Búsqueda por texto en videos/imágenes

Recuperación de Información y Recomendaciones en la Web

Hernán Esteves – hernan.esteves.rosano@fing.edu.uy
Nicolás Tomeo – nicolas.tomeo@fing.edu.uy
Matías Rodal – matias.rodal@fing.edu.uy

Grupo 20

15 de diciembre, 2015
Facultad de Ingeniería UdelaR

Índice general

1. Introducción

1.1 Motivación

Hoy en día sitios web los cuales le permiten al usuario almacenar, buscar y reproducir videos utilizan fundamentalmente como información para indexar metadatos del audiovisual, como ser título, descripción, categoría y etiquetas, los cuales deben ser introducidos por el usuario.

Sin embargo, no encontramos ningún sitio en el cual el contenido mismo de los videos sea indexado. Youtube, por ejemplo, realiza una transcripción automática del audio de los videos, pero no la indexa debido a la extrema baja calidad de las mismas. Sí indexa las transcripciones cuando las provee el usuario.

La motivación principal de este proyecto es aprovechar la información incluída en los videos para mejorar los resultados de búsqueda. En este caso, se propone utilizar el texto embebido en los videos.

Múltiples tipos de videos pueden aprovechar en gran manera este tipo de indexación, entre los más aprovechables se encuentran los videotutoriales, los videolyrics[1] y clases con diapositivas. Cualquier video rico en texto es gran candidato para explotar el potencial de este tipo de indexado.

[1] Videos con la letra de las canciones.

1.2 Propuesta y objetivos

Se propone realizar una aplicación web a modo de prototipo que permita buscar videos indexados exclusivamente a partir del texto extraído de los mismos. La aplicación también permitirá subir nuevos videos, y los indexará para incluirlos en futuras búsquedas.

Para obtener el texto de los videos, la idea principal es extraer distintos fotogramas, y aplicarles técnicas de reconocimiento óptico de caracteres[2] para identificar y obtener el preciado valor.

Resumiendo, la aplicación se divide en tres desafíos principales bien marcados:

- Descomponer los videos en fotogramas
- Procesar los fotogramas con técnicas de OCR
- Indexar los videos utilizando el texto reconocido por el OCR

Para poder visualizar los resultados, la aplicación también proveerá una interfaz de búsqueda.

Como objetivo secundario, se planteó la posibilidad de agregar la capacidad de agregar videos a la biblioteca e indexarlos dinámicamente para proveer una experiencia más cercana a los productos reales, donde es posible subir videos.

Por último, si el tiempo lo permite también nos parecería interesante agregar indexado de imágenes además de los videos. Esto sería útil en sitios centrados en la búsqueda de imágenes (por ejemplo imgur.com). Estos sitios tienen una gran cantidad de "memes", los cuales siempre vienen acompañados de algún texto corto, por lo que sería realmente útil indexar con el texto extraído.

[2] El reconocimiento óptico de caracteres (ROC), generalmente conocido como reconocimiento de caracteres y expresado con frecuencia con la sigla OCR (del inglés Optical Character Recognition), es un proceso dirigido a la digitalización de textos, los cuales identifican automáticamente a partir de una imagen símbolos o caracteres que pertenecen a un determinado alfabeto, para luego almacenarlos en forma de datos, así podremos interactuar con estos mediante un programa de edición de texto o similar [7].

2. Desarrollo

2.1 Herramientas

Como framework para el desarrollo de la aplicación web se utilizó *Ruby on Rails* junto a PostgreSQL para la persistencia.

La descomposición del video en fotogramas se realizó utilizando la librería *FFmpeg*[2]. Esta librería provee una gran cantidad de utilidades para el trabajo con videos entre las cuales se encuentra la de extracción de fotogramas. Una característica muy importante de esta librería es la gran cantidad de formatos que soporta. Soporta prácticamente todos los formatos existentes, por lo que la aplicación no estará restringida en este sentido.

Para el procesamiento OCR se utilizó Tesseract, una herramienta Open Source disponible bajo la licencia de Apache 2.0 que puede ser usada directamente, o por programadores utilizando una API [4]. En nuestro caso se utilizó la gema *rtesseract* que hace uso de la API mencionada anteriormente. Esta gema permite gran flexibilidad en configuración, y es la más usada para este tipo de tareas[3].

Para indexar los videos e imágenes se consideraron algunas opciones, pero se decidió finalmente utilizar Apache Solr[6] debido a su gran popularidad, a que hemos trabajado anteriormente con Lucene (librería sobre la que esta escrito Apache Solr), pero sobre todo porque está construido para grandes empresas, lo que garantiza su alto desempeño. La comunicación con Apache Solr se realiza a través de una API REST, por lo que es fácilmente integrable con cualquier lenguaje de programación. Para esta aplicación usamos la gema *Sunspot*[5], que permite una fácil integración de Apache Solr con Ruby.

Una de las etapas de desarrollo incluyo la utilización de la herramienta ImageMagick para modificar las imágenes obtenidas con el objetivo de mejorar la capacidad de reconocimiento del OCR pero no se obtuvieron resultados significativos.

2.2 Diseño y Arquitectura

El sistema funciona totalmente en forma local y en la figura 2.1 se puede observar la arquitectura del mismo.

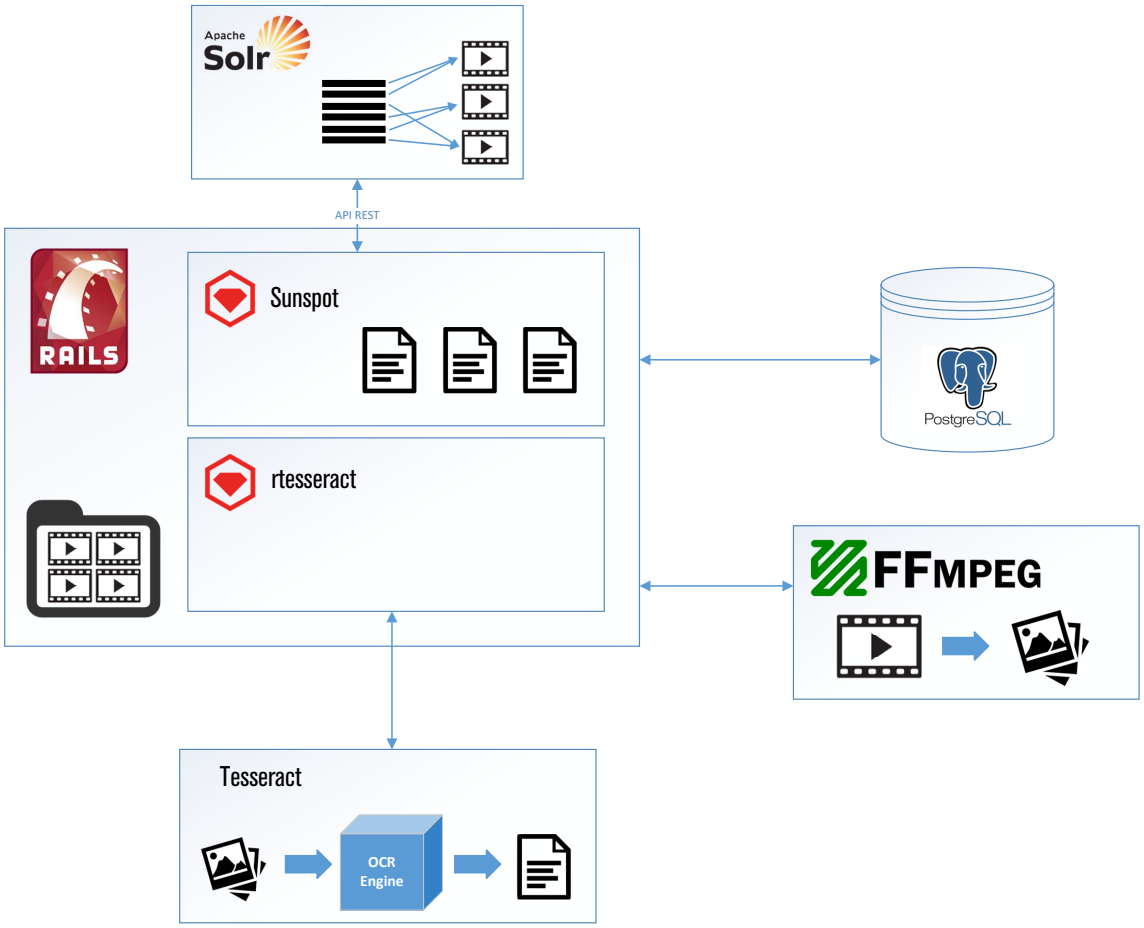


Figura 2.1: Arquitectura de la solución

Por un lado podemos observar el servidor local de Solr (podría utilizarse de forma remota) que se comunica mediante una API REST con la gema Sunspot alojada en la aplicación de Rails.

Cuando un usuario sube un video, el mismo es almacenado para poder ser descargado al realizar una búsqueda, y de inmediato comienza el proceso de indexación. Este proceso consiste inicialmente en la extracción de un fotograma cada 10 segundos, que consideramos es un buen balance entre el costo de extracción de fotogramas y el texto nuevo que se pueda obtener, ya que en el tipo de videos objetivo de esta aplicación, el texto cambia aproximadamente entre cada 5 y 15 segundos.

Luego, los frames son procesados por el motor OCR Tesseract mediante el uso de la gema rtesseract, para luego almacenar el texto extraído en la base de datos e indexar el

video en Apache Solr.

2.3 Indexación

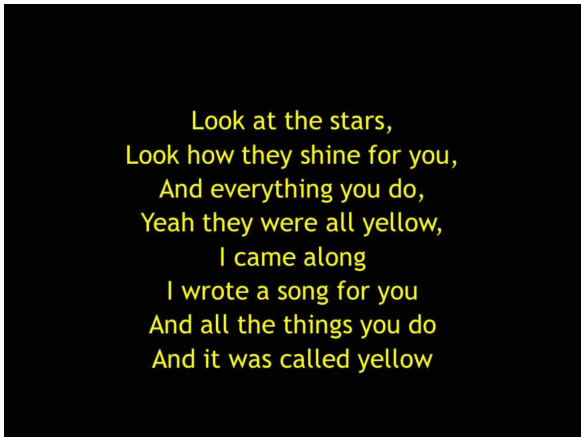
Para la indexación se utiliza el modo de texto completo de Apache Solr sobre el texto extraído de los videos e imágenes. La búsqueda por texto completo se diferencia de otros tipos de búsqueda en que utiliza todas las palabras para buscar. Otras técnicas se basan en metadatos (como en este caso podría ser el título o la descripción) o versiones reducidas del contenido. Este modo permite encontrar cualquier video o imagen buscando cualquier palabra que contenga y haya sido correctamente reconocida por el OCR.

2.4 Resultados

2.4.1 Videolyrics

En esta primera parte de la sección se mostrará un caso de un videolyric de la cancion Yellow de Coldplay. El video es puramente texto en amarillo con fondo negro, y es estático, lo cual lo hace uno de los casos más simples para el reconocimiento de texto.

A continuación se muestran algunos frames extraídos y el texto reconocido en cada uno:



(a) Frame extraído

Look at the stars,

Look how they shine for you,
And everything you do,
Yeah they were all yellow,

I camealong
I wrote '-a song. for you
And all the things- you do
And it was called yellow

(b) Texto reconocido

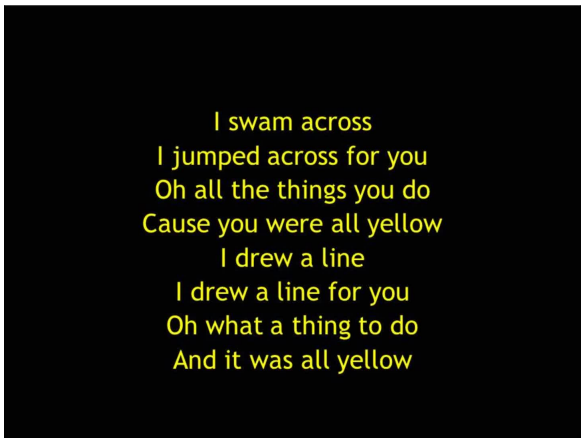
Figura 2.2: Frame 1 - Coldplay Yellow

(a) Frame extraído

So then 'I took my turn
Oh all the things We done
And it was all yellow
Your skin
Oh yeah your skin and bones
Turn into s.omet.hi'ng beautiful
Do you know you know I love you so
You know I love you so

(b) Texto reconocido

Figura 2.3: Frame 2 - Coldplay – Yellow



(a) Frame extraído

I ‘swam across
I jumped ‘across for -you
Oh all the things you do
Cause you were all yellow
I drew a line
I drew a line for you
Oh what ,a thing to do
And it was all yellow

(b) Texto reconocido

Figura 2.4: Frame 3 - Coldplay – Yellow

(a) Frame extraído

Your skin
Oh yeah your skin and bones
Turn into something beautiful
Do you know for you I bleed myself dry
For you i bleed myself dry

(b) Texto reconocido

Figura 2.5: Frame 4 - Coldplay – Yellow

Como se puede observar, el texto fue reconocido sin grandes problemas. Ningún frame fue perdido, es decir, todo lo que se podía reconocer fue procesado. Esto es debido obviamente a la naturaleza del video, que es estático y cambia de frame cada 30 segundos aproximadamente.

Si se da el caso en el que nos acordamos de alguna frase de la canción pero no del título, podremos buscar la misma por cualquier frase, por ejemplo *“I drew a line for you”* y se mostrará en los resultados de la búsqueda.

Luego se presenta el caso de un videolyric con mayor animación y en idioma español. Se presentan algunos frames:

V’ ‘*1
Y ME MUE R PEDIR
- ’ ..,3-45;‘.

(a) Frame extraído (b) Texto reconocido

Figura 2.6: Frame 1 - La Franela – Fue tan bueno

Y HACERME BRILLAR
H ._ A-Q.D-. 4-

(a) Frame extraído (b) Texto reconocido

Figura 2.7: Frame 2 - La Franela – Fue tan bueno

UE ME GUiAN UANDO

(a) Frame extraído (b) Texto reconocido

Figura 2.8: Frame 3 - La Franela – Fue tan bueno

(a) Frame extraído

Y HACERME BRILLAR

(b) Texto reconocido

Figura 2.9: Frame 4 - La Franela – Fue tan bueno

Debido al mayor dinamismo existente en este video se nota una clara pérdida en la precisión. Sin embargo, es posible reconocer un número importante de palabras correctamente y hasta algunas frases sin ningún error.

2.4.2 Videos de clases

En esta sección, introducimos como ejemplo un video obtenido desde OpenFing[1]. Se utilizó un fragmento del mismo a modo de prueba. El fragmento contiene tanto fotogramas que incluyen diapositivas de la clase como la grabación del docente. A continuación se pueden observar los resultados para algunos fotogramas:

(a) Frame extraído

Agenda Inlroduccuon Mudelos de Proceso de Softwa-re Herramlentas y TÃ©cnlcas Modelo de Process RUP Prooesos Aglles Modelo de Proceso MUM (Pro-yecio de Ingenreria de Softwa-re) lnlloducclwn lngerrreua de Soï,ware

(b) Texto reconocido

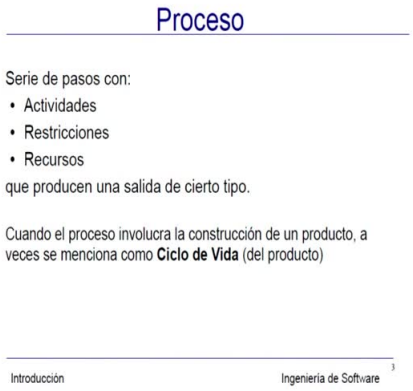
Figura 2.10: Frame 1 - Clase

(a) Frame extraído

«Texto ilegible»

(b) Texto reconocido

Figura 2.11: Frame 2 - Clase



(a) Frame extraído

Proceso
Serie de pasos con:
- Actividades
- Restricciones
- Recursos
que producen una salida de cieno Iipo.
Cuando el proceso Involucra 1a conslruccuon de un produc-to, a
veoes se menclona coma Cicla de Wda (del produc1o)
x
Immducclon mgemeua 4: Summit

(b) Texto reconocido

Figura 2.12: Frame 3 - Clase

En el frame de la figura 2.11 se obtuvo un texto ilegible. Podría haberse esperado obtener algún texto reconocido a partir de las diapositivas, pero la calidad de la imagen no fue suficiente como para que así sea o el algoritmo OCR utilizado por Tesseract no fue lo suficientemente bueno para este tipo de imágenes.

Si bien en los frames de las diapositivas se obtuvieron buenos resultados, aún se encontraron algunos desvíos importantes, teniendo en cuenta la claridad del texto en la imagen. Aún así la indexación que se logró a partir de estos textos fue útil, ya que por ejemplo buscando por la frase "*Modelo de proceso*" se obtiene el video como resultado.

2.4.3 Utilización de ImageMagick

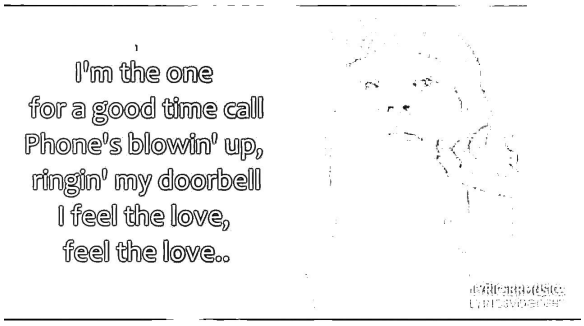
Se investigó la herramienta ImageMagick para mejorar las imágenes utilizando un filtro para resaltar el texto de las mismas:

(a) Frame extraído

%h1..e one (ad time call

(b) Texto reconocido

Figura 2.13: Imágen sin filtro ImageMagick



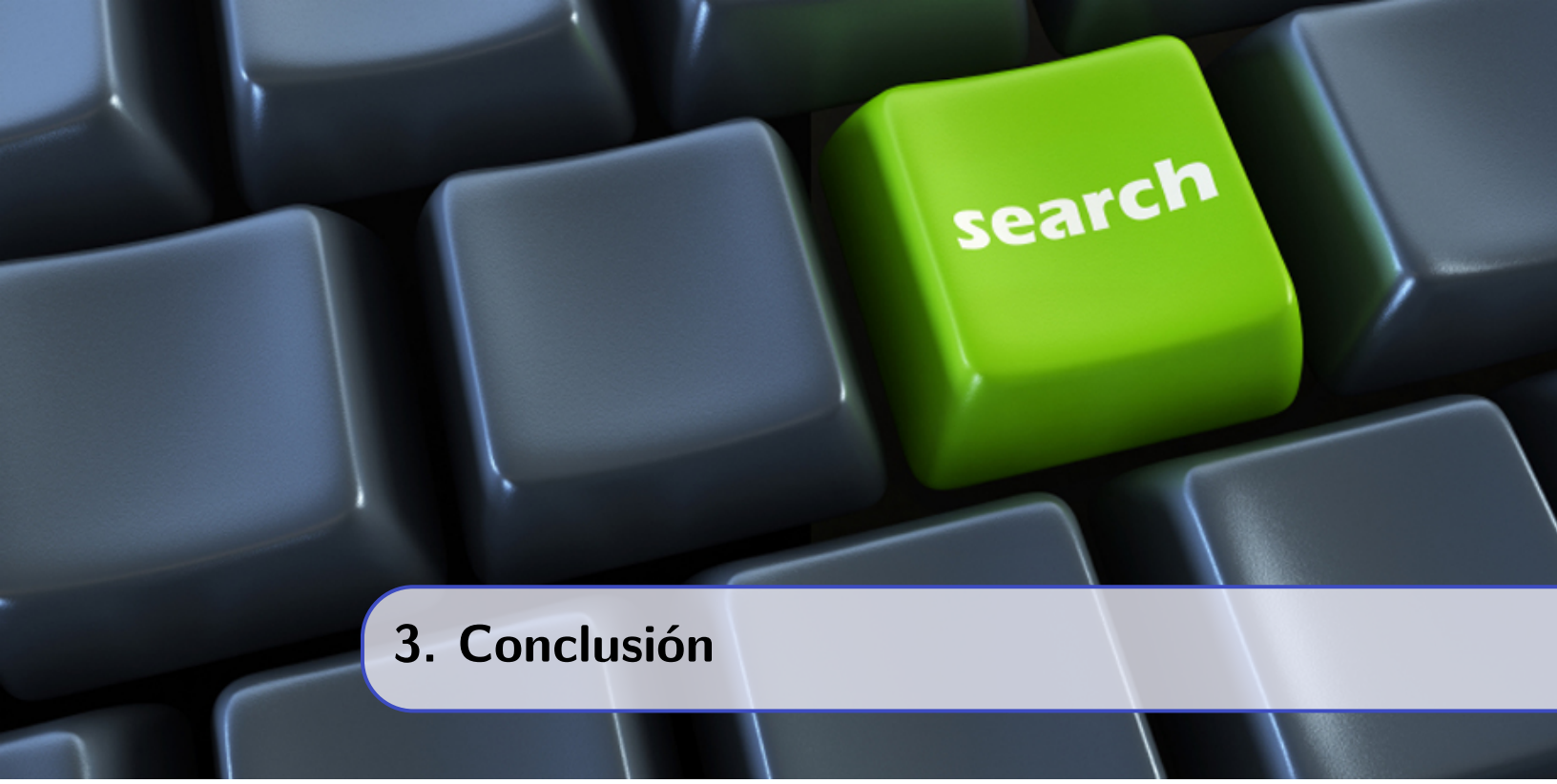
(a) Frame extraído

[caracteres sin sentido alguno]

(b) Texto reconocido

Figura 2.14: Imágen con filtro ImageMagick

Sin embargo, no se obtuvieron buenos resultados en general y fue descartada.



3. Conclusión

3.1 Conclusiones

Desde el comienzo la idea nos entusiasmó mucho. Sobre todo nos gustó porque no encontramos antecedentes de aplicaciones de este tipo, y creemos que es algo de gran utilidad, por lo que el potencial es grande.

La aplicación final cumple con todos los objetivos propuestos al principio. El tiempo permitió complementar el manejo de videos con el manejo de imágenes, y así cubrir un abanico más amplio de sistemas objetivo.

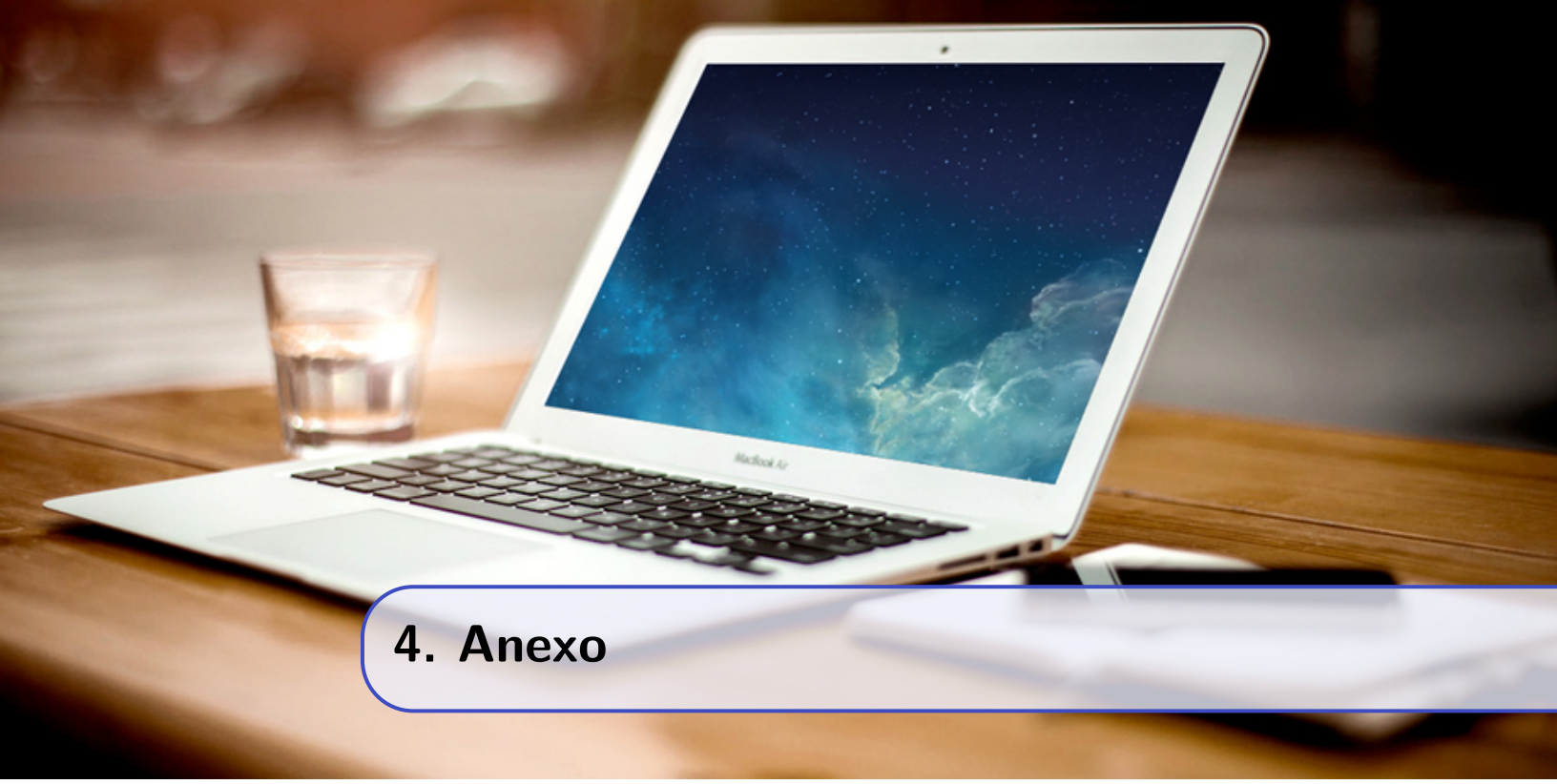
3.2 Trabajo a futuro

Lo primero que debería mejorar en un futuro es el reconocimiento de texto. Si bien muchas veces reconoce correctamente el texto, en algunas ocasiones Tesseract reconoce algunos caracteres extraños que claramente no están en la imagen. Probamos distintas configuraciones del motor y distintas herramientas de mejora de imagen concebidas específicamente para resaltar el texto en imagenes, pero no logramos eliminar estos problemas ocasionales. Otros motores de pago que probamos no presentaban este problema, pero no logramos conseguir licencias para utilizarlos.

Otra cosa que ayudaría mucho es la corrección de errores ortográficos. Utilizando algún sistema de corrección de errores se solucionaría gran parte de los errores generados por el OCR.

Una idea que quedó fuera del alcance del proyecto, y que optimizaría el procesamiento, es la de utilizar algún tipo de heurística para determinar con algún nivel de probabilidad si la imagen contiene texto. Si fuera posible de forma poco costosa determinar esto, podría reducirse considerablemente los fotogramas a ser procesados por el OCR, que es la etapa más costosa del procesamiento. Otra forma de reducir la cantidad de fotogramas a procesar,

es determinar si el fotograma actual es el mismo o muy parecido al anterior. De esta forma se podría reducir el costo de extraer fotogramas mas contiguos y consecuentemente perder menos contenido.



4. Anexo

4.1 Uso de la aplicación

El flujo de la aplicación es bastante sencillo. El primer paso consiste en la subida de nuevo contenido (videos y/o imágenes).

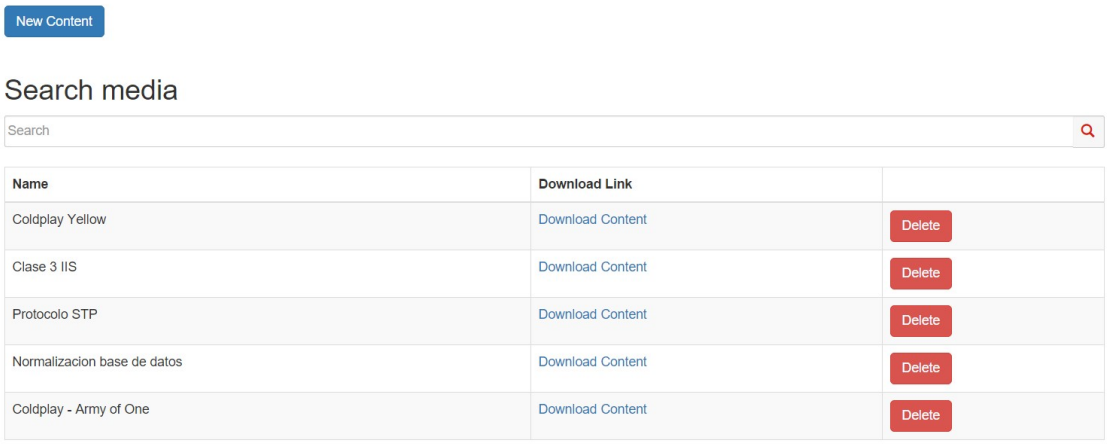


Figura 4.1: Pantalla principal

Como se puede observar, el nuevo contenido es agregado mediante el botón "*New Content*"; al agregar un nuevo contenido es necesario especificar el nombre del mismo.

Luego de subirlo, se realizará todo el proceso descrito anteriormente y luego se podrá recuperar el video/imágen mediante el campo de búsqueda.

Desde la misma pantalla principal es posible tanto descargar como borrar contenido

alojado en la aplicación.

Bibliografía

[1] *Clase 3 de IIS en OpenFing*. URL: http://open.fing.edu.uy/iis/clase/3 (véase página 12).

[2] *FFmpeg*. URL: https://www.ffmpeg.org/ (véase página 6).

[3] Danilo Jeremias da Silva. *rtesseract*. URL: https://github.com/dannnylo/rtesseract (véase página 6).

[4] Ray Smith. *Tesseract*. 2015. URL: https://github.com/tesseract-ocr/tesseract (véase página 6).

[5] *Sunspot*. URL: https://github.com/sunspot/sunspot (véase página 6).

[6] Wikipedia. *Apache Solr*. URL: https://en.wikipedia.org/wiki/Apache_Solr (véase página 6).

[7] Wikipedia. *Reconocimiento óptico de caracteres*. URL: https://es.wikipedia.org/wiki/Reconocimiento_%C3%83%C2%B3ptico_de_caracteres (véase página 5).