UNIVERSITAT POLITÈCNICA DE VALÈNCIA

etsinf

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

Sistema avanzado de resumen de vídeos

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Autor: Daniel Turner Cebriá

Tutor: Lluis Felip Hurtado Oliver
Cotutor: Encarna Segarra Soriano

Curso 2019-2020

Resum

Aquest treball consisteix en el desenvolupament d'una aplicació amb interfície d'usuari capaç de resumir vídeos proporcionats per l'usuari, en funció de diferents paràmetres configurables, per a obtindre resums que s'ajusten a les peticions de l'usuari.

Per a realitzar el resum, el treball es basa en tres fonts d'informació: els subtítols, les escenes i els objectes.

Els subtítols, proporcionats per l'usuari, s'analitzen utilitzant tècniques de resum de text per a obtindre les frases que són més rellevants.

Les escenes s'obtenen calculant els moments en els quals la imatge del vídeo canvia prou com per a considerar-se una nova escena, i s'utilitzen per a adaptar el resum a aquestes escenes obtenint talls naturals i per a optimitzar la detecció d'objectes.

Els objectes detectats s'utilitzen per a remarcar certes parts del vídeo. L'usuari indica quins objectes vol visualitzar i aquests es detecten en el vídeo i es mostren les escenes que els contenen en el resum.

Finalment es realitzen proves per a validar l'elecció de certes opcions com l'optimització per escenes.

Paraules clau: vídeo, resum, intel·ligència artificial, interfície

Resumen

Este trabajo consiste en el desarrollo de una aplicación con interfaz de usuario capaz de resumir videos proporcionados por el usuario, en función de distintos parámetros configurables, para obtener resúmenes que se ajusten a las peticiones del usuario.

Para realizar el resumen, el trabajo se basa en tres fuentes de información: los subtítulos, las escenas y los objetos.

Los subtítulos, proporcionados por el usuario, se analizan utilizando técnicas de resumen de texto para obtener la frases que son más relevantes.

Las escenas se obtienen calculando los momentos en los que la imagen del vídeo cambia lo suficiente como para considerarse una nueva escena, y se utilizan para adaptar el resumen a estas escenas obteniendo cortes naturales y para optimizar la detección de objetos.

Los objetos detectados se utilizan para remarcar ciertas partes del vídeo. El usuario indica qué objetos quiere visualizar y estos se detectan en el vídeo y se muestran las escenas que los contienen en el resumen.

Finalmente se realizan pruebas para validar la elección de ciertas opciones como la optimización por escenas.

Palabras clave: vídeo, resumen, inteligencia artificial, interfaz

Abstract

This work consists of developing an application with a user interface capable of summarizing a video provided by the user, based on various configurable parameters, to obtain a video that meets the user's requests. This work consists in the development of an application with a user interface capable of summarizing videos provided by the user, based on different configurable parameters, to obtain summaries that adjust to the user's requests.

To make the summary, the work is based on three sources of information: the subtitles, the scenes and the objects.

Subtitles, provided by the user, are analyzed using text summarization techniques to obtain the phrases that are most relevant.

The scenes are obtained by calculating the moments when the video image changes enough to be considered a new scene which are used to adapt the summary to these scenes obtaining natural cuts and to optimize the detection of objects.

Detected objects are used to highlight certain parts of the video. The user indicates which objects she wants to visualize and these are detected in the video and their scenes are shown in the summary.

Finally, tests are carried out to validate the choice of certain options such as optimization by scenes.

Key words: video, abstract, artificial intelligence, interface

Índice general

Índice general VII
Índice de figuras IX
Índice de tablas IX

Agradecimientos XI
1 Introducción 1
1.1 Objetivos . . . 2
1.2 Estructura . . . 2
1.3 Antecedentes . . . 3
2 Estado del arte 5
2.1 Resumen de texto . . . 5
2.1.1 Latent Semantic Analysis . . . 6
2.2 Detección de escenas . . . 9
2.2.1 Diferencias *inter-frame* . . . 9
2.3 Detección de objetos . . . 10
2.3.1 Redes neuronales . . . 10
2.3.2 Redes convolucionales . . . 12
2.3.3 You Only Look Once . . . 13
3 Diseño de la solución 15
3.1 Arquitectura del sistema . . . 16
3.2 Diseño detallado . . . 16
3.2.1 Interfaz . . . 16
3.2.2 Modelos . . . 17
3.2.3 Contextos . . . 18
3.2.4 Procesos . . . 19
3.3 Tecnologías utilizadas . . . 23
3.3.1 FFmpeg . . . 23
3.3.2 MoviePy . . . 23
3.3.3 Natural Language Toolkit . . . 23
3.3.4 NumPy . . . 24
3.3.5 OpenCV . . . 24
3.3.6 PyQt . . . 24
3.3.7 Pysrt . . . 24
3.3.8 Python . . . 25
3.3.9 Scikit-Learn . . . 25
3.3.10 Yolo . . . 25
4 Análisis de la interfaz 27
5 Pruebas 33
5.1 Detección de escenas . . . 33

5.2 Detección de objetos 34
5.3 Optimización por escenas 35
6 Conclusión **37**
6.1 Trabajos futuros 37
Bibliografía **39**

Índice de figuras

2.1 El flujo de la técnica de resumen LSA. 6
2.2 Representación visual de la función SVD. 8
2.3 Representación visual de un cambio entre escenas. 9
2.4 Estructura de una red neuronal. 11
2.5 Representación del resultado de aplicar un filtro a una imagen. . . . 12
2.6 Representación de una convolución. 13
2.7 Representación de una red convolucional. 13

3.1 Esquema básico de la solución propuesta. 15
3.2 La arquitectura multi-capa de la aplicación. 16
3.3 El flujo detallado de la aplicación. 22

4.1 Ventana principal de la aplicación. 27
4.2 Ventana de configuraciones generales. 28
4.3 Ventana de configuración del análisis de objetos. 29
4.4 Ventana de configuración del análisis de subtítulos. 30
4.5 Ventana de resumen de las configuraciones. 31
4.6 Ventana de resumen del vídeo. 32
4.7 Ventana de descarga del vídeo resumido. 32

Índice de tablas

5.1 Resultados de la prueba de detección de escenas. 34
5.2 Resultados de la prueba de detección de objetos. 35
5.3 Resultados de la prueba de optimización por escenas. 36

Agradecimientos

Después de un intenso período, hoy es el día: escribo este apartado de agradecimientos para finalizar mi trabajo de fin de grado. Ha sido un período de aprendizaje tanto en lo personal como en lo académico. Realizar este trabajo ha tenido un gran impacto en mí y es por eso que me gustaría agradecer a todas aquellas personas que directa o indirectamente han estado involucradas en este proceso.

A mi madre Julia, a mis tíos Jorge y Mari, y a toda mi familia, gracias a quienes soy quien soy y hacia quienes sólo puedo expresar mi más sincero agradecimiento por apoyarme durante la etapa académica que hoy culmina.

A mis amigos, tanto de dentro como de fuera de la universidad, que no solo habéis estado ahí para apoyarme, sino que también me habéis hecho vivir grandes momentos juntos.

A mis tutores Lluis y Encarna por su valiosa ayuda y a la universidad por brindarme la oportunidad de cursar esta carrera.

A Mª José, mi profesora de informática en Bachillerato por inculcarme la admiración y respeto a la informática.

Y finalmente, a mi abuela Julieta, fallecida en 2018, por su apoyo incondicional sin el cual no estaría hoy aquí. Seguro que estaría muy orgullosa.

Hoy acaba una etapa muy importante de mi vida dando paso a nuevos desafíos para los cuales espero seguir contando con vosotros.

CAPÍTULO 1
Introducción

Internet está en alza. Desde sus inicios en 1983 de manos del Departamento de Defensa de los Estados Unidos, y el posterior anuncio público de la *World Wide Web* en 1991[13], internet lo ha revolucionado prácticamente todo. En Enero de 2020, la agencia We Are Social tasaba el número de usuarios conectados a internet en 4.54 billones[1], con un incremento de 298 millones respecto al año anterior. Es innegable la influencia que internet tiene en nuestras vidas. Tanto para trabajar como para ocio estamos constantemente conectados.

Según este mismo estudio, el 90 % de los usuarios que utilizan internet, entre otras cosas, lo hacen para consumir videos. Ya sean películas, series, documentales o videos de Youtube, por ejemplo, cada vez más usuarios acuden a estos en sus ratos de ocio. Las estadísticas de Youtube, con 2 billones de usuarios conectados cada mes, son un buen ejemplo de este aumento en la demanda de videos online.

Cada vez hay más demanda y con ello aumenta la oferta. Plataformas como Youtube o Netflix, entre otros, aumentan día a día su contenido. Pero para la mayoría de los usuarios es imposible ver todo lo que les gustaría. Es por esto que las empresas líderes en el sector están destinando gran cantidad de fondos para hacer esta enorme oferta de videos más accesible al usuario.

No es raro encontrarse con sistemas de recomendación en muchas plataformas que intentan mostrar lo que más les va a gustar a los usuarios. Aún así, estos no resuelven el problema que surge de la excesiva oferta y la enorme ambición de los usuarios por ver el máximo posible. Quieren ver más. Quieren saber más. Pero tienen poco tiempo. Y es ahí donde surge la necesidad de ofrecer más información al usuario en menos tiempo.

1.1 Objetivos

Es en el marco de esta necesidad donde surge la idea de este trabajo. El objetivo principal que nos marcamos en este trabajo es el de desarrollar una aplicación capaz de hacer un resumen de un vídeo proporcionado por el usuario.

La subjetividad del usuario es importante tenerla en cuenta para el resumen, por lo que la aplicación tiene que ser lo suficientemente configurable como para adaptarse a las preferencias de cada usuario.

Se pretende realizar el resumen en base a tres fuentes de información. La primera son los subtítulos, los cuales contienen todo el texto del vídeo y nos los deberá de proporcionar el usuario de forma opcional.

La segunda que pretendemos tener en cuenta son las escenas del vídeo. Se deberán detectar las distintas escenas del vídeo para que, en caso de que el usuario lo indique de esta forma, poder tomar dichas escenas como la unidad mínima de resumen con el objetivo de minimizar los cortes repentinos en el vídeo final.

Finalmente tomaremos también una lista de objetos importantes para el resumen que nos podrá indicar el usuario y que deberemos detectar a lo largo del vídeo y garantizar que estén presentes en el resumen final.

Las distintas fuentes de información deberán de poder ser combinadas permitiendo realizar un resumen basándonos en los subtítulos, las escenas o los objetos que aparecen.

1.2 Estructura

El trabajo se va a repartir en cuatro partes fundamentales:

1. Estado del arte (Capítulo 2): Analizaremos el estado del arte actual en los distintos ámbitos sobre los que nos vamos a apoyar para el desarrollo de la aplicación.

2. Diseño de la solución (Capítulo 3): Profundizaremos en el diseño de la estructura empleada para el desarrollo de la aplicación y las tecnologías utilizadas en esta.

3. Análisis de la interfaz (Capítulo 4): Recorreremos la interfaz de la aplicación para entender el funcionamiento a nivel de usuario, así como las distintas opciones que se ofrecen.

4. Pruebas y conclusión (Capítulo 5 y Capítulo 6): Analizaremos las distintas pruebas realizadas durante el desarrollo de la aplicación, analizaremos algunas decisiones tomadas para esta y terminaremos con una conclusión y varias propuestas de trabajos futuros a partir de este.

1.3 Antecedentes

Alberto Donet, estudiante de la Universidad Politécnica de Valencia, desarrolló una solución al problema de resumen de vídeos en 2016[6]. Esta solución servirá de punto de partida para el desarrollo del presente trabajo.

No son pocos los intentos de resumir la información de vídeos de una u otra forma. Algunas empresas están empezando a ofrecer servicios que utilizan la detección de objetos o de movimiento para resumir los vídeos de cámaras de seguridad[11].

Otro caso destacable es el del gobierno Chino, el cual ha instalado una red de cámaras de vigilancia a lo largo del país[1]. Toda la información de dichas cámaras es procesada gracias a la inteligencia artificial para obtener la información que el gobierno considera interesante. Una de las técnicas utilizadas es la detección de objetos en tiempo real.

El actual estado del arte en detección de objetos en tiempo real es YOLOv4[2]. Dicho algoritmo es capaz de procesar aproximadamente 65 *frames* por segundo con un acierto promedio del 45 %. Lo veremos en detalle en la Subsección 2.3.3.

En cuanto al ámbito educativo, podemos relacionar este trabajo con varias asignaturas de la Universidad Politécnica de Valencia como 'Interfaces persona computador', 'Sistemas inteligentes', 'Aprendizaje automático', 'Percepción' y 'Sistemas de almacenamiento y recuperación de información'.

[1] https://www.youtube.com/watch?v=uReVvICTrCM&t

CAPÍTULO 2
Estado del arte

A lo largo del trabajo vamos a descubrir algunas de las técnicas y algoritmos más relevantes en el campo del resumen de vídeos. En este capítulo vamos a profundizar en los tres antecedentes más importantes para nuestro trabajo, pues son sobre los que se basa nuestra aplicación de resumen automático.

El capítulo se divide en tres secciones: el resumen de texto (Sección 2.1), donde descubriremos el estado del arte en este campo y nos centraremos en el *Latent Semantic Analysis*; la detección de escenas (Sección 2.2), profundizando en la detección de la diferencia *inter-frame*; y finalmente nos pondremos al día con los avances en detección de objetos (Sección 2.3) así como en la importancia de las redes neuronales en este campo.

2.1 Resumen de texto

El resumen de textos es uno de los desafíos más relevantes actualmente en el campo del procesamiento del lenguaje natural (NLP por sus siglas en inglés). En los últimos años el número de estudios relacionados con el resumen de textos ha ido en aumento[31]. Constantemente aparecen nuevas técnicas y metodologías para afrontar este interesante desafío.

En cuanto a la forma del resumen existen dos importantes aproximaciones. La primera de estas es el resumen por extracción, la cual consiste en extraer las partes más importantes del texto para formar el resumen final. La aproximación contraria a esta es el resumen por abstracción, la cual se basa en extraer la información más importante del texto y reconstruir el resumen a partir de esta información sin sacar las frases directamente del texto original como se hace en la primera aproximación.

En paralelo a estas aproximaciones tenemos distintas formas de enfocar el resumen, según el número de documentos que se resumen a la vez. Respecto a esto, en los últimos años se ha investigado mucho sobre el resumen multi-documento, el cual realiza el resumen no sobre un documento en concreto, sino sobre una colección de documentos sobre un tema concreto.

Finalmente, para realizar el resumen existen multitud de técnicas. En este trabajo vamos a tratar la conocida LSA o *Latent Semantic Analysis* (Subsección 2.1.1).

2.1.1. Latent Semantic Analysis

LSA o *Latent Semantic Analysis*[28], es una técnica de resumen de texto muy conocida desde su publicación en 1990. Podemos distinguir tres partes importantes en el proceso.

La primera es la creación de una matriz que relaciona las frases con las palabras. Para esta relación existen distintas formas de calcular los valores que veremos en la primera parte de esta sección.

El siguiente paso es la ejecución del conocido algoritmo SVD o *Singular Value Decomposition* el cual también veremos explicado en la segunda parte de la sección.

Finalmente veremos las distintas formas en las que podemos seleccionar las frases más importantes para nuestro resumen en la parte final de la sección.

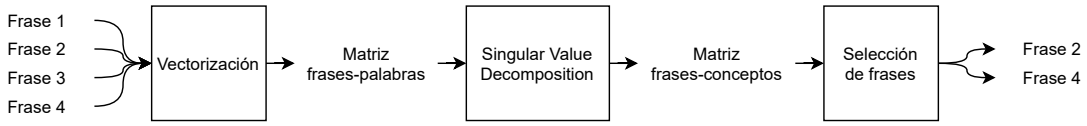

Figura 2.1: El flujo de la técnica de resumen LSA.

En la figura 2.1 podemos observar el flujo de la técnica de resumen LSA y las tres partes importantes que vamos a describir a continuación.

Matriz frases-palabras

En este primer paso tenemos que crear una matriz que relacione las frases del texto a resumir con las palabras. Para crear estas matrices podemos obtener los valores de dos formas: por conteo o por frecuencia de aparición.

Partimos, por ejemplo, de estas tres frases: ”El coche rojo está aparcado enfrente”, ”Este coche es el mejor coche del mundo” y ”El transporte más comun en el mundo es el coche”. Antes de generar las matrices, las frases son optimizadas eliminando los *stop words*.

En la primera forma realizamos un conteo de las palabras de cada frase creando la relación entre frases y palabras según el número de veces que estas aparecen en cada frase. Esto se conoce como *bag of words* o bolsa de palabras en español.

$$\begin{array}{c} \\ F1 \\ F2 \\ F3 \end{array} \begin{array}{c} \begin{array}{cccccccc} coche & rojo & aparcado & enfrente & mejor & mundo & transporte & común \end{array} \\ \begin{pmatrix} 1 & 1 & 1 & 1 & 0 & 0 & 0 & 0 \\ 2 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 1 & 1 & 1 \end{pmatrix} \end{array}$$

Esta primera forma admite una pequeña modificación, la cual, en lugar de contar las veces que aparece cada palabra solo tiene en cuenta si aparece o no,

indicando un 1 o un 0 en la matriz, similar a los llamados *one hot vector*.

	coche	*rojo*	*aparcado*	*enfrente*	*mejor*	*mundo*	*transporte*	*común*
*F*1	1	1	1	1	0	0	0	0
*F*2	1	0	0	0	1	1	0	0
*F*3	1	0	0	0	0	1	1	1

Podemos añadir también una mejora basada en el algoritmo *n-gram* [3], la cual intercambia las palabras (*1-gram*) por conjuntos de palabras sucesivas. *N-gram* hace referencia a los tokens (unidad mínima de información), que en nuestro caso son las palabras. En el caso de 3-gram, por ejemplo, serían conjuntos de tres palabras sucesivas y contaríamos la aparición de ese conjunto de tres palabras.

	coche rojo aparcado	*rojo aparcado enfrente*	...	*común mundo coche*
*F*1	1	1	...	0
*F*2	1	0	...	0
*F*3	1	0	...	1

Por otro lado podemos optar por crear una relación basada en la frecuencia de los términos/palabras (TF por sus siglas en inglés). Este tipo de relaciones se basan en calcular la frecuencia con la que aparece cada palabra en la frase. Para calcular estas frecuencias tenemos dos tipos distintos de normalización a nuestro alcance.

La primera (L1) busca obtener las menores desviaciones absolutas asegurándose de que la suma de los valores absolutos de cada frase sea 1.

	coche	*rojo*	*aparcado*	*enfrente*	*mejor*	*mundo*	*transporte*	*común*
*F*1	0,25	0,25	0,25	0,25	0	0	0	0
*F*2	0,5	0	0	0	0,25	0,25	0	0
*F*3	0,25	0	0	0	0	0,25	0,25	0,25

La segunda (L2) busca obtener los menores cuadrados, y funciona asegurándose que la suma de los cuadrados de cada frase sea 1.

	coche	*rojo*	*aparcado*	*enfrente*	*mejor*	*mundo*	*transporte*	*común*
*F*1	0,5	0,5	0,5	0,5	0	0	0	0
*F*2	0,7	0	0	0	0,5	0,5	0	0
*F*3	0,5	0	0	0	0	0,5	0,5	0,5

Una actualización del TF sería el TF-IDF o *Term Frequency Inverse Document Frequency*[22], cuyo objetivo es el de atenuar el peso de aquellas palabras que tengan presencia en muchos documentos, entendiendo en nuestro caso un documento como una frase.

Para calcular el peso de cada palabra se multiplica la frecuencia de la palabra por un factor de descuento que viene determinado por el número de documentos dividido por la cantidad de apariciones de dicha palabra en el conjunto de todas

las frases. La formulación matemática referente a lo descrito se puede expresar de la siguiente forma:

$$w_{ij} = tf_{ij} * log(\frac{N}{df_i})$$

siendo w_{ij} el peso de la palabra i en la frase j; tf_{ij} la frecuencia de la palabra i en la frase j; N el número de frases; y df_i la frecuencia de la frase de la palabra i en el total del texto.

	coche	*rojo*	*aparcado*	*enfrente*	*mejor*	*mundo*	*transporte*	*común*
F1	0,24	0,24	0,24	0,24	0	0	0	0
F2	0,48	0	0	0	0,24	0,24	0	0
F3	0,24	0	0	0	0	0,24	0,24	0,24

Singular Value Decomposition

Tras crear la matriz que nos relaciona las frases y las palabras según la relación que hayamos decidido aplicar, pasamos a la siguiente fase.

En esta vamos a aplicar SVD o *Singular Value Decomposition*[9]. El objetivo de esta es la factorización de una matriz rectangular (nuestra matriz de frases-palabras) en tres matrices más pequeñas tal y como podemos ver en la figura 2.2.

Una de estas matrices nos mostrará la relación de las frases con los "conceptos":

$$A = U\Sigma V^T$$

siendo A nuestra matriz original frases-palabras; U una matriz que relaciona las palabras con los conceptos; Σ una matriz cuya diagonal representa los conceptos; y V^T una matriz que relaciona las frases con los conceptos.

Estos conceptos estarán ordenados por importancia y esto nos sera muy útil para obtener las frases más importantes.

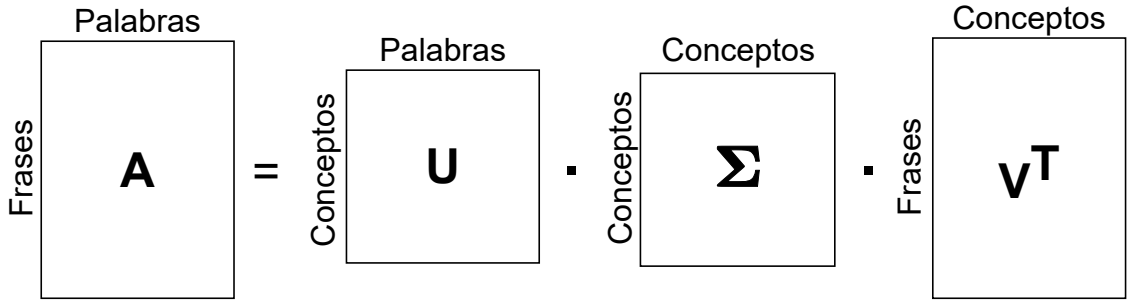

Figura 2.2: Representación visual de la función SVD.

Selección de frases

Finalmente, tras obtener la matriz frases-conceptos debemos elegir las frases más importantes para añadirlas a nuestro resumen. Esto no va a ser complicado puesto que tenemos los conceptos ordenados de más a menos importante.

Sería tan simple como coger la frase con mayor puntuación en cada uno de los conceptos más importantes hasta superar el porcentaje de resumen que hayamos decidido aplicar al resumen. Habría que tener en cuenta que la frase que estemos cogiendo no se encuentre ya añadida al resumen para evitar duplicidades, pero sigue siendo trivial.

2.2 Detección de escenas

A la hora de hacer un resumen, si solo tenemos en cuenta parámetros como los subtítulos nos arriesgamos a que los cortes del resumen sean bruscos y a perder el contexto de la parte del vídeo original que queremos mostrar. Para evitar esto, podemos hacer el resumen en función de las escenas, siendo estas las unidades mínimas de resumen. De esta forma lograremos unos cortes naturales y escenas completas que completen la información ofrecida en el resumen.

Un vídeo esta compuesto por multitud de imágenes o *frames* consecutivos. Si mostramos suficientes *frames* por cada segundo, conseguimos ese efecto de fluidez que nos brindan los videos. Sin embargo, en la mayoría de los videos, los cambios entre escenas ocurren de un frame a otro, con lo cual no tenemos una transición fluida. Esta es la clave del método que vamos a utilizar para detectar las distintas escenas.

Figura 2.3: Representación visual de un cambio entre escenas.

Como podemos ver en la figura 2.3, es posible dividir dos escenas si encontramos el punto en que dos *frames* consecutivos son lo suficientemente distintos. El momento en que eso ocurre es el que tomaremos para dividir las dos escenas.

2.2.1. Diferencias *inter-frame*

Para detectar las escenas lo que vamos a hacer es detectar los cambios de escenas, es decir, el momento exacto en el que acaba una escena y empieza la siguiente. Una de las características de las escenas es que la mayoría de lo que ocurre en la imagen (los fondos principalmente) son los mismo a lo largo de toda la escena. Al cambiar de escena todo esto cambia y de repente tenemos una imagen muy distinta a la anterior.

La forma de detectar este cambio es analizar cada frame del vídeo en función del frame anterior y detectar este cambio. Detectar cuando la diferencia entre los dos *frames* (imágenes) es elevada.

Una imagen es una matriz tridimensional, sobre la cual, por cada posición se dispone del correspondiente color en RGB (rojo, verde y azul). Estos tres canales tienen un valor entre 0 y 255 según su intensidad y la mezcla de estos tres es lo que genera el color final de cada pixel.

A partir de todo esto, para calcular la diferencia entre dos *frames* solo tenemos que calcular la suma de los valores RGB de todos los píxeles de una imagen y restarlo a los de la imagen anterior. Podemos obtener así el porcentaje de diferencia entre dos imágenes. A partir de esto podemos elegir un porcentaje a partir del cual asumimos que las dos imágenes son lo bastante diferentes como para interpretar que hay un cambio de escena.

2.3 Detección de objetos

En el campo de la inteligencia artificial, más concretamente en el campo del reconocimiento de patrones, uno de los desafíos más conocidos es el de clasificar imágenes. Existen multitud de algoritmos para clasificar imágenes, los más utilizados en los últimos años son los basados en redes neuronales. Estas redes se utilizan tanto para clasificar imágenes como para detectar los objetos que se encuentran en una imagen.

Hasta la aparición de estas redes convolucionales se utilizaban algoritmos basados en representaciones globales o locales. Dichas redes irrumpieron con mucho éxito y se convirtieron en el actual estado del arte[25].

En este apartado vamos a profundizar en el funcionamiento de las redes neuronales y redes neuronales convolucionales para la clasificación de imágenes. Después nos centraremos en YOLO[23], una red pre-entrenada de software libre que utilizaremos en nuestro proyecto y se basa en una red convolucional.

2.3.1. Redes neuronales

Las redes neuronales[24] son un modelo computacional muy utilizado en detección de objetos y procesamiento del habla, entre otros campos, debido a la eficacia que pueden lograr con un buen entrenamiento. Dichas redes han conseguido sistemas *end-to-end* gracias a cubrir el *gap* entre los procesos de representación y los procesos de clasificación, ya que estas realizan todo el trabajo a la vez.

Estas están formadas por pequeñas unidades llamadas neuronas, inspiradas en el comportamiento de las neuronas del cerebro del ser humano. Estas neuronas están conectadas a las demás y ordenadas en distintas capas, de forma que las neuronas de la capa n se conectan con las capas vecinas.

Cada neurona recibe un conjunto de *inputs* o entradas y producen un *output* o salida realizando una combinación lineal de los pesos de entrada por los valores que tiene asociado la neurona. La operación que se realiza dentro de la neurona es lo que tendremos que entrenar.

Dados los valores de entrada, la red neuronal es capaz de, mediante el algoritmo *forward*, realizar las operaciones oportunas y devolver en la salida de la red la solución al problema propuesto.

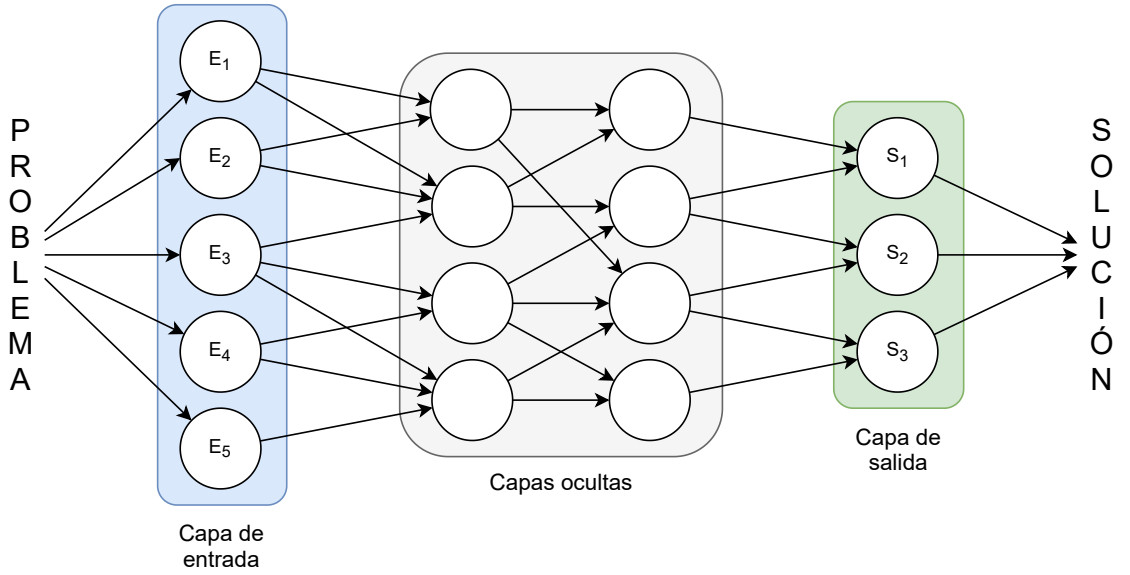

Figura 2.4: Estructura de una red neuronal.

En la figura 2.4 podemos observar un conjunto de las mencionadas neuronas organizadas en cuatro capas. Una de estas es la de entrada, la cual solo contiene los valores de entrada para la red. Otra es la capa de salida, la cual contiene los resultados de la red. Y entre estas se encuentran las llamadas capas ocultas, pues funcionan como una caja negra. Estas capas son las que tienen el potencial de la red neuronal. Podemos observar también que las neuronas no tienen por qué estar conectadas a todas las neuronas de la capa siguiente, pudiendo algunas de estas estar incluso conectadas solo con una neurona.

La parte más importante de la construcción de una red neuronal es el diseño de esta. En esta etapa, el diseñador tiene que decidir el número de capas, el número de neuronas y la estructura de la red. En cuanto a esto, la capa de entrada y la capa de salida es lo único que está predefinido por el problema a resolver, dejando el resto de características en manos del diseñador.

Una vez definida la red se pasa a la fase más costosa en términos computacionales, la fase de entrenamiento. Para esta, se prepara un gran conjunto de casos de muestra con el problema y la solución que se desea obtener. Estos casos se van pasando uno a uno por la red neuronal, cuyas neuronas estarán configuradas de forma aleatoria al inicio.

En cada caso se obtiene la solución de la red y se compara con la solución que se desea obtener. A partir de esto se obtiene el error que la red ha cometido y se envía hacia atrás en lo que se llama *back-propagation* a través de las distintas capas de neuronas. La red es capaz de asignar a cada neurona un nivel de responsabilidad sobre el error producido, lo que hará que la neurona se ajuste automáticamente para la próxima vez acercarse más a la solución que se pretende obtener. Al realizar esto con una gran cantidad de muestras se obtiene una red neuronal entrenada para resolver el problema que se le propone.

Finalmente se le dan unos casos de muestra distintos a los del entrenamiento (conocido como conjunto de test o validación) y se observan los errores que

presenta entre lo esperado y lo que obtenemos para obtener la precisión de la red.

2.3.2. Redes convolucionales

En el campo de las redes neuronales existen gran variedad de modelos. Uno de estos modelos son las redes neuronales convolucionales[12], utilizadas ampliamente para tratar imágenes.

Estas redes destacan por la aplicación de filtros a la imagen de entrada, de forma que se destacan las características mas importantes de esta. Estos filtros son un conjunto de operaciones que se aplican sobre cada pixel de la imagen original. Los valores del filtro se inician aleatoriamente y se van ajustando automáticamente por la propia red durante el entrenamiento, de forma que la red aprenda que filtro es el que le ayuda a destacar la información que le resulta mas interesante.

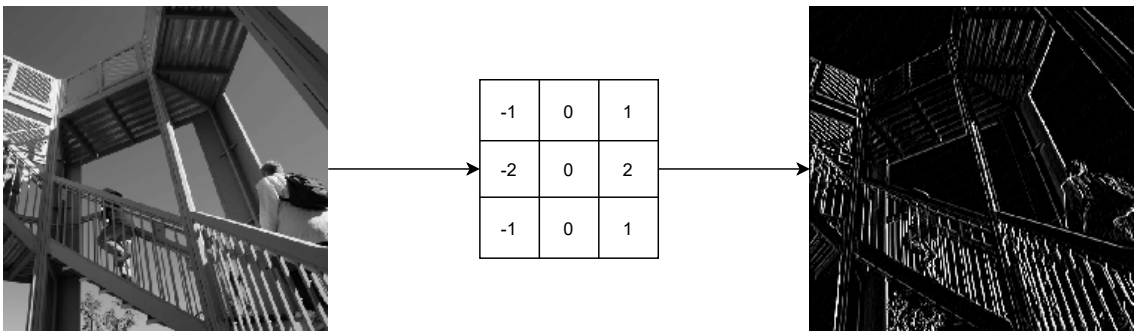

Figura 2.5: Representación del resultado de aplicar un filtro a una imagen.

En la figura 2.5 podemos observar como al aplicar el filtro adecuado a una imagen podemos obtener otra en la que se resalta la información importante. En este caso se observa como el filtro resalta los bordes de los objetos que aparecen en la imagen.

Después de la aplicación de las capas convolucionales se suele utilizar lo que se conoce como *pooling*. El uso de pooling permite reducir la dimensionalidad de la imagen que recibe como entrada. Para esto se divide la imagen en distintas secciones de tamaño $n * n$ para, de cada sección, obtener un valor que sea el representante de este en la imagen reducida. Por ejemplo, en caso de dividir la imagen en secciones de $2 * 2$, la imagen original se reduciría a un 25 %. Para esto se pueden aplicar varias técnicas distintas de *pooling*.

1. *Max pooling*: se obtiene el valor máximo de los píxeles de dicha sección.

2. *Average pooling*: se obtiene el valor medio de los píxeles de dicha sección.

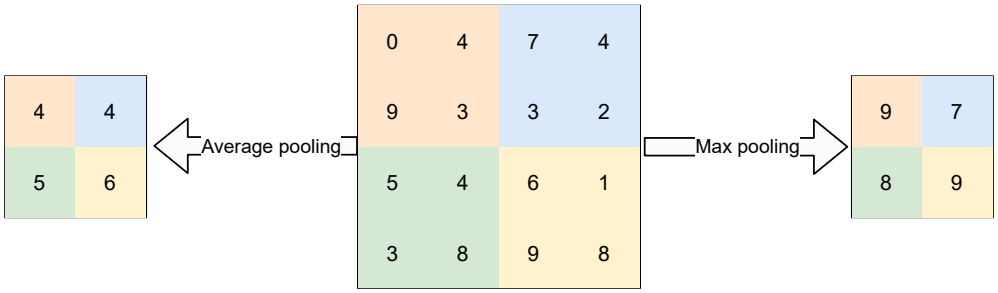

Figura 2.6: Representación de una convolución.

En la figura 2.6 podemos observar como a través del *pooling* se reduce la dimensionalidad de la matriz original obteniendo el valor máximo o medio de cada sección.

Se puede aplicar un conjunto de operaciones convolucionales y pooling hasta que llegas a un tamaño lo suficientemente reducido como para ser introducidas en una capa *fully connected* (capas conectadas al completo con las de la capa siguiente) para obtener la salida de dicha red.

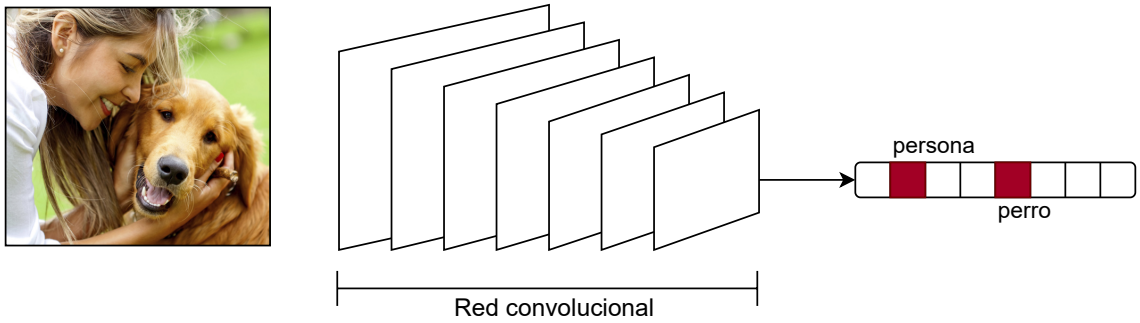

Figura 2.7: Representación de una red convolucional.

En a figura 2.7 se observa una red convolucional que gracias a la reducción de dimensionalidad logra obtener un vector que nos indica que en la imagen hay un perro y una persona.

2.3.3. You Only Look Once

You Only Look Once[23] (YOLO por sus siglas) es un algoritmo de código abierto para detección de objetos en imágenes. Destaca por ser extremadamente rápido y casi igual de fiable que sus competidores.

Para lograr esto, sus creadores han decidido, en lugar de separar la detección de objetos en distintas partes especializadas como hacen los demás modelos (R-CNN)[8], agruparlo todo en una única red neuronal convolucional, permitiendo de esta forma realizar un entrenamiento *end-to-end*. Dicha red está compuesta de 24 capas convolucionales y 2 capas *fully connected*. La red divide la imagen de entrada en partes más pequeñas y calcula las probabilidades de esas partes de ser alguno de los objetos para los que se entrena.

Esta red devuelve los objetos detectados, junto a la probabilidad y su posición, lo cual es ampliamente utilizado para controlar el movimiento de ciertos objetos en vídeos. A su vez, gracias a la velocidad de esta red (aproximadamente 45 *frames* por segundo), puede ser utilizada para la detección en tiempo real (por ejemplo, a través de una cámara), con un reducido tiempo de respuesta.

CAPÍTULO 3
Diseño de la solución

La solución que se va a desarrollar este trabajo consiste en una aplicación con interfaz gráfica. Esta aplicación permite al usuario configurar el resumen que va a realizarse. Durante la ejecución de la aplicación, el usuario podrá ajustar los parámetros del resumen de forma que este se adapte a su criterio. Entre las distintas configuraciones disponibles para el usuario se destaca que podrá indicar una lista de objetos que deben estar presentes en el resumen final.

Se toman como entrada el vídeo original y los subtítulos de este. A partir del vídeo la aplicación detectará los objetos y las escenas, y gracias a los subtítulos obtendrá la transcripción del audio de este. A partir de toda esta información, la aplicación resumirá el vídeo original permitiendo obtener el resultado de este en un fichero con formato '.mp4'.

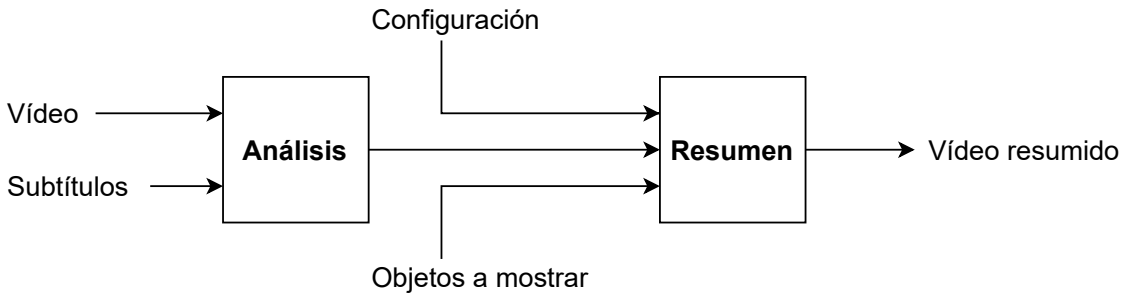

Figura 3.1: Esquema básico de la solución propuesta.

En la figura 3.1, podemos observar un esquema básico de la aplicación con el vídeo y los subtítulos como parámetros de entrada, la configuración y los objetos a mostrar en el resumen como parámetros configurables durante la ejecución de esta, y el vídeo resumido como salida final.

A continuación veremos la arquitectura que se ha elegido para la aplicación en la Sección 3.1.

También se hará una descripción detallada del diseño de la aplicación en la Sección 3.2 y finalmente veremos las tecnologías empleadas en la Sección 3.3.

3.1 Arquitectura del sistema

Las clases de la aplicación podemos dividirlas en cuatro tipos según su función:

1. Interfaz: permite al usuario configurar el resumen a realizar, así como el proceso de este, y descargar el resultado del resumen.
2. Modelo: asociado a la interfaz correspondiente, se encargará de gestionar las respuestas a las acciones que tome el usuario sobre esta.
3. Contexto: guarda en disco las configuraciones y la información importante para el resumen.
4. Proceso: rutina que se ejecuta en segundo plano y realiza una de las funciones básicas de la aplicación como analizar los subtítulos, detectar las escenas, etc.

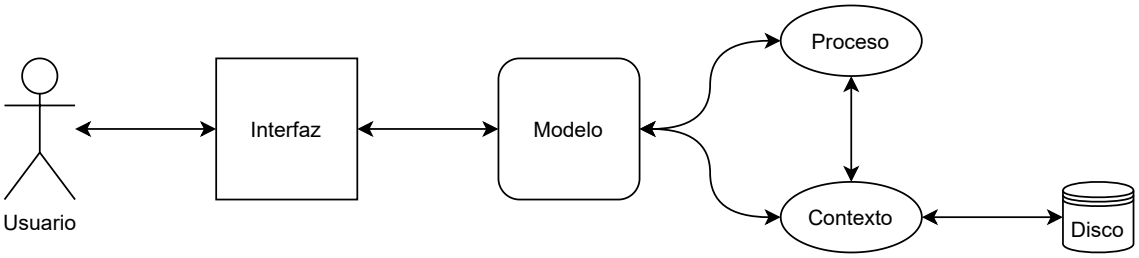

Figura 3.2: La arquitectura multi-capa de la aplicación.

Podemos observar en la figura 3.2 la arquitectura multi-capa de la aplicación, donde la interfaz interactúa únicamente con los modelos y estos son los que acceden a los contextos y controlan la ejecución de los procesos. Los procesos también acceden a los contextos, pues necesitan de su información para ejecutarse.

3.2 Diseño detallado

Tal y como se muestra en la Sección 3.1, la arquitectura del sistema se puede dividir en cuatro partes esenciales. Estas son la interfaz, los modelos, los procesos y los contextos.

3.2.1. Interfaz

La interfaz es la parte de la aplicación con la que interactúa el usuario. Dicha interfaz está desarrollada en PyQt5 utilizando la herramienta QtCreator dando como resultado un fichero '.ui' por cada ventana.

En la aplicación tenemos un total de siete ventanas con su correspondiente fichero '.ui':

1. MainWindow.ui: ventana principal, la cual te permite reiniciar la configuración de la aplicación a la definida por defecto y seleccionar la ruta del vídeo que se quiere resumir.

2. GeneralOptions.ui: ventana desde la que se pueden configurar los parámetros generales del resumen (el tipo de resumen, la detección de escenas y el porcentaje de diferencia entre escenas para considerarlas distintas).

3. ObjectsOptions.ui: si el tipo de resumen incluye la detección de objetos, la aplicación te mostrará esta ventana. Esta permite configurar la detección (objetos a detectar, periodicidad de los análisis en milisegundos o por escenas y los ficheros de YOLO que conforman la red neuronal que detectará los objetos).

4. SubtitlesOptions.ui: si el resumen configurado incluye el análisis de los subtítulos, se mostrará esta ventana, que permite configurar el análisis de los subtítulos (la ruta de los subtítulos, el tipo de vectorizador para estos, el porcentaje de resumen de los subtítulos y distintos preprocesados para dichos subtítulos como el borrado de *stop words*).

5. ResumeOptions.ui: ventana a modo de resumen de las configuraciones elegidas para el resumen del vídeo para que se puedan revisar antes de indicarle a la aplicación que realice el resumen del video.

6. ResumeWindow.ui: en esta ventana se mantendrá al usuario mientras se completan los procesos de análisis y resumen del vídeo, donde simultáneamente se puede ver el avance de estos gracias al uso de las barras de progreso que se muestran en la ventana.

7. ResumeResult.ui: finalmente en esta última ventana se le da al usuario la opción de descargar el vídeo resumido en la ruta que indique, así como ver el progreso de esta descarga o guardado del video.

Además, para los cambios entre ventanas, la aplicación dispone de un controlador de ventanas (windows_controller.py), el cual se encargará del flujo de la aplicación, ya que algunos cambios entre ventanas son condicionales a la configuración.

3.2.2. Modelos

Los modelos son los encargados de relacionar la interfaz con los procesos. Hay un modelo por cada una de las ventanas de la aplicación descritas en la Subsección 3.2.1.

Estos modelos comparten algunas de las funcionalidades principales como son pasar a la siguiente ventana, o actualizar la barra de progreso. Es por eso por lo que se ha hecho que todos extiendan de un modelo que implementa toda esta funcionalidad para simplificar la aplicación.

Los modelos desarrollados son los siguientes:

1. model_interface.py: modelo 'padre' que implementa la funcionalidad de actualizar los valores de las barras de progreso de los procesos en segundo plano y el movimiento hacia adelante o hacia atrás a través de los botones.

2. main_window_model.py: modelo de la ventana principal que se encarga de desplegar la ventana para seleccionar la ruta del vídeo a cargar, reiniciar las configuraciones del contexto cuando lo pida el usuario y de desbloquear el botón 'Next' de la ventana una vez la ruta sea correcta.

3. general_options_model.py: modelo de la ventana 'GeneralOptions.ui' que se encarga de cargar la ventana y guardar en el contexto general las configuraciones generales, así como lanzar o reiniciar el proceso de análisis de escenas si la configuración lo requiere.

4. objects_options_model.py: modelo que gestiona la ventana de configuración del análisis de objetos, ya sea carga o guardado de esta configuración en el contexto de objetos; gestiona la lista de objetos a buscar (añadiendo y borrando elementos de la lista); muestra al usuario la ventana de selección de rutas para cargar los tres ficheros de YOLO necesarios; activa el botón 'Next' de la ventana cuando la configuración es correcta; y lanza o reinicia el proceso de detección de objetos.

5. subtitles_options_model.py: modelo de la ventana de configuración de subtítulos, el cual se encarga de cargar y guardar la configuración del contexto de subtítulos, verificar la configuración para activar el botón 'Next' de la ventana y lanzar o reiniciar el proceso de análisis de subtítulos.

6. resume_options_model.py: modelo de la ventana que muestra un resumen de las configuraciones y se encarga de cargar las configuraciones de todos los contextos de la aplicación y lanza o reinicia el proceso de resumen del video.

7. resume_window_model.py: modelo de la ventana 'ResumeWindow.ui', el cual se encarga de activar el botón 'Next' de la ventana una vez el proceso de resumen del vídeo ha finalizado.

8. resume_result_model.py: modelo de la ventana final que permite al usuario seleccionar una ruta para la descarga del vídeo resumido y lanzar o reiniciar los procesos de corte y descarga del video.

3.2.3. Contextos

Los contextos son los encargados de almacenar y recuperar la información en el disco para compartir información entre modelos y mantener la configuración del usuario aun cerrando la aplicación.

En la aplicación tenemos cuatro contextos diferenciados según sus datos:

1. general_context.py: contexto que guarda la configuración general de la aplicación (ruta del vídeo original, ruta de descarga del resumen, modo de resumen, detección de escenas, diferencia mínima entre escenas y los tiempos en base a los cuales hay que cortar el vídeo para hacer el resumen).

2. scenes_context.py: contexto en el cual se gestiona la información asociada a las escenas (la lista de tiempos que definen las distintas escenas del video).
3. objects_context.py: contexto que gestiona la configuración de los objetos a detectar en el vídeo (un diccionario con los objetos que aparecen a lo largo del vídeo y sus tiempos, la lista de objetos a detectar, las rutas de los ficheros de YOLO, el tipo de análisis y la periodicidad).
4. subtitles_context.py: contexto que almacena la configuración sobre el análisis de subtítulos (la ruta de los subtítulos, la lista de subtítulos ya extraídos y procesados, el porcentaje de resumen, el tipo de vectorizador y los distintos preprocesados disponibles para los subtítulos).

3.2.4. Procesos

Los procesos son la parte más importante de este trabajo. Son los encargados de analizar el vídeo y los subtítulos, así como de hacer el resumen del vídeo. También se utilizan para el guardado del vídeo resumido.

Los procesos están desarrollados de forma que se ejecutan en un hilo separado de la aplicación principal para no bloquearla mientras dura la ejecución de estos. A su vez, estos procesos mantienen actualizada una señal con la que indican su progreso para mostrarlo en la interfaz y permitir que el usuario pueda ver como avanzan estos procesos.

Análisis de escenas

El proceso de análisis de escenas es el encargado de, a partir del vídeo original, obtener una lista de escenas para guardarla en el contexto de escenas y así permitir al proceso de detección de objetos y resumen usar dicha información.

Lo primero que hacemos es obtener el porcentaje de diferencia de cada frame con su predecesor tal y como se ha explicado en la Sección 2.2. Para esto la aplicación utiliza la herramienta FFmpeg (Subsección 3.3.1).

A partir de la lista de diferencias obtenidas y el porcentaje mínimo de diferencia indicado por el usuario, se obtiene la lista de escenas que componen el vídeo con sus tiempos de inicio y fin en milisegundos.

Análisis de objetos

El análisis de objetos es el encargado de analizar los *frames* del vídeo para detectar los objetos que aparecen en este.

Para ello la aplicación obtiene ciertos *frames* del vídeo para analizarlos. Esta selección se hace en función de los parámetros que configura el usuario.

Se puede configurar cada n milisegundos o se puede activar la optimización por escenas, lo que aprovecha la detección de escenas para obtener un especificado por el usuario de *frames* por escena. La idea de la optimización es minimizar los *frames* necesarios a analizar, ya que los objetos de una escena no cambian mucho.

Una vez obtenidos los *frames* que vamos a analizar se pasa uno a uno por la red neuronal de YOLO (Subsección 2.3.3). Esta red nos devuelve los objetos presentes en el *frame* analizado. Con esta información montamos un diccionario que relaciona los objetos que aparecen con el milisegundo en el que lo hacen y se guarda en el contexto de objetos para que lo utilice el proceso de resumen.

Análisis de subtítulos

Este análisis es el encargado de leer los subtítulos y devolver una lista de estos puntuados según su importancia en el vídeo para que el proceso de resumen pueda obtener los más importantes.

Lo primero que se hace es obtener los subtítulos que nos ofrece el usuario gracias a la herramienta Pysrt (Subsección 3.3.7). Con esto creamos una lista de objetos 'Subtitle' que es un objeto propio que se ha creado para la aplicación y se compone del texto, el tiempo de inicio y fin, y la puntuación del subtítulo.

Después de esto, se pasa cada subtítulo por algunos pre-procesos:

1. Fusión: se fusionan los subtítulos que son consecutivos, es decir, que forman la misma frase y no contienen signos de puntuación final entre ellos.
2. Mayúsculas: se convierten las mayúsculas en minúsculas para evitar que la misma palabra se considere dos distintas al tener una mayúscula.
3. *Stop words*: se eliminan los *stop words* en función del idioma que elige el usuario. Estas son aquellas palabras que se repiten, por lo general, mucho en una lengua y no aportan información sobre el discurso del autor. Esto se hace para eliminar ruido en el posterior análisis de los subtítulos.
4. Puntuaciones: se eliminan los signos de puntuación, pues la misma palabra podría detectarse de forma distinta al ir una unida a un signo de puntuación.
5. Acentos: se eliminan los signos de acentuado para simplificar el posterior resumen.

A excepción de la fusión, el resto de opciones son configurables por el usuario.

Tras el preprocesado se procede a el conteo de las palabras que se encuentran en los subtítulos creando una matriz que relaciona las palabras y las frases a las que pertenecen en función del tipo de vectorización elegido por el usuario.

Una vez creada la matriz se ejecuta el algoritmo SVD gracias a la librería NumPy (Subsección 3.3.4). Esto nos devuelve la matriz que relaciona frases y contextos tal como se explica en un apartado de la Subsección 2.1.1.

Después, a partir de dicha matriz, de cada contexto se obtiene la frase más relevante, puntuándolas en función del orden en que se van escogiendo, de forma que las frases más importantes obtienen una mayor puntuación, ya que los contextos están ordenados de mayor a menor importancia en la matriz que obtenemos.

Finalmente se guarda en el contexto la lista de subtítulos con su puntuación para que el proceso de resumen pueda utilizar esta información.

Resumen

El proceso de resumen es el encargado de, a partir de la información que le ofrecen los demás procesos, obtener los tiempos que indican cómo hay que cortar el vídeo para hacer dicho resumen.

Si el usuario demanda un resumen de subtítulos, este proceso obtendrá los tiempos que se almacenaron en el contexto de subtítulos. A estos le añadirá los tiempos del diccionario de objetos almacenado en el contexto de objetos de los objetos que estén en la lista que proporciona el usuario, si el tipo de resumen lo contempla.

Después de obtener dichos tiempos, si el usuario activa esta función, se procede al ajuste de los tiempos a las escenas. Es decir, se amplían los tiempos de los subtítulos y objetos hasta cuadrar con los cambios de escena.

Finalmente se normalizan los tiempos, pues puede que algunos tiempos se sobrepongan entre ellos, lo que haría que se duplicaran partes del video. La normalización obtiene los cortes correctos, asegurándose que no hayan repeticiones.

Finalmente se guarda en el contexto general los tiempos del resumen para que el proceso de guardado del vídeo haga los cortes necesarios antes de guardarlo.

Guardado de video

Una vez el usuario indica la ruta en la que quiere guardar el vídeo se lanza el proceso de guardado. Este proceso será el encargado de cortar el vídeo original según los tiempos generados por el resumen y guardar el resultado de dichos cortes en la ruta indicada.

En primer lugar se obtienen dichos tiempos y con las herramientas que nos ofrece MoviePy (Subsección 3.3.2) se obtienen los sub-clips que formarán el resumen.

Finalmente se guarda el audio y posteriormente el vídeo del resumen final en la ruta indicada.

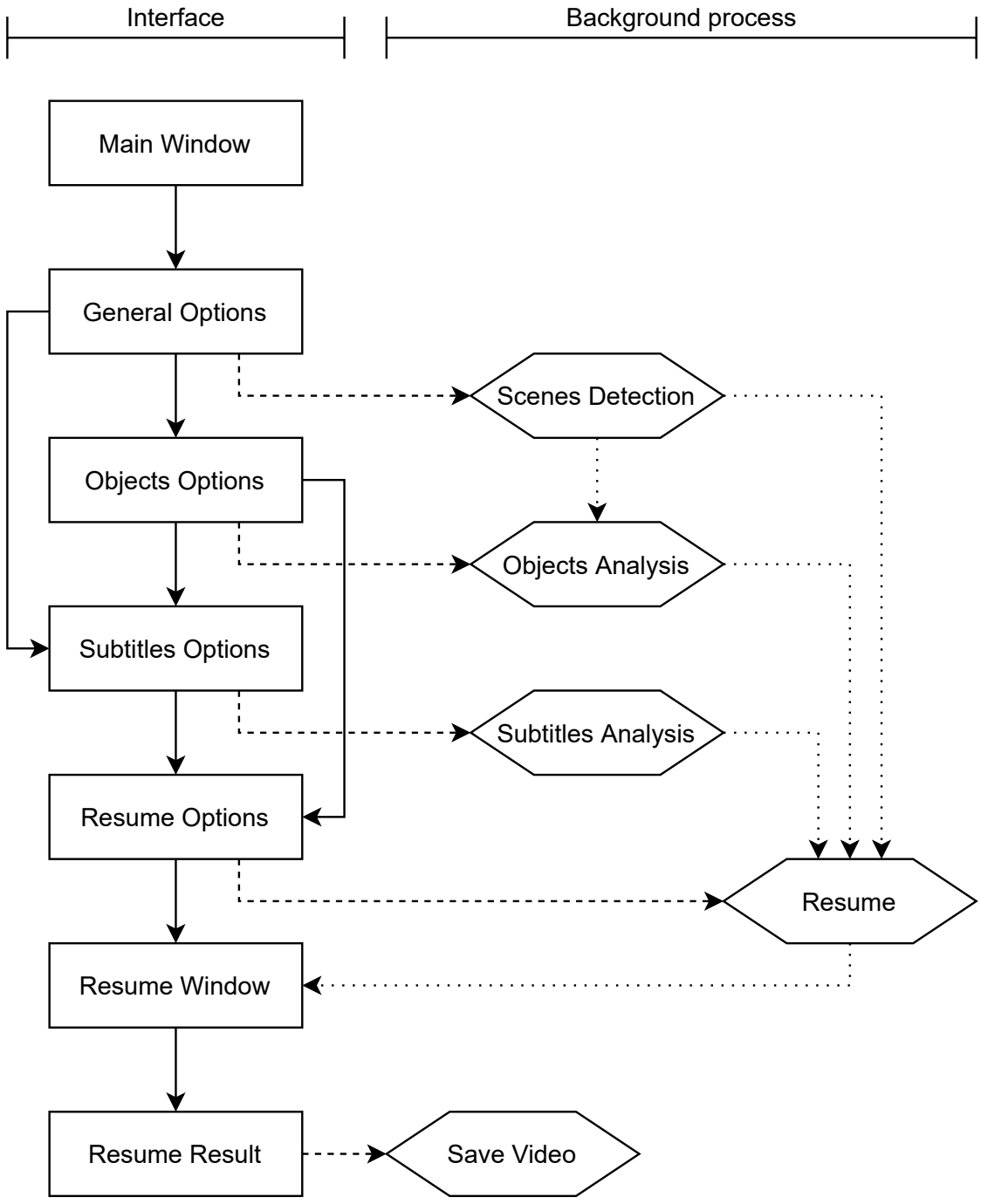

Figura 3.3: El flujo detallado de la aplicación.

En la figura 3.3, podemos observar el flujo completo de la aplicación. Los cuadrados hacen referencia a las distintas interfaces junto a sus modelos. Los rombos hacen referencia a los procesos que se ejecutan en segundo plano. Podemos observar las dependencias que tienen estos procesos, el flujo que sigue la interfaz de la aplicación en función del tipo de resumen elegido y las dependencias entre procesos y modelos.

3.3 Tecnologías utilizadas

En el desarrollo de la aplicación se han utilizado multitud de tecnologías diferentes. En esta sección se enumerarán las tecnologías más importantes utilizadas así como su uso en el desarrollo de la aplicación.

3.3.1. FFmpeg

FFmpeg[7] es el framework multimedia más popular de software libre. Es capaz de decodificar, codificar, transcodificar, retransmitir, filtrar y ejecutar prácticamente cualquier formato de audio y vídeo. Es compatible con una gran variedad de arquitecturas y sistemas operativos como Windows, Linux, Mac OS, BSD, Solaris, etc. FFmpeg fué lanzado por primera vez en el año 2000 y actualmente es un proyecto *open-source*.

En nuestra aplicación utilizamos FFmpeg para filtrar el vídeo original de forma que obtenemos los tiempos en los que la diferencia entre dos *frames* consecutivos es mayor al umbral marcado por el usuario para considerar que se inicia una nueva escena.

3.3.2. MoviePy

MoviePy[14] es un módulo de Python de código abierto diseñado para edición de vídeo que se puede utilizar para realizar operaciones básicas (cortes, concatenaciones, inserciones de títulos), para composición de vídeo, para procesamiento de vídeo y para crear efectos avanzados. MoviePy es compatible con los formatos de vídeos más comunes. Su primera *release* fué en 2017 y es un proyecto de software libre.

MoviePy se ha utilizado en la aplicación para gestionar todos los vídeos. Obtenemos la información necesaria sobre el vídeo original, realizamos los cortes sobre este en funcion de los tiempos que nos indica el resumen, y juntamos los clips obtenidos de los cortes para formar el vídeo final.

3.3.3. Natural Language Toolkit

Natural Language Toolkit[30] o NLTK por sus siglas en inglés, es un kit de herramientas para el procesamiento del lenguaje natural. Este proporciona interfaces fáciles de usar para una gran cantidad de *corpus* y recursos léxicos como WordNet junto a bibliotecas de procesamiento de texto para clasificación, tokenización, etc. Fue lanzado inicialmente en el año 2001 y se mantiene a día de hoy bajo la licencia Apache 2.0.

En nuestra aplicación utilizamos la librería para Python de NLTK para obtener de ella el *corpus* de palabras que vamos a eliminar de los subtítulos por no contener información relevante sobre el significado de la frase, conocidas como *stop words*.

3.3.4. NumPy

NumPy[15] es una librería de Python que pretende brindarle el poder computacional de lenguajes como C o Fortran agregándole un mayor soporte para vectores y matrices. NumPy contiene una extensa biblioteca de funciones matemáticas de alto nivel para trabajar con estructuras matriciales. NumPy fué creado en 2005 basándose en el trabajo inicial de las bibliotecas numéricas y Numarray. Es de código abierto y se mantiene gracias a la enorme comunidad científica que lo respalda.

Para nuestra aplicación utilizamos NumPy para la gestión de matrices y en especial para la ejecución del algoritmo *Singular Value Decomposition* visto en la Subsubsección 2.1.1.

3.3.5. OpenCV

Open Computer Vision[16], es una biblioteca de software libre de visión artificial. Nos permite detectar movimiento, reconocer objetos, reconstrucciones 3D a partir de imágenes, etc. Fué desarrollada inicialmente por Intel en el año 1999 pero sigue siendo a día de hoy la biblioteca más popular para visión artificial.

La API de la biblioteca está desarrollada en C++ pero nosotros trabajamos con su librería para Python. En la aplicación la utilizamos para escalar los *frames* de vídeo a la resolución que requiere YOLO, el cual también utiliza OpenCV internamente.

3.3.6. PyQt

Qt[21], es un *framework* multi-plataforma orientado a objetos muy utilizado para el desarrollo de programas que necesiten una interfaz de usuario. Qt fué desarrollado originalmente por Trolltech en 1995. En 2008 fue comprado por Nokia y a día de hoy se mantiene gracias a la comunidad de *Project Qt*.

Qt está desarrollado en C++ pero nosotros en la aplicación vamos a utilizar su librería para Python llamado PyQt. PyQt[17], esta desarrollada por *Riverbank Computing*, pero Nokia en 2009 desarrolló su versión bajo licencia LGPL llamada PySide[18].

En la aplicación utilizamos PyQt para el desarrollo de toda la interfaz junto a QtCreator[4], el editor para Qt de interfaces con interfaz gráfica.

3.3.7. Pysrt

Pysrt[19] es una librería para Python que nos permite gestionar subtítulos de forma sencilla. Entre sus funciones encontramos la creación de subtítulos, la edición o la carga de subtítulos de un fichero SubRip (.srt). Pysrt es de software libre y su primera *release* fué en 2009.

En la aplicación utilizamos esta librería para cargar y leer los subtítulos desde el fichero .srt que nos ofrece el usuario.

3.3.8. Python

Python[20] es un lenguaje de programación interpretado cuyo desarrollo está orientado a garantizar la legibilidad del código. Es un lenguaje que soporta multitud de paradigmas de programación como programación orientada a objetos o programación imperativa. Python está gestionado por la *Python Software Foundation* y es de código abierto. Fué publicado por primera vez en 1991 por Guido van Rossum.

Nuestra aplicación está desarrollada sobre dicho lenguaje de programación por su simplicidad y por la amplia comunidad que tiene detrás, lo que implica una gran variedad de librerías como las que se mencionan en esta Sección 3.3.

3.3.9. Scikit-Learn

Scikit-Learn[26] es una librería de Python que contiene una biblioteca de herramientas de *machine learning* que da apoyo al aprendizaje supervisado y al no supervisado desarrollada sobre SciPy[27]. Contiene algoritmos de clasificación, regresión y clusterización entre los cuales encontramos algunos como máquinas de vectores de soporte o *K-means*. Scikit-Learn está desarrollada sobre licencia de software libre y su origen se remonta a 2007 cuando David Cournapeau inició el proyecto en el *Google Summer of Code* de ese año. Actualmente es mantenida por la comunidad.

En la aplicación utilizamos los algoritmos de vectorización para obtener las matrices que relacionan las frases con las palabras que contienen a partir de los subtítulos.

3.3.10. Yolo

You Only Look Once[32] o YOLO es el actual estado del arte en sistemas de detección de objetos en tiempo real. Gracias al uso de redes neuronales convolucionales (Subsección 2.3.2) es capaz de procesar imágenes a 30 *frames* por segundo con un acierto medio de casi un 60 % en su versión 3 y a 65 *frames* por segundo con un acierto medio de casi un 45 % en su versión cuatro. YOLO trabaja sobre Darknet[5], un framework para redes neuronales de código abierto escrito en C y CUDA. La primera versión de YOLO se publicó en 2016 por investigadores de la universidad de Washington, el Allen Institute for AI y Facebook AI Research.

En el trabajo utilizamos la versión 3 de YOLO como se explica en la Subsección 2.3.3, puesto que la versión 4 se publicó este mismo año ya con este trabajo en desarrollo. Lo utilizamos para la detección de los objetos que aparecen en los distintos *frames* que se van a analizar para obtener la información necesaria para nuestro resumen.

CAPÍTULO 4
Análisis de la interfaz

En este capítulo se realiza un análisis de la interfaz de la aplicación, explicando las opciones que esta ofrece al usuario.

Ventana principal

Al iniciar la aplicación se nos muestra la ventana principal (figura 4.1). Esta nos ofrece las siguientes opciones:

1. *Load vídeo*: Permite al usuario seleccionar el vídeo que quiere resumir lanzando la ventana de selección de archivos.
2. *Reset config*: Permite reiniciar la configuración de la aplicación a las configuraciones por defecto, ya que el sistema guarda el estado de la aplicación en la ultima ejecución aunque la aplicación se cierre.
3. *Next*: Permite avanzar a la siguiente ventana y se activa si se ha seleccionado un vídeo de entrada y este es formato mp4.

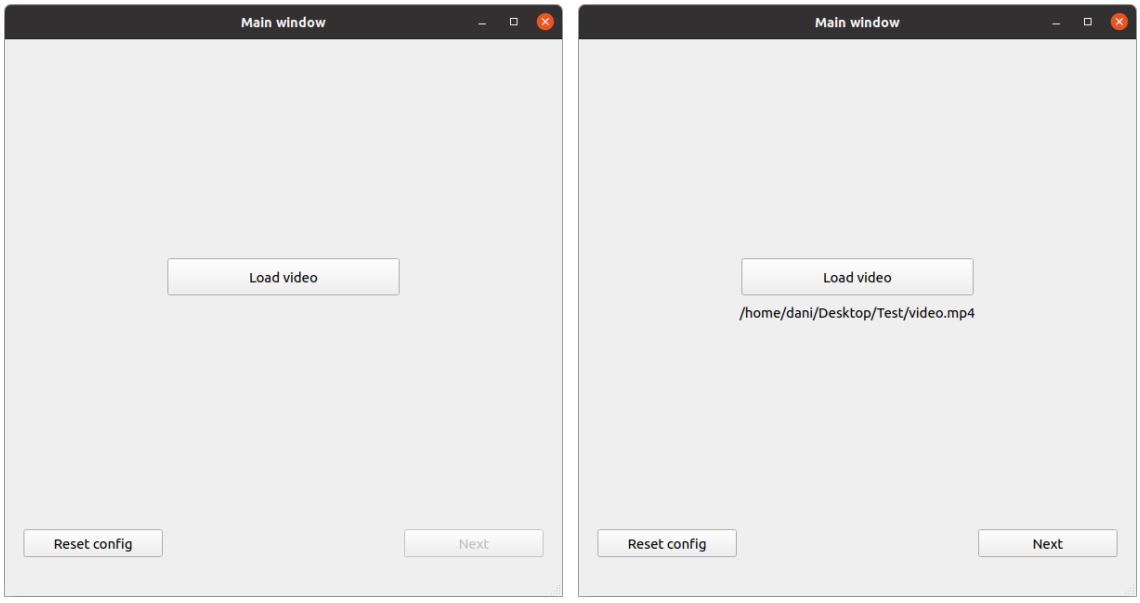

Figura 4.1: Ventana principal de la aplicación.

Opciones generales

Una vez configurado el vídeo de entrada, se avanza a la ventana siguiente donde se configuran las opciones más generales del resumen (figura 4.2). En esta se muestran varias opciones:

1. *Resume mode*: permite seleccionar el modo de resumen que se desea emplear, ya sea solo teniendo en cuenta los objetos, solo los subtítulos o en función de los dos.

2. *Scenes detection*: permite activar o desactivar la detección de escenas, lo cual va a permitir adaptar el resumen a los cambios de escena y optimizar la detección de objetos.

3. *Scene difference*: permite configurar el porcentaje mínimo de diferencia que debe haber entre dos *frames* para considerar un cambio de escena.

4. *Previous*: permite volver a la ventana principal guardando la configuración actual.

5. *Next*: permite avanzar a la ventana siguiente iniciando el proceso de detección de escenas si esta opción está activada y redirigiendo a la ventana de opciones de objetos o de subtítulos en función del modo del resumen.

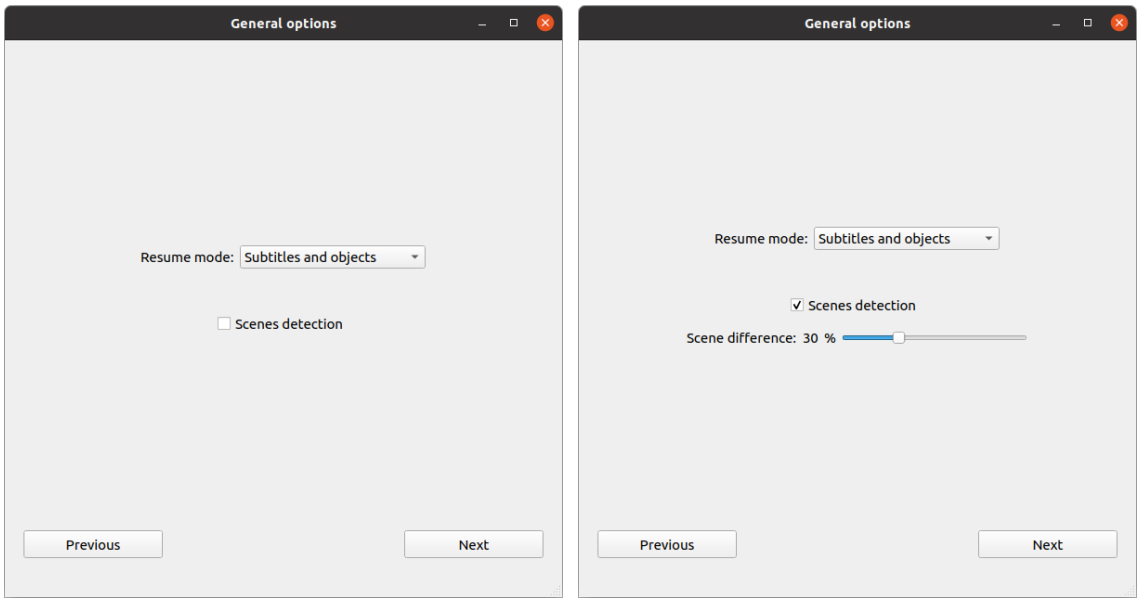

Figura 4.2: Ventana de configuraciones generales.

Opciones de objetos

En caso de que el modo de resumen incluya la detección de los objetos del vídeo, se muestra la ventana de configuración del análisis de objetos (figura 4.3). En esta se muestran multitud de opciones:

1. *Objects to detect*: lista en la que podemos añadir y borrar los objetos que queremos que la aplicación detecte para incluirlos en el resumen.

2. *Optimization per scenes*: permite activar la optimización por escenas del análisis, ahorrando una gran cantidad de tiempo a cambio de un ligero aumento en la fiabilidad.

3. *Nº of Analysis*: en caso de activar la optimización por escenas, se permite ajustar el número de *frames* que se tienen que analizar por escena.

4. *Periodicity*: si la optimización por escenas está desactivada, permite configurar cada cuantos milisegundos queremos analizar los *frames*.

5. *Select Yolo weights*: permite seleccionar el fichero '.weights' de YOLO con los pesos de la red a utilizar.

6. *Select Yolo cfg*: permite seleccionar el fichero '.cfg' de YOLO con la configuración de la red a utilizar.

7. *Select Yolo names*: permite seleccionar el fichero '.names' de YOLO con la lista de nombres de los objetos que la red es capaz de detectar.

8. *Previous*: permite volver a la ventana anterior guardando la configuración seleccionada, pero forzando el reinicio del proceso de análisis de escenas.

9. *Next*: si la configuración es correcta, permite seguir el flujo de la aplicación a la siguiente ventana, ya sea la de análisis de subtítulos o la del resumen de las opciones según el modo de resumen, lanzando el análisis de objetos en segundo plano.

Se muestra también la barra de progreso del proceso de análisis de escenas en caso de que este se haya configurado en la ventana anterior.

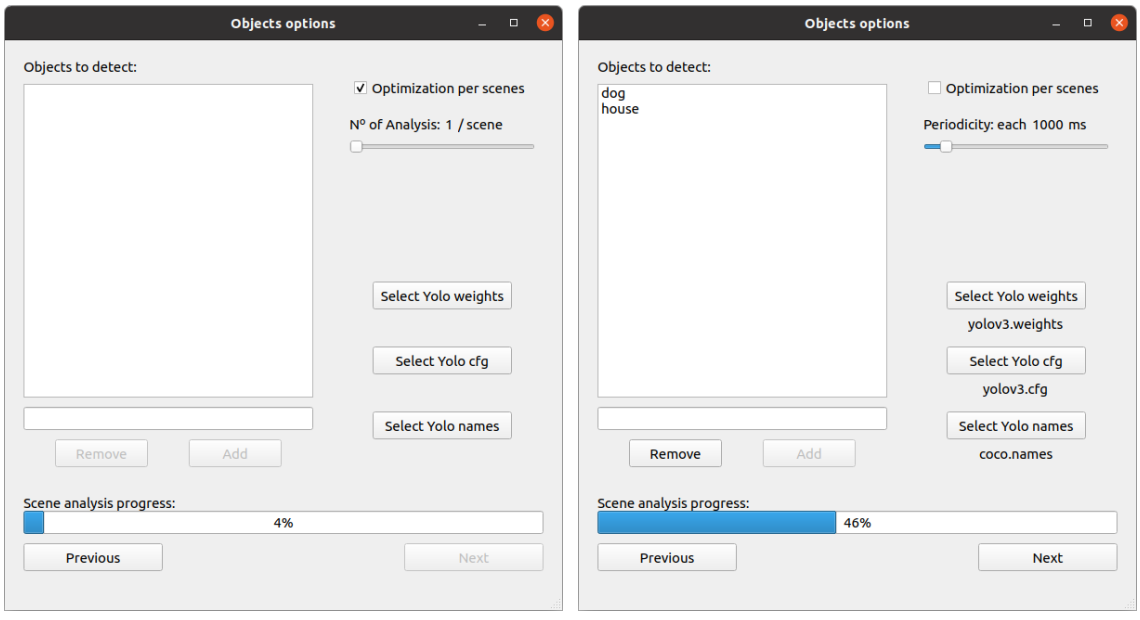

Figura 4.3: Ventana de configuración del análisis de objetos.

Opciones de subtítulos

Si el modo de resumen incluye los subtítulo, se mostrará la ventana de configuración del análisis de subtítulos (figura 4.4). Encontramos estas opciones:

1. *Load subtitles*: permite seleccionar los subtítulos que vamos a analizar.
2. *Vectoring type*: permite seleccionar el tipo de vectorización que se va a utilizar para analizar los subtítulos.
3. *Summary percentage*: permite seleccionar en qué porcentaje queremos resumir los subtítulos.
4. *Remove capital letters*: permite la conversion de los subtítulos a minúsculas.
5. *Remove stop words*: permite eliminar las *stop words* de los subtítulos.
6. *Language*: en caso de activar el borrado de *stop words* se puede seleccionar el idioma de los subtítulos, pues varían según el idioma.
7. *Remove accents*: permite eliminar los acentos de los subtítulos.
8. *Remove punctuations*: permite eliminar los signos de puntuación de los subtítulos.
9. *Previous*: permite volver a la ventana anterior guardando la configuración seleccionada, pero forzando el reinicio del proceso de análisis de objetos.
10. *Next*: si la configuración es correcta, permite avanzar a la siguiente ventana activando el proceso de análisis de subtítulos en segundo plano.

En la parte inferior se muestran las barras de progreso de los procesos de análisis de escenas y análisis de objetos si están activados.

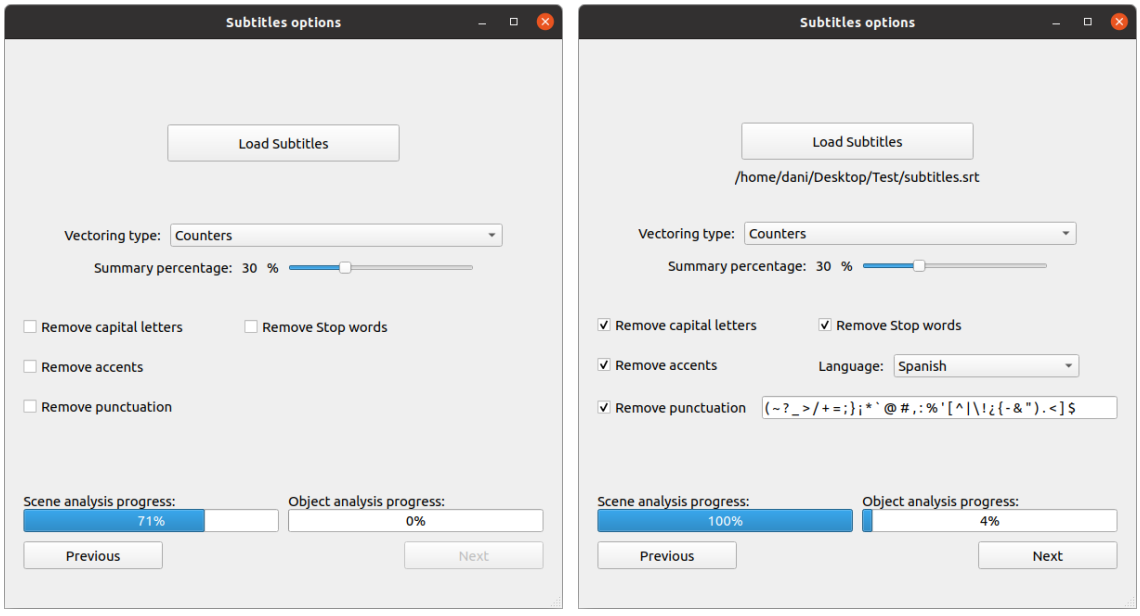

Figura 4.4: Ventana de configuración del análisis de subtítulos.

Resumen de opciones

Una vez configurados los distintos análisis que queremos que se realicen a nuestra aplicación, se nos muestra la ventana de resumen (figura 4.5). En ella podremos ver y confirmar que las configuraciones son correctas. Se ofrecen dos opciones:

1. *Previous*: permite volver a la ventana anterior, pero forzando el reinicio del proceso de análisis de objetos o subtítulos, según el modo de resumen.
2. *Next*: permite avanzar a la ventana de resumen, una vez hayamos acabado de revisar las configuraciones, activando el proceso de resumen.

Se muestran en la parte inferior las barras de los procesos en segundo plano de análisis de escenas, análisis de objetos y análisis de subtítulos que estén activados según el modo del resumen.

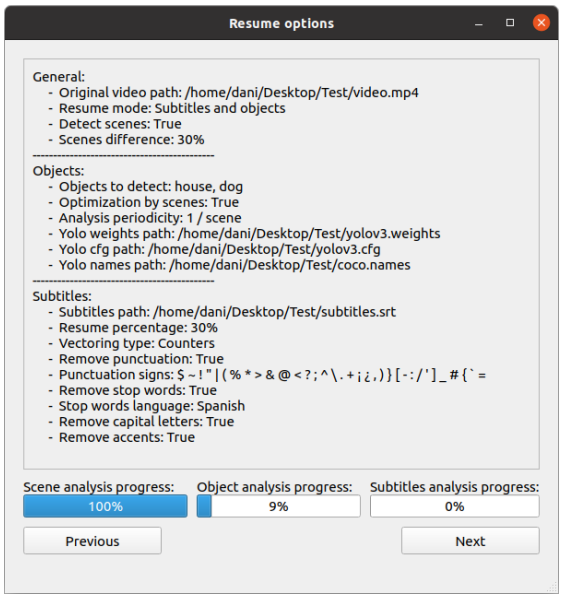

Figura 4.5: Ventana de resumen de las configuraciones.

Ventana de resumen

Una vez se confirman las configuraciones del resumen se avanza a la ventana de resumen (figura 4.6). En ella se nos muestran las barras de progreso de los procesos que se estén ejecutando en segundo plano, o estén activados esperando que se terminen los procesos de los que dependen. Se ofrecen dos opciones:

1. *Previous*: permite volver a la ventana anterior, pero forzando el reinicio del proceso de resumen.
2. *Next*: una vez el resumen se ha completado, permite avanzar a la pagina de descarga del resumen.

Una vez pasemos a la ventana siguiente no se podrá volver, ya que se considera finalizado el resumen.

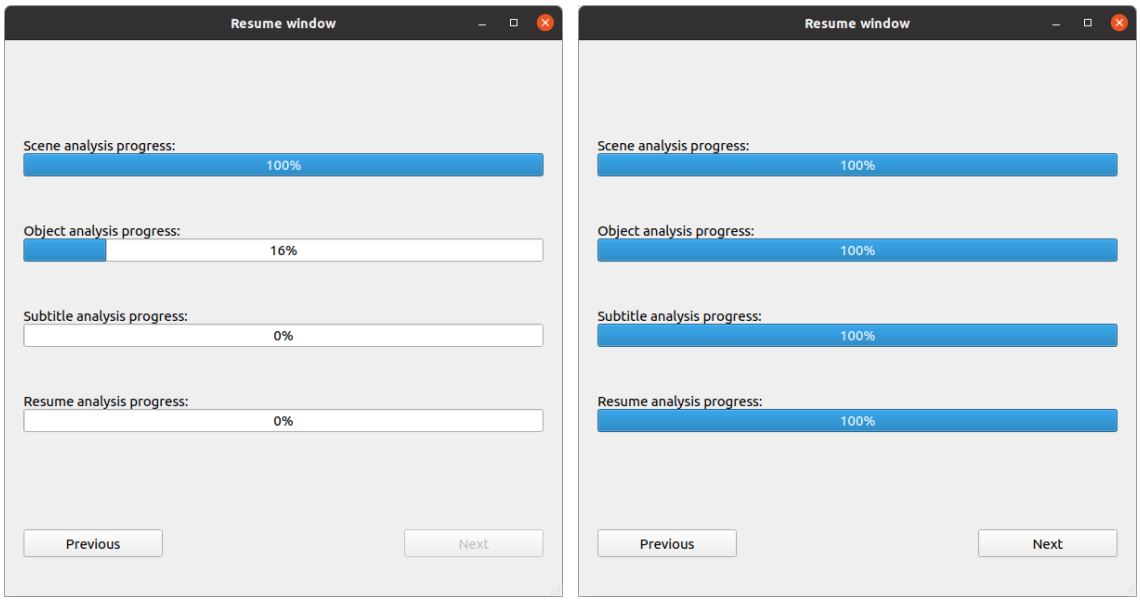

Figura 4.6: Ventana de resumen del vídeo.

Resultado del resumen

Finalmente, una vez se ha completado el resumen, se muestra la ventana final (figura 4.7). En ella se ofrece una única opción:

1. *Download*: permite seleccionar la ruta en la que queremos guardar el vídeo resumido y, si esta es correcta, lanza el proceso de guardado del vídeo en segundo plano.

Una vez se lanza el proceso de guardado, se muestran tres barras de progreso: el progreso del cortado del vídeo, el progreso del guardado del audio, y el proceso de guardado el vídeo. Una vez completado, el resumen estará en la ruta indicada.

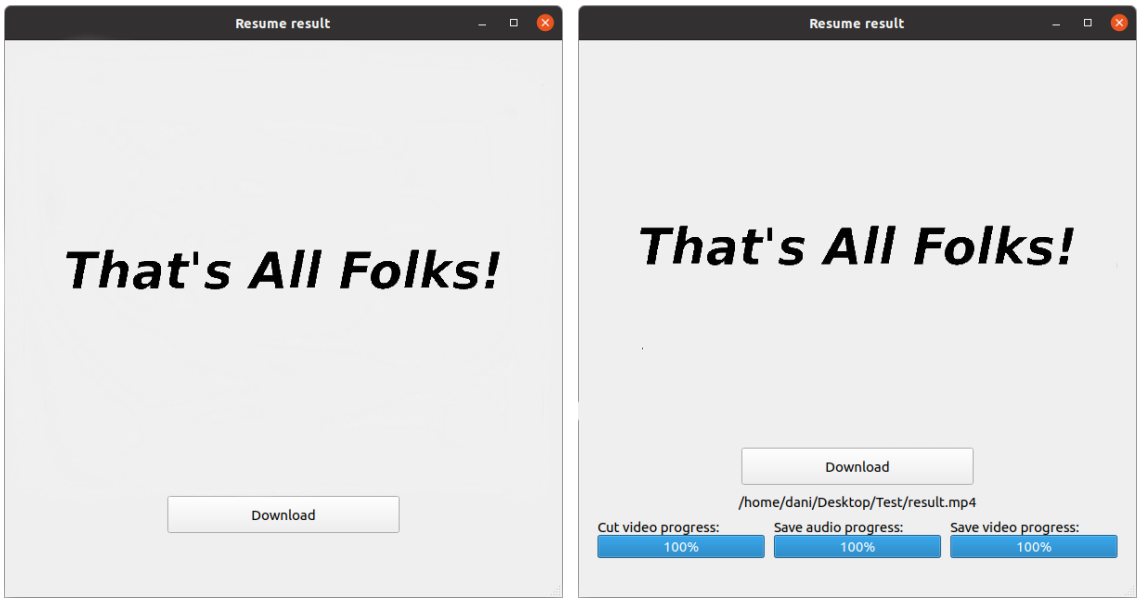

Figura 4.7: Ventana de descarga del vídeo resumido.

CAPÍTULO 5
Pruebas

El resumen de vídeos no es una tarea sencilla. Aparte de la evidente complejidad de esta tarea, nos enfrentamos, también, a la subjetividad humana. Es lógico pensar que dos personas no harán el mismo resumen partiendo de un mismo vídeo. Esto es debido a que los humanos somos subjetivos.

Esta subjetividad es la mayor dificultad a la hora de probar la aplicación que hemos desarrollado. No podemos medir el nivel de éxito del resumen sin tener que recurrir a un enorme estudio estadístico en el cual pongamos a prueba a muchas personas resumiendo el mismo vídeo para compararlo con el resumen realizado por la aplicación.

Realizar este tipo de pruebas sería muy costoso en cuanto a tiempo y recursos, y excedería el alcance de este proyecto. Es por esto que en este capítulo no vamos a intentar medir la calidad global del resumen, sino que vamos a intentar medir partes más concretas de la aplicación. Estas van a ser la detección de escenas, la detección de objetos y la optimización por escenas.

5.1 Detección de escenas

La aplicación es capaz de basar su resumen en las distintas escenas que aparecen en el vídeo. Para esto necesitamos detectar los cambios de escena tal y como se explica en la Sección 2.2.

Para determinar en qué punto del vídeo hay un cambio de escena se calcula la diferencia entre dos *frames*. El porcentaje mínimo de diferencia entre dos *frames* para considerar el cambio de escena puede ser configurado por el usuario.

En esta prueba se escoge un vídeo y se ejecuta la detección de escenas realizando variaciones de este porcentaje, viendo así cuantas escenas se obtienen en función de dicho valor. Para la prueba se ha utilizado un vídeo de 5 minutos con un gran número de cambios de escena para probar correctamente dicha función. Una vez obtenidas las distintas escenas se han revisado manualmente para confirmar si realmente hay un cambio de escena o no. Este vídeo es un corte de 5 minutos del documental 'El rey de la sabana' de Televisión Española.

Tabla 5.1: Resultados de la prueba de detección de escenas.

	Diferencia mínima	# Escenas	# Correctas	# Incorrectas	Acierto
P1	10 %	126	89	37	70 %
P2	20 %	93	82	11	88 %
P3	30 %	84	75	9	89 %
P4	40 %	67	60	7	90 %
P5	50 %	44	42	2	95 %

En la tabla 5.1 podemos observar cómo varía la cantidad de escenas que detectamos en función del porcentaje mínimo de diferencia entre *frames* que le indicamos a la aplicación.

De los resultados se puede concluir que cuanto mayor es el porcentaje de diferencia mínimo, necesario para detectar un cambio de escena, mayor acierto en el cambio de escenas obtenemos, pero sacrificando el número de escenas totales que detectamos. Se nota un aumento del acierto considerable al cambiar entre un 10 % y un 20 %, siendo el resto de pruebas similares en acierto.

5.2 Detección de objetos

La detección de objetos es una funcionalidad importante de la aplicación. Además es fácil de medir su correcto funcionamiento. Para ello vamos a probar con un vídeo la detección de objetos variando los parámetros configurables y posteriormente realizaremos el visionado de los resultados comprobando manualmente que sean correctas las detecciones de los objetos.

Es importante tener en cuenta los dos factores que pueden influir en los resultados de esta prueba. La primera es el porcentaje medio de acierto de YOLOv3, pues es el encargado de detectar los objetos en escena. La segunda es la optimización del análisis por escenas, pues con esta opción activada es previsible que el error pueda aumentar. Frente al primer factor no podemos realizar variaciones, pero sí vamos a probar con la optimización por escenas tanto activada como desactivada.

En esta prueba pretendemos obtener la sensibilidad (porcentaje de objetos indicados por el usuario que se muestran correctamente en el resumen) y la especificidad (porcentaje de objetos indicados por el usuario sobre el total de objetos que se muestran en el resumen) de la aplicación en la detección de escenas con las distintas configuraciones posibles. Para ello vamos a utilizar una metodología estadística que consiste en realizar un conteo de las distintas posibilidades:

1. Verdaderos positivos (VP): se trata de los objetos que aparecen y son bien detectados por nuestra aplicación.

2. Falsos positivos (FP): se trata de los casos en los que la aplicación detecta el objeto pero no se encuentra en la imagen.

3. Verdaderos negativos (VN): se trata de los casos en los que no aparecía dicho objeto y la aplicación no lo ha detectado.

4. Falsos negativos (FN): se trata de los casos en que aparece el objeto en cuestión pero la aplicación no lo detecta.

Para la prueba se utiliza el documental de Televisión Española llamado 'El rey de la sabana' al completo y le pedimos a la aplicación que nos muestre solo las partes del documental en el que aparecen zebras. La duración de dicho documental es de 55 minutos aproximadamente.

Tabla 5.2: Resultados de la prueba de detección de objetos.

	Análisis	**FA**	**VP**	**FP**	**VN**	**FN**	**Sensibilidad**	**Especificidad**
P1	10/seg	33240	193	51	3058	22	0.72 %	0.99 %
P2	1/seg	3324	154	24	3085	61	0.72 %	0.99 %
P3	1/escena	543	175	69	3040	40	0.81 %	0.98 %
P4	2/escena	1086	181	95	3014	34	0.84 %	0.97 %
P5	3/escena	1629	195	110	2999	20	0.91 %	0.96 %

En la tabla 5.2 podemos observar los resultados obtenidos con la aplicación variando la frecuencia con la que se hace la detección de objetos. Se han realizado dos pruebas con una periodicidad del análisis en segundos y tres con la optimización por escenas activada. FA hace referencia al número de *frames* que se han analizado en total. La prueba se ha realizado con un 30 % como diferencia mínima para considerar el cambio de escena.

A partir de los resultados obtenidos se aprecian varias cosas:

1. La aplicación es capaz de detectar la mayoría de las zebras, con una sensibilidad superior al 70 % en todas las pruebas y aumentando dicha sensibilidad aumenta al activar la optimización por escenas. Esto se debe a que al mostrar toda la escena, la cantidad de zebras mostradas correctamente aumenta, al contrario, por ejemplo del caso del análisis de un frame por segundo, el cual, en caso de no aparecer una zebra en dicho *frame*, el resto del segundo desaparece del resultado.

2. Por otro lado, la especificidad es muy alta, pero disminuye entre un 1 % y un 3 % al aplicar la optimización por escenas, pues al mostrar toda la escena, mostramos también *frames* en los que no aparece ninguna zebra.

5.3 Optimización por escenas

Uno de los mayores problemas a la hora de detectar los objetos en el vídeo es la gran cantidad de tiempo que esto requiere. Es por ello que en la aplicación se ha añadido la posibilidad de optimizar esta detección utilizando la información que nos ofrece la detección de escenas.

En esta prueba se quiere medir la diferencia de duración del análisis de objetos en función de la frecuencia de análisis que indicamos y de la activación o no de dicha optimización. Para esta prueba se trabaja con el mismo vídeo del apartado anterior.

Tabla 5.3: Resultados de la prueba de optimización por escenas.

	Análisis	FA	Duración	Sensibilidad	Especificidad
P1	10/seg	33240	01:53:36	0.72 %	0.99 %
P2	1/seg	3324	00:28:09	0.72 %	0.99 %
P3	1/escena	543	00:03:27	0.81 %	0.98 %
P4	2/escena	1086	00:05:59	0.84 %	0.97 %
P5	3/escena	1629	00:09:00	0.91 %	0.96 %

En la tabla 5.3 podemos observar los resultados obtenidos y el tiempo que ha tardado en ejecutarse el análisis variando la frecuencia con la que se hace la detección de objetos. FA hace referencia al número de *frames* que se han analizado en total. La prueba se ha realizado con un 30 % como diferencia mínima para considerar el cambio de escena.

De los resultados de esta prueba se puede comprobar cómo la optimización por escenas disminuye considerablemente el tiempo necesario para realizar la detección de objetos manteniéndose en la línea de sensibilidad y especificidad de los análisis de fuerza bruta e incluso mejorando los resultados de estos.

CAPÍTULO 6
Conclusión

En este último capítulo se va a hacer una recapitulación de los objetivos que se marcaban en este trabajo y de lo que se ha conseguido. Además, se añadirán algunas ideas para los trabajos futuros que podrían surgir de este.

En este trabajo se marcaba como objetivo el crear una aplicación con interfaz de usuario, capaz de, a partir de un video proporcionado por el usuario y unos parámetros indicados por este, realizar un resumen que se adaptase a las preferencias de dicho usuario. Para ello se pretendía basar el resumen en las escenas del vídeo, los subtítulos de este y los objetos que aparecen en el.

En cuanto a la aplicación, se logra desarrollar una aplicación sobre PyQt con una interfaz consistente y con una buena gestión de hilos, asi como la monitorización del estado de los distintos procesos que se ejecutan en segundo plano gracias a las distintas barras de progreso que se muestran en la interfaz.

Por otra parte, en el resumen se logran buenos resúmenes basados en los subtítulos, con unos cortes muy naturales gracias a la utilización de las escenas como unidades mínimas de resumen y la posibilidad de mostrar ciertos objetos que el usuario considere importante.

Además, se ha desarrollado una optimización en la detección de objetos que reduce significativamente el tiempo que la aplicación necesita para analizar todo el vídeo en busca de los objetos indicados por el usuario.

Finalmente la aplicación es capaz de cortar el vídeo original para formar el resumen y guardar este en una ruta indicada por el usuario a través de la interfaz.

Por último, el código esta disponible en el repositorio público de Github[29].

6.1 Trabajos futuros

Para concluir este trabajo, se comentan algunos de los posibles trabajos futuros que se podrían realizar siguiendo en la línea de este trabajo:

1. Proximidad entre objetos detectados: se podría ampliar la información que se le transmite al proceso que realiza el resumen realizando otro análisis previo a dicho resumen que consista en, a partir del tiempo en que aparecen en pantalla (es decir, en el video) intentar calcular la proximidad entre

los objetos, ya sean estos personas, si se logra un reconocimiento facial adecuado, o simplemente entre objetos para obtener patrones que el usuario pueda indicar como importantes para el resumen.

2. Sonidos fuertes: podríamos añadir también la detección de sonidos fuertes como música, explosiones o demás sonidos que suelen indicar que dicha parte del vídeo es importante.

3. Relación entre personajes: en caso de poder hacer uso de un adecuado reconocimiento facial y querer enfocarse en vídeos que muestren una relación entre distintas personas, se podría intentar obtener una representación de la relación que tienen los personajes entre sí a partir de los sentimientos que expresan las frases que pronuncian cada uno de ellos cuando hablan, pudiendo así obtener qué personajes mantienen una relación de amistad o una relación de odio, pudiendo incluso detectar las variaciones en estas relaciones y así fortalecer el resumen al poder mostrar partes del vídeo que muestren ese avance en la relación.

4. Transcripción a partir del audio: se podría, en lugar de pedir la transcripción del vídeo al usuario a través de los subtítulos, realizar la transcripción a partir del audio del vídeo, pudiendo incluso, separando las distintas tonalidades de voz, obtener qué persona está diciendo dicha frase para así profundizar en el punto anterior de las relaciones entre personajes.

5. Detección de escenarios: podríamos también intentar detectar qué escenas pertenecen al mismo entorno o escenario, a través de librerías que transcriben, por ejemplo, lo que hay en escena, pudiendo así obtener todas las escenas pertenecientes al interior de una casa o a un parque aunque estas no vayan seguidas y pudiendo obtener una cronología de los sitios por los que avanza el vídeo y los retornos que se hacen a dichos sitios.

Bibliografía

[1] Digital 2020, global digital overview. Technical report, We Are Social, 2020.

[2] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[3] William B Cavnar, John M Trenkle, et al. N-gram-based text categorization. In *Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval*, 1994.

[4] Qt Creator. https://www.qt.io/product/development-tools.

[5] Darknet. https://pjreddie.com/darknet.

[6] Alberto Donet Olmeda. *Sistema de resúmenes de vídeos*. PhD thesis, 2016.

[7] FFmpeg. https://ffmpeg.org.

[8] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2014.

[9] Gene H Golub and Christian Reinsch. Singular value decomposition and least squares solutions. In *Linear Algebra*. 1971.

[10] Thomas Hofmann. Probabilistic latent semantic analysis. *arXiv preprint arXiv:1301.6705*, 2013.

[11] IntelliVision. https://www.intelli-vision.com/video-summary.

[12] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, 2012.

[13] Barry M Leiner, Vinton G Cerf, David D Clark, Robert E Kahn, Leonard Kleinrock, Daniel C Lynch, Jon Postel, Larry G Roberts, and Stephen Wolff. A brief history of the internet. *ACM SIGCOMM Computer Communication Review*, 2009.

[14] MoviePy. https://zulko.github.io/moviepy.

[15] NumPy. https://numpy.org.

[16] OpenCV. https://opencv.org.

[17] PyQt. https://www.riverbankcomputing.com/software/pyqt.

[18] PySide. https://wiki.qt.io/Qt_for_Python.

[19] Pysrt. https://github.com/byroot/pysrt.

[20] Python. https://www.python.org.

[21] Qt. https://www.qt.io.

[22] Juan Ramos et al. Using tf-idf to determine word relevance in document queries. In *Proceedings of the first instructional conference on machine learning*, 2003.

[23] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016.

[24] Brian D Ripley. *Pattern recognition and neural networks*. Cambridge university press, 2007.

[25] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, et al. Imagenet large scale visual recognition challenge. *International journal of computer vision*, 2015.

[26] Scikit-Learn. https://scikit-learn.org.

[27] SciPy. https://www.scipy.org.

[28] Josef Steinberger and Karel Jezek. Using latent semantic analysis in text summarization and summary evaluation. *Proc. ISIM*, 2004.

[29] Video Summary. https://github.com/untalturner/VideoSummary.

[30] Neural Language Toolkit. https://www.nltk.org.

[31] Adhika Pramita Widyassari, Supriadi Rustad, Guruh Fajar Shidik, Edi Noersasongko, Abdul Syukur, Affandy Affandy, et al. Review of automatic text summarization techniques & methods. *Journal of King Saud University-Computer and Information Sciences*, 2020.

[32] YOLO. https://pjreddie.com/darknet/yolo.

[33] Wen Zhang, Taketoshi Yoshida, and Xijin Tang. A comparative study of tf* idf, lsi and multi-words for text classification. *Expert Systems with Applications*, 2011.