisis.

Luego de ver el foco de este trabajo de memoria, que en definitiva es obtener una transcripción coherente y fiel al contenido de la presentación, se pasará a desarrollar ciertos retos del proceso, específicamente de la obtención y preparación de la información:

Primero, toda esta etapa es dependiente de la creatividad puesta en la exposición y el sin fin de variaciones que pueden existir, pero como se mencionó el foco es en un tipo específico de exposiciones, estas son de tipo *video pitch* que hasta cierto punto acotan variaciones como la extensión, la formalidad en el lenguaje usado, temas abordados, entre otros, a pesar de esto aún existen algunas características del diseño importantes que pueden llegar a hacer engorroso el proceso, como lo son la inclusión de elementos con propiedades similares a un

texto, tales como diseños o dibujos con patrones continuos como una secuencia de símbolos o incluso letras como tal, usadas como fondo de las láminas o diapositivas.

Desde la variación en la tipografía y/o caligrafía surgen algunos errores en la transcripción, tales como tildes innecesarias o mal posicionadas, la invención de nuevas palabras creadas por la unión de dos contiguas dado su cercanía o por cortes debido al espaciado, letras ilegibles asimiladas por otra con mayor semejanza para el programa, generando de esta forma palabras con faltas de ortografía o incluso fuera del alfabeto utilizado. Sobre esto último para este trabajo se considera poder utilizar español e incluir el inglés debido al alto uso de palabras en este idioma dentro de presentaciones.

En relación con la estructura de las diapositivas, el reconocimiento de texto también se ve afectado cuando el orden de lectura no es estándar (occidental), de arriba hacia abajo y de izquierda a derecha, ya sea porque el texto dentro de las láminas tiene cortes, separaciones horizontales de párrafos o el hecho de que haya encapsulación en cuadros de texto.

Considerando el proceso de análisis de la data, si se quisiera mantener una transcripción bruta de cada diapositiva, aunque esta fuera correcta, fiel al contenido escrito y no tuviera los problemas mencionados anteriormente, el hecho de que esta no tenga una estructura ordenada y le falte conexión sobre el texto de cada bloque, pierde su valor al momento de extraer información, por tanto, encontrar y preservar las relaciones de cada bloque de información es una tarea esencial.

A fin de cuentas, el lograr entregar una transcripción con las características mencionadas y la estructura deseada, permitirá hacer viable conocer información que de otra forma se habría perdido, o, por otro lado, este proceso puede llegar a facilitar o hacer más preciso el trabajo realizado por el análisis posterior.

1.3. Objetivos

1.3.1. Objetivos Generales

Obtención y estructuración de información de presentaciones a partir de imágenes de un video.

1.3.2. Objetivos Específicos

1. Obtener información bruta de las imágenes de un video.
2. Limpiar la información extraída, minimizando la información irrelevante.
3. Estructurar la información, entregándole un formato estándar para análisis.

A continuación, se presenta el árbol del problema planteado, donde se detallan las causas que llevan al problema principal de este trabajo, así mismo se muestran los efectos o consecuencias de este, relacionados con el contexto del problema.

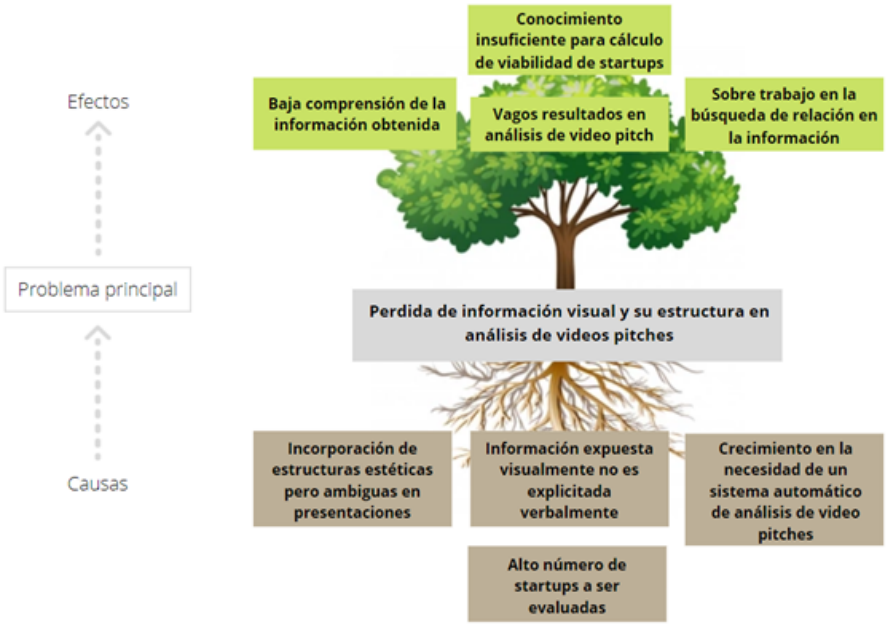


Figura 1: Árbol del problema.

CAPÍTULO 2
MARCO CONCEPTUAL

A continuación se presentarán las bases técnicas sobre la cuales se trabajará para esta memoria, se incluyen herramientas, métricas, metodologías de trabajo, entre otros:

- Reconocimiento de texto en frames de video.
- Reconocimiento óptico de caracteres.
- Métricas de calidad de imagen.
- Modelos de *upscaling*.
- Métrica de semejanza de textos: tf-idf.
- Lematización.

2.1. Reconocimiento de texto en frames de video

Inicialmente en el marco de definir un proceso para mejorar la transcripción obtenida de un video, el trabajo [Chen *et al.*, 2004] nos entrega un esquema general de como abordar este problema partiendo con imágenes o un video, y terminando con un texto. En este trabajo ellos remarcan que para obtener el texto contenido en un video se debe tener dos grandes procesos cubiertos. Primero, la detección del texto y, segundo, el proceso en sí de reconocimiento de texto. Su trabajo puede ser resumido en 4 etapas,

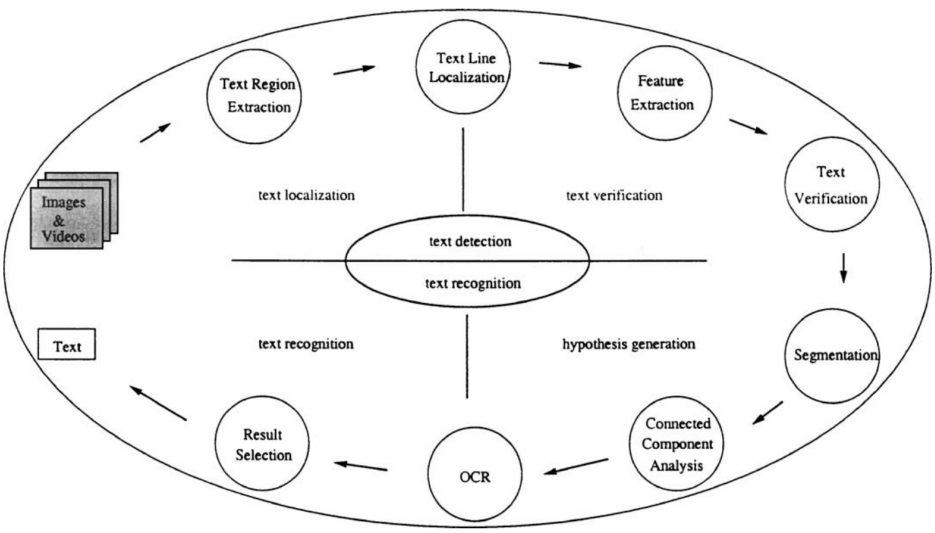


Figura 2: Algoritmo para la detección y reconocimiento de texto.
[Chen *et al.*, 2004]

En la figura 2 se muestra la división de las 4 etapas y el algoritmo completo usado por Chen y compañía para lograr extraer imágenes de un frame de un video.

- Localización consiste en la detección de bloques de texto, para esto se hace una estimación de probabilidades de que un píxel de la imagen sea parte del bloque y luego se procede a agrupar los que tienen mayor probabilidad. Una segunda parte de esta etapa localiza las líneas de texto en el bloque, primero ubicando la parte superior e inferior de cada línea, para después con una definición de parámetros sobre los píxeles se restringe las dimensiones de esta.
- Verificación se realiza debido al carácter empírico de la localización de las líneas y bloques de texto. Esto se logra usando métodos de minimización de riesgo empírico de *machine learning* llamados: *multilayer perseptrons* (MLP) y *support vector machines* (SVM).
- Generación de hipótesis: primero esta etapa trabaja con las imágenes en escala de grises y está enfocada en encontrar la escala de gris que toma todo el texto y asume que todo el resto es del fondo de la imagen. Para esto tiene un número predefinido de sistemas haciendo este proceso de búsqueda para obtener el mejor candidato de escala de gris y luego aplicar un algoritmo llamado *análisis de componentes conectadas*, que marca regiones dentro de la imagen como texto y no-texto basado en propiedades geométricas, reglas de dimensiones y proporciones de altura y ancho. Finalmente, solo se mantienen las componentes que cumplan lo anterior.
- Reconocimiento: contempla la evaluación de los componentes anteriores con un sistema OCR y la selección entre los resultados generados, con esta decisión basada en un valor de confianza estadística o grado de certeza que define el sistema OCR.

2.2. Reconocimiento óptico de caracteres

Como ya fue mencionado, es necesario incluir una forma de reconocimiento de caracteres, para esto están los sistemas *Optical character recognition* (OCR) o en español reconocimiento óptico de caracteres, los cuales hacen una conversión desde una imagen con contenido manuscrito o texto en letra imprenta hacia un texto digital disponible para ser manipulado en algún sistema computacional. Como se menciona en [Patel *et al.*, 2012] lo que hacen los sistemas OCR es una replicación de lo que logra el ojo y la mente humana en el proceso de reconocer texto.

El surgimiento de estos sistemas se remonta a Emanuel Goldberg, quien ya en 1914 había creado una máquina capaz de leer caracteres. Posteriormente, en los años 80, la empresa Hewlett-Packard desarrolló una versión avanzada de estos sistemas después de una fase de investigación y desarrollo en el reconocimiento de patrones en caracteres [Jesús, 2022]. Durante este proceso, para lograr una mejor distinción de los caracteres, se utilizaron modelos matemáticos inspirados en el funcionamiento del cerebro humano, conocidos como redes neuronales.

En la actualidad, los sistemas de reconocimiento óptico de caracteres (OCR) están estrechamente vinculados al aprendizaje automático (*machine learning*). Según indica [Dilmegani, 2021], algunas de las implementaciones más destacadas y eficientes en este campo son:

- ABBYY FineReader 15
- Amazon Textract
- Google Cloud Platform Vision API
- Microsoft Azure Computer Vision API
- Tesseract OCR Engine

Estos se centran en entregar resultados directamente como texto, en la figura 3 se muestra una comparativa hecha por Dilmegani [Dilmegani, 2021] en donde usa 3 tipos de datos o categorías para obtener resultados: capturas de pantalla tomadas de páginas Web con texto (*Category 1*), escritura hecha a mano (*Category 2*) y, por último, recibos, facturas y contratos escaneados (*Category 3*).

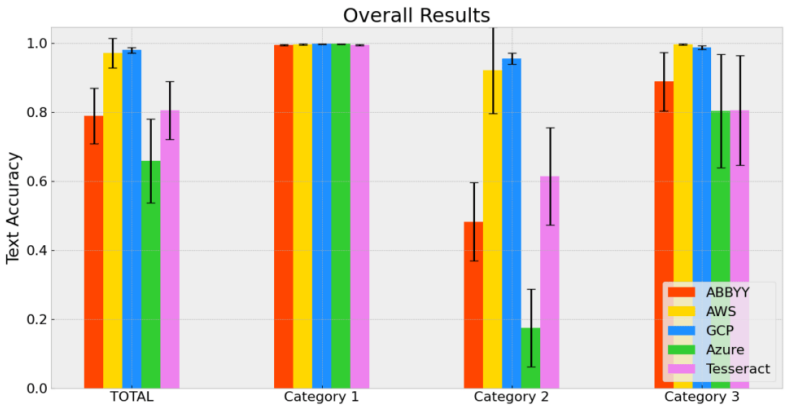


Figura 3: Comparativa entre sistemas OCR.
[Dilmegani, 2021].

Como se puede observar en la comparativa, uno de los sistemas más destacados en la actualidad es la API Vision desarrollada por Google en la plataforma *Google Cloud Platform* (GCP). Aunque se considera uno de los mejores, [Hosseini *et al.*, 2017] evaluó la robustez de esta API en situaciones donde las imágenes presentan alta contaminación.

Si bien la API Vision de Google es altamente efectiva, es fundamental considerar las condiciones específicas de las imágenes y aplicar técnicas de preprocesamiento, como los filtros propuestos por Hosseini y compañía, para garantizar resultados precisos y confiables.

La API Vision de Google también proporciona un amplio conocimiento sobre una imagen, incluyendo la capacidad de identificar la posición del texto encontrado. Esto permite delimitar bloques y determinar la distancia entre diferentes líneas de texto, lo que a su vez posibilita establecer relaciones entre distintos bloques de texto en una misma imagen.

Con esta funcionalidad, es posible obtener una comprensión más completa de la estructura y organización del texto en una imagen, lo que resulta especialmente útil para tareas como el análisis de documentos, la extracción de información y la comprensión de la disposición visual de los elementos textuales.

Otro sistema OCR no incluido en el anterior ranking es EasyOCR, que está disponible para el uso de manera libre, dado que es un software open-source y está a la par con uno de los mejores, Tesseract, que es otro sistema open-source ya mencionado. Estos fueron sujetos a una comparación por [Liao, 2020], donde se sometió a prueba a ambos sistemas con 1000 ejemplos con texto y 1000 con números, y se obtuvieron los siguientes resultados: para la precisión, Tesseract obtuvo una tasa de error en números de 5,5 % y letras del 0,7 %, mientras que para EasyOCR las tasas de error fueron 1,9 % en números y 4,3 % en letras. Para la velocidad de procesamiento por imagen se compararon diferenciando el empleo de GPU y CPU, usando solo CPU Tesseract procesa 2,7 veces más rápido con 0,3 segundos por imagen

y usando GPU EasyOCR sobrepasa 3,6 veces con una velocidad de 0,07 segundos por imagen.

2.3. Métricas de calidad de imagen

Considerando que se harán comparaciones de imágenes durante este trabajo, se revisaron formas de evaluar tales diferencias, utilizando métricas de calidad de imagen. La evaluación de calidad de imagen consiste en comparar características propias de una imagen contra las de otra, características como resolución o color de cada píxel. Actualmente, no existe una solución óptima que detecte o de información de los muchos parámetros que hay para analizar en las imágenes, pero la necesidad de poder definir la calidad o diferencias de una imagen llevan a tener varias métricas, algunas de estas son [Raval, 2021]:

- *Mean Squared Error* (MSE): es una función de bajo costo computacional, la cual satisface la interpretación de similitud y valores bajos de esta métrica indican alta similitud. Esta métrica funciona mejor en casos donde la distorsión de una imagen a otra se debe a ruido aditivo, este último entendiéndolo como una variación aleatoria en los píxeles en la imagen.
- *Peak Signal-to-Noise Ratio* (PSNR): Corresponde a la relación entre el máximo poder de una señal con el máximo poder de la señal de ruido, esta métrica presta mayor utilidad con imágenes HDR o alto rango dinámico. Los valores altos indican mayor similitud y da buenos resultados en casos de distorsión por ruido blanco
- *Structural Similarity Index* (SSIM): es una métrica enfocada en la similitud de dos imágenes que trae mejoras frente a PSNR y MSE. Los valores resultantes están normalizados, por tanto, entre más cercanos a 1 mayor es la similitud [Wang *et al.*, 2004]
- *Universal Quality Image Index* (UQI): es una métrica basada en modelos HVS o *Human visual system* la cual puede presentar inestabilidad, ya que en varias evaluaciones hechas por Wang et al. no presentó correlación [Wang y Bovik, 2002].
- *Feature Similarity Index* (FSIM and FSIMc): esta entiende una imagen por sus características de bajo nivel como la congruencia de fase que es robusta en casos de cambio de contraste e iluminación. FSIM es usada en imágenes en escala de grises o en la componente lumínica, mientras que FSIMc es para imágenes a color.

Y existen muchas otras más, como variaciones de las anteriores o con focos en otros tipos de errores o ruidos de imagen.

- *Multi-scale Structural Similarity Index* (MS-SSIM, [Wang *et al.*, 2003])
- *Root Mean Squared Error* (RMSE)

- *Erreur Relative Globale Adimensionnelle de Synthèse* (ERGAS)
- *Spatial Correlation Coefficient* (SCC, [Zhou *et al.*, 1998])
- *Relative Average Spectral Error* (RASE, [Gonzalez-Audicana *et al.*, 2004])

Algunos de los tipos de ruidos en los cuales se enfocan las métricas son los siguientes: difuminado (blur), desenfoque Gaussiano, interferencia granular (speckle), ruido de impulso o *Salt and pepper* (S&P), ruido de Poisson, entre otros. Y frente a estos las métricas tienen diferentes comportamientos, como se puede ver en la tabla comparativa 4, donde por ejemplo a grandes rasgos para el caso de S&P las métricas con mejor desempeño son SSIM, MSSSIM y UQI.

Metric/Image	Original	Blur	Gaussian	Speckle	S&P	Poisson
MSE	0	685.012	3955.396	6294.409	251.843	88.346
RMSE	0	26.173	62.892	79.337	15.869	9.399
PSNR	inf	19.774	12.159	10.141	24.119	28.669
SSIM	(1.0,1.0)	(0.689,0.690)	(0.319,0.329)	(0.386,0.391)	(0.855,0.856)	(0.924,0.924)
UQI	1	0.923	0.689	0.771	0.948	0.992
MSSSIM	1	0.916	0.7734	0.7491	0.9829	0.9875
ERGAS	0	14994.295	33527.153	42277.242	9561.001	5502.346
SCC	1	-0.022	0.2047	0.2561	0.7928	0.7823
RASE	0	2087.62	4801.734	5810.969	1400.691	764.301
SAM	0	0.1915	0.4568	0.5941	0.1038	0.0681
VIFP	1	0.3113	0.0786	0.0973	0.5706	0.5308

Figura 4: Comportamiento de diferentes métricas bajo varios tipos de ruido en imagen [Raval, 2021].

Para el caso particular de la diferenciación de frames en video se puede usar cualquiera de estas, aunque se debe considerar el tipo de ruido presente en la data a usar como también la robustez de cada métrica frente a estos.

2.4. Modelos de *upscaling*

En casos donde las imágenes disponibles tienen baja resolución o cuando se desea mejorar la resolución actual, se utiliza el proceso de *upscaling*. Inicialmente, este proceso se llevaba a cabo mediante algoritmos básicos que intentaban completar los píxeles intermedios en una imagen para obtener más detalle. Sin embargo, en la actualidad se ha adoptado un enfoque basado en nuevas herramientas como el *deep learning*, una rama del *machine learning* que busca desarrollar algoritmos que permitan a las máquinas o computadoras manejar tareas más complejas y aprender de ellas.

Una de estas tareas es el *upscaling*, también conocido como súper-resolución de imágenes. En este contexto, las redes neuronales convolucionales se destacan como una de las principales herramientas utilizadas. Algunos modelos relevantes en este campo, abordados en [Anwar *et al.*, 2020] son:

- EDSR (*Enhanced Deep Super-Resolution*)
- FSRCNN (*Fast Super-Resolution Convolutional Neural Network*)
- ESPCN (*Efficient Sub-Pixel Convolutional Neural Network*)
- LapSRN (*Laplacian Pyramid Super-Resolution Network*).

En un estudio realizado por [Bhartia *et al.*,], se compararon estos modelos en diferentes escalados de imagen. Para evaluar la calidad de las imágenes resultantes, se volvieron a las dimensiones originales y se utilizó el indicador PSNR en comparación con la imagen original. A continuación, se presentan los resultados obtenidos para escalados x2 y x3:

Nombre modelo	Escalado x2	Escalado x3
EDSR	28.550	26.484
ESPCN	28.380	25.961
FSRCNN	28.167	26.128
LapSRN	28.098	-

Tabla 1: Resultados de comparación de cuatro modelos de *upscaling*, para aumento del doble (x2) y triple (x3) de la escala original [Bhartia *et al.*,]

De los resultados anteriores señalar que el modelo LapSRN no posee escalado x3 y se destaca que el mejor desempeño fue obtenido por el modelo EDSR.

2.5. Métricas de semejanza de textos

Existen maneras simples de comparar textos, obteniendo el porcentaje o el número de palabras que coinciden entre dos textos, esto implica directamente tomar cada palabra y revisar la existencia en otros textos sin tener ninguna consideración relativa al uso o relevancia de cada palabra en el texto mismo. Tres métodos que si consideran la semántica de los textos son: similitud coseno usando vectores tf-idf, similitud coseno usando *word embedding* y similitud soft coseno usando *word embedding*, donde tf-idf o *Term Frequency-Inverse Document Frequency* y *word embedding* (Word2Vec es una de las implementaciones más usadas de *word embedding*) son dos métodos para obtener vectores de características, que a su vez estos son un grupo de características de un objeto representadas numéricamente y la similitud coseno y soft coseno son dos funciones para obtener la similitud entre dos objetos. Las

tres fueron comparadas en el trabajo [Sitikhu *et al.*, 2019], del cual se extraen los siguientes resultados:

Métodos usados para calcular similitud de texto	**Precisión Top-1 (%)**
Similitud de coseno usando vectores tf-idf	76.8
Similitud de coseno usando vectores Word2Vec	75.9
Similitud coseno suave usando vectores Word2Vec	76.0

Tabla 2: Resultados de comparación de tres métodos de calculo de similitud de texto[Sitikhu *et al.*, 2019].

Sin ahondar en el proceso usado para la comparación se tiene que los mejores resultados, aunque con leve diferencia, los obtiene la similitud de coseno usando vectores tf-idf.

La similitud de cosenos usando vectores tf-idf (*term-frequency times inverse document-frequency*) disminuye el peso de palabras que tienen una ocurrencia demasiado elevada en un conjunto predefinido de documentos, por lo que se considera que estas no aportan mayor información al documento principal [Stecanella, 2019].

Esta métrica ya tiene una implementación disponible en la librería de *scikit-learn* [Pedregosa *et al.*, 2011], la cual incluye muchas otras herramientas para trabajar en el aprendizaje automático, pero la relevante para este trabajo corresponde a *TfidfVectorizer* importada desde el módulo *sklearn.feature_extraction.text*, la cual permite obtener un valor para medir la semejanza de textos directo desde dos párrafos o textos.

2.6. Lematización

Con el foco de llevar la transcripción final a una estructura idónea para el análisis se considera la lematización, esto considerando que es un proceso de normalización que agrupa las inflexiones de una misma palabra con el fin de reducir la complejidad de un texto y hacer análisis automatizado de manera más eficiente, este proceso considera un vocabulario dado y características morfológicas de las palabras, tales como el tiempo verbal, género, entre otras.

Existen varias herramientas para aplicar lematización y una de estas es el presentado en el trabajo [Qi *et al.*, 2020], un *toolkit* llamado Stanza con herramientas disponibles en varios lenguajes, incluyendo inglés y español. Esta herramienta facilita el proceso de lematización, el manejo de etiquetas del POS (*part-of-speech*, indica donde se encuentra la palabra dentro del discurso), y también permite obtener características morfológicas directas de un texto, siendo estas dos últimas relevantes al otorgar mayor precisión en la lematización.

El *toolkit* Stanza está implementado en Python, hace uso de redes neuronales para un análisis robusto de texto y entrega variadas opciones para el análisis de lenguaje natural. A continua-

ción, en la figura 5 se muestra un resumen del camino para el procesamiento de texto que tiene Stanza.

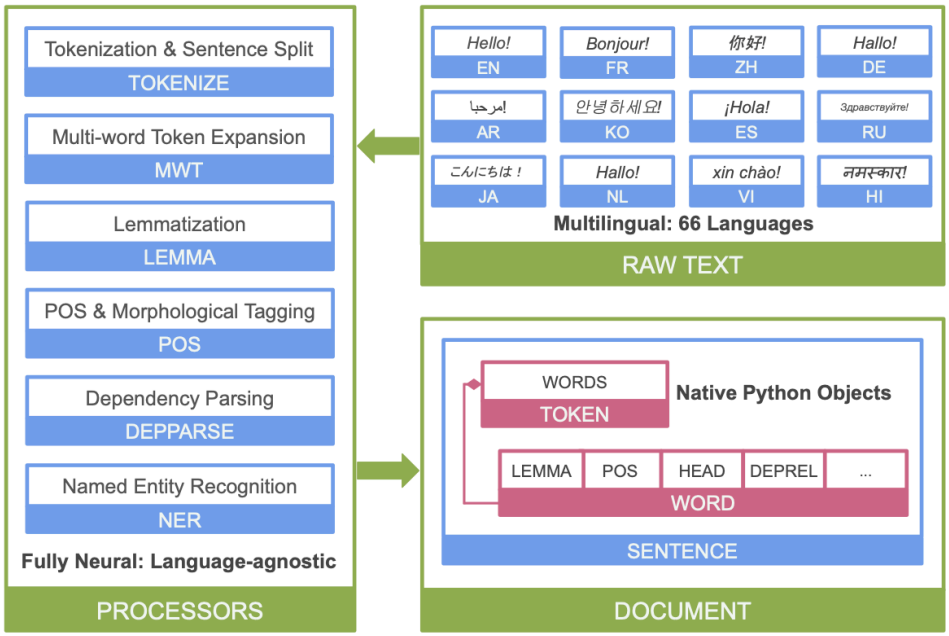


Figura 5: Resumen de los servicios que entrega el paquete de Stanza [Qi *et al.*, 2020].

Algunas definiciones relevantes de lo que se dispone en el *toolkit*:

Etiquetado gramatical o *POS tagging*: corresponde al proceso de asignar etiquetas a palabras dentro de un texto o *corpus*, estas etiquetas consideran la definición, como también el contexto en el que se encuentra. Ejemplos de etiquetas pueden ser el tiempo verbal, número gramatical, relaciones gramaticales como verbos, sustantivos, etc.

Análisis Morfológico: es la definición y clasificación de palabras según su estructura interna o componentes, tales como la base léxica, raíz o lexema de la palabra, como también el morfema que es la unidad mínima con significado.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

Para el problema y contexto planteado, se crea una librería que contiene tanto la manipulación de los *frames* de un video como la obtención de la transcripción final inmediatamente disponible para ser analizada o manipulada según se necesite. Se elige esta opción dado que este problema se origina dentro de un proyecto más grande en el cual se puede usar directamente esta librería o también puede servir de apoyo como base para la manipulación y extracción de texto en videos. En esta se incluye una serie de dependencias a paquetes de acceso libre, tales como Tesseract, EasyOCR, OpenCV y sus modelos, entre otras. Esta librería se desarrolla para el lenguaje de programación Python 3.6.8 y queda disponible para su instalación con el comando "*pip install*" gracias a que se aloja en los repositorios de PYPI (*Python Package Index*), sistema el cual es masivamente conocido y aloja *software* libre para el lenguaje de programación Python. También se pone a disposición el código completo en otra plataforma con el objetivo de visualizar o importar este para trabajar directamente desde una copia local, el repositorio se llama *VideoSlides* y esta disponible en [Palma Muñoz, 2023].

Esta librería permite trabajar desde un video ya sea como archivo (formato mp4, webm, entre otros) o desde un *link* de Youtube, incorporando esta última opción, dado es una de las principales plataforma libres para compartir videos y en el contexto de emprendimientos es posible compartir un video a empresas financiadoras directamente con el *link*. Se permite extraer los *frames* desde cualquier fuente de video, dando la opción de guardarlos o simplemente eliminarlos al terminar su uso. Con estas imágenes o *frames* extraídas se puede hacer una limpieza con el fin de minimizar las imágenes que contengan la misma información o así también posean menos información que alguna antecesora o sucesora, esto se logra usando alguna de las métricas de calidad de imagen mencionadas en la sección 2.3, con las que se puede obtener un valor comparativo para cuantificar que tanto se asemejan o diferencian dos imágenes, en este caso siendo *frames* contiguos extraídos del video.

Luego de esta limpieza, continuando con el objetivo de reducir la redundancia en la información extraída, se procede a localizar el número de diapositivas, tanto el frame inicial como el final asociado a una misma diapositiva en la presentación. En estos mismos conjuntos se aplicará una nueva limpieza teniendo en consideración no eliminar datos relevantes, por lo tanto, minimizando la perdida de información.

Posteriormente, se implementa una mejora en la calidad de los frames seleccionados utilizando modelos de *upscaling*. Esta técnica tiene como objetivo mejorar los resultados en el reconocimiento de los sistemas OCR, lo que contribuye significativamente a la precisión y eficiencia del proceso.

Ya teniendo los *frames* finales, se procede con la transcripción de imagen a texto, para lo que se usaran distintos sistemas OCR. En este caso, se utilizarán dos opciones: EasyOCR y Tesseract, dos sistemas de uso libre, con implementaciones disponibles en Python y con ambas se

tiene la opción de extraer la posición del texto reconocido.

Haciendo uso de técnicas para la detección de plagio en texto se comparará lo transcrito, esto con el fin de aplicar una última etapa de limpieza donde se busca eliminar redundancia, aunque esta vez la comparación es sobre la transcripción misma y no sobre las imágenes o *frames*, esto mediante valores de distancia/semejanza en los textos obtenidos de las distintas diapositivas.

Por último, el proceso de estructuración, que incluye agrupación de textos según, la división espacial en las diapositivas y el uso de cambios en la transcripción: lematización y/o *steeming*. Estos últimos procesos guiados con el fin de facilitar un futuro análisis de la información, sea utilizando modelos pregunta-respuesta u otro que necesite lematización.

En la figura 6 se muestra el flujo del proceso, comenzando con el único usuario, quien proporciona el video en alguna de las opciones disponibles. A medida que avanza por diferentes etapas, el video se transforma en un conjunto de *frames*, una agrupación de subconjuntos de *frames* o diapositivas y, finalmente, en un texto o transcripción. El proceso se divide en 5 secciones: recolección de *frames* en verde, agrupación de diapositivas en azul, mejora de calidad de imagen en negro, transcripción y estructuración en rojo, y etapas de limpieza en amarillo. Cada una de estas secciones será detallada más adelante en profundidad.

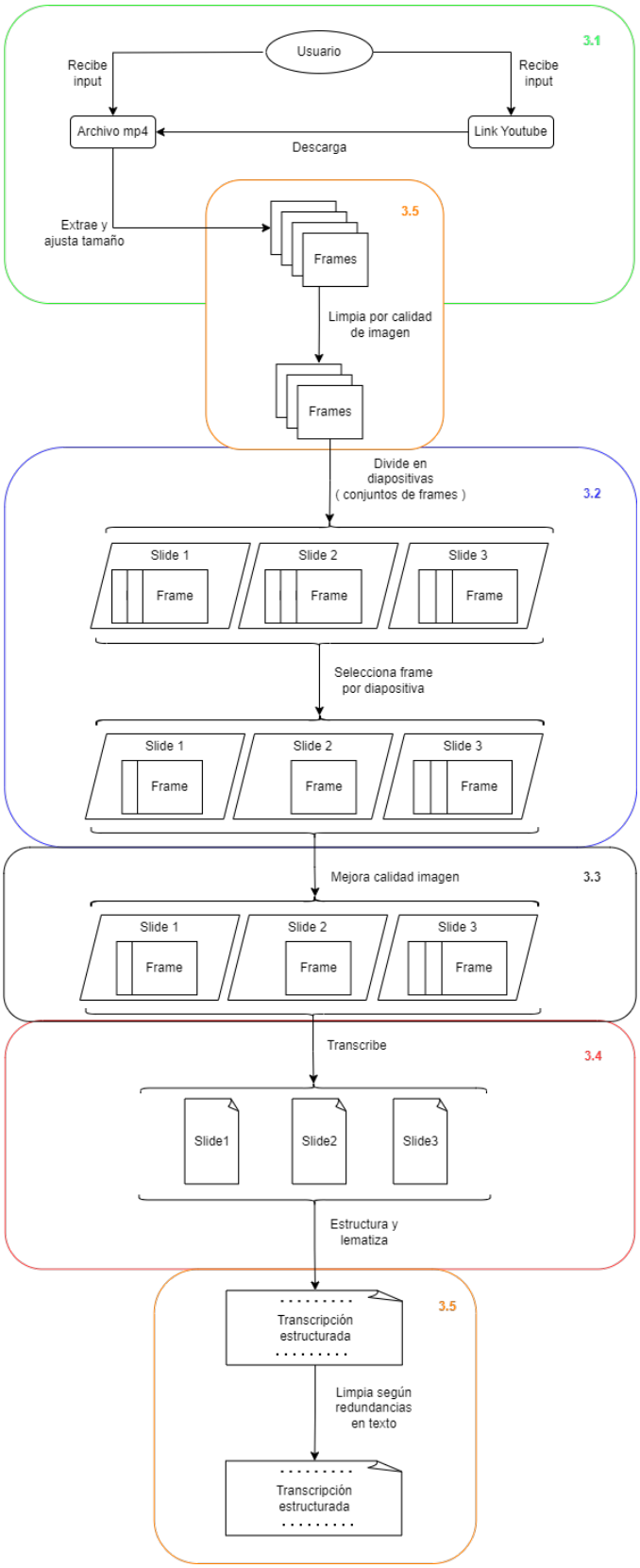


Figura 6: Esquema del flujo del proceso y gestión de información en 5 secciones.

3.1. Recolección de *Frames*

Para este proceso se usó la librería CV2 que permite extraer cada frame desde un video y también da más información como las dimensiones de cada *frames*, así como también la tasa de *frames* por segundo. Estos datos son relevantes, ya que a la hora de manejar la limpieza, se consideró y aplicó una disminución sobre la cantidad total de *frames* que se leen por segundo. Se definió JUMPS, un parámetro de salto en la lectura de frames, que indica cada *n*fps* veces se lee un frame desde el video, este parámetro de saltos en la lectura se define estático para idealmente mantener una tasa de lectura constante de *frames* por segundo, suficientemente pequeño para no perder información y grande para evitar comparaciones innecesarias. Este método de limpieza es de mayor utilidad para casos de altas tasas de *frames* por segundo, que es generalmente una característica positiva, ya que otorga mayor fluidez y mejor calidad al video, pero no aportar a nuestro objetivo de tener la mayor cantidad de información, por lo mismo no presta un mayor valor tener grandes cantidades de *frames* cada segundo. En esta etapa de extracción, también se consideró el caso de necesitar aumentar la velocidad del proceso, disminuyendo las dimensiones de los *frames* a expensas de disminuir la calidad de los resultados en la transcripción.

A continuación en 7 se tiene un ejemplo de este proceso de recolección de *frames* usando el parámetro JUMPS igual a 1, con un video de 30 fotogramas por segundo, en donde se puede ver en el nombre de los *frames* una numeración que indica su posición dentro del total de *frames* del video, y para este caso solo hay lecturas cada 30 *frames*.

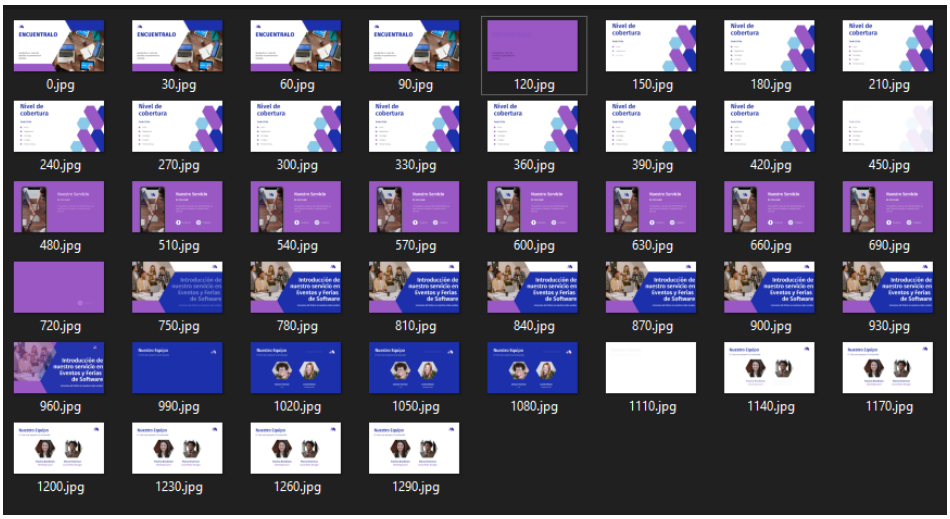


Figura 7: Ejemplo de extracción de frames para un video de 30 fps con parámetro JUMPS igual a 1.

3.2. Agrupación por diapositivas

Continuando en el flujo del esquema de la figura 6 sin considerar la limpieza que se abordara más adelante, se llega a la etapa de identificación de diapositivas, que implica tomar la totalidad de los *frames* y separarlos en grupos donde cada grupo representa solo una diapositiva, siendo este un beneficio no encontrado en la literatura, ya que en los trabajos encontrados se debía considerar el texto de cada uno de los *frames* lo que aumentaba considerablemente la carga y tiempo de procesamiento.

Para lograr esta selección de diapositivas se aplicaron primero dos métodos de agrupación o identificación de transiciones, uno usa la similitud de imágenes y el otro está basado en el volumen del audio del video. Luego de tener la agrupación se pasa a la selección de un único frame dentro de cada grupo con el foco que sea el representante y, por lo tanto, incluya idealmente toda la información expuesta en el resto del grupo.

En esta etapa y en las que consideran el uso de la imagen como la data principal, se habilita la opción de usar cada frame con dos códigos de color: blanco-negro y código rojo, verde y azul (RGB). Esto dado que en algunos trabajos se obtienen diferentes resultados según se cambie este parámetro, como es el caso del trabajo de [Mancas-Thillou y Gosselin, 2006] donde se afirma que el código RGB puede dar mejores resultados manejando variabilidad en cierto tipo de imágenes.

3.2.1. Agrupación por similitud imagen

Este método se centra en encontrar las divisiones entre distintas diapositivas usando métricas de similitud de imagen, enfocándose en el punto donde los *frames* tienen alguna transición, ya sean de tipo barrido, disolvencia o desenfoque, que son algunas de las más comunes. De esta manera poder marcar el inicio y fin de una diapositiva y por consiguiente crear grupos con alta similitud, relacionada con altos valores para las métricas de calidad de imagen.

Para encontrar estos puntos se evalúan las similitudes entre los *frames* contiguos usando algunas de las métricas mencionadas en la sección 2.3, la elección de la métrica específica a usar, se deja a elección del usuario como un parámetro libre, esto debido a los diferentes ruidos que podrían existir en la data usada, las opciones disponibles son SSIM, MSE, PSNR, UQI y dif, donde está última es una métrica simple y directa de la resta entre las matrices de ambas imágenes divido en el total de píxeles para normalizar el valor final. Con los resultados de las evaluaciones/comparaciones de todos los *frames* con sus vecinos se genera una lista de tasa de similitud, con la cual se puede graficar cada par de *frames* versus una similitud normalizada y, finalmente, con la lista y su representación gráfica se pueden ubicar los mínimos locales que indicaran puntos donde la similitud tiene un descenso marcado seguido de un aumento, aquí es donde se toma la asunción de que estas bajadas corresponden a un cambio de diapositiva.

A continuación se muestra este tipo de gráfico para dos casos, primero en la figura 8 el caso pesimista donde se obtiene una función con 15 mínimos locales, lo que indicaría que hay 16 diapositivas en este video, mientras que realmente tiene 21 diapositivas, lo cual se puede deber a las numerosas animaciones o contenidos multimedia y los saltos inconsistentes con los cambios de diapositivas se deben a las bajas en similitud de imagen no marcadas.

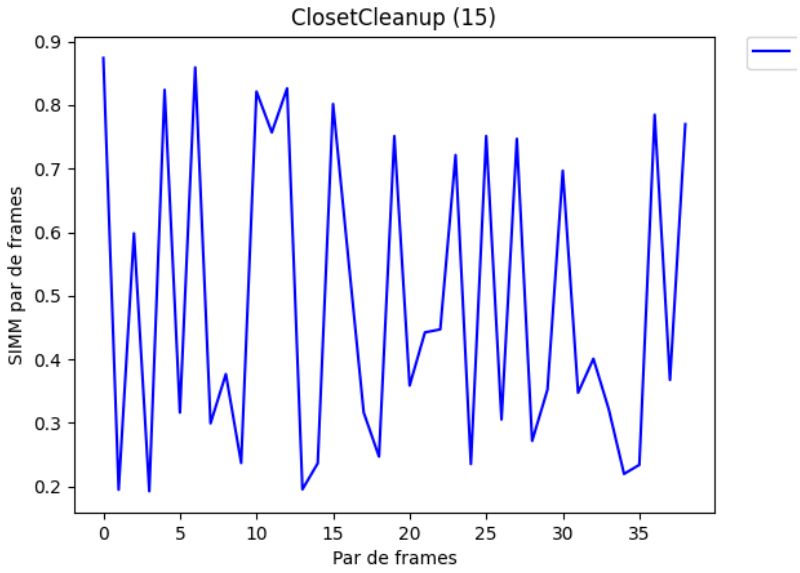


Figura 8: Gráfico de similitud de imagen versus par de *frames* contiguos para caso pesimista de agrupación.

El gráfico 9 es el caso optimista, donde se obtienen 35 mínimos locales, lo que indicaría un total de 36 diapositivas en el video, cuando realmente contiene 30 diapositivas, dando un 20 % de más. Este caso se considera optimista dado que en una situación como esta se incluirán textos redundantes, los cuales se podrían eliminar en una etapa de limpieza posterior. Contrario al caso anterior cuando se tendrían menos diapositivas por transcribir, causando la eliminación de información luego de la selección.

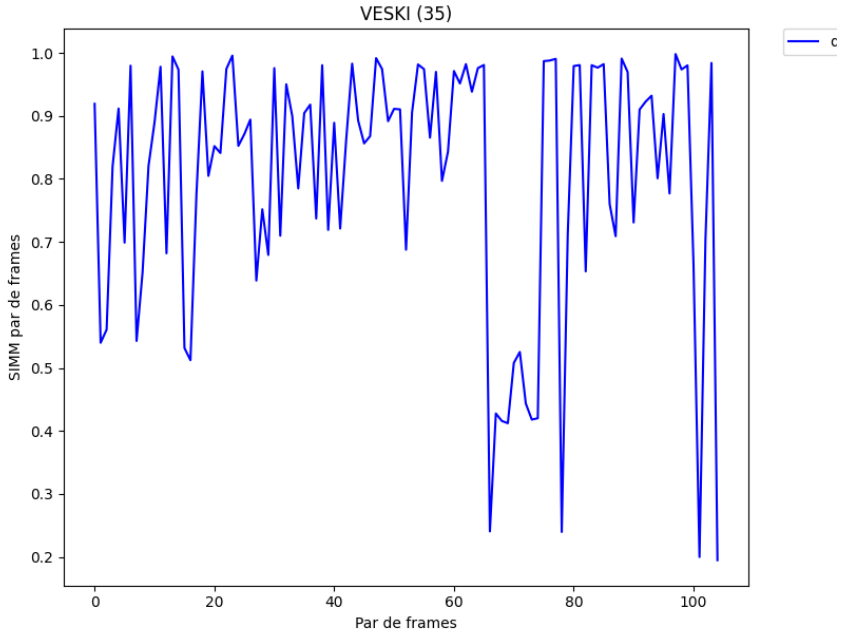


Figura 9: Gráfico de similitud de imagen versus par de *frames* contiguos para caso optimista de agrupación.

Considerando que existen cambios dentro de una misma diapositiva que puedan provocar caídas en la similitud, se crea un parámetro que permita tomar en consideración estas posibles variaciones, permitiendo solo contabilizar las diferencias que superen este factor. A continuación se tiene el gráfico 10 para un video de 10 diapositivas, que para el caso normal obtiene 17 mínimos, lo que serían 18 diapositivas. Al aplicar un coeficiente 0.8, línea punteada roja, quedan fuera algunos mínimos quedando solo 8 y, por lo tanto, un total de 9 diapositivas.

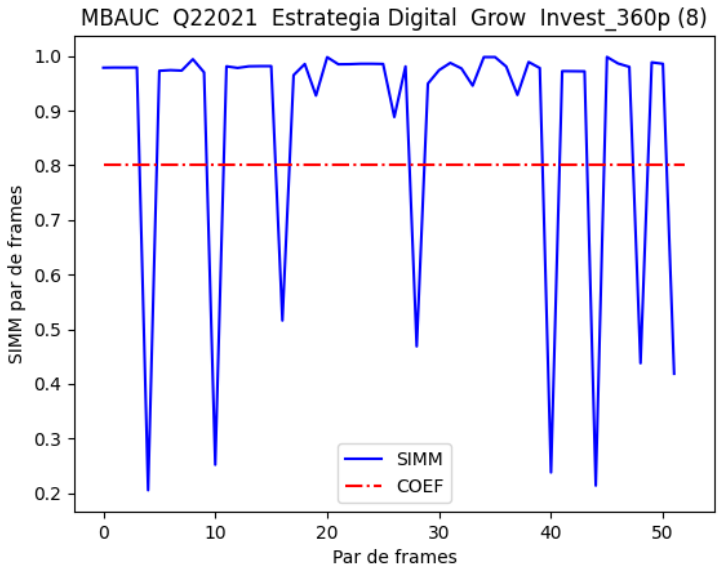


Figura 10: Ejemplo de identificación de mínimos/diapositivas para un video de 10 diapositivas con parámetro COEF igual a 0.8.

Como fue mencionado, el código de colores puede afectar los resultados del análisis de imágenes, como es el caso en la comparación de imágenes con métricas de similitudes, por esto se incluye un campo RGB que en caso de estar activo (con valor *True*) se aplica código rojo-verde-azul y en caso contrario se usa código blanco-negro. A continuación se muestra un caso en el cual se ejemplifica esta diferencia procesando un video con 14 diapositivas, con ambas bandas de colores. Con RGB falso se aprecia en el gráfico 11 un menor número de diferencias, con 16 mínimos que indican 17 diapositivas en total. Mientras que en el gráfico 12 para RGB verdadero se estiman 26 mínimos, denotando un conteo de 27 diapositivas.

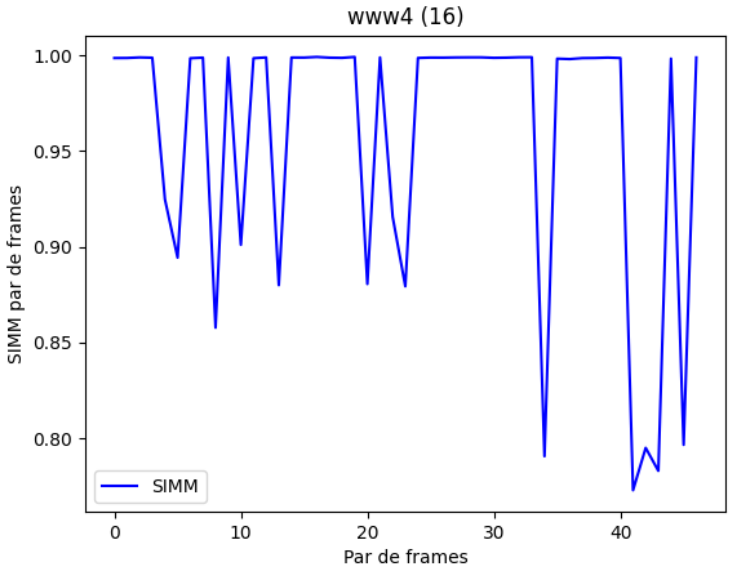


Figura 11: Ejemplo de identificación de mínimos/diapositivas para un video de 14 diapositivas con parámetro RGB igual a *False* o banda blanco-negro.

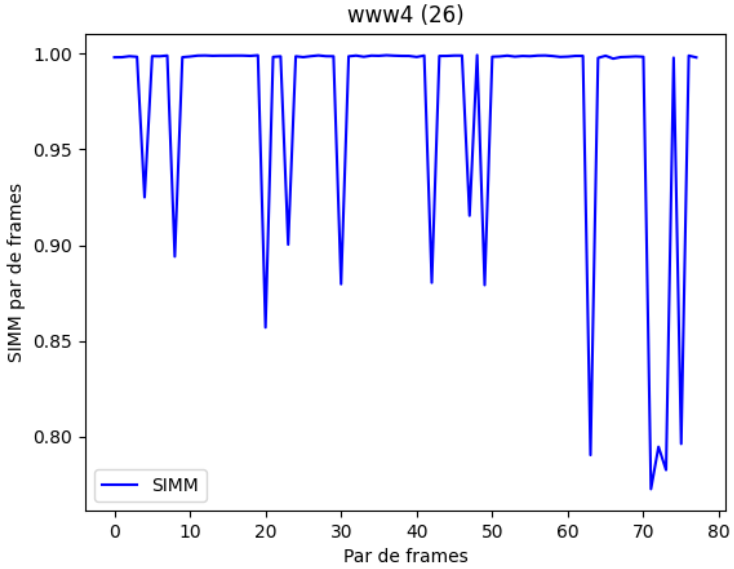


Figura 12: Ejemplo de identificación de mínimos/diapositivas para un video de 14 diapositivas con parámetro RGB igual a *True* o banda rojo-verde-azul.

Considerando las diferentes opciones de métricas mencionadas, cada una con distintos fuertes frente a ruidos en las imágenes, se define un parámetro METRICA que puede tomar un

número entero, índice para indicar estas métricas. En la siguiente tabla se ejemplifica el desempeño de cada una en la etapa de agrupación por diapositivas, considerando un video con 14 diapositivas, se estimaron los siguientes valores:

METRICA	Diapositivas estimadas
SSIM	27
dif	28
mse	28
psnr	2
uqi	26

Tabla 3: Número de diapositivas estimadas para un video de 14 diapositivas con parámetro METRICA tomando los valores 1, 2, 3, 4 y 5, correspondientes a los índices de las métricas usadas.

3.2.2. Agrupación por audio

En el siguiente método abordado para agrupar diapositivas se usa el audio del video, esto con base en la asunción de que durante los cambios de diapositivas existirá unos segundos de silencio que permitirán separar los saltos y, por lo tanto, ubicar cambios en diapositiva, lo cual implicaría mantener registro temporal del instante al que pertenece cada frame dentro del video y coincidir con el audio. Estas pruebas no entregaron buenos resultados, por lo que se puede visualizar en los siguientes gráficos donde se toma el audio en el tiempo y para tener una mayor claridad se aplican funciones de suavizado para idealmente destacar cada cambio.

Para los mismos ejemplos expuestos con el método anterior se tiene:

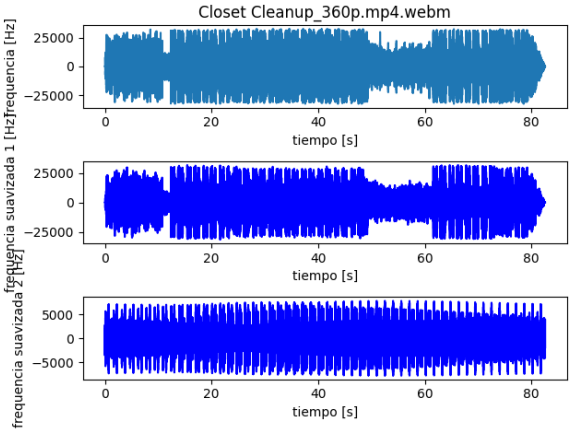


Figura 13: Gráfico de frecuencia versus tiempo del audio del video “closet cleanup”.

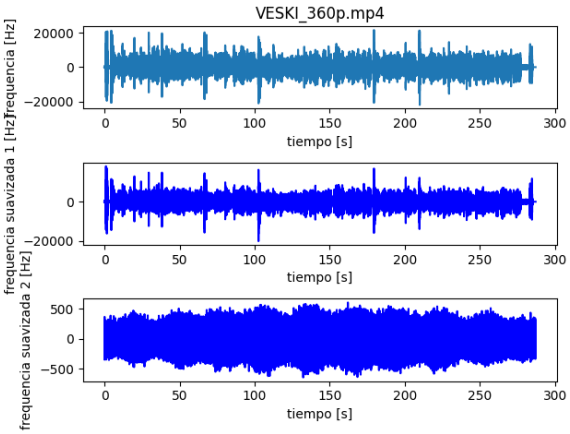


Figura 14: Gráfico de frecuencia versus tiempo del audio del video “veski”.

Como se mencionó, para este caso se probó aplicando transformaciones para suavizar la curva del audio, con el objetivo de destacar las subidas y bajadas en el volumen e idealmente cambios en las diapositivas. No dio buenos resultados, ya que no se obtuvieron mínimos distintivos que denotaran silencios previos a cambios de diapositiva, dada esta baja correlación este método de agrupación fue descartado. Una de las principales causas pueden ser los audios/música de fondo, usados comúnmente en presentaciones de *video pitch* y también los casos que el presentador mantenía un discurso fluido independiente existiera un cambio de diapositiva.

3.2.3. Selección de frame representante

Luego de obtener la división en grupos de *frames* que comparten información de una misma diapositiva, se debe elegir cuál o cuáles de estos *frames* son quienes contienen la mayor cantidad de información por cada agrupación:

Un primer acercamiento heurístico a la solución es seleccionar el último frame de cada conjunto, que en general puede dar buenos resultados considerando transiciones instantáneas, donde no existe frame intermedio que involucre las diapositivas contiguas, pero para casos donde las transiciones de diapositivas contiene difuminación de imagen/información de algún tipo, entonces el último frame de cada grupo puede contener menos información que *frames* intermedios.

El segundo método utilizado considera el problema mencionado anteriormente, consiste en obtener la transcripción de cada frame y comparar el texto obtenido, el parámetro a considerar para diferenciar es el número de palabras contenidas en cada texto. De esta forma, al buscar el máximo dentro de cada grupo se podría indicar que se tiene el frame con mayor cantidad de información escrita. Este método implica un mayor costo en tiempo de procesamiento, pero así mismo una mayor precisión al momento de la elección.

3.3. Mejora de calidad de imagen

En la presente etapa se busca mejorar la calidad de los *frames* manipulados, con el fin de optimizar los recursos que serán utilizados por las herramientas OCR en una etapa posterior. Para esta mejoría se utilizaron dos de los cuatro modelos de súper resolución cubiertos en la sección 2.4, los elegidos fueron EDSR y FSRCNN, ya que mantuvieron comparativamente buenos resultados en ambos casos, escalando x2 y x3, donde EDSR fue él con mejor desempeño y FSRCNN el siguiente mejor.

Para incluir estos modelos se utilizó la librería OpenCV, en donde se incluyen implementaciones para ambos sin necesidad de alguna otra dependencia, solo implica una descarga de estos modelos, ya que por defecto no se incluyen en la instalación de la librería, puesto que los archivos tienen un peso considerable y no cualquier usuario que desee utilizar OpenCV va específicamente por estos modelos.

Luego de descargar los modelos que tienen extensión '.pb' se pueden agrupar en una carpeta con nombre sugerido 'models' y así indicar al proceso esta única ruta, de donde se extraerán todos los modelos. El nombre de los archivos tiene la siguiente estructura '*<model>_x<ratio>.pb*' por lo que es posible descargar otros modelos disponibles y cargarlos entregando los parámetros '*model*' y '*ratio*' correspondientes.

Considerando que esta etapa puede implicar un costo extra en procesamiento y tiempo de ejecución, este mejoramiento se aplica justo antes de pasar a la siguiente etapa de transcripción

En el proceso para seleccionar entre los distintos modelos y ratios disponibles, se han dispuesto los parámetros MODEL y RATIO. A continuación se ejemplifica con un caso diferenciador, en la primera figura se tiene la porción de un *frame* directamente extraído de un video de prueba y luego la misma porción, pero luego de haber pasado por un escalado con el modelo *fsrcnn* con RATIO igual a 2, en ambas imágenes se puede ver una diferencia en la claridad del texto.

Figura 15: Porción de un *frame* extraído de un video de prueba, sin aplicación de modelo de aumento de calidad de imagen.

Figura 16: Porción de un *frame* extraído de un video de prueba, con valores *fsrcnn* y 2, para MODEL y RATIO respectivamente.

3.4. Transcripción y estructuración

Para la transcripción, como herramienta OCR se consideraron Tesseract e EasyOCR, ambas opciones *open-source* comentadas en la sección 2.2, fueron elegidas dado que no limitan el uso para incluirlas como dependencias de manera abierta en una nueva librería, en la decisión también se considera el buen desempeño de ambas confirmado por trabajos mencionados en la misma sección 2.2.

Como se ha hecho para otros casos, ahora también se da la opción al usuario de elegir cuál utilizar dado las correspondientes ventajas de cada uno, según si el contenido sea cargado hacia una mayor cantidad de escritura o una mayor cantidad de dígitos numéricos.

Primero, para definir el objetivo de esta estructuración; esencialmente, se puede resumir en facilitar una posterior etapa de análisis de la transcripción para así poder extraer la información más relevante para el investigador, este objetivo se puede abordar según como vaya a hacerse el posterior análisis, por lo que se tomarán dos caminos:

El primero buscará mantener la información relacionada en bloques, definiendo limitadores para diferenciar diapositivas entre sí, como también para textos dentro de una misma diapositiva, esta separación se basa en la distancia de los objetos detectados, el tipo de objeto varía según el sistema OCR usado, ya sea cada párrafo detectado para EasyOCR o cada palabra detectada para Tesseract, se considerarán como objetos relacionados los cuales tienen una distancia menor o igual a la distancia media en la diapositiva actual, esto con el fin de tomar en consideración párrafos que puedan llegar a tener diferentes tamaños de letra y/o espaciado entre palabras/objetos.

El segundo camino será implementar técnicas de transformaciones del texto como lo son la normalización, *steeming* y la lematización. Para este caso la normalización fue posible aplicarla en conjunto con la lematización, la primera implica la reducción o transformación de caracteres especiales a formas comprensibles por una máquina. Por ejemplo, si se tiene un porcentaje “4 %”, se convertirá en “4/100” y para la lematización se usó el conjunto de herra-

mientas dentro de la librería de Stanza para Python, mencionado en la sección 2.6, con este se obtienen las bases léxicas de cada palabra transcrita anteriormente, esto continuando con el objetivo de transformar los textos transcritos a un lenguaje procesable y/o analizable de manera directa por una máquina.

Para este proceso, se definieron parámetros que permiten seleccionar el sistema OCR a usar en la transcripción y otro parámetro que da la libertad de aplicar o no un proceso de lematización, los nombres de estos son OCR y LEMA respectivamente. A continuación se ejemplifica la transcripción utilizando una diapositiva de prueba, el resultado del proceso de transcripción utilizando EasyOcr y finalmente el resultado de aplicarle la lematización:

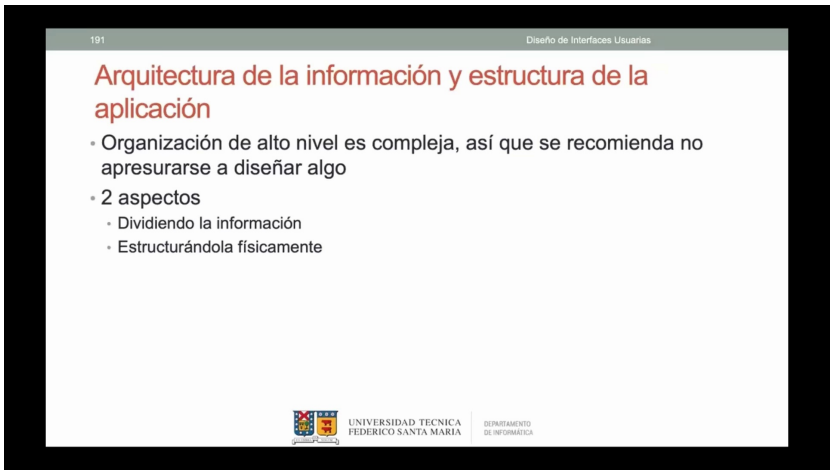
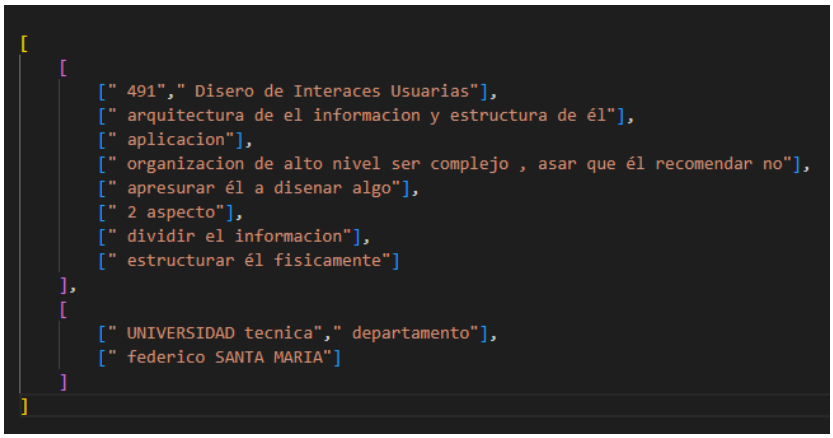


Figura 17: Diapositiva extraída de un video de prueba.


```
[
    [
        ["491", "Disero de Interaces Usuarias" ],
        ["Arquitectura de la informacion y estructura de la"],
        ["aplicacion"],
        ["Organizacion de alto nivel es compleja, asi que se recomienda no"],
        ["apresurarse a disenar algo"],
        ["2 aspectos"],
        ["Dividiendo la informacion"],
        ["Estructurandola fisicamente"]
    ],
    [
        ["UNIVERSIDAD TFCNICA","DEPARTAMENTO"],
        ["FEDERICO SANTA MARIA"]
    ]
]
```


Figura 18: Transcripción de diapositiva en figura 17, utilizando parámetro OCR igual a 1 o EASYOCR.



```
[
    [
        [" 491"," Disero de Interaces Usuarias"],
        [" arquitectura de el informacion y estructura de él"],
        [" aplicacion"],
        [" organizacion de alto nivel ser complejo , asar que él recomendar no"],
        [" apresurar él a disenar algo"],
        [" 2 aspecto"],
        [" dividir el informacion"],
        [" estructurar él fisicamente"]
    ],
    [
        [" UNIVERSIDAD tecnica"," departamento"],
        [" federico SANTA MARIA"]
    ]
]
```


Figura 19: Transcripción lematizada de diapositiva en figura 17, utilizando parámetro OCR igual a 1 o EASYOCR y parámetro LEMA igual a *True*.

3.5. Etapas de limpieza

Esta etapa es considerada en dos instancias del flujo mostrado en la figura 6, las dos mantienen el objetivo de eliminar redundancia sin perder información durante el proceso, aunque varían en el tipo de data que manejan para hacer la limpieza, a continuación se detalla cada una según ambos *inputs*:

1. Eliminación de redundancia sobre imágenes: consiste en una eliminación de *frames* idénticos o con una similitud objetivamente alta, donde se deja como parámetro la similitud límite. La limpieza avanza haciendo comparaciones de calidad de imagen con métricas de calidad entre cada frame y su vecino contiguo, y en el caso de tener dos *frames* considerados similares/iguales se mantendrá solo el último frame ”leído”.

2. Eliminación en base al texto: considera los casos en que el filtro anterior no logró eliminar imágenes que contenían el mismo texto, por esto se buscan coincidencias luego de haber pasado por la transcripción y así comparar directamente en los textos de diapositivas contiguas, que dada la estructuración aplicada en la etapa anterior, es posible reconocer diapositivas según los bloques o niveles de encapsulación de los párrafos. El método usado para decidir si dos textos son similares es una comparación basada en el porcentaje de palabras que coinciden en los textos. El porcentaje límite para esto es definido por el usuario, pero como predeterminado se deja en 90 %, lo que implicaría que al menos el 90 por ciento de los textos coinciden y en el caso de que dos textos cumplan esta condición se elimina el contenido del frame o bloque que contenga menos información.

3.6. Dificultades del proceso

Durante la creación de la librería se presentaron algunos obstáculos, la mayoría de ellos relacionados con las métricas usadas en los procesos que implican comparar o agrupar imágenes. Esto debido al límite intermedio durante la limpieza de la data, al minimizar la redundancia y por el otro lado querer mantener la suficiente data para no perder información importante e irreemplazable. En el proceso de limpieza de *frames* se necesitaba identificar cuáles *frames* tenían una mayor representatividad sobre sus vecinos, considerándose a estos los que contienen la mayor cantidad de información. Inicialmente solo pensando en métricas de calidad de imagen, se limitó las características comparables que validarán que un frame fuera de mayor aporte que otro vecino. Al momento de incorporar el reconocimiento de caracteres es cuando se obtiene acceso a métricas que aportan mayor información de un frame sobre otro, permitiendo comparar el contenido de cada uno. En sí la dificultad fue encontrar las métricas correctas y los límites ideales para que tanto los *frames* eliminados no fueran relevantes o la información de estos ya fuera cubierta por otro frame que se haya mantenido en el conjunto.

Luego de haber disminuido el conjunto de *frames* considerablemente se entró a agrupar los *frames*, para lo cual igual que el caso anterior necesitaba alguna métrica de comparación, en este caso se consideró utilizar el audio del *video pitch* como otro método que incluya otra dimensión fuera de la imagen o *frames*, en el audio se probaron algunas técnicas de eliminación de ruido y suavización de curvas para obtener mínimos en el volumen de la voz y determinar así según pausas los cambios en las diapositivas, pero dado el uso de música de fondo, el abordarlo por este medio se complicó. No teniendo buenos resultados, en esta etapa se consideró solo la imagen donde para variaciones mínimas en los frames se considera como la misma diapositiva, pero para casos donde la variación es suficientemente alta entonces es considerada como un máximo y, por lo tanto, un cambio de diapositiva.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

4.1. Arquitectura

La figura a continuación muestra el diagrama de componentes para el proceso, en el cual se detallan las entradas necesarias para que cada componente ejecute correctamente y las salidas obtenidas del procesamiento interno, las dependencias más relevantes por cada componente y los requerimientos entre componentes. En verde se destacan los componentes encargados de mejorar el proceso base (componentes en blanco) y en amarillo un componente no esencial que modifica el resultado base.

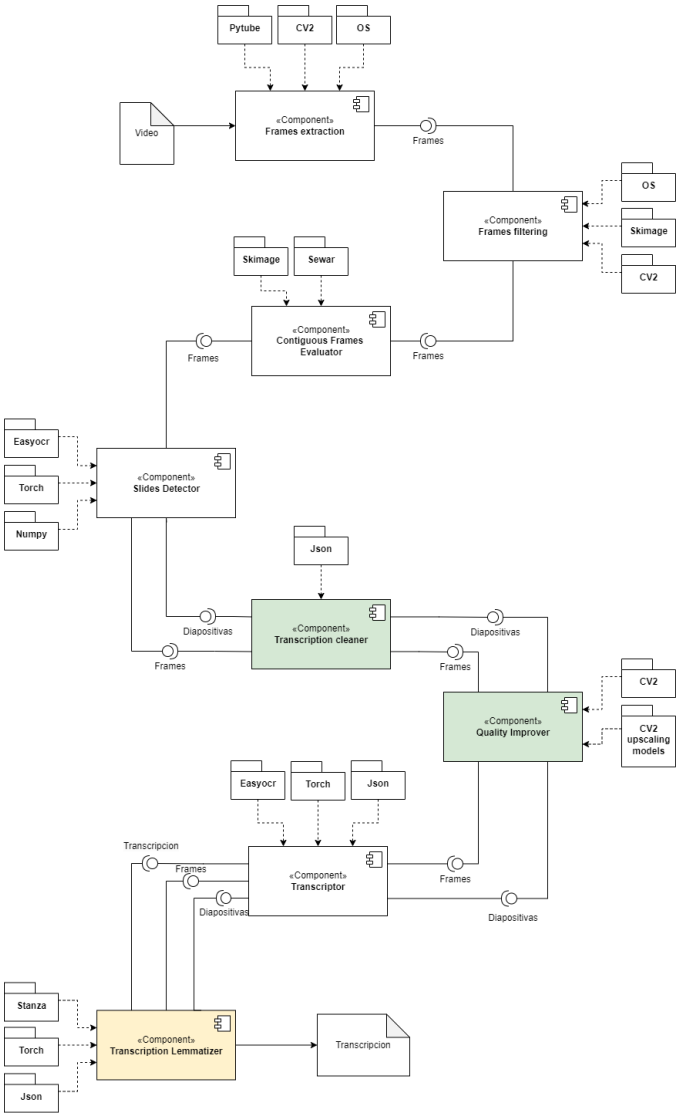


Figura 20: Diagrama de componentes de la solución.

El anterior flujo esquematizado inicia en la parte superior con un video en varios formatos posibles, es procesado por cada componente recibiendo, tanto transformaciones entre diferentes tipos de datos como del contenido del mismo y termina en la base entregando una transcripción del texto contenido en las imágenes del video. Las salidas y entradas que manejan los componentes son solo tres: *frames* o imágenes, diapositivas como grupos de imágenes y transcripciones como texto simple o estructurado. Las entradas y salidas representan las dos formas de manejar la información en el proceso: la información estática o efímera, que incluye listas y variables en tiempo de ejecución, y la información persistente, que se refiere a archivos de imagen y carpetas organizadas en la unidad de almacenamiento de datos.

Las dependencias más relevantes son representadas en paquetes, solo se indican las princi-

pales por cada componente, sobre las cuales se concentra la mayor parte del procesamiento del mismo componente.

4.2. Pruebas y métricas de evaluación

En esta sección se encuentran las pruebas, tanto para ajustar parámetros, como para validar el proceso, aplicando diferentes evaluaciones con el fin de conocer la calidad de los resultados; los casos o variaciones de parámetros a comparar son:

1. **GPU**: uso de la GPU y CPU versus solo haciendo uso de la CPU (*True*: CPU + GPU y *False*: solo CPU)
2. **RGB**: variación en la lectura de imágenes, una banda: blanco y negro o tres: RGB (*True*: RGB y *False*: B/W)
3. **COEF**: diferentes valores para los límites de las métricas de calidad de imagen (rango de valores: [0, 0.8, ..., 0.97625, 1])
4. **JUMPS**: número de *frames* evitados por cada *frame* leído (1, 2 y 3)
5. **RUNTIME**: manejo de data de forma persistente versus manejo en tiempo de ejecución. (*True*: en ejecución y *False*: memoria persistente)
6. **LEMA**: aplicación o no de lematización al texto transcrito. (*True*: con lematización y *False*: sin lematización)
7. **METRICA**: tipo de métrica utilizada (SSIM, dif, MSE, PSNR y UQI)
8. **OCR**: tipo de herramienta de reconocimiento de caracteres (1:EASYOCR, 2:TESSERACT)
9. **MODEL**: modelo de aumento de escala de imagen (fsrcnn y edsr)
10. **RATIO**: corresponde al multiplicador usado por el modelo anterior (2 y 3)

Las métricas usadas para comparar ordenadas por prioridad de mayor a menor relevancia son las siguientes:

1. **Similitud coseno usando Tf-idf (SCTF-idf)** toma valores entre 0 y 1, con 0 para transcripciones que no poseen ninguna similitud al texto real y 1 para una transcripción que coincide exactamente con la transcripción real.
2. **Porcentaje de palabras reconocidas por video (PRPV)** al igual que la métrica anterior, toma valores entre 0 y 1, con 0 para transcripciones que no poseen ninguna similitud al texto real, mientras que un valor de 1 indica que todas las palabras de la transcripción coinciden con el texto real.

3. **Porcentaje de perdida de diapositivas** mide el porcentaje de los casos en que el número estimado de diapositivas es menor al número real, causando que exista una perdida de información en la transcripción final, por esto mientras menor sea este porcentaje mejor es el desempeño en la etapa de agrupación de *frames*.

4. **Velocidad de ejecución**, medida en segundos y expresa el tiempo tomado al ejecutar el proceso completo para un video.

5. ***Normalized Mean absolute error* (NMAE)** corresponde al promedio de la diferencia absoluta entre el número estimado de diapositivas y el número real de diapositivas, dividido en el promedio del número real de diapositivas, el cual busca identificar en que porcentaje se aleja el estimado del valor real sin importar el signo y, por lo tanto, incluyendo sobre-estimaciones y subestimaciones del valor, el cual esta normalizado en los positivos y para 0 se indica que los valores estimados son exactamente el número real de diapositivas y otro factor indica el porcentaje de desviación respecto al valor real.

4.2.1. Descripción de las muestras

Para el proceso de pruebas se tendrán dos conjuntos de muestras, el primero se conforma de diez videos con tiempos entre 43 segundos y 5 minutos con 44 segundos, con contextos de emprendimientos. El segundo conjunto de 28 videos, un compilado de 5 clases en modalidad *online* con diapositivas en el fondo, los cuales fueron recortados/divididos para poder obtener el total de 28 videos, los largos de los extractos de video van entre los 2 y 6 minutos.

De los conjuntos anteriores se obtiene el 30 % de cada uno para formar el conjunto de "entrenamiento", en total 11 videos que serán usados para ajustar las variables anteriormente descritas, este ajuste con el objetivo de optimizar los valores según la priorización definida y el resto 70 % o 27 videos será usado para el testeo.

Este enfoque de asignar aproximadamente el 30 % de la muestra al entrenamiento y el 70 % a la evaluación, difiere de las proporciones utilizadas comúnmente. Sin embargo, se elige esta distribución debido a la reducida cantidad de datos generados en total. El objetivo es priorizar una evaluación más realista y generalizada de la precisión del proceso, evitando el sobre ajuste a la muestra utilizada.

Y al tener una muestra mayor para el entrenamiento se tendería a reducir la variabilidad de los resultados ya que no habría una gran dependencia sobre los videos elegidos aleatoriamente. Además, esto nos permitiría considerar un caso desfavorable para el proceso, ya que se limita el aprendizaje previo sobre la muestra.Finalmente teniendo esta base, con aumentar el porcentaje de entrenamiento se podría mejorar la calidad de los resultados.

Algunas consideraciones previas para regular las pruebas:

- Se considera **diapositiva** al *frame* que contiene información completa, con esto refiriéndose a que las palabras contenidas están claras y legibles (sin difuminación, ni cortes por transiciones u otro ruido en el texto).
- Para el **conteo de diapositivas** completas se consideran solo los *frames* que cumplan la definición anterior y que tengan texto distinto a la diapositiva antecesora y la predecesora.
- En el caso que para dos diapositivas contiguas el texto de una sea contenido en otra, se considerara solo la que tenga mayor información.

Para la comparación final se obtuvo un punto de referencia para cada video, el cual incluye el conteo de las diapositivas reales y también la transcripción de estas diapositivas, siendo ambos procesos realizados manualmente para cada video.

Para evitar variaciones por causas distintas a las de interés durante las pruebas, los parámetros que no sean testeados se mantendrán fijos en los siguientes valores:

- GPU = *True*
- JUMPS = 1
- RGB = *True*
- RUNTIME = *False*
- LEMA = *False*
- MODELO = *False sin modelo aplicado inicialmente*
- RATIO = 0, *sin modelo aplicado inicialmente, por lo tanto, sin ratio*
- METRICA = ssim
- OCR = 1, *EasyOCR*
- COEF = 1

4.2.2. Resultados pruebas

A continuación se muestra un resumen de cada prueba y se especifican los valores de cada parámetro con mejor desempeño según las métricas y para el caso del *porcentaje de perdida de diapositivas* solo se exponen los datos de parámetros que afectan la agrupación de *frames*. Algunas consideraciones usadas para discernir casos sin un claro ganador son: la métrica SCTF-idf tiene mayor peso versus la PRPV, esta diferencia hecha dado el mayor ajuste hecho con base en el aporte o relevancia de cada palabra.

Definición GPU: Dada la notable mejoría en eficiencia en la ejecución del proceso, se asigna *True* al parámetro GPU.

GPU	Tiempo promedio	Tiempo medio
TRUE	898.1	945.0
FALSE	2762.1	3071.6

Tabla 4: Promedio y media del tiempo en segundos de ejecución para diferentes valores del parámetro GPU

Definición RGB: Para este parámetro se elige *RGB* igual a *False*, dado que se obtuvieron mejores resultados con este valor tanto para las métricas de similitud de texto como para el valor de NMAE.

RGB	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
TRUE	0.597	0.757	0.649	0.782
FALSE	0.600	0.755	0.650	0.781

Tabla 5: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro RGB

RGB	Porcentaje de perdida	NMAE
TRUE	0 %	3.94
FALSE	0 %	3.62

Tabla 6: Porcentaje de perdida de diapositivas y NMAE para valores del parámetro RGB

Definición OCR: Para este parámetro, se ha elegido el módulo EasyOCR debido a su mejor desempeño en ambas métricas de similitud de imagen. Sin embargo, es importante tener en cuenta que la diferencia no es significativa, por lo que la elección podría variar según la estructura de la muestra. Como se mencionó en la comparación de ambos módulos en la sección 2.2, cada uno tiene sus fortalezas en diferentes características de los textos evaluados.

OCRs	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
Easyocr	0.597	0.757	0.649	0.782
Tesseract	0.583	0.555	0.637	0.573

Tabla 7: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro OCR

Definición Modelo de aumento de escala: Desde el inicio se descarta el modelo EDSR debido a su excesivo tiempo de ejecución, el cual supera más de 15 veces el tiempo obtenido por el modelo FSRCNN, como se muestra en la tabla 8.

En cuanto al parámetro RATIO, se ha elegido el valor 2 debido a su mejor desempeño en los resultados. Aunque no muestra una diferencia significativa en comparación con el valor 3. Esta elección también se ve respaldada por los tiempos de ejecución mostrados en la tabla 10, donde se observa que el valor 2 logra tiempos menores, aunque de igual forma con una diferencia mínima.

MODEL	RATIO	Tiempo promedio	Tiempo medio
fsrcnn	2	241.5	214.4
fsrcnn	3	260.0	235.9
edsr	2	3999.2	2745.3
edsr	3	3997.2	2744.8

Tabla 8: Promedio y media en segundos del tiempo de ejecución para diferentes valores del parámetro RATIO para los modelos FSRCNN y EDSR, solo considerando el 30 por ciento de la primera muestra de 10 videos

MODELS	RATIO	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
fsrcnn	2	0.598	0.781	0.636	0.791
fsrcnn	3	0.596	0.776	0.645	0.788

Tabla 9: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro RATIO para el modelo fsrcnn

MODELS	RATIO	Tiempo promedio	Tiempo medio
fsrcnn	2	1107.6	1134.1
fsrcnn	3	1105.6	1197.5

Tabla 10: Promedio y media en segundos del tiempo de ejecución para diferentes valores del parámetro RATIO para el modelo fsrcnn

Definición Métrica: Para este parámetro, se ha elegido la métrica UQI (Índice de Calidad Universal) principalmente debido a que muestra resultados ligeramente superiores en los valores de SCTF-idf en comparación con otras métricas. Además, UQI genera una pérdida del 0 % y tiene valores de NMAE (Error Absoluto Medio Normalizado) dentro de un mismo rango en comparación con las demás métricas, exceptuando el caso de PSNR.

En cuanto a PSNR, aunque tiene los mejores valores de NMAE, obtuvo los peores resultados en la similitud de texto y en el porcentaje de pérdida. Por otro lado, una métrica que se acerca mucho a UQI en términos de rendimiento es SSIM (Índice de Similitud Estructural). SSIM se posiciona como el segundo mejor en cuanto a similitud, con una pérdida del 0 % y solo un 14 % de diferencia en NMAE en comparación con UQI.

METRICAS	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
SSIM	0.597	0.757	0.649	0.782
dif	0.586	0.736	0.628	0.764
mse	0.587	0.746	0.637	0.775
psnr	0.334	0.131	0.330	0.130
uqi	0.601	0.751	0.653	0.777

Tabla 11: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro MÉTRICAS

METRICA	Porcentaje de perdida	NMAE
SSIM	0 %	3.94
dif	2 %	3.76
mse	0 %	3.80
psnr	83 %	0.86
uqi	0 %	3.80

Tabla 12: Porcentaje de perdida de diapositivas y NMAE para valores del parámetro METRICA

Definición Saltos: Para este parámetro, se ha elegido un solo salto, lo que implica considerar cada frame capturado. Esta opción muestra una mejora leve en comparación con las otras dos opciones, pero también resulta en un aumento en el valor de NMAE. Esto representa un "*trade-off*"que debe considerarse según el tipo de datos que tenga el usuario y el contexto de implementación. Ya que si se cuenta con una muestra que incluye videos con numerosas diapositivas en cada uno, utilizar JUMPS igual a 1 implicaría manejar aproximadamente el doble de datos en comparación con usar JUMPS igual a 3.

JUMPS	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
1	0.597	0.757	0.649	0.782
2	0.586	0.740	0.651	0.777
3	0.574	0.722	0.639	0.742

Tabla 13: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro JUMPS

JUMPS	Porcentaje de perdida	NMAE
1	0 %	3.94
2	0 %	2.72
3	0 %	1.67

Tabla 14: Porcentaje de perdida de diapositivas y NMAE para valores del parámetro JUMPS

Definición Lema: Se realizó una comparación entre las transcripciones obtenidas al aplicar lematización, el parámetro *LEMA* establecido en *True* y las transcripciones manuales previamente lematizadas. En el caso de las transcripciones sin lematización, se compararon directamente con las transcripciones manuales sin lematizar. Se encontró diferencias de aproximadamente 3 % en las métricas de similitud. Caso particular, ya que aunque esta mejora puede ser mínima, la lematización también ofrece la ventaja de adaptar el lenguaje para su posterior análisis. Es por esto que se selecciona aplicar la lematización en el caso final.

LEMA	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
TRUE	0.626	0.785	0.643203	0.819
FALSE	0.597	0.757	0.649	0.782

Tabla 15: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro LEMA

Definición coeficientes: Para esta elección se hacen dos iteraciones de pruebas, en la primera se toman coeficientes equidistantes entre 0 y 1, incluyendo los límites. De la cual se encuentra 1 como el mejor coeficiente tanto para la similitud como para el porcentaje de perdida. Ya que se encuentra un aumento en la similitud y también un descenso considerable en el porcentaje de perdida en valores cercanos a 0.905, se definen nuevos coeficientes entre los valores contiguos, un nivel inferior y el máximo, 0.81 y 1 respectivamente, sin incluirlos, porque fueron incluidos en esta primera prueba.

COEFS	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
0	0.315	0.125	0.314	0.130
0.08	0.315	0.125	0.314	0.130
0.16	0.320	0.127	0.314	0.130
0.245	0.318	0.130	0.314	0.130
0.33	0.317	0.134	0.314	0.130
0.41	0.323	0.141	0.314	0.130
0.49	0.341	0.152	0.314	0.143
0.57	0.348	0.159	0.354	0.143
0.65	0.370	0.208	0.354	0.149
0.73	0.431	0.258	0.396	0.208
0.81	0.566	0.505	0.601	0.521
0.905	0.609	0.685	0.585	0.718
1	0.597	0.757	0.649	0.782

Tabla 16: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro COEFS

COEF	Porcentaje de perdida	NMAE
0	83 %	0.86
0.08	83 %	0.86
0.16	83 %	0.86
0.245	82 %	0.85
0.33	80 %	0.83
0.41	79 %	0.82
0.49	78 %	0.81
0.57	76 %	0.80
0.65	71 %	0.78
0.73	63 %	0.71
0.81	32 %	0.36
0.905	4 %	0.22
1	0 %	3.94

Tabla 17: Porcentaje de perdida de diapositivas y NMAE para valores del parámetro COEFS

Para la segunda elección se tiene la siguiente tabla con los valores de similitud para cada coeficiente y luego el gráfico 21 con la similitud para cada valor usado como coeficiente, de estos se consideraron los mejores resultados señalados en verde en la tabla 18 y los dos máximos valores destacados en el gráfico. Por lo tanto, en esta segunda iteración se eligen dos coeficientes: 0.8575 y 1.

COEFS	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
0.81	0.566	0.505	0.601	0.521
0.83375	0.593	0.584	0.638	0.615
0.8575	0.603	0.635	0.659	0.663
0.88125	0.609	0.675	0.590	0.699
0.905	0.609	0.685	0.585	0.718
0.92875	0.609	0.695	0.585	0.726
0.9525	0.608	0.701	0.585	0.726
0.97625	0.604	0.706	0.585	0.726
1	0.597	0.757	0.649	0.782

Tabla 18: Promedio y media para ambas métricas de similitud de texto para diferentes valores del parámetro COEFS, para la segunda iteración de coeficientes.

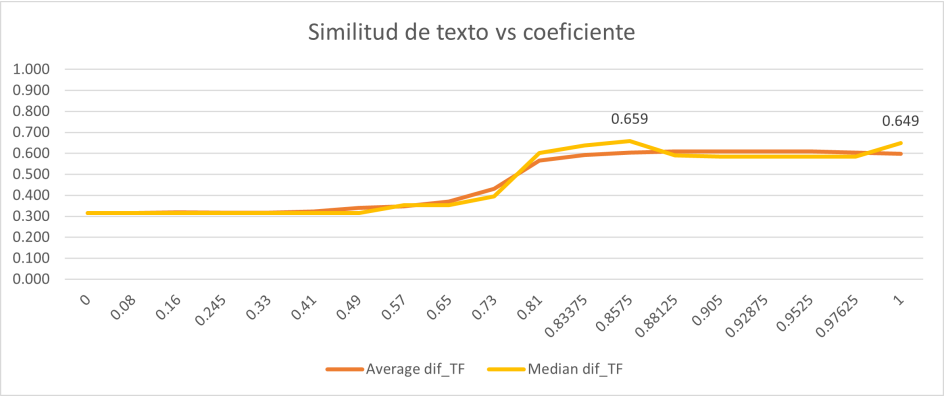


Figura 21: Gráfico de similitud de texto versus valores del parámetro coeficiente

A continuación se presenta el listado de parámetros y los valores definidos para cada uno durante las pruebas hechas con el primer conjunto de videos:

- GPU = *True*
- RGB = *False*
- OCR = EasyOCR
- MODELO = FSRCNN
- RATIO = 2
- METRICA = SSIM
- JUMPS = 1
- LEMA = *True*
- COEF = 0.8575 y 1.

Enseguida se presentan los resultados de la ejecución completa del proceso utilizando la muestra de evaluación y los valores asignados para cada parámetro, considerando ambas opciones de COEF.

En relación al último parámetro, se observa un desempeño notablemente mejor utilizando el valor 1, lo cual se confirma al analizar los resultados de similitud de texto. Además, al evaluar el porcentaje de pérdida, se observa que el valor 0.8575 genera una pérdida considerablemente mayor de diapositivas, aproximadamente 30 veces mayor. A pesar de que el valor NMAE es más del doble en comparación con el coeficiente 0.8575, se prioriza evitar la pérdida de información que no podrá ser recuperada en futuros análisis o procesamientos.

Considerando los resultados mencionados, se ha seleccionado el valor 1 como el parámetro COEF para su implementación.

COEF	SCTF-idf promedio	PRPV promedio	SCTF-idf media	PRPV media
0.8575	0.516	0.522	0.511	0.572
1	0.610	0.761	0.619	0.794

Tabla 19: Promedio y media para ambas métricas de similitud de texto para los valores seleccionados de cada parámetro, incluyendo ambos valores de COEFS

COEF	Porcentaje de perdida	NMAE
0.8575	29 %	0.88
1	1 %	3.13

Tabla 20: Porcentaje de perdida de diapositivas y NMAE para la prueba final comparando parámetro COEF

4.2.3. Especificaciones equipo usado para pruebas

Procesador	Intel(R) Core(TM) i5-7400 CPU @ 3.00GHz
RAM	16.0 GB
GPU 0	NVIDIA GeForce GTX 1060 6GB
Driver version:	31.0.15.1640

CAPÍTULO 5
CONCLUSIONES

5.1. Conclusiones generales

En este trabajo se logró desarrollar un proceso de libre acceso, disponible como librería para el lenguaje Python. Una librería con la cual es posible tomar un video y entregar una transcripción estructurada de las diapositivas usadas para exponer una presentación de un emprendimiento en formato *video pitch*. En este software se habilitan diferentes métodos para mejorar la calidad de la transcripción, en el reconocimiento del texto, aplicando modelos de escalado de imagen para mejorar el *input* usado en la transcripción, y en la eliminación de redundancia gracias a diferentes etapas de limpieza aplicada tanto en los *frames* extraídos como directamente en la transcripción obtenida. En el proceso se da la libertad de elegir ciertos parámetros que permitirían ajustar el procesamiento a variadas características de los datos entregados.

Considerando que el desarrollo de este trabajo fue impulsado con el fin de mejorar las transcripciones directamente tomadas de los videos y evitar perder información en instancias donde la principal fuente de información es visual. Se definieron objetivos que abordaran esta necesidad y sus mejoras; el primero fue obtener información directa desde el video, para abordarlo se planteó hacerlo usando servicios/APIs de Google, pero dadas las restricciones de su uso y la alta dependencia que generaría en el resultado se terminó decidiendo alcanzar este primer objetivo con librerías *open-source* tales como EasyOCR y Tesseract las que dieron buenos resultados, aunque la primera dando un mejor desempeño.

El segundo objetivo; limpiar la información, minimizando lo irrelevante como son los casos de duplicación de información, dado que varios *frames* cubren una misma diapositiva. Como se mencionó en las secciones 3.2 y 3.3 se minimizaron estos casos, inicialmente agrupando las diapositivas para definir límites, donde se está tomando la misma información, para luego dentro de cada uno obtener el *frame* más completo, para la agrupación se analizaron dos opciones: agrupar por similitud de imágenes y agrupar por variaciones en el audio, de estas se eligió la primera dado que mostraba diferencias más notorias para los cambios de diapositivas. Como segunda limpieza se hicieron eliminaciones por redundancia sobre las imágenes y luego sobre los párrafos ya transcritos desde las imágenes.

El último objetivo, la estructuración, se abordó durante el reconocimiento de bloques de texto gracias a los datos de posición obtenidos desde las librerías OCR, se agruparon oraciones y párrafos, permitiendo generar una jerarquía en la transcripción, haciéndose visible esta agrupación por medio de listas de listas en los archivos de salida en formato json.

Para las pruebas se debe considerar que los resultados tanto para similitud y tiempo variarán según la data y el sistema utilizado. A pesar de esto, los resultados obtenidos tienen buena

representatividad, ya que en la muestra se incluyeron *videos pitch* reales y los videos de las clases mantienen la estructura de una presentación audiovisual. También se debe tener presente que al no encontrar un sistema que aborde la transcripción de un video de inicio a fin con el cual se puedan hacer comparaciones, se decidió aplicar la transcripción de manera manual, teniendo esta intrínsecamente un error humano. La comparación final realizada se limita a analizar las transcripciones completas en lugar de comparar diapositivas individuales. Sin embargo, al comparar directamente una diapositiva con su transcripción manual y la transcripción generada por el proceso, se podrían haber obtenido más detalles para analizar las diferencias. Es importante tener en cuenta que no fue posible realizar una comparación 1 a 1 debido a que no se puede garantizar tener el número exacto de diapositivas transcritas con el proceso, ya que en algunos casos el proceso puede detectar más o menos diapositivas de las que realmente hay.

De los resultados obtenidos en la selección de parámetros se remarcan algunos valores esperados, como el caso de la GPU, notables mejores tiempos habilitando su uso y el caso del parámetro JUMPS que tiene mejores resultados en la medida que se leen un mayor número de *frames*, aunque puede considerarse como un parámetro a transar en caso de requerir mejores resultados en la etapa de agrupamiento de diapositivas, ya para valores mayores a 1, donde aumenta los frames saltados, mejora la aproximación del número estimado de diapositivas.

Los resultados a resaltar son: la gran cercanía de los resultados para las dos opciones de sistemas OCR, lo cual puede dar pie a buscar los mejores aspectos de cada implementación y generar un OCR híbrido mezclando ambos módulos. En el caso de las métricas, resaltar la cercanía en los resultados de similitud de textos y porcentaje de perdida, exceptuando la métrica PSNR, la cual teniendo su fuerte con el ruido blanco podría implicar baja presencia de este tipo de interferencia en las imágenes. Además, para el parámetro COEF los resultados expuestos implican que es preferente no aplicar el límite que definía el coeficiente en la etapa de división de diapositivas, dado que los mejores resultados se obtienen al tomar el valor 1, caso que considera todos los mínimos, por ende manteniendo todas las divisiones encontradas y definiendo mayores cantidades de diapositivas.

Finalmente, al evaluar los resultados obtenidos durante la fase de pruebas y validación, utilizando los parámetros seleccionados después de las pruebas iniciales y una muestra de 27 videos, se observó un mínimo porcentaje de pérdida, indicando que en promedio se subestimó en un 1 % la cantidad de diapositivas presentes en los videos procesados. Al usar la media de SCTF-idf como la métrica más robusta, como se explicó en la sección 2.5, se logró alcanzar una **precisión aproximada del 60 %** al comparar las transcripciones con el texto real en cada video.

5.2. Desarrollos posibles

Durante el desarrollo de este trabajo se dieron una serie de caminos por donde se pudo haber explorado y así quizás mejorar procesos dentro de las etapas expuestas en la propuesta de solución, algunas son:

1. La profundización del reconocimiento de pausas en el audio como indicador de cambios en diapositivas probando con la extracción o aislado de la frecuencia de la voz, lo cual pudo aportar en obtener más certeras divisiones entre las diapositivas.
2. Utilizar el análisis de contexto para identificar las separaciones dentro o entre las diapositivas, especialmente cuando hay párrafos espacialmente cercanos pero con distintos tópicos.
3. La estructuración de transcripción elegida es buena cuando se consideran párrafos de texto, pero para esquemas y relaciones de contenido con símbolos como flechas, líneas punteadas u otros conectores gráficos no es posible extraer tales dependencias, por lo que ahondar en un método de incorporar esta información a la transcripción final se considera como un buen punto para futuros trabajos.
4. En la selección de *frame* representante: considerar la posibilidad de tener más de un máximo, ya que pueden ocurrir cambios dentro de una misma diapositiva, como la eliminación o adición de nuevos párrafos.
5. La creación de un sistema OCR híbrido que considere casos de mejor desempeño de cada OCR, como es el caso de reconocimiento de números o caracteres especiales.

5.3. Palabras finales autor

Durante el proceso de investigación e implementación, se profundizó en diversas temáticas relacionadas con el manejo de videos, imágenes y textos. Se destacó la amplia variedad de herramientas disponibles para manipular cada uno de estos tipos de datos, permitiendo extraer nuevos conocimientos al procesar la información, aplicar transformaciones y realizar comparaciones con el fin de obtener nueva información relevante en un determinado contexto.

Adquirí un valioso aprendizaje al explorar las metodologías utilizadas para manipular datos, así como las técnicas y herramientas empleadas para mejorar y transformar la información según las necesidades específicas. En mi investigación, me basé en el trabajo de otros investigadores y desarrolladores, quienes han perfeccionado estas herramientas e innovado con nuevas ideas para abordar diversas problemáticas. Gracias a su dedicación y avances, se han abierto nuevos horizontes para posibles desarrollos, tal como ha sido el caso de mi

propio trabajo. Este proyecto se ha beneficiado enormemente de todos los trabajos previos que fueron referenciados en este documento.

Es evidente que los avances en cada campo de estudio progresan rápidamente, lo cual resulta aún más notable gracias a los aportes *open-source*. Estos aportes permitieron establecer una base sólida para el desarrollo de este trabajo, beneficiándose de la colaboración y el acceso a recursos compartidos por la comunidad.

ANEXOS

5.1. Descripción clase principal librería : Video

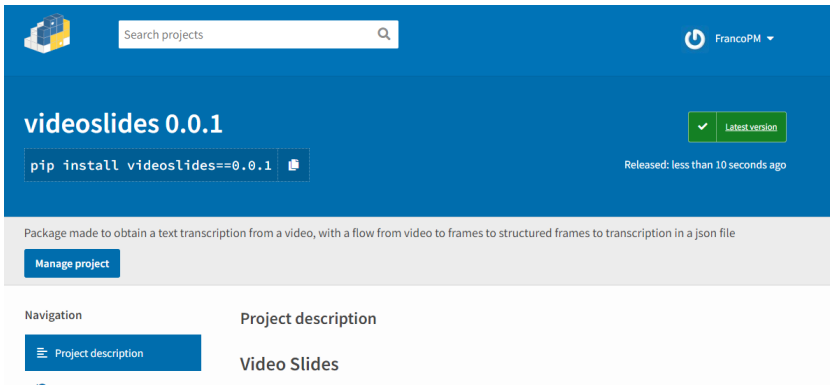


Figura 22: Pagina web de PYPI donde se muestra el paquete creado “VideoSlides”.

Esta clase permite manipular un video, sus *frames* y la transcripción de estos mediante el conjunto de funciones descritas más adelante. Inicialmente, para la creación de esta clase se permite la inserción de un video, ya sea desde una ruta local así como de un url de un video de Youtube, también se tiene como opcional entregar valores para los parámetros: *scale*, porcentaje al cual se quieren reducir los *frames* del video, jumps: corresponde a un indicador de la frecuencia con la que se tomara un *frame* del video en relación con los fps de este (1: se toma un *frame* cada vez que pasen *fps frames* en medio) y finalmente con *runtime* se tiene como opción definir la forma en que se desea manejar la data durante la utilización de esta clase (*True* como predeterminado, indica que el video y los *frames* se mantendrán solo en tiempo de ejecución, *False* el caso contrario indica que se creara una carpeta en la ruta local donde se mantendrán los *frames* del video)

5.1.1. Atributos

A continuación se muestra el listado de atributos que pueden ser obtenidos mediante métodos *getters* con formato: “*video.get_<atributo>*” o directamente con “*video.<atributo>*”, siendo “*video*” un objeto de la clase Video.

1. *path* = ruta del video guardada localmente.
2. *video_name* = nombre del video, obtenido directamente de Youtube o del nombre del archivo “.mp4”.

3. *video_cap* = corresponde a un objeto de clase *<cv2.VideoCapture>* que contiene el video "entregado".
4. *num_frames* = número de *frames* que contiene el video.
5. *fps* = tasa de *frames* por segundo del video.
6. *dim* = dimensiones de los *frames* (*width*, *height*) contenidos en el video luego de ser escalados según el valor de *scale*.
7. *frames* = en caso de runtime igual a *False* se tiene la ruta local de la carpeta que contiene los *frames*, caso contrario una lista con objetos de tipo *<numpy.ndarray>* que contienen la información de los *frames* extraídos.
8. *data* = inicialmente una lista vacía, la cual luego de ejecutar el método *set_data()* se agregan las diferencia entre *frames* ordenadas por *frames* contiguos.
9. *slides* = inicialmente una lista vacía, que con el método *set_slides()* divide los *frames* y obtiene los que contienen la mayor parte de la información de cada diapositiva dentro del video.
10. *transcription* = inicialmente una lista vacía, que luego de ejecutar el método *set_transcription()* contiene el texto obtenido mediante OCR de las diapositivas finales guardadas en el atributo *slides*.

5.1.2. Métodos

1. *set_data* = agrega al atributo *data* las diferencia entre *frames* ordenadas según la secuencia del video.
2. *set_slides* = divide los *frames* y obtiene los que contienen la mayor parte de la información de cada diapositiva dentro del video y los guarda en el atributo *slides* como una lista.
3. *set_transcription* = método que itera sobre los *frames*/imágenes transcribiéndolas usando EasyOCR o Tesseract y estructura el resultado según el orden de lectura occidental, esta se guarda en el atributo *transcription*.
4. *clean_frames* = se encarga de comparar los *frames* de forma ordenada, para el caso con *runtime* igual a *True* mantiene en el atributo *Frames* solo los que se consideren distintos a su *frame* anterior, para el caso *runtime* igual a *False* se elimina de la ruta en *frames* los que sean considerados igual a su *frame* contiguo.
5. *clean_transc* = aplica la limpieza sobre lo contenido en la transcripción actual en *transcription* ya sea un *string* on un archivo *json*, al igual que el caso anterior para *runtime* igual a *True* se limpia desde esta misma variable y en el caso contrario se edita el archivo de la ruta *transcription* exportado anteriormente en el directorio.

6. *improve_quality* = ejecuta el *model* y *ratio* indicado sobre cada imagen contenida en *frames* ya sea una lista estática o archivos persistentes en el disco.

5.2. Ejemplos de uso y resultados

A continuación se muestra un código de ejemplo donde se ejecuta un flujo de uso para el caso de iniciar con un *link* de Youtube, se mantiene las dimensiones originales de los *frames* (*scale* = 100) y se obtiene un frame cada *fps frames* leídos (*jumps* = 1), se usara código de colores blanco y negro (*rgb* = *False*), todos estos parámetros no siendo necesario explicitarlos por ser los valores por defecto y solo se indica mantener las salidas en forma persistente (*runtime* = *False*).

```
import videoslides as vs

link = "https://youtu.be/XD5PYgxGLag"

# 1.- Crear clase de Video
video = vs.Video(link, runtime = False)

# 2.- Filtra los frames
video.clean_frames()

# 3.- Obtiene el conjunto de valores comparando frames contiguos
video.set_data()

# 4.- Obtiene las slides y de estas el conjunto de los frames con la
# mayor parte de la info.
video.set_slides()

# 5.- Obtiene la transcripcion estructurada de las slides.
video.set_transcription()

# 5.- Obtiene la transcripcion estructurada de las slides.
video.set_transcription()

print(video.data)
# [0.62046652 0.76658127 0.98855191 0.99628632 0.90726049 0.66280858
#  0.98182305 0.98622411 0.80762703 0.69132001 0.95423874 0.984781
#  0.86919341 0.7472609  0.91355526 0.40584704 0.89340392 0.98795554
#  0.99709984 0.99015406]
```

```
print(video.slides)
# [0, 5, 9, 13, 15, 20]

print(video.transcription)
# [
# [[[' ENCUENTRALO'], [' ayudar en caso de'], [' perdido de pertenencia'],
#             [' valioso']], [' 85.0096'], [' oo058']],
# [[[' todo Chile'], [' Arica'], [' Magallanes'], [' Santiago'], [' Longavi'],
#              [' Punta Arenas']]],
# [],
# [[' Plbjecf'], [[' introduccion de'], [' nuestro servicio en'],
#              [' evento y Ferias'], [' de Software'],
#              [' conocer alli , fecha en nuestro red social']]],
# [[[' nuestro Equipo'], [' el Team que apoyar tu busqueda']],
#              [[' Antonio Harrison', ' Camila Gibson'], [' CoO',
#               ' Marketing Head']]],
# [[[' nuestro Equipo'], [' el Team que apoyar tu busqueda']],
#             [[' Paulina Bradshaw', ' Marco Bromman'], [' Marketing Expert',
#               ' Social Media Manager']]]
# ]
```

Listing 1: Código de prueba ejemplificando el uso de la librería “VideoSlides”.

Salidas del proceso para el caso *runtime* igual a *False*:

1. Video = desde el *link* se descarga el video en formato mp4 para luego de este obtener los *frames* (para este caso de nombre Caso3.mp4)
2. *Frames* = carpeta creada por el proceso, donde se almacena cada *frame* extraído. Del ejemplo anterior la carpeta toma el nombre “F_Caso3”
3. Transcripción = este archivo de formato *json* contiene la lista estructurada luego de haber ejecutado “*set_transcription()*”

5.3. Descripción funciones

A continuación se presentan el conjunto de funciones que construyen la librería creada para este trabajo, podrán usarse directamente desde los métodos o indirectamente siendo usadas a través de otras funciones. Para cada función se indica una descripción, los *inputs* que puede tomar junto al tipo de estructura y el Output en caso de corresponder con su tipo.

	download_video
Descripción	Descarga un video de Youtube
Input	url (str): link del video de Youtube
Output	(boolean): True para descarga exitosa y en caso contrario False (str): string con el nombre del video descargado, en caso fallido string vacio

	getqua
Descripción	Función que compara dos frames con la métrica que indica el parametro ”me” 1:SSIM, 2:dif, 3:mse, 4:psnr, 5:uqi
Input	frame1 (stro o numpy.ndarray): ruta o información frame frame2 (str o numpy.ndarray): ruta o información frame rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) me (int): métrica a usar para comparar los frames
Output	(float): valor de evaluación obtenido con métrica elegida

	getdata
Descripción	Función que usando getqua() en frames ordenados entrega un array con los valores de similitud de cada par de frames contiguos
Input	frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 me (int): métrica a usar para comparar los frames rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False)
Output	data (list): array ordenado con números enteros obtenidos evaluando frames contiguos

	localmin
Descripción	Función que obtiene los mínimos locales de la data entregada
Input	data (list): array ordenado con números enteros 1D coef (int): factor limitador sobre el número de mínimos considerados
Output	counts[1] (int): número de mínimos locales encontrados pos (list): posiciones correspondiente a los mínimos locales dentro del array data

	dist_2p
Descripción	Obtienen la distancia euclidiana entre dos puntos
Input	pos1 (tuple(int,int)): valores en eje x e y de punto 1 pos2 (tuple(int,int)): valores en eje x e y de punto 2
Output	(float): distancia

	min_dis_sq
Descripción	Función que entrega la distancia entre dos cuadrados, asumiendo diferentes casos reduciendo el cálculo a distancia entre dos puntos
Input	pos1 (arrays(int,int)): valores en eje x e y de punto 1 pos2 (arrays(int,int)): valores en eje x e y de punto 2
Output	(float): distancia

	easy
Descripción	Función que: - Obtiene una transcripción de una imagen y las posiciones de cada bloque de texto - Dadas las posiciones calcula las distancias entre ellos - Con las distancias estructura las transcripciones en orden de lectura occidental (arriba hacia abajo e izquierda a derecha)
Input	reader (class): clase easyocr.easyocr.Reader usada para la transcripción de frames frames (str): ruta a carpeta de frames o imagen (numpy array) detail (str): nombre de la extension de la carpeta de bloques rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) ocr (int): indicador de que OCR es usado para obtner la transcripción, 1 = easyOCR o 2 = teseract debugg (boolean): True ->grafica sobre la imagen los bloques de texto reconocidos
Output	order (list): lista con la transcripción estructurada

	deep_index
Descripción	Función que entrega los índices de puntos a los cuales corresponde la distancia indicada en word, dentro de la lista triangular distance (no flatten)
Input	distance (list(lists(int))): lista de listas con distancias entre bloques de texto, (estructura triangular: [a distancia con b, c, d, e] [b distancia con c, d, e] ...) word (str): palabra/número a indexar en la lista distance
Output	l[0] (tuple(int, int)): índices de puntos a los cuales corresponde la distancia indicada en word

	clustering
Descripción	Función forma grupos según distancias entregadas
Input	array2 (list(lists(int))): lista de listas con distancias entre bloques de texto, (estructura triangular: [a disntacia con b, c, d, e] [b distancia con c, d, e] ...)
Output	ret_array2 (list(list(int))): lista de listas de grupos creados a partir de las distancias (no reundantes)

	select
Descripción	selecciónа un frame por cada sub lista dentro de .ªrray", obteniendo una lista de posiciones de los frames selecciónados types 0 : Obtiene los últimos elementos 1 : Obtiene los que posean mas información (get_trans_slide).
Input	array (list): lista de listas, agrupación de frames por diapositiva frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 types (int): indicador del tipo de selector a usar; 0 para el último de cada lista, 1 para usar get_trans_slide rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) gpu_use (boolean): indicador para activar o no el uso de la GPU
Output	retorno (list): lista del nombre de los frames selecciónados

	get_trans_slide
Descripción	Obtiene una lista de posiciones, con las transcripciones de los frames indicados en array compara para seleccionar
Input	reader (class): clase easyocr.easyocr.Reader usada para la transcripción de frames array (list): array con las posiciones de los frames de una slide frames (str): ruta a carpeta de frames o imagen (numpy array) rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False)
Output	list_ret (list): lista de posiciones de los frames selecciónados

	write_json
Descripción	Función que escribe data en un archivo formato json
Input	data (list o dict): data estructurada en listas o diccionarios filename (str): ruta del archivo con el nombre del mismo, excluyendo la extension
Output	No aplica

	get_transcription
Descripción	Función que itera sobre los frames/imágenes transcribiendolas usando algún OCR 1 = easyOCR 2 = teseract
Input	vname (str): nombre del video procesado frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 data (list): array con posiciones, usadas como filtro en la selección de imágenes rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) runtime (boolean): indicador de modo de manejo de frames (True: en numpy.array, False: en rutas de los frames) gpu_use (boolean): indicador para activar o no el uso de la GPU path (str): ruta destino del archivo json con la transcripción (necesario solo para caso runtime=False) ocr (int): indicador de que OCR es usado para obtener la transcripción, 1 = easyOCR o 2 = tesseract
Output	transcription (str o list): texto recopilado de cada frame unido en una sola estructura, ya sea en formato de array o string de la ruta del archivo

	isame
Descripción	Compara dos frames usando el porcentaje de píxeles que difieren como también el valor para SSIM entre ellos
Input	frame1 (str): ruta del primer frame frame2 (str): ruta del segúndo frame rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) pix_lim (float): indicador de límite para filtrar imágenes según métrica de porcentaje de píxeles cambiantes ssimv_lim (float): indicador de límite para filtrar imágenes según métrica SSIM dbugg (boolean): True en caso de querer visualizar los frames
Output	state (boolean): indicador que indica si son considerados suficientemente similares

	clean
Descripción	Función que usando isame() filtra las imágenes que son consideradas iguales (dejando solo una de ellas) para el caso de runtime falso: se elimina el frame de la ruta caso runtime: se crea una nueva lista con los frames correspondientes y se retorna
Input	frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 rgb (boolean): indicador de uso de 3 bandas de color (RGB, True) o solo una (B/W, False) pix_lim (float): indicador de límite para filtrar imágenes según métrica de porcentaje de píxeles cambiantes ssimv_lim (float): indicador de límite para filtrar imágenes según métrica SSIM
Output	Frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 # Frames (lista): runtime->lista con los frames (array(numpy.array)) y no-runtime->lista con los nombres de los frames en la carpeta

	ploteo
Descripción	Función que grafica data 1D, y en caso de no entregarla la obtiene usando getdata(f_ruta)
Input	nombre (str): nombre de la data (video) data (list): lista con valores obtenidos de la comparación de frames contiguos coef (int): factor limitador sobre el número de mínimos considerados dbugg (boolean): True en caso de querer visualizar el nombre entregado
Output	'OK' (str)

	classic
Descripción	Grafica data 1D, indicando el nombre, mínimo y máximo de la data
Input	data (list): array ordenado con números enteros 1D nombre (str): nombre de la data (video) coef (int): factor limitador sobre el número de mínimos considerados
Output	no aplica

	clean_transc
Descripción	Desde una transcripción en formato de lista se eliminan redundancias y se retorna la nueva lista
Input	transc (list o str): lista de listas con texto transcrito o ruta de archivo json
Output	transc (list o str): lista de listas con texto transcrito filtrado o ruta de archivo json filtrado del (list): lista binaria con 1 en la posición de un frame eliminado

	delete_frames
Descripción	Desde una transcripción en formato de lista se eliminan redundancias y se retorna la nueva lista
Input	frames (str): ruta de carpeta de frames o (list): array de imágenes cv2 lt_delet (list): lista ordenada de frames a ser eliminados (1 en posición de frames a eliminar, 0 sino se elimina) o (número de las posiciones a mantenerse, ej: [0,3,7]) tipo (int): indicador si se usara la primera o segunda estructura para la lista lt_delet
Output	Frames (str): ruta de carpeta de frames o (list): array de imágenes cv2

	lemat
Descripción	Función que lematiza el texto recibido, iniciando internamente el pipeline de stanza
Input	text (str): string con oración o párrafo a ser lematizado gpu_use (boolean): indicador para activar o no el uso de la GPU
Output	ret (str): string con texto lematizado

	lemat2
Descripción	Función que lematiza el texto recibido
Input	nlp (class): stanza.Pipeline text (str): string con oración o párrafo a ser lematizado gpu_use (boolean): indicador para activar o no el uso de la GPU
Output	ret (str): string con texto lematizado

	lematize
Descripción	Función que lematiza transcripción desde un array o una carpeta
Input	transcription (str): string con oración/párrafo a ser lematizado o ruta del archivo json gpu_use (boolean): indicador para activar o no el uso de la GPU dest_file (str): ruta de archivo de salida, incluyendo nombre y extensión archivo
Output	transcription (str o list): string con ruta al json o lista de listas con la transcripción

	tese
Descripción	Función que desde un frame/imagen obtiene una transcripción usando OCR tesseract, entregandolo en un formato estandar usado por el sistema EasyOCR
Input	ruta (str): ruta de frame/imagen a transcribir lim_acc (int): factor mínimo de confianza usado para filtrar las palabras extraídas con el sistema OCR debug (boolean): indicador si se quiere o no mostrar la imagen a transcribir
Output	data (str): transcripción de la imagen a texto

	upscale_img
Descripción	función que mejora imagen según modelo y escala entregada
Input	img (str): ruta de imagen a mejorar pb_path (str): ruta del archivo pb del modelo a usar model (str): nombre del modelo a usar (’edsr’, ’espcn’, ’fsrcnn’ o ’lapsrn’) ratio (int): escala a aplicar a la imagen (2, 3 o 4) runtime (boolean): indicador de modo de manejo de imágenes (True: imagen en numpy.array, False: entonces img es ruta de la imagen) gpu (boolean): indicador de uso de gpu
Output	imagen en array cv2 para runtime True y ’ok’ para caso contrario

REFERENCIAS BIBLIOGRÁFICAS

[Anwar *et al.*, 2020] Anwar, S., Khan, S., y Barnes, N. (2020). A deep journey into super-resolution: A survey. *ACM Computing Surveys (CSUR)*, 53(3):1–34.

[Bhartia *et al.*,] Bhartia, K., Rai, S., Gupta, P., y Agrawal, K. K. Image resolution enhancer using deep learning.

[Chen *et al.*, 2004] Chen, D., Odobez, J.-M., y Bourlard, H. (2004). Text detection and recognition in images and video frames. *Pattern recognition*, 37(3):595–608.

[Dilmegani, 2021] Dilmegani, C. (2021). Best ocr by text extraction accuracy in 2021. Disponible en `https://research.aimultiple.com/ocr-accuracy/`.

[Estrategia y line, 2021] Estrategia, D. y line, E. O. (2021). Emprendedores de inteligencia artificial: Latinoamérica avanza en sus desafíos. Disponible en https://www.diarioestrategia.cl/texto-diario/mostrar/2640005/emprendedores-inteligencia-artificial-latinoamerica-avanza-desafios.

[Gonzalez-Audicana *et al.*, 2004] Gonzalez-Audicana, M., Saleta, J., Catalan, R., y Garcia, R. (2004). Fusion of multispectral and panchromatic images using improved ihs and pca mergers based on wavelet decomposition. *IEEE Transactions on Geoscience and Remote Sensing*, 42(6):1291–1299.

[Hosseini *et al.*, 2017] Hosseini, H., Xiao, B., y Poovendran, R. (2017). Google's cloud vision api is not robust to noise. En *2017 16th IEEE international conference on machine learning and applications (ICMLA)*, pp. 101–105. IEEE.

[Jesús, 2022] Jesús (2022). ¿qué es el reconocimiento Óptico de caracteres (ocr)? - datasmarts.

[Liao, 2020] Liao, C. (2020). Ocr engine comparison — tesseract vs. easyocr. Disponible en `https://medium.com/swlh/ocr-engine-comparison-tesseract-vs-easyocr-729be893d3ae`.

[Mancas-Thillou y Gosselin, 2006] Mancas-Thillou, C. y Gosselin, B. (2006). Spatial and color spaces combination for natural scene text extraction. En *2006 International Conference on Image Processing*, pp. 985–988. IEEE.

[Palma Muñoz, 2023] Palma Muñoz, F. (2023). Videoslides. Disponible en `www.github.com/Zes7one/VideoSlides`.

[Patel *et al.*, 2012] Patel, C., Patel, A., y Patel, D. (2012). Optical character recognition by open source ocr tool tesseract: A case study. *International Journal of Computer Applications*, 55(10):50–56.

[Pedregosa *et al.*, 2011] Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., y Duchesnay, E. (2011). Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research*, 12:2825–2830.

[Qi *et al.*, 2020] Qi, P., Zhang, Y., Zhang, Y., Bolton, J., y Manning, C. D. (2020). Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. En *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pp. 101–108, Online. Association for Computational Linguistics.

[Raval, 2021] Raval, P. (2021). Measuring similarity in two images using python | by param raval | towards data science. Disponible en `https://towardsdatascience.com/measuring-similarity-in-two-images-using-python-b72233eb53c6`.

[Sitikhu *et al.*, 2019] Sitikhu, P., Pahi, K., Thapa, P., y Shakya, S. (2019). A comparison of semantic similarity methods for maximum human interpretability. En *2019 artificial intelligence for transforming business and society (AITB)*, volumen 1, pp. 1–4. IEEE.

[Stecanella, 2019] Stecanella, B. (2019). Understanding tf-id: A simple introduction.

[Sustentable, 2021] Sustentable, D. (2021). Récord de creación de empresas en 2020: se constituyeron 158 mil nuevas compañías. Disponible en https://www.diariosustentable.com/2021/01/record-de-creacion-de-empresas-en-2020-se-constituyeron-158-mil-nuevas-companias/.

[Wang *et al.*, 2004] Wang, Z., Bovik, A., Sheikh, H., y Simoncelli, E. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

[Wang y Bovik, 2002] Wang, Z. y Bovik, A. C. (2002). A universal image quality index. *IEEE signal processing letters*, *9*(3):81–84.

[Wang *et al.*, 2003] Wang, Z., Simoncelli, E., y Bovik, A. (2003). Multiscale structural similarity for image quality assessment. En *The Thrity-Seventh Asilomar Conference on Signals, Systems & Computers, 2003*, volumen 2, pp. 1398–1402 Vol.2.

[Zhou *et al.*, 1998] Zhou, J., Civco, D. L., y Silander, J. A. (1998). A wavelet transform method to merge landsat tm and spot panchromatic data. *International Journal of Remote Sensing*, 19(4):743–757.