

Reconocimiento de conjuntos de caras de actores extraídos de fragmentos de películas mediante Python y SSH

Trabajo Final de Grado

Grado en Multimedia

Apellidos: Ordóñez Brumós Nombre: Roger

Plan: 2009

Director: Raventós Mayoral, Arnau

**Índex**

[Resumen 5](#_Toc18454908)

[Palabras clave 6](#_Toc18454909)

[Enlaces 6](#_Toc18454910)

[Índex de tablas 7](#_Toc18454911)

[Índex de figuras 8](#_Toc18454912)

[Glosario 9](#_Toc18454913)

[Agradecimientos 10](#_Toc18454914)

[1. Introducción 11](#_Toc18454915)

[1.1 Motivación 13](#_Toc18454916)

[1.2 Formulación del problema 14](#_Toc18454917)

[1.3 Objetivos generales del TFG 15](#_Toc18454918)

[1.4 Objetivos específicos del TFG 15](#_Toc18454919)

[1.5 Alcance del proyecto 16](#_Toc18454920)

[2. Estado del arte 17](#_Toc18454921)

[2.1 Detección de caras 18](#_Toc18454922)

[2.2 Reconocimiento de caras 20](#_Toc18454923)

[2.3 Estudio de Mercado 21](#_Toc18454924)

[3. Gestión del proyecto 22](#_Toc18454925)

[3.1 Procedimiento y Herramientas para el seguimiento del proyecto 22](#_Toc18454926)

[3.1.1 GANTT 22](#_Toc18454927)

[3.1.2 Trello 22](#_Toc18454928)

[3.1.3 GitHub 23](#_Toc18454929)

[3.2 Herramientas de validación 24](#_Toc18454930)

[3.3. DAFO 24](#_Toc18454931)

[3.4. Riesgos y plan de contingencias 25](#_Toc18454932)

[3.5. Análisis inicial de costes 25](#_Toc18454933)

[4. Metodología 26](#_Toc18454934)

[5. Desarrollo del proyecto 27](#_Toc18454935)

[5.1 Búsqueda y selección de películas 27](#_Toc18454936)

[5.2 Procesado de las películas mediante API 28](#_Toc18454937)

[5.2.1 Python 29](#_Toc18454938)

[5.2.1.1 Librería de Python: Keras 30](#_Toc18454939)

[5.2.1.2 Librería de Python: Mxnet\_cu92 30](#_Toc18454940)

[5.2.1.3 Librería de Python: Cython 30](#_Toc18454941)

[5.2.1.4 Librería de Python: Numpy 31](#_Toc18454942)

[5.2.1.5 Librería de Python: Setuptools 31](#_Toc18454943)

[5.2.1.6 Librería de Python: Tensorflow\_gpu 31](#_Toc18454944)

[5.2.1.7 Librería de Python: Opencv\_python 31](#_Toc18454945)

[5.2.1.8 Librería de Python: Tqdm 32](#_Toc18454946)

[5.2.1.9 Librería de Python: Pillow 32](#_Toc18454947)

[5.2.1.10 Librería de Python: Logger 32](#_Toc18454948)

[5.2.1.11 Librería de Python: Mxnet 32](#_Toc18454949)

[5.2.1.12 Librería de Python: Tensorflow 32](#_Toc18454950)

[5.2.2 Resultados 33](#_Toc18454951)

[5.3 Extracción de artistas de las diferentes películas 35](#_Toc18454952)

[5.3.1 IMDB 35](#_Toc18454953)

[5.3.2 Themoviedb.org 35](#_Toc18454954)

[5.3.3 Simkl 35](#_Toc18454955)

[5.3.4 Comparativa y resultados 36](#_Toc18454956)

[5.4 Creación de una base de imágenes de actores 37](#_Toc18454957)

[5.5 Análisis y comparación de actores con las imágenes extraídas 38](#_Toc18454958)

[5.5.1 Extracción de características faciales de las imágenes extraídas 38](#_Toc18454959)

[5.5.2 Generación de recursos para comparación de las imágenes extraídas 40](#_Toc18454960)

[5.5.3 Generación de recursos clúster 40](#_Toc18454961)

[5.5.4 Comparación facial 41](#_Toc18454962)

[5.6 Fase de testeo y comprobación 51](#_Toc18454963)

[6. Conclusiones y trabajos futuros 53](#_Toc18454964)

[7. Bibliografía 54](#_Toc18454965)

[8. Anexos 55](#_Toc18454966)

# Resumen

Este proyecto de final de grado consta del desarrollo de un software que permite a partir de una Api que extrae de forma automática imágenes de los actores que participan en una película, comprobar de que actores se trata a partir de un conjunto de scripts en Python que permiten reconocerlos mediante el uso de modelos de reconocimiento facial SSH. Estos archivos se proporcionan en los anexos como una base para su uso en otros proyectos.

# Palabras clave

Python, SSH, detección facial, reconocimiento facial, open source, tensorflow.

# Enlaces

<https://github.com/arkasarius/python-IMDB-TFG>

# Índice de tablas

Tabla 1: DAFO del trabajo de final de grado. Página 24

Tabla 2: Tabla de riesgos y contingencias del proyecto de final de grado. Página 25

# Índice de figuras

Figura 1 Representación de convolución. Página 19

Figura 2: Clúster artista 00 Yo robot (Will Smith). Página 28

Figura 3: Muestra de errores típicos en extracción de caras. Página 34

Figura 4: Muestra de código de IMDBPY. Página 36

Figura 5: Muestra de algunos de los actores de Matrix. Página 36

Figura 6: implementación de descarga de imágenes de actores. Página 37

Figura 7: muestra de imágenes de Keanu Reeves. Página 38

Figura 8: Representación de dos caras V y W en un espacio vectorial N = 128. Página 44

Figura 9: Verdaderos positivos de Linda Fiorentino según coeficiente DM. Página 45

Figura 10: Verdaderos positivos de Rip Torn según coeficiente DM. Página 46

Figura 11: Imagen del sub set de comprobación para el actor Rip Torn. Página 47

Figura 12: Verdaderos positivos de Linda Fiorentino según coeficiente AM. Página 48

Figura 13: Verdaderos positivos de Linda Fiorentino según coeficientes AM

y DM. Página 49

Figura 14: Falsos positivos de Linda Fiorentino según coeficientes AM y DM. Página 50

Figura 15: muestra de resultados de los posibles casos para la película

Ángeles y demonios. Página 52

# Glosario

Aprendizaje supervisado: técnica que permite entrenar un sistema a partir de unos parámetros de entrenamiento que permite devolver unos resultados en función de unos datos de entrada.

Aprendizaje semi supervisado: mismo principio que el aprendizaje supervisado que emplea datos para el entrenamiento etiquetados y no etiquetados.

Etiqueta: información relevante a un contenido concreto, por ejemplo para una película su género o duración.

Machine Learning: sub campo de las ciencias de la computación dentro de la rama de inteligencia artificial. Pertenecen a este los diferentes tipos de aprendizajes citados previamente.

Frame: fotograma o captura de una secuencia de video.

Api: software que devuelve una respuesta en forma de datos a una llamada por parte de una petición.

Base de datos: conjunto de datos organizados en con un mismo criterio, por ejemplo un diccionario.

Algoritmo: conjunto de reglas o instrucciones definidas en un lenguaje de programación

Biometría: características representativas que definen de manera concreta a un ser vivo.

Clúster: conjunto de archivos relacionados dentro de un contexto por un algoritmo automatizado.

Repositorio: banco de datos en la nube con capacidad de seguimiento en el tiempo de los archivos que pertenecen a este.

XML: lenguaje que permite almacenar datos de forma legible en un archivo de texto para su uso como base de datos de texto.

Python: lenguaje de programación

Red neuronal: modelo computacional de un algoritmo de inteligencia artificial.

Framework: entorno de trabajo.

Código abierto: modelo de desarrollo de software que permite acceso de este de manera gratuita.

C / C++: lenguaje de programación

Ruido: muestras de datos que no pertenecen a un conjunto concreto de archivos o procesos que diluye la pureza de la muestra

# Agradecimientos

Quiero dedicar este apartado para agradecer:

Primero, a Arnau Raventós Mayoral, director de este Trabajo de final de grado, por su inestimable propuesta de Trabajo de final de grado, su confianza, disposición y guía en la realización del Trabajo de final de grado.

A mis padres, que me han apoyado durante la etapa de estudios.

A mis amigos, por apoyarme en los momentos difíciles durante el transcurso de este trabajo de final de grado.

A vosotros,

Gracias.

# 1. Introducción

Estamos viviendo en un momento de la historia donde cada vez más estamos más conectados entre nosotros gracias a la tecnología. En poco más de quince años hemos pasado de una sociedad donde teníamos internet solo en casa a poder disponer de dispositivos móviles que nos conectan constantemente con internet i la información allí donde vamos.

Durante este proceso las persones cada vez han tenido un acceso más fácil y rápido a la información, con más contenidos y de mayor calidad, sobretodo en cuanto a material multimedia. La tecnología ha evolucionado a gran velocidad permitiendo dispositivos con mayor capacidad para almacenar archivos y a su vez aumentando la velocidad de las conexiones entre dispositivos.

Esta facilidad que nos ofrece este avance en tecnología ha implicado que aumente la demanda de contenidos multimedia, provocando que se genere una cantidad de información muy grande. En un día en internet se publica más contenido multimedia del que puede consumir una persona en toda su vida. Esto implica que hay mucha información entre la que escoger y encontrar aquello que queremos usar entre todo el conjunto de contenidos disponibles es una de las principales tareas que ofrecen diferentes tipos de buscadores inteligentes a partir de etiquetado de contenido.

El etiquetado en si nos permite clasificar diferentes contenidos para poder filtrarlos según nuestras necesidades de una manera rápida. Este etiquetado comúnmente se ha realizado de forma manual o a partir de mecanizaciones utilizando software diseñado para el etiquetado.

Este trabajo de final de grado propone implementar un sistema de indexación automática de contenidos audiovisuales, reconociendo los actores que participan en series o películas.

Para ello se analizara y se extraerá las caras de los actores a partir de videos de estos contenidos para generar librerías de caras. Estas librerías se utilizaran para detectar que actores son dentro del contexto de la película con la ayuda de la base de datos IMDB.

IMDB nos ofrece información referente a películas y con la cual podemos saber los actores que participan.

Este proceso está planteado a una escala reducida, conociendo inicialmente la película y actores que participan, creando un prototipo escalable que permita reconocer con precisión los actores con unos requisitos de procesado reducidos, tanto en cantidad de imágenes necesarias para procesar como en almacenamiento de esta información.

Este trabajo de final de grado está dividido en ocho apartados:

1. **Introducción:** Resumen de la problemática y contexto del proyecto.
2. **Estado del arte:** Referentes y tecnologías utilizadas actualmente que nos ayudan a resolver el problema.
3. **Gestión del proyecto:** Procesos y herramientas para la comunicación y gestión del proyecto.
4. **Metodología:** fases del proyecto que se realizaran para realizar el trabajo de final de grado.
5. **Desarrollo del proyecto:** Desarrollo de las fases del proyecto documentada.
6. **Conclusiones y trabajos futuros:** Información de cierre del trabajo con la visión personal de la utilidad y posibles aplicaciones futuras del trabajo de final de grado.
7. **Bibliografía**
8. **Anexos**

## 1.1 Motivación

El planteamiento de este proyecto surgió de una práctica durante unas sesiones de clase relacionadas con la biometría, tecnología de identificación basada en el reconocimiento de una característica física e intransferible de las personas, como por ejemplo, la huella digital.

En esta práctica generamos un sistema que comparaba diferentes imágenes y reconocía imágenes parecidas a partir de algoritmos sencillos y comparaciones de cuan diferentes eran las imágenes entre ellas.

Al considerar que esta área muy interesante y con mucha capacidad para crecer decidí ampliar mis conocimientos en sistemas de análisis de la información y Arnau Raventós Mayoral vio este interés y me propuso realizar el trabajo de final de grado bajo su tutela.

Actualmente hay compañías millonarias que analizan contenidos multimedia, pero sus sistemas de procesado y etiquetaje de estos contenidos estás aun mejorándose y no hay un techo en cuanto a un sistema perfeccionado que ofrezca una solución para etiquetar correctamente todo el contenido que se genera. Por eso mismo al ser una tecnología que se está investigando y mejorando da opción a entrar y aportar dentro del conjunto de trabajos que se están realizando actualmente en esta materia.

Por último, quería apartarme del resto de compañeros en la carrera y no quería generar más contenido multimedia sino una herramienta para poder analizar los contenidos que se generan y marcar un punto diferencial respecto a ellos.

## 

## 1.2 Formulación del problema

Actualmente en un día se genera y publica en internet más contenido multimedia del que es capaz de visualizar una persona en toda su vida. **[11]**

Una de las plataformas de video más conocidas, YouTube, recibe videos por parte de sus usuarios y genera en un día 4 millones de horas de contenido, equivalente a 452 años y medio al día.

Solo con el contenido que se genera en un día en YouTube ya se visualiza que es imposible para una persona revisar tal cantidad de contenidos y no es la única plataforma donde se generan.

Esta imposibilidad de revisar todo el contenido que se genera implica que se requiere de sistemas automáticos para poder comprobar estos contenidos y poder etiquetarlos para que los usuarios de las plataformas puedan encontrarlos y visualizarlos de forma rápida.

Estos sistemas de etiquetado están basados en aprendizaje supervisado **[12]** o semi-supervisado **[13]**, permitiendo generar un modelo que genera una función a partir de datos de entrenamiento. Esta función nos permite a partir de datos de entrada como videos recibir datos de salida, por ejemplo, etiquetas para clasificar estos videos.

Este trabajo de final de grado pretende generar un sistema para catalogar los actores que aparecen dentro de una película de forma automática utilizando algoritmos de inteligencia artificial ya existentes y fáciles de implementar gracias a APIs externas.

## 1.3 Objetivos generales del TFG

El objetivo general de este proyecto es la implementación de un sistema capaz de analizar y reconocer conjuntos de caras de actores extraídos de fragmentos de películas para etiquetarlos.

El proceso de etiquetado partirá del uso de algoritmos de Machine Learning, que permiten identificar de forma automática las caras de los actores en frames de películas utilizando datos de Apis externas como IMDB.

Para comprobar la validez del sistema se pretende analizar un conjunto reducido de películas para extraer los actores que participan y determinar quiénes son.

La validez del sistema se obtendrá de métricas extraídas en cuanto al porcentaje de acierto del sistema y del algoritmo en sí.

## 1.4 Objetivos específicos del TFG

Los objetivos específicos de este proyecto son:

1. Establecer una pequeña muestra de cinco películas para hacer pruebas con el algoritmo desarrollado.
2. Recopilar información sobre los actores que participan en las películas seleccionadas.
3. Implementar un sistema que genere automáticamente bases de datos de imágenes con las caras de los actores de las películas seleccionadas.
4. Utilizar Apis externas para analizar las películas seleccionadas y obtener clústeres no supervisados de caras similares.
5. Comparar las bases de datos de actores con los clústeres generados por el sistema no supervisado para reconocer a que caras corresponden los diferentes actores que participan.
6. Establecer y consolidar los conocimientos necesarios para resolver la problemática mediante la programación de un algoritmo capaz de solventar las necesidades previamente expuestas.

## 1.5 Alcance del proyecto

Este trabajo de final de grado pretende generar un algoritmo de análisis de películas para poder etiquetar información referente a los actores que participan en estas.

El algoritmo beneficiara a dos sectores diferentes de usuarios:

* Usuario que consume películas y quiere conocer más información referente a los actores que participan en ella. Actualmente para encontrar esta información se requiere de herramientas de terceros y búsquedas específicas que requieren conocimientos de datos como el nombre de los actores para encontrar esta información.
* Usuarios de plataformas de contenido que quieran catalogar sus series o películas de forma automática sin necesidad de emplear recursos humanos para ello.

La complejidad del proyecto presenta un conjunto de obstáculos en cuanto a refinar el prototipo debido a que las muestras serán limitadas y por lo tanto no se podrá garantizar en ningún caso un acierto del algoritmo del 100%.

# 2. Estado del arte

Este proyecto pretende automatizar la detección de actores dentro de películas. Para ello se requiere en primera instancia detectar si aparece un actor en una escena y encontrar algún rasgo característico que sirva para identificar de quien se trata y diferenciarlo del resto de actores. Para ello se puede emplear la detección facial.

La detección facial es una rama de la biometría, ciencia que mide las propiedades físicas y de comportamiento de los seres vivos.

La biometría nos permite identificar o verificar un individuo del resto. Las características físicas de cada individuo lo caracterizan y en la mayoría de casos no se comparten con otros individuos. El ejemplo más concreto para diferenciar dos individuos sería una comparación de ADN. En el ámbito de la tecnología cuotidiana tenemos sistemas que analizan nuestras características físicas como por ejemplo los sistemas de desbloqueo de huella dactilar de los dispositivos móviles.

Existen diferentes sistemas biométricos que permiten identificar y verificar individuos. Todos ellos requieren de unos requisitos para funcionar, como en el caso del análisis de ADN una muestra de sangre, una imposición intrusiva que requiere el sistema para funcionar. En el caso del ADN además se requiere de un tiempo elevado para procesar la muestra y de un coste elevado de recursos.

Estas complejidades hacen que estos sistemas tan precisos como en el caso del análisis de ADN nos impidan realizar análisis biométricos a gran escala y en tiempos reducidos.

En el caso de este trabajo de final de grado utilizamos un sistema biométrico que trabaja a partir de fotografías de caras para analizar las características de estas, requisito no intrusivo en comparación y con un coste muy pequeño ya que al tener a disposición las películas podemos extraer esta información.

La detección facial no requiere por tanto de muestras físicas de los individuos i no requiere de la voluntad de estos para ser capturada, un claro ejemplo de esto es su aplicación en sistemas de seguridad.

Algunos de los usos comunes de la detección de caras en seguridad son:

* Identificación de criminales
* Registro de asistencia de persones a eventos o en ámbitos laborales.
* Verificación de identidad en aeropuertos, aduanas, entre otros.

Diversas tareas en el ámbito del procesamiento de imagen en caras son:

* Detección de caras
  + La capacidad del modelo para encontrar una cara dentro del contexto de una imagen.
* Verificación de caras
  + La capacidad de discernir si dos caras pertenecen a la misma persona.
* Reconocimiento de caras
  + La capacidad de decir a quien pertenece una cara, por lo tanto identificando a ese individuo.
* Clustering de caras
  + La capacidad de encontrar caras con características parecidas dentro de un gripo de caras.

En este trabajo de final de grado se utilizara la detección de caras conjuntamente con el reconocimiento de caras para localizar los actores dentro de las películas y decir a quien pertenecen esas caras.

A continuación se dará una visión global del estado del arte de estos dos procedimientos.

## 2.1 Detección de caras

La detección facial se ocupa de encontrar una cara dentro de una imagen digital, si es que hay alguna **[3]**. Es un requisito previo de cualquier sistema automático que requiera localizar caras dentro de imágenes. La detección facial está especialmente diseñada para funcionar con imágenes singulares, pero para este proyecto se utilizara para extraer caras de secuencias de video, consiguiendo una muestra de caras más grande y precisa de los diferentes actores que aparecen en las películas.

La detección facial es el proceso que se estudia en el ámbito de la computación de información y tiene un recorrido extenso en la historia. Se podría decir que, desde que existen imágenes digitales, se ha analizado la información que contienen.

Los métodos de detección clásicos están basados en estudios como ***“detección rápida de objetos mediante características sencillas”*** **Viola, P. Jones, M 2004 [5]** donde se analizan caras mediante el uso de filtros secuenciales que permiten encontrar las diferencias de contraste en intensidad de secciones de caras para poder generar unos vectores representativos que las caractericen. Es un sistema en cascada por etapas que trabaja la información por bloques de menor a mayor precisión.

Estos modelos han ido evolucionando y actualmente los más utilizados son:

* MTCNN **[4]**:
  + MTCNN, *multi-task cascaded convolutional network* es un modelo de detección de caras basadas en redes neuronales donde se trabajan las imágenes por fases. Primero a cada imagen que se analiza se le aplican una cadena de trasformaciones generando unas copias de menor tamaño y resolución que después se procesan para encontrar la cara en la imagen. Este proceso se realiza a través de un barrido en la imagen buscando la cara dentro de todas las copias generadas a diferentes escalas. Se generan cortes en las diferentes imágenes generando sectores de menor tamaño que después pasan por un proceso de convolución. Este proceso colapsa la información reduciendo la información de los sectores generados a una muestra reducida de valores a partir de los valores cercanos.

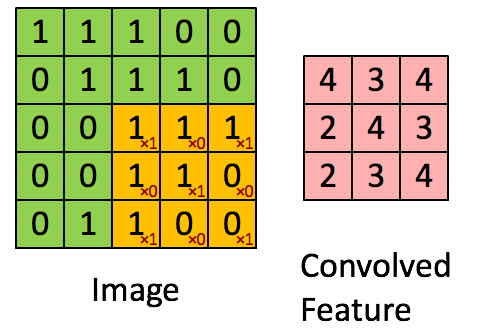


Figura 1: Representación de convolución

El proceso de convolución se repite 2 veces más y se aplican otros procesos para alterar los valores resultantes para procesarlos. Por último el resultado se envía a dos procesos donde se devuelve la probabilidad de que exista una cara en ese sector y la probabilidad de que no exista. Si la probabilidad de que en ese sector exista una cara es grande y por lo tanto de que no exista baja podemos determinar que en ese sector hay una cara.

* SSH **[6]**
  + Sistema más costoso computacionalmente que presenta mejoras respecto a los modelos que procesan las imágenes piramidalmente, como MTCNN. Está basado en métodos de detección de objetos y busca cambios en contrastes en la imagen para localizar las características faciales. Este proceso no pasa por tantas capas de procesamientos por lo que es rápida en devolver resultados, pero requiere de más coste computacional para resolver la problemática.

## 

## 2.2 Reconocimiento de caras

El reconocimiento de una cara es el proceso por el cual podemos analizar las características propias de una cara para identificar o verificar la identidad de una persona. Esta técnica se conoce como reconocimiento facial. El análisis de las imágenes recogen las características faciales que describen una cara en un conjunto de datos que luego podemos comparar con conjuntos de datos conocidos para calcular la diferencia estos.

El uso de versiones clásicas como PCA **[9]** permite reducir la complejidad de la información que se presenta en una imagen reduciendo la redundancia y preservando los cambios notables en la imagen distribuyendo coeficientes de mayor a menor cantidad de información relevante. Estos coeficientes se pueden comparar con pares de imágenes i calcular si las dos imágenes son similares o no.

Actualmente en el sector del reconocimiento facial una de las diferentes opciones que hay dentro del estado del arte es FaceNet **[1]** de Google. La base de funcionamiento de FaceNet es la agrupación de grupos de caras den paquetes de 3 caras, dos parecidas y una diferente. Estos paquetes comparan las características de las caras que lo forman y generan dos vectores de distancia respecto a una de las tres caras generando un ancla. Posteriormente genera un par de distancias con estos vectores, situando las caras que son más parecidas a poca distancia y la cara que es más diferenciada a más distancia. Este método genera comparaciones donde tenemos directamente una salida de caras parecidas y no parecidas y podemos conseguir con mucha velocidad grupos de caras parecidas y diferentes dentro de clústeres de imágenes.

Además de FaceNet, otras empresas como Facebook proponen soluciones como FaceMatch **[8]**, también centrada al reconocimiento de caras con otras aproximaciones, pero que aportan como el resto una salida de información en forma de valores numéricos o descriptores faciales.

Todos estos sistemas usan Redes Neuronales ya que permiten a partir de información conocida, por ejemplo en el caso del reconocimiento de caras clústeres de caras, entrenar un sistema que aprende a realizar tareas de reconocimiento de patrones. En el caso del reconocimiento de caras se enseña a estos sistemas a reconocer patrones dentro de la estructura facial para generar en la salida los valores que determinan los descriptores faciales.

Este proceso de entrenamiento guiado permite definir modelos como los antes citados que generan en poco tiempo y con un porcentaje de error reducido la información necesaria para comparar las caras.

## 2.3 Estudio de Mercado

El propósito de este trabajo de final de grado no pretende crear una herramienta que genere una competencia contra otros productos o soluciones comerciales actuales.

Actualmente existen sistemas que muestran información de los artistas que aparecen en series o películas en plataformas de contenido a la carta como HBO o Netflix, los datos de las cuales son introducidos por operarios de estas empresas para poder ofrecer esta información.

Existen además sistemas comerciales como DIVE **[14]**, antes conocida como Touchvie que actualmente se ha integrado en las televisiones inteligentes de Samsung, capaces de reconocer una película o una seria y mostrar información de los actores o vestuario. Este sistema utiliza el audio de la película o el canal en el que aparece para reconocerla y mostrar información sobre esta. Una de las limitaciones que tiene este sistema es que requiere de datos previos introducidos. Esto se puede obtener de comentarios del CEO José Luis Flórez, cuando indicaba que el funcionamiento de sus sistema utilizaba el top 100 de películas de cada año, por lo tanto denotando que la información no se generaba dinámicamente y que por el contrario se introducía a mano a partir de bases de datos como IMDB o parecidas.

Amazon ofrece dentro de su contenido a la carta la opción X-ray **[10]** que muestra información de los contenidos que ofrece al usuario en diferentes plataformas y dispositivos para la mayoría de sus títulos. Como en los casos anteriores esta información la introducen manualmente operarios de Amazon.

En cuanto a sistemas no específicos de reconocimiento de actores tenemos por ejemplo Clarifai que permite detectar i verificar a que actor pertenece una imagen, pero a diferencia de lo requerido por el sistema no trabaja con video y solo opera con imágenes estáticas.

Estos sistemas en general presentan inconvenientes conocidos, o bien requieren de datos introducidos manualmente o por el contrario están limitados a los datos conocidos y requieren de un mantenimiento y ampliación de las bases de datos con las que operan para trabajar.

El sistema que se propone para este trabajo de final de grado pretende automatizar el reconocimiento y búsqueda de datos extrayendo información de bases de datos fiables como es el caso de IMDB. Esto permite reducir tanto el coste de mantenimiento del sistema como aumentar la velocidad respecto a los métodos donde se introduce la información manualmente y facilita que sea un sistema escalable y que pueda funcionar incluso con películas o series españolas, catalanas o poco conocidas.

# 3. Gestión del proyecto

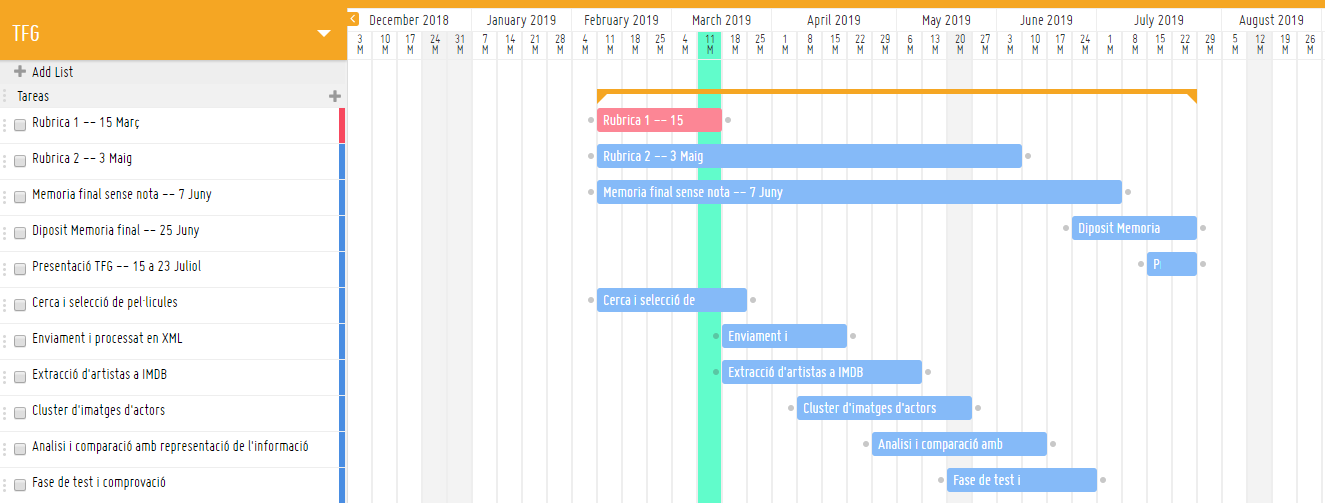
## 3.1 Procedimiento y Herramientas para el seguimiento del proyecto

El procedimiento estipulado para el seguimiento del proyecto consta de sesiones presenciales cada dos semanas y un seguimiento no presencial a partir de Mayo donde se evalúan las tareas con la supervisión del tutor y se revisa la evolución y etapas del proyecto.

A partir de Mayo el seguimiento pasa a ser telemático vía correo electrónico.

El proyecto esta desglosado en un grupo de metas que se describen de forma visual en el diagrama de Gantt del apartado 3.1.1 y que están descritas en profundidad en el apartado 4 Metodología.

### 3.1.1 GANTT



### 3.1.2 Trello

Trello es una plataforma web que ofrece un gestor de tableros basados en Kanban, sistema de información de tarjetas. Se utiliza para generar tareas o tarjetas y generar un seguimiento del desarrollo del proyecto simplificado. Esta metodología se usa ampliamente en el mercado para gestionar la productividad y el flujo de ejecución de las tareas de proyectos complejos.

Para este trabajo de final de carrera se han desglosado unas tareas que posteriormente han servido para situar tiempos de ejecución en un diagrama de Gantt que refleja el tiempo estimado de las tareas para poder realizar el trabajo de final de grado.

### 3.1.3 GitHub

GitHub es una plataforma web que ofrece un servicio de repositorios en la nube. Estos repositorios sirven como una copia tanto de seguridad, permitiendo obtener acceso a la información respaldada, como a su vez como una herramienta de evolución de contenido.

A partir de las herramientas de control de versión mediante Git se tiene no solo el contenido en la nube, sino todo el proceso de evolución del contenido y se puede saltar a cualquier punto del desarrollo del proyecto.

En si una versión de control ayuda a los desarrolladores de contenidos a llevar un registro de todos los cambios tanto en código como en contenido de un proyecto.

Git es un sistema de control de versión creada por Linus Torvalds en 2005.

Para este proyecto de final de grado es imprescindible el uso de un sistema de control de versiones ya que se requiere un control de la evolución de código y la capacidad de poder acceder a este en diferentes máquinas para su edición y testeo, tanto por parte de creación como de pruebas del sistema.

Todos los contenidos del proyecto que formen parte de producción de código estarán alojados en GitHub para mantener un control de versión del código y poderlo ofrecer posteriormente al finalizar como una herramienta de desarrollo y punto de arranque para futuros proyectos.

Los archivos referentes citados en el trabajo pueden encontrarse en la siguiente carpeta pública de GitHub:

<https://github.com/arkasarius/python-IMDB-TFG>

## 

## 3.2 Herramientas de validación

La validación del modelo funcional se realizara a partir del análisis de un grupo reducido de películas (5) en primeras fases de test y posteriormente con un grupo más amplio para calcular la precisión del sistema.

Este test tendrá en cuenta el número de aciertos y errores para calcular el porcentaje de fidelidad del modelo final.

## 3.3. DAFO

|  |  |  |
| --- | --- | --- |
|  | **Positivos** | **Negativos** |
| **Origen**  **Interno** | **Fortalezas**   1. Buen nivel de programación. 2. Buenos resultados en estudios relacionados anteriores. | **Debilidades**   1. Disponibilidad horaria reducida por trabajo |
| **Origen**  **Externo** | **Oportunidades**   1. Gran cantidad de información y referentes 2. Proyectos relacionados open source. 3. Área de trabajo del tutor de final de grado. | **Amenazas**   1. Costes derivados del estudio o infraestructura necesaria para operar el sistema. |

Tabla 1: DAFO del trabajo de final de grado.

## 

## 3.4. Riesgos y plan de contingencias

Los posibles riesgos identificados para este proyecto y sus soluciones son las citadas a continuación ordenadas de menor a mayor importancia:

|  |  |
| --- | --- |
| **Riesgo** | **Solución** |
| Imposibilidad de crear un modelo de reproducción a tiempo real. | Creación de un modelo emulado que muestre los resultados de detección y etiquetado de películas. |
| Costes de mantenimiento web. | Reducción de costes mediante hosting propio y maquinaria propia. |
| Necesidad de más tiempo para elaborar el modelo. | Traspasar la fecha de entrega la segunda entrega de TFG. |

Tabla 2: Tabla de riesgos y contingencias del proyecto de final de grado.

## 3.5. Análisis inicial de costes

El trabajo de final de grado conforma un total de 300 horas de trabajo de desarrollo. Para ello podría situar un coste de producción a la hora y citar un presupuesto de desarrollo pero lo considero una muestra que no representa correctamente el precio final del producto en sí que representa el trabajo final de carrera.

Las metodologías utilizadas como método de trabajo son agiles, no por que representen un beneficio real a la producción del trabajo final de grado, sino por su aplicación real con respecto a la proporción de evolución en el tiempo que tiene que realizarse durante el trabajo de final de grado. El proceso en si se gestiona por etapas que se tienen que realizar dentro de las fechas indicadas no solo para completar el trabajo de forma correcta, sino como una herramienta de producción para evolucionar el proyecto.

El proyecto en sí utilizara herramientas Open Source y librerías públicas, así como hostings gratuitos, por lo que los costes derivados, como antes citado son solo los propios de las horas invertidas en el proyecto, que se requieren por el propio propósito del trabajo de final de grado y que son implícitas de este, por lo que indiferentemente de un posible presupuesto que se pueda aportar, se tienen que realizar.

# 4. Metodología

La metodología propuesta para este proyecto es Scrum. Este proceso se aplica para obtener el mejor resultado posible del proyecto, ya que requiere de unos hitos muy concretos que se requieren en un tiempo de producción reducido. La metodología Scrum plantea la elaboración de hitos e iteraciones que acostumbran a ser de dos semanas, concordando con la propuesta de seguimiento por parte del tutor del trabajo de final de grado. En consecuencia los hitos fundamentales serán:

* Búsqueda y selección de películas para la muestra de trabajo.
  + En este hito se requiere escoger cinco títulos en la mayor calidad posible para su uso en el trabajo de final de grado.
* Procesado de las películas mediante API externa y extracción de datos en XML.
  + En este punto se utiliza una API funcional para analizar y extraer clústeres de caras y ficheros de relación en formato XML.
* Extracción de artistas de las diferentes películas de IMDB
  + En esta fase se requiere dominar el uso de la base de datos IMDB para poder extraer los actores que participan en las películas de la muestra para poder procesar su información a posteriori.
* Creación de una base de imágenes de los actores que forman parte del reparto de las películas.
  + En este hito se pretende generar una base de imágenes de los actores que conforman el set de las diferentes películas como una base para compararlas con los datos extraídos de la API y compararlos para identificar a los diferentes actores.
* Análisis y comparación de actores con las imágenes extraídas y representación de la información de los actores.
  + En este punto se analizan los resultados de las comparaciones y se muestra con gráficos informaciones relevantes a las detecciones e información de los actores mediante el uso de IMDB para conseguir los datos.
* Fase de testeo y comprobación
  + En esta fase se evalúa la capacidad del sistema y se contabiliza el grado de acierto del sistema con las películas de la muestra.

# 5. Desarrollo del proyecto

Teniendo en cuenta la planificación y metodología los puntos que conforman el desarrollo del proyecto son:

1. Búsqueda y selección de películas.
2. Procesado de las películas mediante API.
3. Extracción de artistas de las diferentes películas.
4. Creación de una base de imágenes de actores.
5. Análisis y comparación de actores con las imágenes extraídas.
6. Fase de testeo y comprobación

## 5.1 Búsqueda y selección de películas

Para poder generar un modelo de prueba donde se pueda analizar, extraer y reconocer a los diferentes actores he decidido establecer una pequeña muestra de cinco películas para poder trabajar con ellas.

Estas películas son:

1. Ángeles y demonios
2. El quinto elemento
3. Hombre de negro
4. Matrix
5. Yo robot

En todas y cada una de ellas tenemos a los actores en escenas con iluminaciones diferentes con contrastes tanto en escenas de mucha luz como en escenas de oscuridad y nocturnas generando diferentes matices de tonalidad en las caras de los actores como para tener una muestra no homogénea de imágenes faciales.

Todas ellas disponen de planos frontales donde los actores interpretan su papel con respecto a la cámara, facilitando la captura de imágenes faciales que nos sirvan para determinar los actores que participan en las películas.

En algunas películas tenemos criaturas no humanas con características faciales que el sistema podría reconocer, de manera que pueden generar clústeres que no pertenezcan a ningún actor en concreto y por lo tanto generar información que se tendrá que filtrar.

Todas las películas tienen versiones con una calidad mínima como para operar con ellas, tanto en resolución como en compresión de imagen.

Este hito se concluyó dentro del tiempo establecido en la planificación.

## 5.2 Procesado de las películas mediante API

API es una sigla que alude a *Application Programming Interface.* Una API por lo general no deja de ser una librería de código que atiende a una petición y devuelve un resultado. En cierto modo es una capa de comunicación con tareas programadas que se utiliza para generar contenidos para su uso de una manera más sencilla y rápida.

En este trabajo de final de grado empleo una API proporcionada por el tutor del TFG que permite extraer las caras de las diferentes personas que aparecen en una secuencia de video.

Este proceso es solo de extracción de información y el sistema no indica a quien pertenecen esas caras, pero agrupa caras que son similares dentro de una salida en Clústeres de imágenes.



Figura 2: Clúster artista 00 Yo robot (Will Smith)

Una vez procesadas las películas mediante el uso de esta API el resultado es:

1. Conjunto de caras agrupadas en clústeres.
2. Fotogramas clave de la película detectados por el sistema.
3. XML donde se atribuye cada cara del clúster a un determinado momento de la película en unas coordenadas concretas para localizarlo.
4. Secuencias de navegación donde se separan los diferentes cambios de planos de la película
5. Otros archivos con informaciones varias.

Esta API para su funcionamiento utiliza como base el lenguaje de programación Python.

### 5.2.1 Python

Python es un lenguaje de programación interpretado, multiparadigma y multiplataforma de código abierto.

Un lenguaje de programación interpretado es aquel que no requiere de un proceso de compilación explicito para transformar el código en instrucciones para que el sistema las ejecute. Esto facilita la creación de código ya que el único paso antes de ejecutar un código es simplemente guardar el archivo de código antes de ejecutarlo. Otra de las ventajas de ser un código interpretado es la facilidad de encontrar errores en el código, ya que los fallos en el código no hacen que se pare el programa de forma inesperada y siempre retornan una traza del error al ejecutarse y fallar, facilitando la resolución de estos errores.

Python al ser multiparadigma ofrece diferentes enfoques de programación dentro de sus capacidades como son la programación orientada a objetos, programación imperativa o programación funcional entre ellas.

Python tiene soporte para diferentes sistemas informáticos. Originalmente desarrollado para Unix evoluciono a un estado donde es posible ejecutar código en Python si existe un intérprete adaptado en un sistema operativo cualquiera.

El beneficio de que Python sea de código abierto hace que exista un soporte muy grande en diferentes plataformas y se genere cantidad de código útil que luego se puede reutilizar a partir de librerías.

Las librerías son en su conjunto son implementaciones de código funcional con interfaces y funciones abiertas para su uso que proveen de funcionalidades que se pueden reutilizar y facilitan la creación de contenido al no tener que crearse de cero.

Las librerías que implementa y utiliza la API de extracción de caras son:

1. Keras
2. Mxnet\_cu92
3. Cython
4. Numpy
5. Setuptools
6. Tensorflow\_gpu
7. Opencv\_python
8. Tqdm
9. Pillow
10. Logger
11. Mxnet
12. Tensorflow

#### 5.2.1.1 Librería de Python: Keras

Keras es una biblioteca de redes neuronales de código abierto escrita en Python capaz de ejecutarse sobre TensorFlow, empleada en esta API.

Keras está especialmente diseña para crear redes de aprendizaje profundo. Es una librería modular y extensible diseñada por el ingeniero de Google François Chollet.

En 2017 el equipo de desarrollo de TensorFlow de Google ofreció soporte a la biblioteca Keras.

El concepto principal de Keras es ofrecer un modelo de interfaz en lugar de un framework de aprendizaje automático.

Keras ofrece al usuario implementaciones funcionales de bloques típicos de redes neuronales como por ejemplo Layers, funciones de activación u optimizadores matemáticos.

Su código está alojado en GitHub y existen foros y canales de soporte.

La principal ventaja que aporta para la API es la capacidad de la biblioteca de ofrecer soporte a las redes neuronales convolucionales, como en el caso de uso de esta API.

#### 5.2.1.2 Librería de Python: Mxnet\_cu92

Mxnet\_cu92 es el nombre de la librería Apache MXNet. Esta librería ofrece un framework de Aprendizaje profundo que ofrece gran flexibilidad en su implementación y permite obtener abstracciones de alto nivel en datos diversos.

#### 5.2.1.3 Librería de Python: Cython

Cython es una librería que ofrece la capacidad a Python de llamar a funciones o métodos en C o C++ directamente desde la implementación en Cython.

Cython añade la posibilidad de usar tipos de datos estáticos en las variables como enteros o flotantes. Las ventajas que ofrece al API principalmente son de ancla a códigos en C directamente desde código Python.

#### 5.2.1.4 Librería de Python: Numpy

Numpy es una librería que extiende Python agregando soporte para vectores y matrices, construyendo una librería de funciones matemáticas de alto nivel.

Numpy ofrece herramientas de integración en C/C++ y capacidades de algebra lineal, transformadas de Fourier y randomizadores.

Al trabajar con imagen la forma de representar las imágenes en un sistema numérico es mediante el uso de matrices que representan los pixeles de la imagen, por lo que operar con matrices es un requisito indispensable para la API.

#### 5.2.1.5 Librería de Python: Setuptools

Setuptools es una librería de Python que está desarrollada para crear paquetes de proyectos.

Esta herramienta ayuda a generar archivos en módulos para su posterior envió o instalación.

#### 5.2.1.6 Librería de Python: Tensorflow\_gpu

Tensorflow\_gpu es una librería que permite realizar los cálculos de la librería TensorFlow en los procesadores lógicos de la tarjeta gráfica. Al operar en múltiples subprocesos cambiar de un uso de procesado de procesador a grafica aumenta la velocidad y capacidad de cálculo de las operaciones realizadas por las librerías de cálculo.

Esta librería optimiza el rendimiento de la API ofreciendo la capacidad de cálculo de los procesadores gráficos ya que actualmente tienen más cantidad de núcleos y cantidad de RAM disponible para su uso que en cálculos de procesador.

#### 5.2.1.7 Librería de Python: Opencv\_python

Librería de paquetes de OpenCV preparadas para Python.

#### 5.2.1.8 Librería de Python: Tqdm

Tqdm es una librería sencilla que aporta a Python barras de progreso útiles para visualizar tiempos de respuesta y evoluciones de progreso tanto en entrenamientos de sistemas de inteligencia artificial como para visualización de progresos varios.

#### 5.2.1.9 Librería de Python: Pillow

Pillow es una librería que ofrece soporte para manipulación de imágenes. Ofrece soporte para abrir, manipular y guardar imágenes en diferentes formatos.

Es especialmente útil ya que permite a la API trabajar con las imágenes que forman parte de los flujos de video y extraer y guardar copias de fotogramas o regiones concretas como las caras de los actores.

#### 5.2.1.10 Librería de Python: Logger

Logger ofrece a Python funciones y clases típicas que aumentan la flexibilidad de testeo de Python ofreciendo soluciones de comprobación de fallos más amplias que las propias que implementa Python.

#### 5.2.1.11 Librería de Python: Mxnet

Esta librería ofrece un framework de Aprendizaje profundo que ofrece gran flexibilidad en su implementación y permite obtener abstracciones de alto nivel en datos diversos.

#### 5.2.1.12 Librería de Python: Tensorflow

TensorFlow es una biblioteca de software de código abierto para computación numérica de alto rendimiento. Su arquitectura flexible permite una fácil implementación de cálculos en una variedad de plataformas (CPU, GPU, TPU) y desde computadoras de escritorio hasta clústeres de servidores a dispositivos móviles y de vanguardia.

Originalmente desarrollado por investigadores e ingenieros del equipo Google Brain dentro de la organización de AI de Google, tiene un fuerte soporte para el aprendizaje automático y el aprendizaje automático, y el núcleo de computación numérica flexible se usa en muchos otros dominios científicos.

### 5.2.2 Resultados

Teniendo en cuenta todas las librerías en uso citadas previamente y la selección de películas obtenemos como resultado los siguientes datos:

* Ángeles y demonios:
  + 64 clústers de individuos.
  + 5237 imágenes de caras extraídas.
  + 7761 fotogramas clave.
  + XML de cambios de escena.
  + XML de localización de caras extraídas.
* El quinto elemento:
  + 72 clústers de individuos.
  + 5884 imágenes de caras extraídas.
  + 7320 fotogramas clave.
  + XML de cambios de escena.
  + XML de localización de caras extraídas.
* Hombres de negro:
  + 51 clústers de individuos.
  + 3171 imágenes de caras extraídas.
  + 4266 fotogramas clave.
  + XML de cambios de escena.
  + XML de localización de caras extraídas.
* Matrix:
  + 43 clústers de individuos.
  + 5000 imágenes de caras extraídas.
  + 8143 fotogramas clave.
  + XML de cambios de escena.
  + XML de localización de caras extraídas.
* Yo robot:
  + 21 clústers de individuos.
  + 4338 imágenes de caras extraídas.
  + 6489 fotogramas clave.
  + XML de cambios de escena.
  + XML de localización de caras extraídas.

Los resultados obtenidos proveen de suficiente información para ser una muestra fiable de caras útiles para detectar a los actores que forman parte en las películas.

Uno de los detalles a tener en cuenta es que normalmente se genera un clúster donde se recogen las imágenes que no se pueden identificar a un individuo concreto como un clúster de elementos residuales.

Parte de las caras que aparecen dentro de los clústers tienen caras de actores que no pertenecen realmente a un actor correcto. Un ejemplo de ello es el siguiente:



Figura 3: Muestra de errores típicos en extracción de caras

Como se puede observar para la película “El quinto elemento” para el clúster del actor “1” en este caso Bruce Willis tenemos una imagen incorrecta, la 1042 donde aparece un actor que no es correcto, pero que el sistema ha identificado como correcto en la extracción y lo ha agrupado dentro del clúster.

Posteriormente las imágenes incorrectas se filtraran correctamente ya que dentro del clúster habrá un número de imágenes que tendrá una distancia más grande con respecto a las imágenes del actor y se descartarán como válidas del clúster.

Este hito se concluyó dentro del tiempo establecido en la planificación.

## 5.3 Extracción de artistas de las diferentes películas

En esta fase de producción se requiere extraer de manera automática la información de los actores que forman parte de las películas de la muestra.

Para ello existen diferentes bases de datos que se pueden usar y consultar donde tenemos esta información.

Para este proyecto de final de grado se han tenido en cuenta la existencia de las siguientes bases de datos de películas:

1. IMDB
2. Themoviedb.org
3. Simkl

### 5.3.1 IMDB

IMDB también conocida como Internet Movie Database es una base de datos que almacena información relacionada con películas. Entre la información que ofrece esta personal de producción, actores, actores de doblaje y otras informaciones útiles para este trabajo de final de grado.

IMDB tiene una página secundaria llamada IMDbPro, con muchos más contenidos e información que requiere de una subscripción de pago, ya que está orientada a uso profesional.

Actualmente y desde 1998 pertenece a Amazon.

### 5.3.2 Themoviedb.org

Themoviedb.org es una base de datos parecida en concepto a IMDB ofreciendo prácticamente la misma información que IMDB pero con una mayor complejidad para solicitar la información debido a su API mediante registro.

### 5.3.3 Simkl

No es una base de datos como tal sino más bien un gestor de contenidos visualizados. A partir de las aplicaciones que ofrece se puede saber información de lo que se está visualizando. El sistema está en beta pero una vez visualizado un contenido muestra en la mayoría de los casos información sobre el contenido y genera notificaciones en de nuevos episodios para series visualizadas. Podría ser aplicable como añadido al proyecto, pero no como fuente de información fiable.

### 5.3.4 Comparativa y resultados

Las bases de datos escogidas tienen en común que pueden ser accesibles mediante APIs, tienen consultas gratuitas, pudiendo recoger información de las bases de datos sin incrementar costes en el prototipo y son utilizadas comúnmente en aplicaciones de estudio como fuente de datos.

Entre ellas las dos destacables son IMDB y Themoviedb.org. Las dos pueden ser candidatas para extraer la información de los actores requerida, pero IMDB destaca respecto a Themoviedb.org por su facilidad de uso y por qué existe una librería que implementa las llamadas típicas a IMDB para Python.

Esta librería llamada IMDBPY está escrita completamente en Python versión 3 y tiene definidas las llamadas de todas las funcionalidades que se pueden solicitar a IMDB con llamadas sencillas.

Por este motivo y por las facilidades que ofrece esta librería en el uso de IMDB se considera una opción viable y correcta para obtener la información requerida para este proyecto de final de grado. Una muestra de la sencillez de esta librería la podemos observar en el siguiente código:

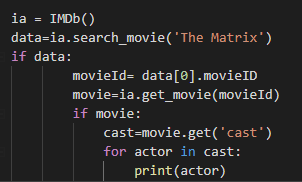


Figura 4: Muestra de código de IMDBPY

En esta muestra se solicita a IMDB la información de la película Matrix. Esta información se recupera en la variable movie que incluye toda la información de actores, reparto y otras informaciones de la película, fácilmente reutilizables posteriormente. Como demostración se puede mostrar el nombre de los actores que participan en la película:

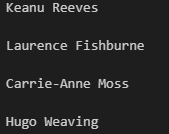


Figura 5: Muestra de algunos de los actores de Matrix

## 5.4 Creación de una base de imágenes de actores

Para este hito se requiere poder obtener de manera rápida y sencilla imágenes de los actores que participan en una determinada película. Para ello y conjuntamente con los resultados del apartado anterior sabemos el nombre real de los actores y necesitamos una manera de encontrar y descargar imágenes de estos actores.

Para tener una fuente de imágenes que se pueda utilizar se requiere de una muestra grande de imágenes de cada actor de cada película.

Teniendo en cuenta que estas imágenes las podemos recuperar de los motores de búsqueda la opción más sencilla para generar estas bases de datos seria descargar las imágenes de una consulta a un buscador.

Para este proceso se puede emplear la librería de Python google\_images\_download que permite descargar de manera sencilla una cantidad de imágenes concreta a partir de una palabra de búsqueda.

Conjuntamente con la implementación anterior obtenemos un código sencillo que resuelve la problemática:

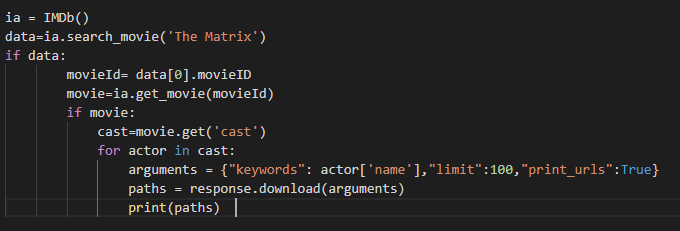


Figura 6: implementación de descarga de imágenes de actores

Este código es una ampliación a lo obtenido en el anterior hito en el que solicitamos 100 imágenes para cada actor que participa en la película.

En este caso para Matrix obtendríamos para el actor Keanu Reeves (Neo) las siguientes imágenes:

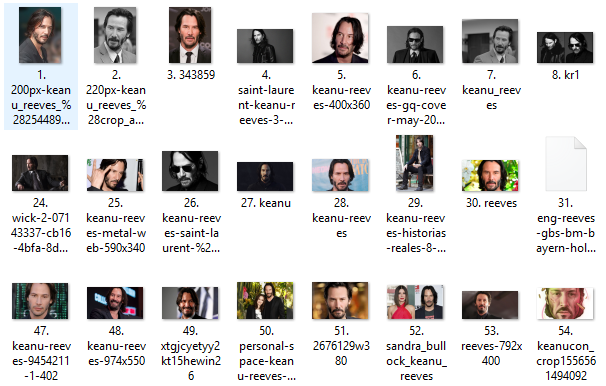


Figura 7: muestra de imágenes de Keanu Reeves

Como se puede apreciar tenemos imágenes en formatos desconocidos además de imágenes donde aparecen otras personas. En la fase de análisis se tendrá que filtrar estas imágenes antes de compararlas con los clústers generados de las películas para identificar los diferentes actores que participan en las películas.

## 5.5 Análisis y comparación de actores con las imágenes extraídas

En este apartado se explicaran los pasos mediante los cuales se extrae la información facial de las imágenes descargadas de los actores mediante el proceso realizado en el punto 5.4.

Para ello se emplean librerías y técnicas de detección facial y se extraen información que se representa en vectores de 128 dimensiones que representan las características faciales de las caras detectadas en las imágenes.

### 5.5.1 Extracción de características faciales de las imágenes extraídas

Para poder extraer la información de las caras de los actores que hemos generado a partir del código de descarga en el punto 5.4 se presentan diferentes opciones con respecto al entorno de desarrollo óptimo para utilizar un modelo de reconocimiento facial.

Lo óptimo en cuestión de control sobre este software es ejecutar estos modelos en un modelo de desarrollo local, mediante una instalación de un entorno en el sistema operativo nativo en el que estemos trabajando o mediante una máquina virtual para ejecutar código especifico si no cumple con los requisitos en el sistema nativo.

Opcionalmente se podría alquilar espacio en un servidor e instalar los requisitos de funcionamiento en este para correr nuestro código de reconocimiento facial.

Existe una alternativa que se emplea en este trabajo de final de carrera para poder ejecutar código para la detección facial. Esta alternativa es Google Colab.

Google Colab es un servicio en la nube que nos proporciona acceso a un entorno de Jupyter notebook, una interface de desarrollo mediante entornos de desarrollo utilizada en múltiples ámbitos de computación de datos en múltiples proyectos de código abierto.

Google Colab permite generar una máquina virtual con la posibilidad de activar GPU para agilizar cálculos, que nos permite correr código tanto en Python 2 como Python 3.

Además de esto el servicio es gratuito e incluye librerías preinstaladas comunes para computación de datos así como la opción de poder instalar nosotros las que podamos necesitar.

Un punto clave de esta plataforma es la capacidad de esta de vincularse con nuestro espacio en Google Drive para poder ejecutar código y poder, como en el caso de este trabajo de final de grado, utilizar imágenes de nuestro espacio en la nube para poder analizarlas.

Para ello y con el fin de realizar un entorno funcional replicable he generado una cuenta nueva de Google Drive con el fin de instalar en esta la librería de reconocimiento facial proporcionada por el tutor de este proyecto y modificarla para extraer la información necesaria para las comprobaciones posteriores.

Replicar este entorno se requiere de los siguientes pasos:

* Descargar el archivo Zip adjunto “roger\_tfg.zip” localizado en GitHub, para ello está el enlace en el punto 3.1.3 GitHub y en 8 Anexos
* Colocar los archivos internos del archivo Zip en la raíz de una unidad de Google Drive
  + Para ello se recomienda vincular una carpeta de nuestra unidad mediante el servicio de copia de seguridad y sincronización de Google para poder operar con los archivos con mayor flexibilidad.
* Colocar las imágenes que se quieran analizar dentro de la carpeta dentro del proyecto en nuestra unidad de Drive localizada en “roger\_tfg/test2”
  + Las imágenes requieren estar en formato JPEG y no estar dentro de carpetas secundarias.
* Ejecutar el archivo “Colab.ipynb” desde nuestra carpeta de Google Drive, pulsando botón derecho se nos dará la opción de abrir este archivo en Google Colab.
* Dentro de Google Colab podemos ejecutar las secuencias de comandos desde el menú “Entorno de ejecución”, pulsando en “ejecutar todas”.
* Se nos requerirá que pulsemos en un enlace que se generara que nos permite generar un código para vincular nuestra unidad de Google Drive para poder ejecutar el código dentro de la carpeta roger\_tfg y poder analizar las imágenes.
* Una vez realizado el proceso se generaran dentro de la carpeta de nuestra unidad “roger\_tfg/outs” la salida correspondiente al análisis de las imágenes en formato de texto.

El funcionamiento de esta secuencia de acciones emplea una versión reducida del api con la que se ha generado y extraído las diferentes características faciales de los actores en las películas. Esta versión simplificada no es tan completa y no devuelve una salida en XML como en la versión facilitada por el tutor en el proceso del punto 5.2.

La versión dada se ha modificado con el fin de extraer la información en formato de texto para poder posteriormente darle un formato concreto para poder comparar la información con la proporcionada por la API.

El resultado de esta extracción de datos genera tantos archivos de texto como imágenes con caras validas estén en la carpeta “roger\_tfg/test2” al ejecutarse. En caso de existir más de una cara valida en estas imágenes, cada una de ellas se extrae en una línea nueva dentro de los archivos de texto, generando tantas líneas con datos como caras validas detectadas.

Una cara valida es aquella que el sistema reconoce como cara y está basada en un detector SSH que forma parte de las técnicas del estado del arte del punto 2.1.

### 5.5.2 Generación de recursos para comparación de las imágenes extraídas

Para poder comparar las caras de los actores primero es necesario convertir los datos que tenemos a un formato que facilite poder comparar la información contra los datos de los clúster obtenidos en el punto 5.2.

Para ello el script desarrollado en Python “data.py” nos permite a partir de una carpeta recoger para cada película el nombre de cada actor al que pertenece y generar un único archivo por actor con la colección de todas las caras que hemos extraído en el punto 5.5.1.

Esto reduce el número de archivos a leer durante el proceso de comparación como paso intermedio, separando la funcionalidad en un bloque secundario especifico solo a este propósito.

Esto nos ofrece la capacidad de recrear la agrupación de los datos en cualquier momento, pudiendo separar la comparación del proceso de preparación de recursos, facilitando la escalabilidad de este a una ampliación de datos sin aumentar el tiempo de procesado durante la comparación.

### 5.5.3 Generación de recursos clúster

Del mismo modo que hemos separado en un módulo la extracción de datos de manera adecuada para su comparación, es necesario generar un grupo de archivos para cada película.

Con el fin de reducir el tiempo de recorrido necesario a la hora de leer los archivos XML generados por la API y eliminar información innecesaria para la comparación, el script en Python “XML\_parser.py” transforma los datos de los archivos XML de la API en unidades individuales correspondientes a cada clúster para cada película.

Este proceso reduce el peso de estos archivos en una media del 45% en bits y facilita su posterior comparación con la información generada en el punto 5.5.2, extrayendo esa complejidad de lectura y escritura del módulo de comparación.

### 5.5.4 Comparación facial

Una vez procesados los dos grupos de datos previamente en los puntos 5.5.2 y 5.5.3 es necesario saltar a la comparación.

Por una parte tenemos para cada película extraído mediante la API proporcionada por el tutor una carpeta que contiene N archivos de texto, siendo N cada uno de los clúster de actores detectados por la API.

Para las películas de la muestra son un total de:

* Ángeles y demonios: 65 clústeres
* El quinto elemento: 65 clústeres
* Hombres de negro: 52 clústeres
* Matrix: 44 clústeres
* Yo robot: 22 clústeres

Cada uno de estos clústeres contiene entre 1 y 2000 vectores correspondientes a una cara detectada por la API.

Por otra parte tenemos de las imágenes descargadas con la ayuda de la base de datos IMDB y el script utilizado en el punto 5.3.4 y posteriormente procesados en el punto 5.5.2 una carpeta para cada película que contiene un archivo por cada actor a comprobar.

En el caso de las películas de la muestra los actores escogidos son los 5 actores mejores pagados para cada película, ya que se refieren principalmente a los actores protagonistas y secundarios principales y son suficientes para demostrar si el sistema funciona ya que escalarlo a todos los actores que participan en las películas sería una tarea posible, pero redundante en cuestión de funcionamiento.

Para las películas de la muestra los actores que conforman este top 5 son:

* Ángeles y demonios
  + Ayelet Zurer
  + Ewan McGregor
  + Pierfrancesco Favino
  + Stellan Skargard
* El quinto elemento
  + Bruce Willis
  + Chris Tucker
  + Gary Oldman
  + Ian Holm
  + Milla Jocovich
* Hombres de negro
  + Linda Fiorentino
  + Rip Torn
  + Tommy Lee Jones
  + Vincent D’Onofrio
  + Will Smith
* Matrix
  + Carrie-Anne Moss
  + Gloria Foster
  + Hugo Weaving
  + Keanu Reeves
  + Laurence Fishburne
* Yo robot
  + Alan Tudyk
  + Bridget Moynahan
  + Bruce Greenwood
  + James Cromwell
  + Will Smith

Cada uno de estos actores conteniendo una docena de vectores que representan caras detectadas en el punto 5.5.1.

Los vectores de los dos sub sets de datos tienen 128 componentes representando las diferentes caras en un espacio de 128 dimensiones donde cada componente describe una característica del espectro facial concreta.

Para poder comparar estos vectores de 128 componentes podemos comparar pares de caras mediante el cálculo de la distancia entre ellos.

La primera aproximación que podemos utilizar para calcular esta distancia es mediante un cálculo vectorial de distancia euclidiana ya que para este caso en concreto estamos hablando de pares de vectores en un espacio N dimensional de N=128.

Por así decirlo dos caras serán parecidas si las distancia entre ellas es muy pequeña, siendo el caso de dos caras exactamente iguales una distancia de 0.

En el caso de este estudio ya que las caras extraídas mediante la API y las obtenidas por internet no son iguales esta distancia nunca será de 0 y es necesario encontrar la distancia mínima aceptada para considerar que dos caras pertenecen a una misma persona.

Para ello podemos utilizar a parte de la distancia entre estos pares de vectores el concepto de ángulo entre vectores. Para un par de vectores podemos obtener la Similitud coseno, medida que representa el ángulo que comprenden un par de vectores en el mismo espacio.

La similitud coseno se emplea frecuentemente para obtener una aproximación de semejanza entre vectores que representan información parecida y es útil en este caso para saber si los pares de caras son parecidos ya que podemos filtrar el ángulo máximo que consideramos valido para decir si dos pares de caras pertenecen a una misma persona.

Para comprobar si dos pares de caras pertenecen a una misma persona podemos decir que serán la misma persona si se cumple que tengan una Similitud coseno lo más próxima a 1, conformando el menor ángulo entre los dos pares de vectores que la forman y a su vez teniendo una distancia euclidiana próxima a 0.

Para ello es necesario obtener 2 coeficientes que serán el ángulo máximo entre dos vectores y la distancia máxima entre vectores, los coeficientes DM y AM.

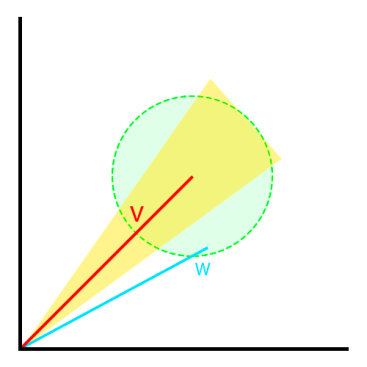


Figura 8: Representación de dos caras V y W en un espacio vectorial N = 128.

En la figura siguiente podemos ver la representación de dos caras, la cara V y la cara W.

La distancia máxima que se aceptara se simboliza con el área verde donde todos los vectores que entren dentro de ese radio serán considerados válidos. En el primer caso ya que su distancia euclidiana entra dentro del área máxima establecida se considera que V y W pueden pertenecer a la misma persona.

El ángulo máximo entre los dos vectores se simboliza con el área amarilla, en este caso en ángulo que conforman la cara V con respecto a W supera el umbral y por lo tanto se considera no valido.

Para que dos caras puedan considerarse que pertenezcan a la misma persona tienen que cumplir que existan dentro del espacio comprendido entre el área de distancia máxima y el área de ángulo máximo.

Para la distancia máxima entre vectores validos nos referiremos a ella como DM y en el caso del ángulo máximo entre vectores como AM.

En las comparaciones de pares de caras según los coeficientes tendremos los siguientes cuatro casos:

1. **Falso negativo:** Cuando los pares de caras pertenecen al mismo actor pero el sistema las reconoce como caras que no se parecen.
2. **Verdadero negativo:** Cuando los pares de caras no pertenecen al mismo actor y el sistema las reconoce como caras que no se parecen.
3. **Verdadero positivo:** Cuando los pares de caras corresponden al mismo actor y el sistema las reconoce como caras que se parecen.
4. **Falso positivo:** Cuando los pares de caras no corresponden al mismo actor y el sistema las reconoce como caras que se parecen.

En función de los coeficientes surgen los siguientes resultados en las comparaciones:

* Si los coeficientes DM y AM son muy grandes tendremos falsos positivos donde dos caras que no pertenecen a un mismo actor serán reconocidas como la misma persona. Para este caso en concreto tendremos el máximo de posibles verdaderos positivos correspondientes a las caras y el máximo de positivos falsos.
* Si los coeficientes DM y AM son muy pequeños tendremos el mínimo de verdaderos positivos y de falsos positivos pero aumentaremos proporcionalmente los falsos negativos.

Para calcular los coeficientes correctos donde obtenemos el mayor número posible de verdaderos positivos y el menor de falsos negativos a la vez de obtener el menor número de falsos positivos se requiere hacer un cálculo partiendo de un valor DM y AM inicial y modificar progresivamente estos valores para encontrar el punto de equilibrio para poder decir con certeza si dos caras pertenecen a la misma persona.

Para ello podemos simplificar el proceso de búsqueda de los coeficientes buscando los valores individuales de estos y elaborar una gráfica de error para cada una.

Si tomamos como ejemplo a la actriz Linda Fiorentino para la película Hombres de negro y analizamos solo los positivos verdaderos, modificando el coeficiente DM, obtenemos la siguiente gráfica:

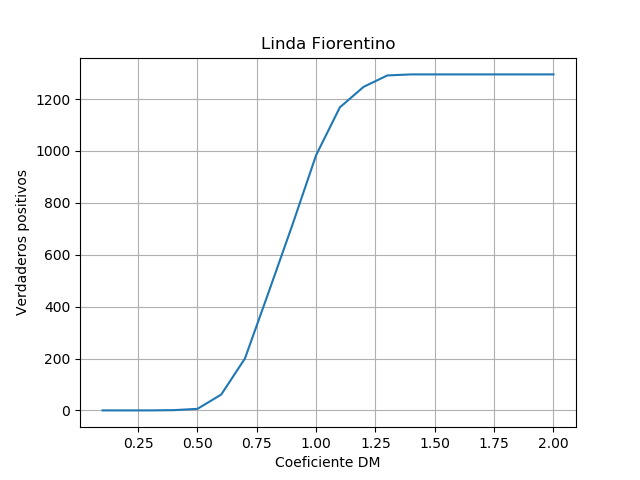


Figura 9: Verdaderos positivos de Linda Fiorentino según coeficiente DM

Como se puede observar en la gráfica no encontramos caras idénticas para un coeficiente de distancia máxima o DM de 0. Esto se debe a que las imágenes que utilizamos para comparar contra el clúster no pertenecen a fotogramas de la película.

Conforme el coeficiente crece comenzamos a obtener los primeros casos de coincidencia entre las caras del clúster y las imágenes de comparación para un coeficiente DM=0.4, llegando en este caso a su máximo para un coeficiente DM=1.39, donde alcanza el límite de 1295 verdaderos positivos.

En este caso el rango de valores útiles para el coeficiente DM está comprendido entre 0.4 y 1.39, pero cada actor tiene un coeficiente mínimo y máximo diferente, como en el caso del actor Rip Torn en la misma película:

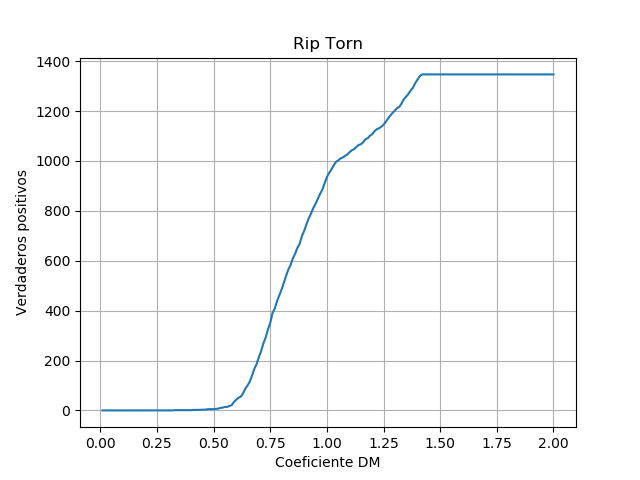


Figura 10: Verdaderos positivos de Rip Torn según coeficiente DM

En este caso el rango para el coeficiente DM de Rip Tom es [0.33, 1.42]. Como se puede apreciar en la gráfica de verdaderos positivos para Rip Tom, el crecimiento de esta se divide en dos curvas, la primera con tendencia exponencial y la segunda, para un DM > 1 con tendencia lineal. Esto es causado por las imágenes que forman parte del set de comprobación, ya que parte de las imágenes que se descargan para los actores contienen imágenes de otros actores y a partir de cierto umbral estas caras que no pertenecen al actor en concreto el sistema las detecta como válidas.

En este caso una de las imágenes que afecta esta curva es la siguiente:



Figura 11: Imagen del sub set de comprobación para el actor Rip Torn

Como se puede apreciar existen casos donde el sistema de búsqueda automático de imágenes devuelve imágenes donde hay otros actores, introduciendo parcialmente ruido al sistema.

Este ruido que podemos apreciar se puede limitar indicando un coeficiente DM que nos permita distinguir entre caras que de verdad pertenecen al actor de elementos externos como obtenemos en esta imagen.

Como es un error que solo se presenta para valores de DM elevados, podemos justificar que el hecho de tener elementos en las muestras de comprobación que contengan información extra de otras caras no afecta al resultado. Si utilizamos un coeficiente DM ajustado esto no supone un problema a la hora de detectar los actores de las películas.

En el caso del coeficiente AM, si nos volvemos a referir a la actriz Linda Fiorentino, obtenemos la siguiente grafica para el coeficiente AM:

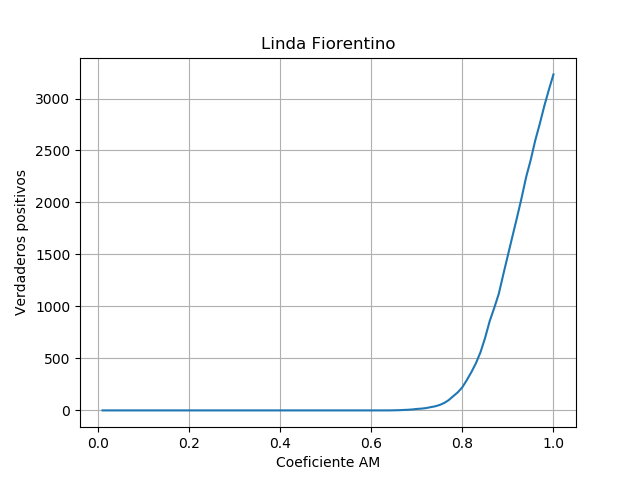


Figura 12: Verdaderos positivos de Linda Fiorentino según coeficiente AM.

Como se puede apreciar en la gráfica, para un coeficiente AM = 1 todas las caras presentes en el clúster se reconocen como verdaderos positivos. En este caso en concreto incluso si las imágenes que comprobásemos contra el clúster no pertenecieran a esta actriz el coeficiente AM = 1 no filtraría ninguna cara y obtendríamos graficas parecidas.

Si comparamos entonces la progresión de la curva de verdaderos positivos para los dos coeficientes, DM y AM obtenemos la siguiente gráfica:

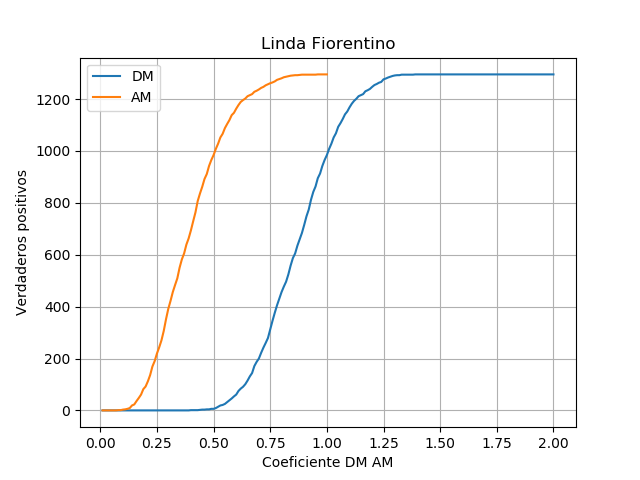


Figura 13: Verdaderos positivos de Linda Fiorentino según coeficientes AM y DM.

En este caso las gráficas tienen similar variabilidad pero diferente tendencia central. Esto es debido a que el área contenida para los vectores de caras que forman el conjunto es prácticamente el mismo ya que las caras son parecidas entre si y no tenemos ruido en la muestra.

La curvatura para el coeficiente AM presenta una curvatura que muestra una progresión inicial y final lenta, característica de las funciones sigmoides. Este comportamiento es normal ya que los dos grupos de valores que se comparan pertenecen a caras de la misma persona y por ello podemos separar el crecimiento de esta en 3 fases:

1. Fase plana: para los coeficientes comprendidos entre 0 y 0.12
2. Fase de aceleración: para los coeficientes entre 0.12 y 0.5.
3. Fase de desaceleración: para los coeficientes entre 0.5 y 1.

Esta forma es dada por el ajuste mínimo del coeficiente para empezar a aceptar caras como válidas al comparar los pares de caras. Una vez sobrepasado este valor de ajuste mínimo se dispara exponencialmente los pares de caras que el sistema reconoce como válidas y posteriormente este entra en la tercera fase donde la mayoría de las caras ya han sido evaluadas como válidas y aunque aumente el coeficiente el crecimiento es mucho más lento y entra en desaceleración.

Si comparamos el sub set de imágenes de la actriz Linda Fiorentino con otro actor que no es ella respecto a los coeficientes AM y DM obtenemos la siguiente gráfica:

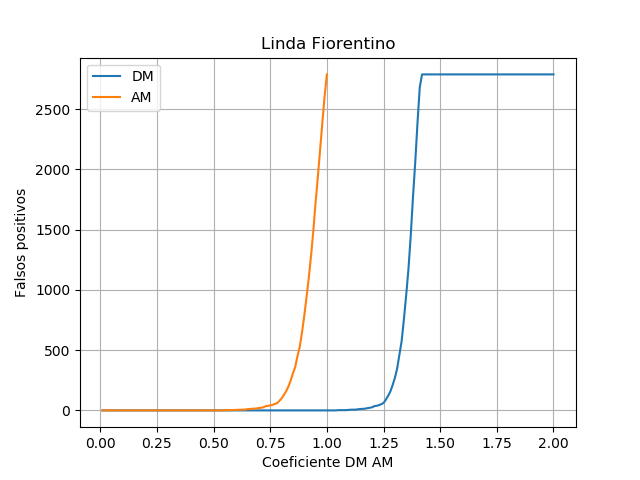


Figura 14: Falsos positivos de Linda Fiorentino según coeficientes AM y DM.

A diferencia del caso anterior podemos apreciar que la progresión de las dos curvas es exponencial pura y no sigmoidea, debido principalmente a que conforme el coeficiente incrementa se consideran caras que no pertenecen a la actriz como resultados positivos.

El primer dato que se resalta en este caso al comparar las dos graficas es el valor de inicio de los coeficientes DM y AM ya que para los casos de positivos verdaderos en los que las caras corresponden a la actriz tenemos que el coeficiente inicial en AM es de 0.12.

Si tenemos en cuenta el coeficiente AM como una representación que va entre los rangos de valores 0 y 1 podemos decir que en el primer caso comenzamos a aceptar como caras positivas aquellas que tienen un valor AM muy pequeño. Al tratarse de un valor comprendido entre 0 y 1 podemos decir que este valor se puede representar como un porcentaje de parecido entre caras. Para un valor de AM = 0.12 podemos decir que su representación porcentual equivaldría a 1-AM 🡺 0’88, siendo entonces los pares de caras parecidas al 88%.

En el segundo caso el porcentaje para los falsos positivos considera que las caras comienzan a parecerse a partir de que se parecen un 36%. Además de que el porcentaje de parecido es muy pequeño, la gráfica que conforma es puramente exponencial, por lo que podemos justificar que se trata en efecto de dos personas que diferentes.

Por ello, para valorar si dos caras pueden considerarse parecidas podemos decir que los coeficientes AM y DM tienen que estar dentro de unos valores concretos donde el filtro combinado de los dos devolverá en la mayoría de los casos suficientes positivos que pertenecerán al conjunto de verdaderos positivos.

Calcular estos valores depende de saber en cada caso que actores corresponden a cada clúster por lo que si queremos obtener un resultado automático aproximado de a quien corresponden esos clústeres podemos designar un valor para los coeficientes AM y DM para filtrar los resultados donde obtendremos que para un mismo clúster tendremos verdaderos positivos y falsos positivos de más de un actor por clúster, pero el número de positivos para los casos donde un actor corresponda a un clúster excederá en gran medida a los actores que al compáralos generan falsos positivos.

Para ello teniendo en cuenta la media de AM y DM de parte de los actores el cálculo para los coeficientes para la muestra de las 5 películas es de:

* AM = 0.4
* DM = 0.75

Los cálculos y gráficos se pueden extraer mediante el uso de los archivos “interpreter\_AM.py”, “interpreter\_DM.py” y “interpreter\_DM\_AM.py” correspondientemente.

## 5.6 Fase de testeo y comprobación

En esta fase del trabajo de final de grado se pone en práctica para los valores obtenidos del estudio de los actores del punto anterior, la comprobación de funcionamiento del algoritmo de comparación.

El código de este corresponde al archivo “Comparador.py”. Este contiene los coeficientes AM y DM previamente tratados con los valores AM = 0.4 y DM = 0.75. Al introducir dentro de sus parámetros el nombre de la película este recorre todos los clústeres y los compara con cada uno de los actores y devuelve una lista de los actores que para estos coeficientes concretos devuelven posibles positivos verdaderos.

A la hora de devolver estos valores se puede observar los siguientes casos:

1. Para un clúster no re reconocen caras positivas.
2. Para un clúster se reconocen como positivos solo las caras de un único actor.
3. Para un clúster se reconocen como positivos las caras de diversos actores pero siempre existe un caso donde un actor tiene más positivos que el resto.

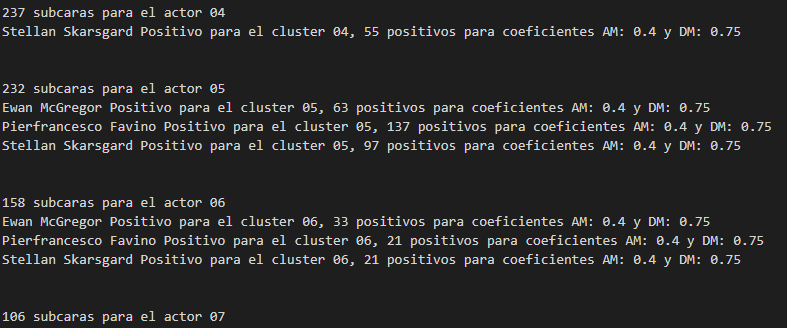


Figura 15: muestra de resultados de los posibles casos para la película Ángeles y demonios.

En el primer caso como se puede observar al final de la figura para el actor del clúster 07 no obtenemos ningún positivo. Esto significa que las caras de ese actor en concreto no se parecen suficientemente al resto de los actores que estamos evaluando contra el clúster para los valores AM y DM establecidos. Esto es correcto ya que ninguno de ellos pertenecen al actor de este clúster, por lo que el filtro funciona correctamente.

En el segundo caso, como se puede apreciar para el clúster 04, obtenemos positivos solo de un actor, en este caso de Stellan Skargard, que en efecto corresponden al actor del clúster 04.

En el tercer caso existen diversos actores que dan positivos en diversos actores, como en el casos de los clústeres 05 y 06 pero los actores correspondientes a estos clústeres son en efecto aquellos que tienen más positivos, siendo los actores del clúster 05 Pierfresco Favino y del clúster 06 Ewan McGregor.

Las principales causas de estos falsos positivos provienen del ruido que genera el generar los sub sets de los actores ya que las búsquedas en los motores de búsqueda no nos ofrecen algunas veces imágenes de los actores correctos o en algunos casos nos muestran más de un actor en la misma imagen.

Esto se podría mitigar en parte calculando diferentes coeficientes para diferentes actores y procesando aquellas imágenes que tengan diversas caras comprobándolas contra el resto del sub set, pero para el caso concreto de discernir quienes son los actores este ruido en la mayoría de casos no afecta a la comprobación de las caras de los clústeres.

Otro efecto de que afecta a los resultados es el hecho de que hay caras que están dentro de los clústeres que no pertenecen a la misma persona. Estos casos como se comenta en el punto 5.2.2 introducen pequeñas muestras de ruido en los clústeres que pueden ofrecer falsos positivos en las comprobaciones.

Teniendo en cuenta estos factores de ruido inducidos tanto en la generación de los clústeres dados por la Api como por el ruido inducido por el sistema de búsqueda de imágenes automática, tenemos unos resultados fiables que corresponden a los actores que participan en las películas.

Podemos decir que en un modelo de extracción de imagen donde las muestras tuvieran ruido prácticamente nulo y con coeficientes suficientemente pequeños, obtendríamos teóricamente siempre casos donde para un clúster solo obtenemos positivos verdaderos de un único actor.

# 6. Conclusiones y trabajos futuros

Para concluir este trabajo de final de grado podemos remarcar el objetivo principal de este:

El objetivo general de este proyecto es la implementación de un sistema capaz de analizar y reconocer conjuntos de caras de actores extraídos de fragmentos de películas para etiquetarlos.

Este objetivo se ha completado y todos los módulos que se anexan en el apartado 3.1.3 y 8 permiten replicar la mayoría de los procesos de extracción y análisis de caras utilizando tecnologías del estado del arte actualmente utilizadas en productos de reconocimiento de elementos como Google Lens o Tesla, en sus coches de conducción autónoma.

El hecho de acercar la tecnología de forma abierta a la gente y que pueda utilizarla para como en el caso de este trabajo de final de grado para entenderla mejor es una forma de hacer que el conocimiento esté en manos de aquellas personas que lo necesiten y puedan con ello crear nuevos proyectos.

En este trabajo de final de carrera se analizan pequeñas muestras de un sistema que con un poco más de trabajo abre las puertas a resolver problemáticas de detección de personas en diversos ámbitos.

Un claro ejemplo de esto si nos referimos a la formulación del problema analizada en este trabajo es la cantidad de datos que se generan cada segundo.

Este algoritmo podría ampliarse por ejemplo para analizar videos de plataformas de contenido como YouTube para comprobar infracciones de copyright, tanto para pequeños como grandes usuarios, ya que actualmente las disputas por copyright están automatizadas y existen casos de contenidos que infringen copyright que el sistema actual no reconoce. También existen casos donde videos reportados por copyright falsamente son eliminados de la plataforma sin revisión ya que no es posible analizar todos y cada uno de los casos.

Otra posible aplicación de esto sería un detector de personas en videos, pudiendo a partir de una base de datos de caras previas con suficiente información etiquetar información relevante de estos videos en relación a las personas que aparecen en este. Las aplicaciones de este son muy grandes y se podría utilizar para recrear el funcionamiento de Shazam **[15]** para películas y obtener todos los datos de los artistas que participan. Otras opciones podrían incluir prevención de riesgos y lucha contra el crimen. Las opciones son amplias.

# 7. Bibliografía

1. *FaceNet: A Unified Embedding for Face Recognition and Clustering. 17 Juny, 2015* [*https://arxiv.org/abs/1503.03832*](https://arxiv.org/abs/1503.03832)Consultado*. 8 Marzo, 2019*
2. *SSH: Single Stage Headless Face Detector. 18 Oct, 2017* [*https://arxiv.org/abs/1708.03979*](https://arxiv.org/abs/1708.03979)Consultado*. 8 Marzo, 2019*
3. *Face detection* [*http://faculty.ucmerced.edu/mhyang/papers/face-detection-chapter.pdf*](http://faculty.ucmerced.edu/mhyang/papers/face-detection-chapter.pdf)Consultado*. 12 Marzo, 2019*
4. *Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks* [*https://arxiv.org/abs/1604.02878*](https://arxiv.org/abs/1604.02878)Consultado*. 12 Marzo, 2019.*
5. *Rapid Object Detection Using a Boosted Cascade of Simple Features* [*http://www.merl.com/publications/docs/TR2004-043.pdf*](http://www.merl.com/publications/docs/TR2004-043.pdf)Consultado*. 12 Marzo, 2019.*
6. *SSH: Single Stage Headless Face Detector* [*http://openaccess.thecvf.com/content\_ICCV\_2017/papers/Najibi\_SSH\_Single\_Stage\_ICCV\_2017\_paper.pdf*](http://openaccess.thecvf.com/content_ICCV_2017/papers/Najibi_SSH_Single_Stage_ICCV_2017_paper.pdf)Consultado*. 12 Marzo, 2019.*
7. *The return of AdaBoost.MH: multi-class Hamming trees* [*https://arxiv.org/abs/1312.6086*](https://arxiv.org/abs/1312.6086)Consultado*. 12 Marzo, 2019.*
8. *Extracto de la página* [*https://www.rankred.com/face-recognition-algorithms-techniques/*](https://www.rankred.com/face-recognition-algorithms-techniques/)Consultado*. 12 Marzo, 2019.*
9. *Eigenfaces for Recognition* [*http://www.face-rec.org/algorithms/PCA/jcn.pdf*](http://www.face-rec.org/algorithms/PCA/jcn.pdf)Consultado*. 13 Marzo, 2019*
10. *X-ray* [*https://www.amazon.com/adlp/xray*](https://www.amazon.com/adlp/xray)Consultado*. 14 Marzo, 2019*
11. *Información de contexto extraída de* <https://www.forbes.com/sites/bernardmarr/2018/05/21/how-much-data-do-we-create-every-day-the-mind-blowing-stats-everyone-should-read/#683d7b7760ba> Consultado. 20 Abril, 2019
12. *Aprendizaje supervisado* <https://es.wikipedia.org/wiki/Aprendizaje_supervisado> Consultado, 20 Abril, 2019
13. *Aprendizaje semi-supervisado* <https://es.wikipedia.org/wiki/Aprendizaje_semisupervisado> Consultado, 20 Abril, 2019
14. *Dive* <https://dive.tv/> Consultado, 1 Mayo, 2019
15. *Shazam* <https://es.wikipedia.org/wiki/Shazam_(aplicaci%C3%B3n)> Consultado 4 Agosto,2019

# 8. Anexos

GitHub de los archivos del proyecto <https://github.com/arkasarius/python-IMDB-TFG>