Universidad Zaragoza

TRABAJO FIN DE GRADO

DE INGENIERÍA INFORMÁTICA

Anotación semántica de grandes colecciones de vídeos a partir de su audio

Directores

Sandra Baldassarri

Pedro Javier Álvarez Pérez-Aradros

Autor

Daniel Delgado Llamas

Escuela de Ingeniería y Arquitectura

—

Zaragoza, Septiembre de 2015

Resumen

En Internet existen numerosos servicios de alojamiento y visualización de vídeos. Este tipo de plataformas permiten realizar búsquedas entre las grandes colecciones de vídeos que almacenan. Estas búsquedas comparan la consulta realizada por el usuario con la información que disponen de cada uno de sus vídeos. La fuente de información para este tipo de búsquedas puede ser muy distinta dependiendo de cada servicio de alojamiento. En general, los elementos más importantes que clasifican un vídeo son: el título y las etiquetas asociadas a cada vídeo y que describen su contenido.
El título y las etiquetas de un vídeo son, normalmente, asignados de forma manual por el usuario que aloja el vídeo en el servidor. El etiquetado de vídeos consiste en asociar o añadir a la información del vídeo un conjunto de palabras o metadatos (datos que describen otros datos) que permita clasificar o describir el contenido del vídeo. Esta tarea, con colecciones de vídeos muy grandes, resulta muy costosa y a veces difícil de realizar por un usuario por desconocimiento del contenido del vídeo. En general, el etiquetado manual produce un conjunto de etiquetas que describen el contenido de los recursos multimedia de forma genérica.
Este proyecto pretende conseguir que este proceso de etiquetado manual, por parte del usuario, se realice de forma automática, permitiendo así reducir costes en el tiempo de etiquetado y generando unas etiquetas que describan mejor el contenido del vídeo utilizando información presente en el audio del mismo, mejorando de esta forma la relevancia de las etiquetas con respecto al etiquetado manual.

Este proceso de etiquetado consiste, en primer lugar, en obtener o descargar el vídeo solicitado, siguiendo con la extracción de su audio y el posterior reconocimiento de voz del mismo para obtener el mayor número de palabras, de este audio, en forma de texto. A continuación, a partir de un proceso de votación, en el que a partir de los resultados obtenidos mediante el uso de algoritmos de extracción de términos, se determinan las palabras más relevantes que serán utilizadas como etiquetas para clasificar el vídeo. Estas etiquetas obtenidas pasan un proceso de validación, por parte de usuarios, para comprobar que son adecuadas y corresponden al contenido del vídeo. Se decidió que este proceso de validación se realizase de forma externa al sistema de etiquetado de vídeos para que el sistema pudiese continuar aunque la validación no hubiera finalizado. Finalmente, estas etiquetas pasan por un proceso en el que se relacionan estas palabras más relevantes con conceptos de una ontología (esquema conceptual dentro de varios dominios con la finalidad de facilitar el intercambio de información entre diferentes sistemas y entidades) de referencia, obtenida del repositorio de información estructurada llamado *dbpedia* para obtener mayor significado e información sobre el vídeo procesado.
Este proceso de etiquetado de vídeos se encapsula dentro de un servicio que permite la entrada de vídeos y genera como resultados un conjunto de metadatos que describen los vídeos en base a su contenido. Este proceso de etiquetado consta de unas etapas con una alta complejidad, como puede ser la extracción del audio o el reconocimiento de voz, que genera la necesidad de una búsqueda de herramientas a integrar en este proceso que resuelvan cada etapa. Finalmente, se lleva a cabo una evaluación de este proceso de etiquetado automático, valorando aspectos como el rendimiento (costes de computo) o la calidad (tasa de aciertos) de algunos de los elementos que intervienen en ese proceso.

Agradecimientos

En primer lugar, agradecer a toda mi familia el apoyo recibido en especial a mi madre y a mi padre, sin su ayuda la realización de este proyecto no habría sido posible.

También quería tener una especial mención para Verónica, por todo su apoyo, ayuda y atención durante la realización del proyecto y la carrera.

Agradecer también a todos mis amigos y compañeros que directa o indirectamente me han ayudado o me han hecho más sencillo el transcurso de estos años.

Finalmente, y no por ello menos importante, agradecer a mis dos directores la oportunidad de llevar a cabo este proyecto, la paciencia que han tenido conmigo y toda la ayuda recibida.

Índice de contenidos

Índice de figuras

Índice de tablas

1 Introducción

En esta sección se describen los motivos que han llevado a desarrollar este trabajo, su contexto, el problema que aborda y los objetivos que se han planteado. Finalmente se indican las distintas partes de las que consta este documento.

1.1 Motivación y contexto

Con el aumento de la popularidad de los servidores de alojamiento de contenido audiovisual en Internet, se disponen de grandes colecciones de vídeos y surge la necesidad de documentar o clasificar estos recursos. El etiquetado de vídeos consiste en asociar o añadir a la información del vídeo un conjunto de términos o metadatos (datos que describen otros datos) que permita clasificar o describir el contenido del vídeo. Lo normal es asignarles ciertos metadatos que describan el vídeo de forma general. En este proyecto se pretende que el etiquetado se realice en base a su contenido.
Esta clasificación tiene como finalidad, dentro del proceso de recuperación de información, mejorar y agilizar la tarea de búsqueda de vídeos y/o facilitar el acceso a ellos. Actualmente esta clasificación se realiza de forma manual por parte de los usuarios. La motivación de este proyecto es conseguir diseñar un sistema capaz de realizar ese proceso de clasificación automáticamente en base a su contenido.
Por último, otra de las motivaciones de este proyecto es conseguir que los recursos estén entrelazados (*linked data*). Cada etiqueta asociada a cada vídeo estará relacionada con distintos conceptos que tendrán una relación semántica entre ellos y que podrán o no estar relacionados con otras etiquetas de otros vídeos, consiguiendo así la interrelación de etiquetas y por lo tanto la interrelación de vídeos, una característica muy importante dentro de la situación actual de la web (Web Semántica [1]).

1.2 Problema a resolver y objetivos de alto nivel

El problema que resuelve este proyecto es el etiquetado o clasificación de grandes colecciónes de vídeos de forma automática en base a la información contenida en el audio. Para llevar a cabo el etiquetado se necesita obtener información sobre el contenido del vídeo. Para ello se propusieron dos métodos para conseguirlo: utilizar los subtítulos del vídeo (si estuvieran disponibles) o procesar la información contenida en el audio. Finalmente se ha optado por el uso de la información contenida en el audio debido a que no siempre se dispone de los subtítulos de los vídeos. Por otro lado, para comprobar que este etiquetado es correcto y las etiquetas generadas corresponden al contenido del audio, se realizará un proceso de validación de los metadatos obtenidos. Al realizar un análisis previo del problema se observó que estaba compuesto por varios subproblemas: la desgarga del vídeo, la extracción del audio, el reconociemiento de voz, la extracción de términos clave, la validación de términos clave y un subproblema final de obtención de más metadatos relacionando las etiquetas con conceptos pertenecientes a ontologías (esquema conceptual con la finalidad de facilitar el intercambio de información entre diferentes sistemas y entidades).
Para resolver el problema planteado, se han definido varios objetivos generales. El primer objetivo a tratar es buscar y analizar tecnologías ya existentes que resuelvan parcial o completamente algunos de los subproblemas del sistema de etiquetado.
El siguiente objetivo es realizar la implementación de dos aplicaciones de soporte para el sistema, así como de la propia implementación del sistema: desarrollar una aplicación web que permita la interacción de los usuarios con el sistema y desarrollar un interfaz web que permita a los usuarios realizar una validación de los resultados del sistema. También se realizará la incorporación

y adaptación (si fuera necesario) de las tecnologías analizadas que han sido seleccionadas para resolver alguna de las tareas del sistema.
Por último se llevará a cabo una evaluación de las prestaciones del sistema en cuanto a rendimiento y calidad. También se realizarán evaluaciones intermedias para tomar decisiones al seleccionar herramientas.

1.3 Estructura de la memoria

La estructura de esta memoria se compone de seis capítulos o apartados:

Capítulo 1: Breve introducción del proyecto, exponiendo el problema que aborda y el contexto por el cual surge.
Capítulo 2: Análisis del proyecto. Descripción de los objetivos concretos y sus requisitos y las tecnologías analizadas.
Capítulo 3: En este apartado se explica la fase de diseño del sistema. Descripción del entorno, sus componentes y la arquitectura utilizadas.
Capítulo 4: Implementación del sistema. Descripción detallada de la implementación del sistema de etiquetado.
Capítulo 5: Resultados obtenidos respecto a la calidad y el rendimiento del sistema.
Capítulo 6: Gestión del proyecto. Descripción de la metodología utilizada, junto a la planificación y los esfuerzos dedicados.
Capítulo 7: Conclusiones y trabajo futuro. Conclusiones sacadas sobre el proyecto y el futuro trabajo que se puede hacer a partir del mismo.

2 Análisis

En esta sección se describen los objetivos del sistema propuestos. También se explicarán las herramientas/tecnologías que han sido analizadas para integrarlas o no, razonando por qué han sido descartadas o por qué han sido seleccionadas.

2.1 Objetivos más concretos y requisitos

El sistema de etiquetado ha desarrollar se divide de distintas tareas para obtener los resultados deseados: generar las etiquetas de los vídeos de forma automática en base a la información contenida en el audio. El objetivo inicial es llevar a cabo una búsqueda de herramientas o tecnologías que puedan resolver los subproblemas detectados del sistema de etiquetado: la descarga del vídeo, la extracción del audio, la extracción de términos clave, la validación de términos clave y la adición de nuevos términos con una relación semántica a los ya obtenidos.
Se realizará un análisis de las características de las herramientas y según unos criterios que se definirán en la siguiente subsección serán seleccionadas. Si varias herramientas han sido seleccionadas para resolver un mismo subproblema se realizará una evaluación de su rendimiento y la calidad de los resultados, y se seleccionará la que mejores resultados muestre.

El siguiente objetivo es realizar la implementación de todos los componentes de los que consta el sistema de etiquetado de vídeos por contenido. Por un lado, desarrollar un componente que permita a los usuarios enviar al sistema de etiquetado el conjunto de vídeos a etiquetar y recibir los resultados generados.
Por otro lado, realizar la implementación del sistema de etiquetado compuesto por las distintas etapas o subproblemas detectados. Cada uno de estos subproblemas será resuelto por alguna de las tecnologías seleccionadas, por lo tanto, otro objetivo será la integración de estas tecnologías al sistema.
Otro objetivo es realizar la implementación del sistema de validación de términos clave. Este sistema deberá permitir a los usuarios validar los términos obtenidos por el sistema, indicando qué términos son correctos y cuáles no.

Por último se realizará una evaluación del sistema de etiquetado teniendo en cuenta aspectos del rendimiento y la calidad. Para aspectos de rendimiento se realizarán medidas de tiempo, tanto de procesamiento o de latencias en escritura y lectura. Para aspectos de la calidad de los resultados se medirá la tasa de aciertos del sistema.

2.2 Requisitos funcionales

RF1: La interfaz de la aplicación web dispondrá de un formulario para que los usuarios interactúen con el sistema.
RF2: La aplicación web debe permitir al usuario iniciar el proceso de etiquetado de vídeos suministrando una o o un conjunto de URLs que identifican uno o varios recursos (vídeos).
RF3: La aplicación web deberá devolver al usuario los resultados del etiquetado de vídeos una vez haya acabado el proceso.
RF4: El sistema deberá ser capaz de descargar el vídeo que identifica una URL y almacenarlo localmente.
RF5: El sistema deberá ser capaz de extraer el audio de un vídeo y almacenarlo.
RF6: El sistema será capaz de transcribir las palabras habladas contenidas en un fichero de audio y almacenar el resultado en un fichero de texto.

RF7: El sistema será capaz de determinar y extraer los términos más significativos de un texto y almacenar los resultados en un fichero de texto.
RF8: El sistema deberá ser capaz de validar términos mediante encuentas creadas automáticamente vía web con la participación de usuarios y almacenar los resultados de la validación.
RF9: El sistema será capaz de generar nuevos términos con una relación semántica a otro conjunto de términos.

2.3 Requisitos no funcionales

RFN1: El sistema deberá ser capaz de procesar grandes colecciones de vídeos.
RFN2: La aplicación web deberá comportarse de la misma manera en todos los navegadores web.
RFN3: La aplicación web podrá manejar de forma concurrente peticiones de varios usuarios.

2.4 Análisis de herramientas

El etiquetado de un vídeo es el proceso por el cual un usuario o entidad asigna un conjunto de palabras o términos significativos que describan su contenido, permitiendo así tener disponible información asociada y clasificarlo.
Este etiquetado se lleva a cabo en distintas etapas o subproblemas tal y como se observa en la figura 1. En primer lugar se descarga el vídeo alojado en algún servidor de almacenamiento. Después se extrae el audio, con el que utilizando alguna herramienta de reconocimiento de voz, se extraen las palabras que contiene. A continuación se determinan cuáles son las palabras más significativas o que más ocurrencias tienen dentro de todo el conjunto. Estas palabras más significativas pasan un proceso de validación por parte de usuarios que verifican que esas palabras ciertamente aparecen en el vídeo y describen su contenido. Finalmente, las palabras más relevantes se "mapean" a conceptos de una ontología de referencia.
Cada una de estas etapas consume unos datos y genera otros, formándose una cadena o flujo que se puede observar representado mediante flechas en la figura 1. A este flujo entran URLs que desencadenan el flujo de eventos y finalmente se consiguen como resultado los vídeos etiquetados.

Para el análisis que se ha realizado de las tecnologías a usar se han tenido en cuenta las siguientes características:

- Funcionalidad.
- Formatos de entrada y salida.
- Desarrollado en Java o facíl integración en una aplicación Java.
- Políticas y precios de uso.

Una vez se ha hecho este primer análisis, si para un subproblema en concreto, se tiene más de una herramienta aceptada se realizan unas pruebas de rendimiento y calidad para finalmente seleccionar la que mejor resultados ofrezca:

- Rendimiento.
- Tasa de fallos/errores.

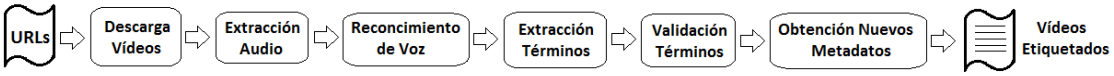

Figura 1: Etapas del sistema de etiquetado.

2.4.1 Descarga de vídeo

Para la descarga de vídeos se hizo una búsqueda en Internet y como resultado se encontraron gran cantidad de aplicaciones web y de escritorio que resolvían este problema, y que no es posible integrarlas al resto del sistema, como por ejemplo: *ClipConverter* [2] (Aplicación Web) y *aTube Catcher* [3] (Aplicación de escritorio). Como el objetivo es poder integrar una tecnología que resuelva el problema dentro del sistema se descartaron. Se hizo una nueva búsqueda más refinada en busca de alguna librería o módulo que se pudiera incorporar sin problemas al sistema. Se encontraron únicamente dos herramientas con ese propósito: *VGET* y *YoutubeDL*.
La idea que reside por debajo de estas dos herramientas es igual que la forma de visualizar vídeos a través de un navegador web en algún servidor de alojamiento de vídeos. Al visualizar un vídeo en un navegador web lo que realmente se hace por debajo es una petición HTTP (Hypertext Transfer Protocol) que permite la comunicación entre un cliente (el navegador web) y un servidor (el servidor de alojamiento de vídeos). Concretamente se realiza una petición HTTP de tipo GET, que solicita información o datos al servidor, en este caso solocita un recurso vídeo. Un navegador al realizar esta petición recibe el recurso vídeo y lo almacena para poder visualizarlo cuando el usuario desee dentro de la web. Por lo tanto, las dos herramientas encontradas utilizan esta idea para capturar el vídeo. Realizan una petición GET contra el servidor y al recibir el vídeo lo copian desde su ubicación a otro directorio completando así la descarga.

La herramienta *VGET* [4] es una librería escrita en *JAVA*. Solo soporta descargas de los servidores de *Youtube* y *Vimeo*. Como entrada se requiere la URL del recurso y como salida se obtiene el vídeo en formato *FLV*.
La herramienta *YoutubeDL* [5] es un *script* escrito en *Python*, con soporte para cientos de servidores de alojamiento de vídeos (que la comunidad de desarrolladores de *GitHub* aumenta constantemente). Como entrada se requiere la URL del vídeo y como salida se obtiene el vídeo en el formato con el que esté almacenado en el servidor. En la tabla 1 se pueden visualizar las características principales de ambas herramientas.

Herramienta	Leng. Progr.	Entrada	Salida	Servidores soportados
VGET	JAVA	URL	Fichero FLV	YouTube y Vimeo
YouTubeDL	Python	URL	Fichero con formato de origen	Más de 100 servidores

Tabla 1: Herramientas analizadas para la descarga de vídeos.

Finalmente se seleccionó la herramienta *YoutubeDL* principalmente por la gran cantidad de servidores a los que da soporte. Otra de las razones fue que la herramienta *VGET* no funciona correctamente y devuelve mensajes de error en su funcionamiento, por otro lado no recibe actualizaciones desde hace un año lo que puede provocar este mal funcionamiento.

2.4.2 Extraer el audio

Para la tarea de extracción del audio, solo se encontraron dos herramientas: *JAVE* y *Xuggler*, las cuales se pueden ver en la tabla 2.
La herramienta *JAVE* [6] (*Java Audio Video Enconder*) es una librería escrita en *JAVA* que encapsula el *framework FFmpeg* [7]. *FFmpeg* es un *software* de código abierto escrito en lenguaje *C*. Para que la librería de *JAVE* funcione correctamente se necesita tener instalada una versión precompilada de *FFmpeg* en la máquina local. *JAVE* permite transcodificar ficheros de audio y vídeo desde un formato a otro. Soporta multitud de formatos y permite redimensionar el tamaño de los vídeos, así como sus proporciones.
La herramienta *Xuggler* es una librería también escrita en *JAVA* que permite descomprimir, modificar y recomprimir cualquier fichero multimedia desde *JAVA*. Esta herramienta además permite capturar en tiempo real imágenes del escritorio de la máquina local o de una cámara conectada a la máquina. Asimismo hace uso por debajo del *framework FFmpeg*. Tiene soporte para multitud de formatos. En la tabla 2 podemos observar que ambas herramientas tienen características semejantes, salvo en aspectos de funcionalidad, como la captura de imágenes de una cámara en tiempo real, que *Xuggler* sí que lo permite, pero *JAVE* no.

Herramienta	Leng. Progr.	Entrada	Salida
JAVE	JAVA	Vídeo: MP4, AVI, FLV, M4V, ...	Audio: MP3, WAV, AC3, FLAC, ...
Xuggler	JAVA	Vídeo: MP4, AVI, FLV, M4V, ...	Audio: MP3, WAV, AC3, FLAC, ...

Tabla 2: Herramientas para la extracción de audio.

Como conclusión, se ha seleccionado finalmente la herramienta *JAVE* debido a que su funcionalidad es más sencilla que *Xuggler* y cubre con las necesidades del subproblema a tratar: extraer el audio de un vídeo. La librería de *JAVE* es por ello mucho más sencilla, con menos clases o componentes y su utilización consta de pocas instrucciones para resolver la tarea, en comparación con la herramienta *Xuggler* que ofrece gran cantidad de funcionalidades que para resolver el subproblema planteado no se necesitan.

2.4.3 Reconocimiento de voz

En general existen multitud de soluciones o herramientas que abordan este problema: el reconocimiento de la voz humana. Existen distintos usos de aplicación de este campo, como pueden ser: transcribir la voz de una persona a texto, reconocer la voz para interpretar órdenes, acciones o autenticar accesos restringidos, etc. En este proyecto se busca en este tipo de herramientas conseguir transformar la voz en texto. En una entrada en la enciclopedia libre *Wikipedia*, se puede encontrar un listado de algunas de estas herramientas enfocadas cada una a distintos entornos y sistemas [14]. De ese listado sólo se seleccionó una herramienta para analizar: *CMU Sphinx*. El motivo fue que era la única herramienta escrita en *JAVA* y que diera soporte a un entorno *Linux*.
Aparte de *CMU Sphinx*, se seleccionaron otras dos herramientas realizando otras búsquedas en Internet. Una de ellas fue *Google Speech API*, un servicio web proporcionado por *Google* para desarrolladores del grupo *Chromium-dev* [22], al cual se puede ingresar libremente. La última herramienta seleccionada para analizar es un servicio web de reconocimiento de voz llamado *Speech to Text* desarrollado por la compañia *IBM* [11].
El motivo principal para la selección de estas tres tecnologías fue que dentro de algunas restricciones no eran de pago, mientras que la gran mayoría requerían de una compra prevía o pago por

su uso. Otro de los aspectos de su selección fueron los comentarios de la comunidad haciendo recomendaciones y reseñas positivas hacia ellas.

La herramienta *CMU Sphinx* [10] es una librería escrita en *JAVA* que permite realizar este proceso de reconocimiento de voz de forma *offline*, sin depender de un servicio web externo. Esta herramienta se puede configurar para ser utilizada por un lenguaje u otro, basta con modificar tres elementos: el modelo acústico, el modelo del lenguaje y el diccionario del lenguaje a reconocer por el sistema.
La herramienta *Google Speech API* [9] es un servicio web externo desarrollado por *Google* que permite realizar peticiones enviando un fichero de audio y recibiendo como respuesta el resultado de la transcripción en formato JSON. El audio enviado no puede superar los 12-15 segundos de duración sino devuelve un resultado vacio. Por otro lado la versión gratuita de este servicio tiene limitado su uso a 50 peticiones diarias.
La herramienta de reconocimiento de *IBM*, *Speech to Text* [13] es un servicio web externo igual que la anterior herramienta, el cual permite realizar peticiones de reconocimiento de voz enviando un fichero de audio y recibiendo como respuesta los resultados en formato JSON. El audio enviado deberá tener una duración máxima de 3 minutos, de lo contrario devolverá una respuesta errónea. Por otro lado, al contrario que la herramienta de *Google*, no tiene límite de peticiones.
Los dos servicios web presentados parece que tienen unas características muy semejantes pero como se puede observar en la tabla 3 cada una tiene unas restricciones distintas en su uso. En la tabla se muestra una comparativa entre las tres herramientas descritas.

Herramienta	Acceso/Leng.Prog	Entrada	Salida	Políticas/Restricciones
Google Speech API	Servicio Web (online)	Audio FLV	Fichero JSON	· 50 peticiones/día · 12-15 seg. audio de entrada
CMU Sphinx	Librería JAVA (offline)	Audio WAV	String	Sin límite
IBM Watson Developer Cloud	Servicio Web (online)	Audio WAV	Fichero JSON	· Sin límite de peticiones · 3 min. audio de entrada

Tabla 3: Herramientas para el reconocimiento de voz.

Como conclusión, se descartó la herramienta proporcionada por *Google* debido a sus fuertes restricciones en la duración del audio de entrada en la versión gratuita. Por consiguiente, quedaban otras dos herramientas. Para determinar cuál era la que mejores resultados ofrecía se llevo a cabo una evaluación intermedia de caliadad y rendimiento cuyos resultados se pueden analizar en el apartado de evaluación intermedia 5.1 de este documento. A partir de la evaluaciónn se decició seleccionar el servicio web proporcionado por *IBM* ya que mostraba un coste de computo de la mitad de tiempo que la otra alternativa (*CMU Sphinx*) y obtenía unos resultados con una tasa de aciertos muy superior (de media un 30% mejor).

2.4.4 Extracción de términos clave

Para la extracción de términos clave se va a hacer uso de la herramienta *JATE* [15]. *JATE* es un *framework* compuesto por una colección de algoritmos destinados a la extracción de términos dentro de un documento o un *corpus* de documentos. Al tratarse de una librería escrita en *JAVA* no supone ningún problema el integrarla dentro del sistema.
La versión que se ha utilizado de *Jate* (v1.1) implementa 8 algoritmos para la extracción de

términos clave: *Simple term frequency*, *TF.IDF*, *Weirdness*, *C-Value*, *GlossEx*, *TermEx*, *RIDF* y *Average Term Frequency in Corpus*. Cada uno de estos algoritmos realiza la extracción de términos según distintas estrategias. La mayoría de estos algoritmos tratan cada palabra como un candidato a término clave mientras que, por ejemplo, el algoritmo *C-Value* utiliza conjuntos de palabras (*multi-words*) como candidatos a términos clave.
Una funcionalidad importante de este *framework* es que aparte de la colección de algoritmos para la extracción de términos, también implementa un algoritmo de votación por pesos para poder utilizar un subconjunto de los algoritmos de extracción de manera conjunta asignándole a cada uno de esos algoritmos un peso de mayor o menor relevancia.
Como trabajo de este proyecto se implementó un algoritmo de votación por mayoría que se utilizó con el mismo propósito que el de votación por pesos: utilizar de manera conjunta los distintos algoritmos de extracción del *framework*. Este algoritmo de votación por mayoría consiste en recuperar los resultados de aplicar cada algoritmo de extracción por separado y puntuar los primeros veinte términos de cada algoritmo con valoraciones de entre veinte a uno en orden, por ejemplo: el término en la posición uno le corresponde una valoración de veinte, al segundo de diecinueve y así hasta llegar al término veinte con una valoración de uno. Al final de esta puntuación se almacenan los veinte términos con mejores valoraciones y son devueltos como los términos más significativos del documento o *corpus* de documentos.
El resultado final de esta etapa de extracción de términos clave son los resultados obtenidos de la votación por mayoría implementada de los 8 algoritmos de extracción de términos que dispone esta versión de *Jate*.

2.4.5 Validación de términos clave

Para la validación de términos clave se va a utilizar la herramienta *Pybossa* [16] debido a que permite realizar esta tarea de validación de forma colaborativa. La herramienta es una aplicación web con la que se pueden crear proyectos y asignarles tareas a resolver por los usuarios. Para realizar la validación se crea una tarea nueva por cada vídeo que entra al sistema, y en esa tarea se pregunta a los usuarios (anónimos o registrados) si los términos obtenidos corresponden o aparecen en el vídeo. Las tareas tienen que ser contestadas por un número mínimo de usuarios (determinados al crear la tarea) para que los resultados se puedan contrastar y detectar indicios de respuestas incorrectas o incoherencias en los resultados. Al completarse la tarea, el creado del proyecto y de la tarea tienen acceso a la información de los resultados obtenidos.
El objetivo final es poder preguntar, mediante este tipo de encuentas, a usuarios reales si los términos generados por el sistema tienen alguna relación con el contenido del vídeo correspondiente.

2.4.6 Obtención de metadatos

Para la última tarea del sistema, se quiere completar el etiquetado de los vídeos obteniendo y enlazando metadatos con una relación semántica con los términos obtenidos en etapas anteriores. Los metadatos son datos que describen otro datos. Con estos metadatos se quiere ampliar semánticamente el significado de los términos asignados al vídeo.
La herramienta que se va a utilizar para realizar o resolver esta tarea es: *Pisixde*. Esta herramienta es la implementación de los algoritmos expuestos en el artículo de investigación *'Semantic linking of learning object repositories to dbpedia'* [17], y dicha herramienta fue desarrollada por los propios autores del artículo.
Esta herramienta permite, dados unos términos de entrada, generar un grafo de instancias de la *dbpedia* [18] que describe semánticamente un concepto. Esta herramienta realiza las búsquedas haciendo consultas contra la *dbpedia*, un repositorio de información estructurada que ha

generado durante mucho tiempo información semántica a partir de la información almacenada en la *Wikipedia*.

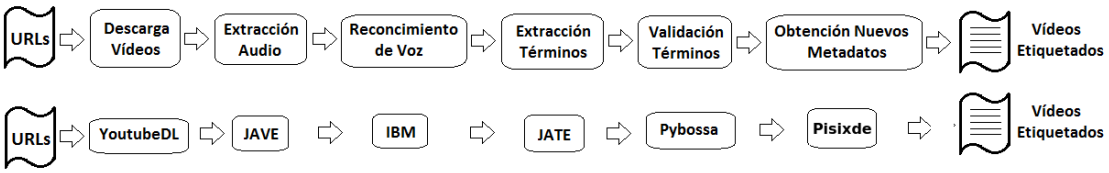

Figura 2: Herramientas seleccionadas para implementar el proceso del sistema de etiquetado.

En la figura 2 se puden observar las herramientas seleccionadas para cada una de las etapas mostradas al principio de la sección.

3 Diseño

En este apartado se va a explicar la estructura del sistema, de que componentes está formado, qué patrones arquitecturales se han utilizado y por último se expondrá la implementación de las distintas partes del proyecto.

3.1 Entorno del sistema

En la figura 3 se puede observar la estructura y componentes del sistema: los actores del sistema, los subsistemas de los que está compuesto y las interacciones entre los distintos componentes.
Antes de pasar a describir el entorno del sistema se van a describir dos elementos que se mencionarán durante las explicaciones: servicio web y aplicación web.
Un servicio web es un módulo que exporta un conjunto de funciones o métodos a aplicaciones a través de la web (Internet) proporcionando independencia de plataformas *hardware/software*.
Una aplicación web es una herramienta o aplicación *software* que los usuarios pueden utilizar accediendo a un servidor web a través de Internet mediante un navegador web.

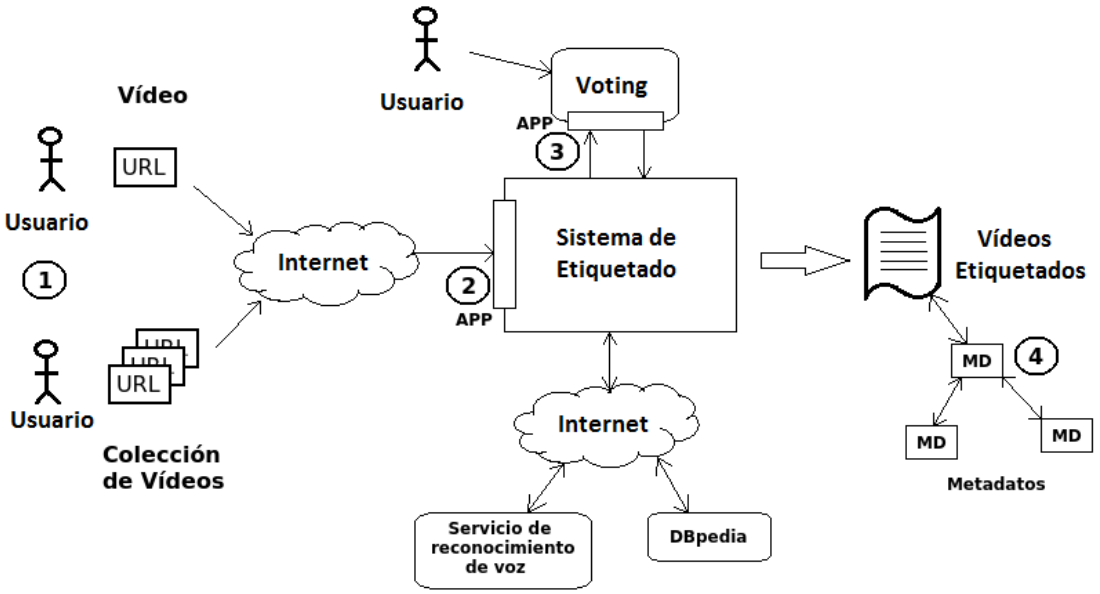

Figura 3: Entorno del sistema.

El sistema de etiquetado interacciona con otros elementos externos para llevar a cabo su objetivo principal: el etiquetado de vídeos.
Los usuarios (1) proporcionan los recursos necesarios al sistema de etiquetado: un vídeo o una colección de vídeos. Estos vídeos se pasan al sistema en forma de URL o un fichero que contiene un conjunto de URLs.
Para realizar la comunicación entre los usuarios y el sistema de etiquetado se ha desarrollado una aplicación web (2) que hace de *frontend* (en diseño *software* es la parte del *software* que interactúa con el o los usuarios). Esta aplicación web se compone de dos sencillos formularios con los que poder enviar al sistema de etiquetado un vídeo (una URL) o una colección de vídeos (un fichero con un listado de URLs). La aplicación web al recibir las peticiones de los usuarios

comienza el proceso de etiquetado de vídeos. Una vez finalizado el proceso de etiquetado, el usuario recibe los resultados como respuesta a la petición realizada en la aplicación web.
En el proceso de etiquetado de vídeos intervienen otros componentes externos necesarios para el sistema de etiquetado: un servicio de reconocimiento de voz, un sistema de *voting* (votación) y por último el repositorio de información descrita semánticamente llamado *dbpedia*.
El servicio de reconocimiento de voz se trata de un servicio web externo, al cual el sistema de etiquetado accede o se comunica a través de Internet. Este servicio de reconocimiento devuelve los resultados obtenidos como respuesta a la petición de reconocimiento de voz sobre un fichero audio realizada.
El sistema de *voting*, para la validación de términos o palabras clave, consta de dos partes: un servicio web y una aplicación web. Este servicio web del sistema de *voting* (3) es el medio por el cual el sistema de etiquetado crea nuevas tareas que se almacenan en la aplicación web con la finalidad de validar los términos o palabras clave de un vídeo. Estas tareas se crean mediante peticiones realizadas por el sistema de etiquetado contra este servicio web.
Por otro lado, la aplicación web del sistema de *voting* permite a los usuarios visualizar las tareas de validación, creadas por el sistema de etiquetado, y participar en ellas a través de la web.
El último componente del entorno es el repositorio de información estructurada *dbpedia* [18]. La *dbpedia* es el resultado, por parte de la comunidad, de extraer información estructurada de la *Wikipedia* [19] y poner esta información disponible en la web. La *dbpedia* permite realizar consultas contra la información contenida en la *Wikipedia* y enlazar los diferentes conjuntos de datos obtenidos a los datos de la propia *Wikipedia*. Este componente es utilizado por el último proceso del sistema de etiquetado, donde se realizan consultas a la *dbpedia* para obtener datos enlazados (*linked data*) sobre los términos clave obtenidos al final del proceso de etiquetado. Estos datos enlazados también se denominan *metadatos* (4) ya que son datos que describen a su vez otros datos.

3.2 Diseño del sistema de etiquetado

El sistema de etiquetado que se ha desarrollado está compuesto de distintas etapas o procesos. Cada proceso necesita datos de entrada proporcionados por procesos o componentes previos y genera datos de salida utilizados por procesos o componentes posteriores.
Esta secuencia de etapas o acciones para llegar a etiquetar un vídeo dentro del sistema de etiquetado tiene una estructura de *'tubería'* en la que los datos van atravesando las distintas etapas en orden.
El sistema de etiquetado consta de distintas etapas o tareas que se han comentado en el capítulo anterior: la descarga del vídeo, la extracción del audio, la extracción de términos clave, la validación de términos clave y la adición de nuevos términos con una relación semántica a los ya obtenidos. Para el diseño del sistema de etiquetado se ha buscado una estructura que permitiera separar estas etapas y a la vez permitir que trabajen conjuntamente para cumplir el objetivo final: el etiquetado de vídeos.
Los recursos que entran al sistema de etiquetado pasan por las etapas mencionadas anteriormente. Cada etapa procesa los datos de entrada, genera unos datos resultantes tras realizar su designada tarea y se los pasa a la etapa siguiente que realiza la misma secuencia de acciones, hasta llegar a la última etapa que genera los resultados finales del sistema de etiquetado.
Con este propósito y estas características se decidió utilizar el patrón *Pipes and Filters* [20] (*pipeline*). Las ventajas de utilizar este patrón son: se adapta a la idea de separar distintas tareas enlazando las salidas de datos de una etapa con la entrada de datos de la siguiente y permite cambiar fácilmente una tarea por otra sin tener que modificar el resto de tareas.
Por otro lado, el incoveniente que tiene es: el tiempo de procesado total es la suma de todos los

tiempos de cada una de las etapas, lo que significa que si una etapa es costosa en tiempo retrasa al resto ya que dependen de los resultados de etapas anteriores.

El patrón *Pipes and Filters* provee una estructura para sistemas que procesan un flujo de datos. Cada etapa de procesado es encapsulada en un componente *Filtro*. Los datos pasan a través de los *Pipes* (o *buffers* intermedios) que se encuentran entre filtros adyacentes. La fuente de datos del sistema es el componente *DataSource* de donde el primer filtro lee los datos y da comienzo el procesamiento de cada dato. El último filtro del sistema genera los datos resultado finales el cual los almacena en el colector de datos llamado *DataSink*.

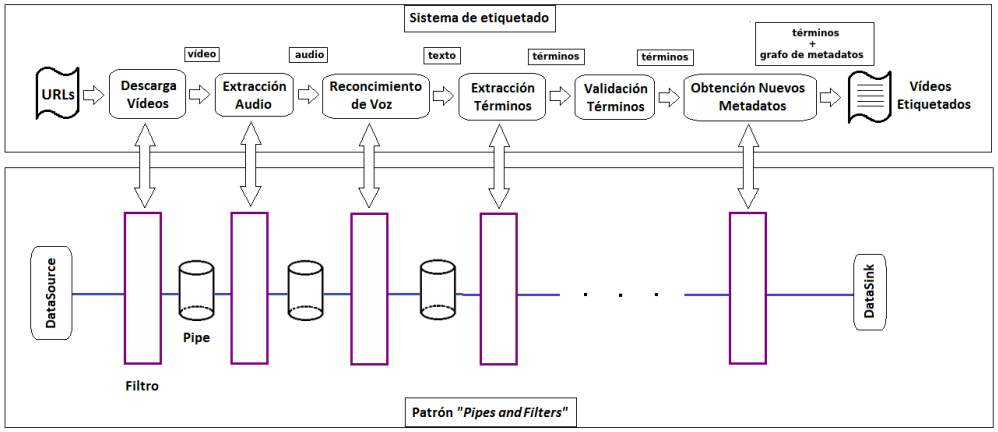

Figura 4: Comparación estructural entre el patrón *Pipes and Filters* y el sistema de etiquetado.

Por lo tanto, el patrón *Pipes and Filters* de forma general está compuesto por estos cuatro componentes: *DataSource*, filtros, *Pipes* y *DataSink*. En la figura 4 se puede observar la correspondencia entre los componentes del patrón y las etapas y elementos del sistema de etiquetado. El componente *DataSource* corresponde con las URLs proporcionadas por los usuarios en la aplicación web. Cada una de las etapas del sistema de etiquetado corresponde al componente *Filtro* del patrón. Finalmente, el componente *DataSink* corresponde con el conjunto de vídeos etiquetados que se devuelven al usuario a través de la aplicación web.

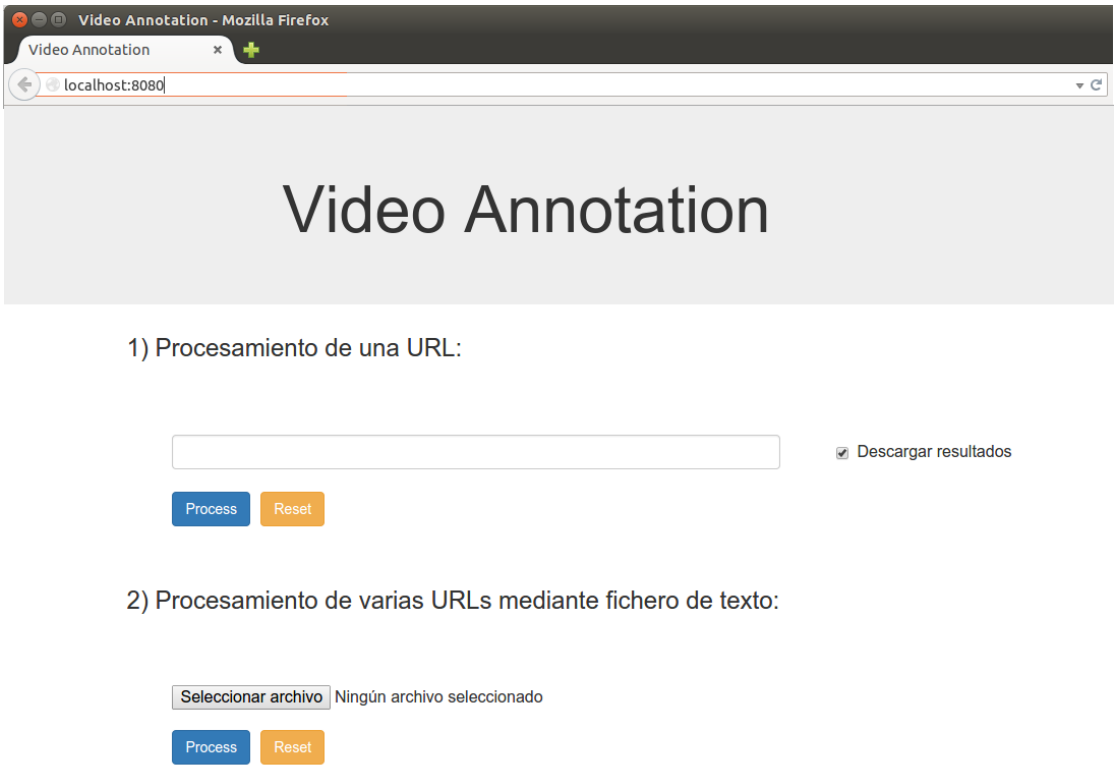

Figura 5: Interfaz de la aplicación web del sistema de etiquetado.

La interfaz web que se puede ver en la figura 5 permite a los usuarios interaccionar con el sistema de etiquetado descrito. Se optó por una interfaz sencilla y clara, separando en dos apartados las dos opciones de envío de URLs al sistema de etiquetado.

Más adelante se explicará como se ha llevado a cabo la implementación de este patrón de diseño *Pipes and Filters*, para instanciar el sistema de etiquetado utilizando la estructura que ofrece, encapsulando las herramientas de cada una de las etapas del sistema de etiquetado en el componente filtro.

3.3 Diseño del sistema de validación de términos

El sistema de validación de términos o *voting* que se ha implementado utiliza el *framework* llamado *Pybossa*. Este *framework* permite crear una aplicación web (ya creada por los desarrolladores del *framework*) donde crear y mostrar tareas a resolver por los usuarios.

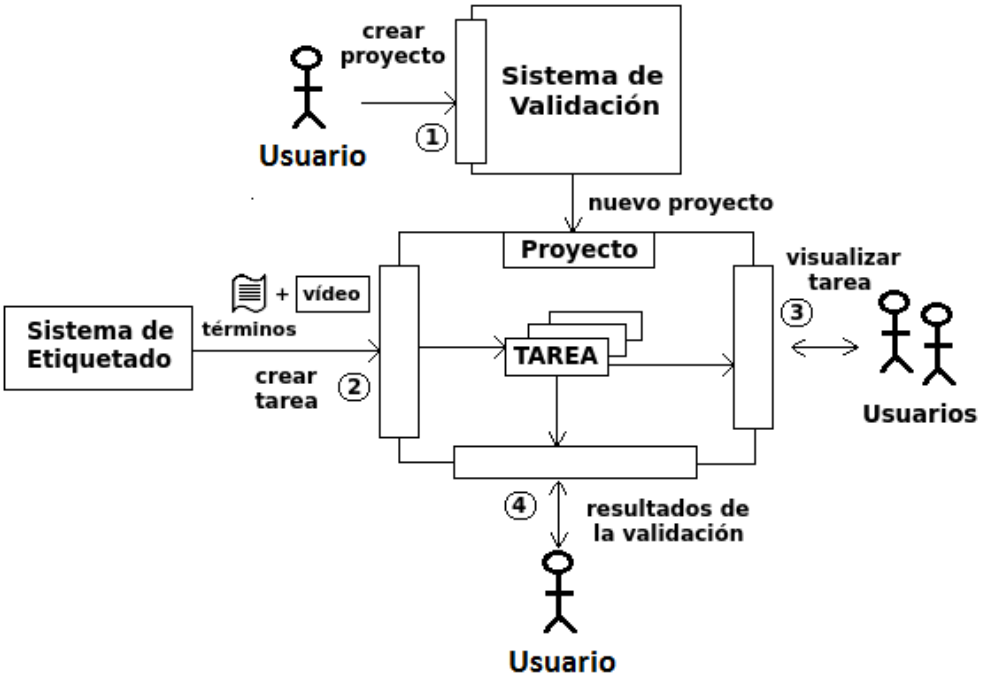

Figura 6: Sistema de validación de términos.

En la figura 6 se puede observar el entorno del sistema de validación. La creación de un nuevo proyecto se realiza, por parte de un usuario, a partir de las opciones que ofrece la aplicación web del sistema de validación (1) al seleccionar en la opción "crear un proyecto". Una vez está creado el proyecto el sistema de etiquetado ya puede añadir nuevas tareas mediante peticiones POST contra su servicio web (2). Para que los usuarios puedan visualizar las tareas disponibles era necesario crear un "presentador de tareas". Este "presentador de tareas" es un fichero HTML que se modifica o edita en un editor interno embebido dentro de la aplicación web de *Pybossa*. El "presentador de tareas" (3) recupera la información de cada tarea y la muestra en formato web. Los usuarios que participan o colaboran en la validación de términos visitan el "presentador de tareas" y completan el formulario mostrado. Una vez la tarea se ha completado, los resultados son accesibles por los usuarios a través de otra interfaz web (4) que permite descargarlos.
En la figura 7 se puede ver un ejemplo de tarea creado por el sistema de etiquetado. Se puede observar: el listado de términos a validar de los cuales hay que seleccionar los términos que aparecen en el vídeo (1), el enlace al vídeo (2) y un botón para enviar el resultado de la validación (3). También está presenta (4) información referente a qué tarea se está resolviendo, y cuál es el progreso de tareas completadas hasta ese momento dentro del proyecto.
En la figura 8 se puede observar un listado de tareas creadas (completadas y en progreso) dentro de un proyecto en la aplicación web de *Pybossa* y junto a cada tarea de la lista se encuentra un botón que da la opción de descargar los resultados.

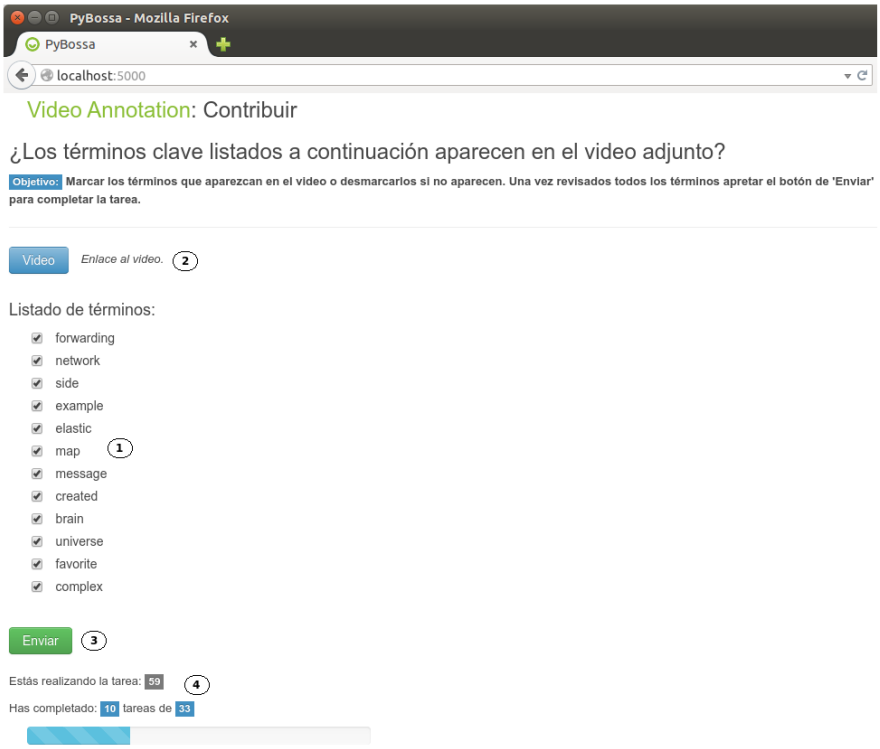

Figura 7: Ejemplo de tarea creada por el sistema de etiquetado en Pybossa.

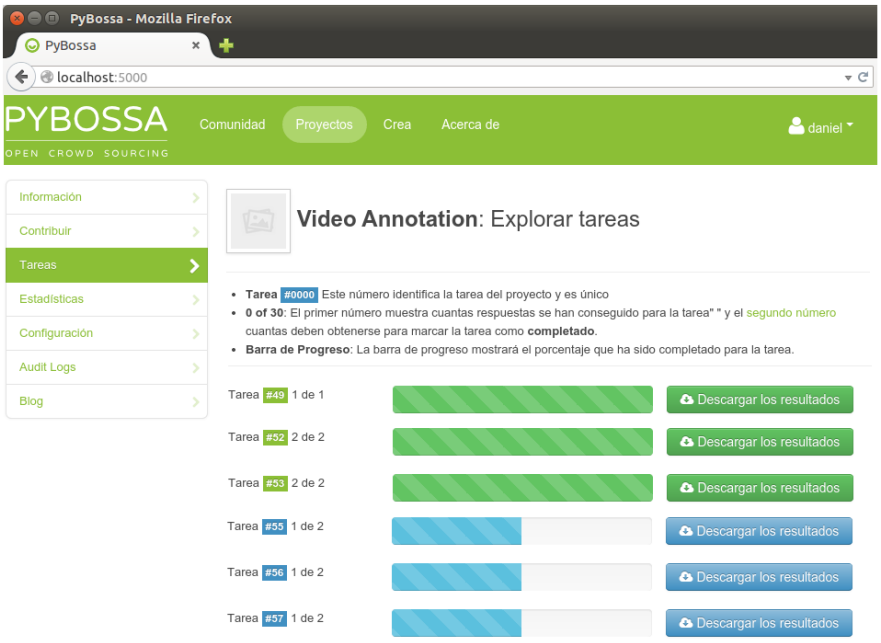

Figura 8: Listado de tareas dentro de un proyecto.

4 Implementación

En este apartado se describe detalladamente la implementación llevada a cabo para desarrollar el sistema de etiquetado.

4.1 Implementación del patrón

En la implementación del patrón de diseño *Pipes and Filters* se han definido un conjunto de interfaces que representan los componentes generales del patrón y que serán el 'esqueleto' de la estructura del sistema de etiquetado. En el diagrama de clases de la figura 10 se pueden observar tres paquetes que conforman todo el sistema de etiquetado. En el paquete *pipeline* se encuentran las interfaces que definen la estructura del patrón: *DataSource*, *Filter*, *BufferPipe*, *DataSink*, *Pipeline* y *Data*.
La interfaz *DataSource* se encarga de leer datos y suministrarlos al primer filtro del sistema. Esta interfaz define un método de lectura (*read()*) que devuelve un elemento de tipo *Data*. La interfaz *Data* representa el flujo de datos que atraviesa el sistema de etiquetado. En la interfaz *Data* se encapsula el tipo de dato o conjunto de datos que utiliza el sistema. La interfaz *Filter* representa cada una de las etapas de procesamiento del sistema de etiquetado. En la interfaz *Filter* se define el método *'doWork()'* cuyo único objetivo es realizar la tarea designada para el filtro del sistema. Cada una de las etapas redefinirá este método para llevar a cabo su tarea concreta. La interfaz *BufferPipe* representa los *buffers* intermedios ubicados entre dos filtros. Su objetivo es almacenar los resultados de la salida de un filtro y proporcionárselos como entrada al filtro siguiente. En la interfaz *BufferPipe* se definen los métodos *'read()'* y *'write()'* que permitirán las lecturas y escrituras en el *buffer*. La interfaz *DataSink* se encarga de almacenar los resultados generados por el último filtro del sistema. En la interfaz *DataSink* se define el método *'write()'* que permite escribir o almacenar los resultados en la ubicación destino.
Por último, la interfaz *Pipeline* encapsula todos los componentes, la estructura y la gestión del patrón utilizado. En la interfaz *Pipeline* se describen los métodos *'design()'* y *'run()'*. El método *'design()'* permite instanciar los componentes del sistema y a continuación con el método *'run()'* se consige iniciar la ejecución *pipeline*.
Una vez definidas las interfaces que describen los componentes del sistema se procede a definir las clases que los implementan.

4.2 Implementación de los *buffers*

La clase *Pipe* implementa la interfaz *BufferPipe*. Esta clase está compuesta por una estructura cola que permite almacenar los resultados de los filtros. Se ha elegido una estructura de datos FIFO (First In First Out) para que los datos circulen por el sistema de etiquetado en el orden que van entrando al *buffer*. La clase *Pipe* implementa los métodos definidos en la interfaz *BufferPipe* (read y write) que permite leer y escribir en la estructura de datos FIFO.
Por otro lado, la clase *Pipe* está compuesta también por un semáforo. Un semáforo es un mecanismo de sincronización que constituye un método para restringir o permitir el acceso a un recurso compartido. En este caso, el recurso compartido es la estructura de datos FIFO (cola). El *buffer* es utilizado por dos filtros: el que genera los datos (y escribe en el *buffer*) y el que los consume (y lee los datos del *buffer*). Por lo tanto es necesario un elemento de sincronización que gestione la entrada y salida de datos del *buffer*: el semáforo. La variable semáforo internamente es un contador. Los semáforos, de forma general, disponen de dos operaciones: *'release()'* y *'acquire()'*. Mediante la operación *'release()'* se incrementa el valor del semáforo en una unidad mientras que con la operación *'acquire()'* se decrementa en una unidad. Si el contador del

semáforo se encuentra a cero y se intenta realizar una operación *'acquire()'*, la entidad que ha realizado la operación queda en un estado de suspensión (o bloqueo) hasta que otra entidad distinta realice la operación de *'release()'* e incremente en una unidad el contador interno del semáforo, en cuyo momento quedará "liberada" la entidad bloqueada. Este mecanismo de sincronización se utiliza con la finalidad de bloquear procesos (filtros) que quieran leer de *buffers* sin datos y se queden a la espera de nuevos datos para poder leer.
En la figura 9 se puede observar la secuencia de acciones que lleva a cabo un filtro y como interacciona con el *buffer*. En primer lugar el filtro 2 realiza la operación *'acquire()'* y queda bloqueado porque todavía no hay datos en el *buffer* (Pipe). El filtro 1 al terminar de realizar sus tareas (*'doWork()'*) escribe sus resultados en el *buffer* mediante la operación *'write()'*. El *buffer*, al recibir un nuevo dato a escribir, realiza internamente la operación *'release()'* liberando al filtro 2 de su bloqueo. A continuación el filtro 2 realiza la lectura de datos del *buffer* mediante la operación *'read()'* y lleva a cabo sus tareas con la operación *'doWork()'*.

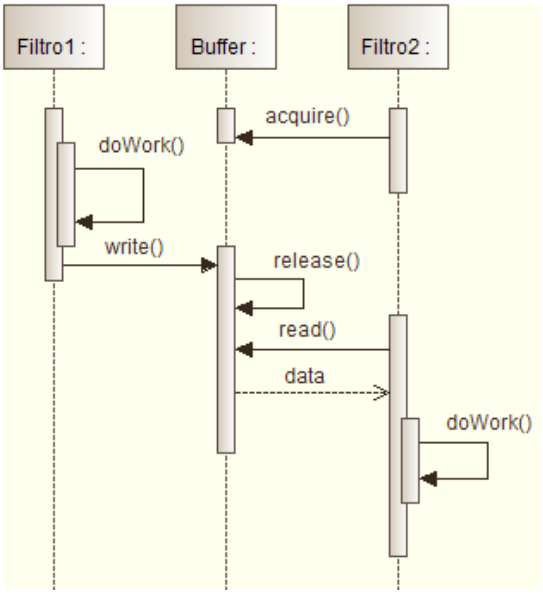

Figura 9: Secuencia de acciones de los *Buffers* y los *Filtros*.

4.3 Implementación de las etapas

La primera fase del sistema de etiquetado es la obtención de las URLs de los vídeos a procesar. Estas URLs son proporcionadas por los usuarios a través de una aplicación web, cuya interfaz ya se ha visto previamente en la figura 5.
Esta aplicación web se ha implementado dentro del paquete *init* que se puede observar en la esquina superior izquierda de la figura 10. La aplicación web está compuesta por una clase llamada *initVT* que hereda de la clase *HTTPServlet*. Un *Servlet* es una clase que permite que una página web se pueda modificar dinámicamente a partir de los parámetros que se envien a través de una petición desde un navegador web. Esta clase recibe las peticiones que realizan los usuarios a través de la interfaz web. El método *'doGet()'* responde a peticiones GET y devuelve el fichero HTML de la interfaz web a los usuarios.
El método *'doPost()'* captura las peticiones POST y procesa la petición. En el procesamiento de la petición POST es donde se inicia la ejecución del sistema de etiquetado y al finalizar, se contesta al usuario con los resultados obtenidos.

La ejecución del sistema de etiquetado se realiza instanciando la clase *VideoLabelling* que implementa la interfaz *Pipeline*. Esta clase define internamente los componentes del sistema (método *'design()'*). A su constructor se le pasan las URLs obtenidas de la petición POST (realizada por el usuario a través de la aplicación web) y el destino deseado de los resultados finales. Para iniciar la ejecución se llama a la operación *'run()'*.
Dentro del método *'design()'* se definen los componentes que forman el sistema. En primer lugar se instancia la clase *URLInputSource* que implementa la interfaz *DataSource*. A la clase *URLInputSource* se le pasan (en el constructor) las URLs que se han recibido por parte de los usuarios, y mediante el método *'read()'* el primer filtro podrá leer cada una de las URLs.
Por otro lado se instancia la clase *OutputSource* que implementa la interfaz *DataSink*. Al constructor de la clase *OutputSource* se le pasa la ubicación donde almacenar los resultados del sistema de etiquetado, que se llevará a cabo con el método *'write()'* que implementa.
El resto de componentes que se definen dentro del método *'design()'* son: todos los *buffers* del sistema, todos los filtros del sistema y todos los semáforos que gestionarán los *buffers*. Antes de acabar se crea un hilo de ejecución (Thread) para cada filtro y finalmente, en el método *'run()'*, se ejecutan todos en paralelo dando comienzo a la ejecución del sistema de etiquetado.

A continuación se va a describir la implementación de las distintas etapas o procesos principales del sistema de etiquetado junto a algunos fragmentos de código relevantes para cada una de las etapas.
Todas las herramientas utilizadas tienen una clase que las implementa o gestiona sus librerías asociadas. Todas estas clases heredan de una misma clase: la clase *Filtro*. La clase *Filtro* es una clase abstracta que a su vez implementa la interfaz *Filter*. Esta clase *Filtro* se ha diseñado para encapsular atributos y métodos comunes a todas las herramientas. En esta clase se almacenan atributos, entre ellos booleanos (*ind y outd*), que indican si el filtro tiene que leer o escribir de un *buffer*, de un *DataSource* o en un *DataSink*. Esta clase *Filtro* permite abstraer la acción de leer de un *buffer* o de un *DataSource* y de escribir en un *buffer* o en un *DataSink*, se consigue de esta forma reutilizar la clase para poder realizar distintas acciones. También se definen métodos de lectura y escritura: *'readFromInput()'* y *'writeFromInput()'*. Estos métodos permiten que dependiendo de los atributos mencionados antes se pueda leer de un *buffer* o del *DataSource* y se pueda escribir en un *buffer* o en un *DataSink*.

YoutubeDL (Descarga de vídeo): Para la herramienta *YoutubeDL* se ha implementado la clase *YoutubeDLTool*. Esta clase ejecuta el *script* de la herramienta (*youtube-dl.py*) para realizar la descarga de vídeos. Para la ejecución de este *script* se utiliza la clase *Process* de JAVA.

```
1 Process p = Runtime.getRuntime().exec(command);
```

El método *'exec()'* (línea 1) permite ejecutar un comando externo, en este caso el *script* de la herramienta *YoutubeDL* con los parámetros que necesite.

JAVE (Extraer el audio): Para la herramienta *JAVE* se ha implementado la clase *JaveTool*. Esta clase hace uso de las librerías de la herramienta para llevar a cabo la extracción del audio de los vídeos.

```
1  AudioAttributes audio = new AudioAttributes();
2  audio.setCodec("pcm_s16le");                //signed 16 bit little endian format
3  audio.setSamplingRate(new Integer(16000)); //sampling rate
4  audio.setChannels(1);                       //number of channels
5
6  EncodingAttributes attrs = new EncodingAttributes();
7  attrs.setFormat("wav");                     //audio format
8  attrs.setAudioAttributes(audio);
9
10 Encoder encoder = new Encoder();
11 encoder.encode(source, target, attrs);     //transcoding file
```

Para llevar a cabo la extracción del audio, en primer lugar se definen los atributos del audio final que se quiere obtener (líneas 1-4). A continuación se especifica el formato de salida del audio (líneas 6-8) y finalmente se realiza la transcodificación (conversión de un codec a otro) del vídeo a audio (líneas 10-11).

IBM - SpeechToText (Reconocimiento de voz): Para la herramienta *SpeechToText* de IBM se han implementado dos clases: *IBMWS* e *IBMThread*. La clase *IBMWS* analiza el fichero de audio obtenido de la etapa anterior (*JAVE*). Esta clase realiza un troceado del audio en fragmentos de hasta tres minutos. Se realiza este troceado debido a que el servicio de reconocimiento de voz de IBM solo permite procesar ficheros de audio con una duración no superior a tres minutos. Cada uno de los trozos se pasa a la clase *IBMThread*.

```
1 File file = splitAudio();
2 IBMThread ibmt = new IBMThread(file,numPart,transText);
3 Thread T = new Thread(ibmt);
4 T.start();
```

Con la función *'splitAudio()'* (línea 1) se trocea el fichero original y cada trozo se pasa a un objeto *IBMThread* junto con un identificador del número de trozo *numpart* y una referencia a un vector de *strings transText* (línea 2).

La clase *IBMThread* se trata de una clase que se ejecuta en un hilo de ejecución aparte (líneas 3-4), ya que implementa la interfaz *Runnable*.

```
1 String recognizeURL = url_api;
2 URL obj = new URL(recognizeURL);
3 HttpsURLConnection con = (HttpsURLConnection) obj.openConnection();
4 con.setRequestMethod("POST");
5 responseCode = con.getResponseCode();
```

La clase *IBMThread* se encarga de realizar una petición al servicio de reconocimiento de voz de IBM para extraer el texto de un fichero de audio. Esta clase al ejecutarse en un hilo de ejecución en paralelo al principal, permite realizar multiples peticiones contra el servidor de IBM a la vez, una para cada trozo o fragmento resultante. El resultado obtenido del reconocimiento de voz se almacena en la variable *transText*.

JATE (Extracción de términos clave): Para la herramienta *JATE* se ha implementado la clase *JateTool*. Esta clase hace uso de las librerías de la herramienta para llevar a cabo la extracción de términos.

```
1   Term[] tf = executeAlgorithm(new FrequencyAlgorithm(),
2                                new FrequencyFeatureWrapper(termCorpusFreq));
3 \\ ... Para cada uno de los algortimos ...
4
5   String[] mresult = majority_voting(tf,avg,ridf,gloss,weird,cvalue);
```

Con el método *'executeAlgorithm()'* (línea 1) se ejecuta cada uno de los algoritmos que se quieren utilizar y se almacenan los resultados de la extracción de terminos en un *array* de términos (*Term*). Al finalizar la ejecución de cada uno de los algoritmos, se realiza una votación por mayoría utilizando la función *'majority_voting()'* que haciendo uso de los resultados obtenidos por todos los algoritmos de extracción de términos, devuelve en un *array* de *strings* los 20 términos más significativos del documento o *corpus* de documentos utilizado.

Pybossa (Validación de términos clave): Para la herramienta *Pybossa* se ha implementado la clase *PybossaTool*. Esta clase se encarga de crear nuevas tareas en la aplicación web de *Pybossa* montada previamente de forma local, realizando peticiones POST con la información de la tarea.

```
1 String url = "http://localhost:5000/api/task?api_key=" + api_key;
2 URL obj = new URL(url);
3 HttpURLConnection con = (HttpURLConnection) obj.openConnection();
4 con.setRequestMethod("POST");
5 DataOutputStream wr = new DataOutputStream(con.getOutputStream());
6 String data = buildJSONData(terms,videoURL);
7 wr.writeBytes(data);
8 int responseCode = con.getResponseCode();
```

Para la creación de la nueva tarea se realiza una petición *POST* a la dirección almacenada en la variable *url* (línea 1). A esta petición *POST* se le añaden la información de los términos del vídeo y su enlace (líneas 5-7) que conforman los datos de la tarea a crear al finalizar correctamente la petición.

Pisixde (Obtención de metadatos): Para la herramienta *Pisixde* se ha implementado la clase *PisixdeTool*. Esta clase ejecuta un *script* (*pisixde.sh*) externo al entorno de ejecución del sistema. Se hace uso de la clase *Process* de JAVA para su ejecución igual que con la herramienta *YoutubeDL*.

```
1 Process proc = Runtime.getRuntime().exec(command);
```

Igual que en la herramienta *YoutubeDL* el método *'exec()'* (línea 1) permite ejecutar un comando externo, en este caso el *script* de la herramienta *pisixde* con los parámetros que necesite.

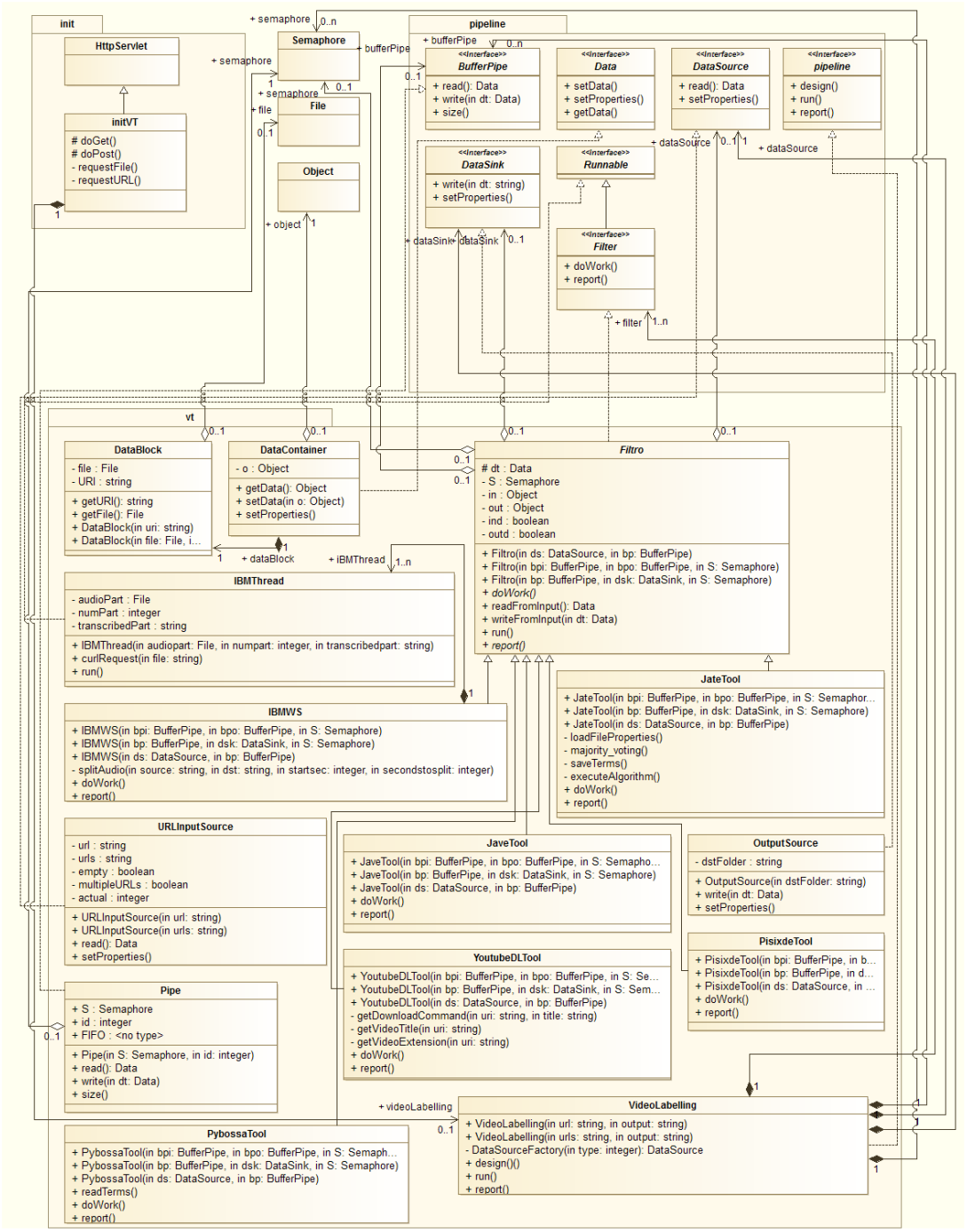

Figura 10: Diagrama de clases del sistema de etiquetado.

5 Evaluación

En esta sección se van a exponer las pruebas intermedias realizadas para la elección de la herramienta a utilizar en el reconocimiento de voz del sistema y por otro lado se van a describir los resultados obtenidos de evaluar el rendimiento general del sistema de etiquetado.

5.1 Evaluación intermedia

A continuación se va a realizar una evalución de las herramientas de reconocimiento de voz que pasaron un primer análisis en la sección 2.4.3 (*IBM* y *CMU Sphinx*). Para determinar qué reconocedor de voz se iba a utilizar, se han realizado pruebas en relación a su rendimiento y calidad, realizando una prueba de reconocimiento de voz por separado, para ver que herramienta tenía mejores resultados. La prueba de rendimiento realizada consiste en comprobar los costes de computo de ambas herramientas, mientras que la prueba de calidad consiste en comprobar qué porcentaje de aciertos tiene cada una de las herramientas comparando el texto obtenido del reconocimiento de voz con los subtitulos originales del vídeo.

5.1.1 Calidad

En primer lugar se van a describir los resultados obtenidos en la prueba de calidad. Para ello, se seleccionaron 20 vídeos (charlas) de la comunidad *TED* [21], cuya finalidad es divulgar conferencias sobre tecnología, entretenimiento y diseño. De cada vídeo se seleccionaron los 3 primeros minutos (dado que para realizar la comparativa era necesario que todos tuvieran la misma duración). Los vídeos fueron introducidos al sistema utilizando por un lado el reconocedor de *IBM* y por otro lado el reconocedor de *CMU Sphinx*. A continuación se compararon los resultados obtenidos en el reconocimiento en cada uno, con los subtitulos de los vídeos originales, permitiendo así determinar cuántos términos o palabras reconocidas eran correctas y cuántas no. En la tabla 5 se puede observar la tasa de aciertos para cada herramienta y para cada vídeo.

En la tabla 5 se puede ver el listado de cada uno de los veinte vídeos seleccionados (cuya duración se ha acortado a tres minutos) junto a la tasa de aciertos (valores porcentuales) obtenida por ambas herramientas en el reconocimiento de voz. Se ha comparado cada una de las palabras obtenidas por cada herramienta con las palabras existentes en los subtítulos originales de los vídeos (descargados de la web de procedencia de los vídeos: *TED*).
También se puede observar como el conjunto de vídeos de la colección utilizada obtiene unos valores en las tasas de aciertos cercanos a la media, a excepción de tres vídeos (*senses* (buenos resultados), *microbes* (malos resultados) y *hidden_objects* (buenos resultados)) que tienen unos valores muy por encima o muy por debajo de la media. Estos resultados atípicos, en estos tres vídeos, estan causados por distintos aspectos: características del ponente de la charla (acento, origen, edad, etc) y características del lugar de la conferencia (existencia o no de ruido de fondo) que puedan afectar a reconocer la voz del ponente. Estos aspectos intervienen en los resultados obtenidos por los reconocedores de voz. Las características del ponente de la charla, como el acento o la nacionaliadd de origen, afectan al resultado del reconocimiento de voz negativa o positivamente dependiendo de los datos de entrenamiento utilizados por el reconocedor. Si el acento del ponente encaja con el acento utilizado en el entrenamiento del reconocedor, se conseguirán resultados positivos, de lo contrario el reconocedor perderá precisión. En las características del entorno de la conferencia afecta negativamente la presencia de ruido de fondo, complicando la tarea de reconocimiento de voz, por el contrario, sin ruido de fondo el reconocimiento de voz será más satisfactorio.

Vídeo	Nombre	URL
1	mars	https://www.ted.com/talks/nathalie_cabrol_how_mars_might_hold_the_secret_to_the_origin_of_life
2	bill gates ebola	https://www.ted.com/talks/bill_gates_the_next_disaster_we_re_not_ready
3	3d printing	https://www.ted.com/talks/joe_desimone_what_if_3d_printing_was_25x_faster
4	comets	https://www.ted.com/talks/fred_jansen_how_to_land_on_a_comet
5	virtual reality	https://www.ted.com/talks/chris_milk_how_virtual_reality_can_create_the_ultimate_empathy_machine
6	capitalism	https://www.ted.com/talks/paul_tudor_jones_ii_why_we_need_to_rethink_capitalism
7	go to space	https://www.ted.com/talks/angelo_vermeulen_how_to_go_to_space_without_having_to_go_to_space
8	senses	https://www.ted.com/talks/david_eagleman_can_we_create_new_senses_for_humans
9	brain control	https://www.ted.com/talks/greg_gage_how_to_control_someone_else_s_arm_with_your_brain
10	quasars	https://www.ted.com/talks/jedidah_isler_how_i_fell_in_love_with_quasars_blazars_and_our_incredible_universe
11	smart computers	https://www.ted.com/talks/nick_bostrom_what_happens_when_our_computers_get_smarter_than_we_are
12	engineering food	https://www.ted.com/talks/pamela_ronald_the_case_for_engineering_our_food
13	brain communication	https://www.ted.com/talks/miguel_nicolelis_brain_to_brain_communication_has_arrived_how_we_did_it
14	butterflies	https://www.ted.com/talks/jaap_de_roode_how_butterflies_self_medicate
15	computer vision	https://www.ted.com/talks/fei_fei_li_how_we_re_teaching_computers_to_understand_pictures
16	bacterias	https://www.ted.com/talks/tal_danino_we_can_use_bacteria_to_detect_cancer_and_maybe_treat_it
17	micro robotics	https://www.ted.com/talks/sarah_bergbreiter_why_i_make_robots_the_size_of_a_grain_of_rice
18	microbes	https://www.ted.com/talks/rob_knight_how_our_microbes_make_us_who_we_are
19	bees	https://www.ted.com/talks/anand_varma_a_thrilling_look_at_the_first_21_days_of_a_bee_s_life
20	hidden objects	https://www.ted.com/talks/abe_davis_new_video_technology_that_reveals_an_object_s_hidden_properties

Tabla 4: Referencias de cada vídeo de la colección utilizada.

En la tabla 4 se muestran las URLs donde se pueden encontrar cada uno de los vídeos de la colección utilizada para realizar la evaluación.

Nombre del vídeo	IBM (%)	CMU Sphinx (%)	Nº palabras	Ruido	Origen
mars	82.72	48.61	463	Nada	Francia
bill gates ebola	88.63	49.02	431	Nada	EE.UU.
3d printing	79.67	53.25	418	Nada	EE.UU.
comets	80.39	44.40	515	Nada	Holanda
virtual reality	85.23	43.99	352	Poco	EE.UU.
capitalism	83.82	55.26	445	Nada	EE.UU.
go to space	79.92	48.42	503	Nada	Bélgica
senses	90.52	65.78	464	Nada	EE.UU.
brain control	77.09	42.59	585	Poco	EE.UU.
quasars	68.99	45.24	416	Poco	EE.UU.
smart computers	79.30	46.59	430	Nada	Suecia
engineering food	89.87	47.76	385	Nada	EE.UU.
brain communication	68.81	50.23	420	Nada	Brasil
butterflies	80.84	50.18	574	Nada	EE.UU
computer vision	82.35	54.59	357	Nada	China
bacterias	79.45	44.83	506	Nada	EE.UU.
micro robotics	86.36	55.68	572	Nada	EE.UU.
microbes	71.59	34.69	521	Poco	EE.UU.
bees	85.48	52.60	372	Nada	EE.UU.
hidden objects	94.29	69.79	473	Nada	EE.UU.
Media	81.77	50.18	460.1	-	-

Tabla 5: Tasa de aciertos (%) entre los reconocedores de voz *CMU Sphinx* e *IBM*.

En la tabla 5 también podemos observar: el número de palabras reales que se pronuncian en cada vídeo (obtenido a partir de los subtítulos originales del vídeo), la existencia o no de ruido (Nada, Poco o Mucho) y el país de origen del ponente de la conferencia.
A partir de estos datos, se pone en el punto de mira los dos vídeos de la colección que han conseguido unas tasas de aciertos por debajo de la media con la herramienta de *IBM*: *quasars* y *brain communication*. La razón por la que el primer vídeo tienes estos malos resultados es debido a que en el vídeo la voz de la ponente se escucha algo distorsionada con una ligera existencia de ruido de fondo. Por otro lado, el segundo vídeo tiene una tasa de aciertos por debajo de la media debido a que el ponente de la conferencia es de origen brasileño y en su charla utiliza terminología portuguesa y tiene un acento brasileño, lo que dificulta el reconocimiento de todas las palabras de su charla.

5.1.2 Rendimiento

A continuación se van a describir los resultados de las pruebas en relación al rendimiento mostrado (costes de computo). Para ello, utilizando los 20 vídeos seleccionados en la subsección anterior, se van a ejecutar las dos herramientas de reconocimiento de voz (*IBM* y *CMU Sphinx*) para analizar el tiempo de procesado empleado por cada una de ellas. Las especificaciones del procesador de la máquina utilizada son: *Intel® Core™ i7-4700MQ CPU @ 2.40GHz x 8* y *8GB de RAM*. Los resultados obtenidos se pueden observar en la figura 11, que muestra el tiempo en realizar el proceso de reconocimiento de voz para cada vídeo y por cada herramienta.

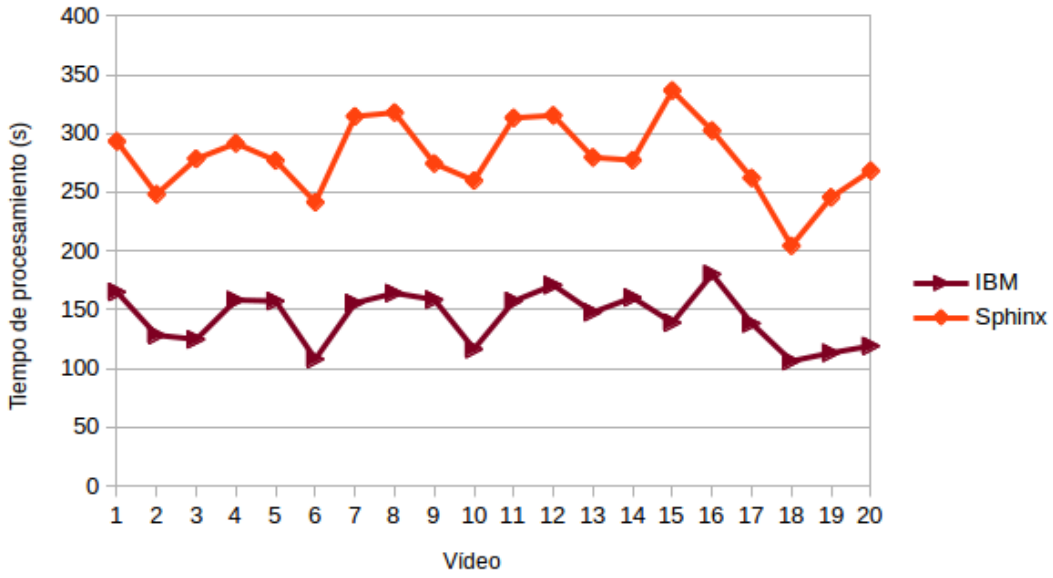

Figura 11: Comparativa del tiempo de procesado entre las herramientas de reconocimiento de *IBM* y *Sphinx*.

	IBM (s)	CMU Sphinx (s)
T. de procesado medio	143.37	280.01

Tabla 6: Tiempo de procesado medio (en segundos) entre los reconocedores de voz *IBM* y *CMU Sphinx*.

En primer lugar, en la figura 11 se puede apreciar que a lo largo de los veinte resultados (de los veinte vídeos) para cada herramienta se muestra una tendencia similar entre ambas herramientas. Como se puede observar en la tabla 6, la herramienta *CMU Sphinx* tarda de media 1.95 veces más tiempo en procesar el reconocimiento de voz de un audio, que la herramienta análoga de *IBM*. Este hecho se puede apreciar también en la figura 11, en la que la línea que representa la herramienta *CMU Sphinx* tiene valores que, en general, doblan a los que tiene la línea que representa a la herramienta de *IBM*.
Por otro lado, teniendo en cuenta que se ha explicado antes que en la colección de veinte vídeos,

tienen todos una duración de tres minutos, los tiempos de procesado varían entre cada vídeo. Este hecho se debe a que, al margen de la duración del vídeo, cada vídeo es la grabación de una charla distinta, con una estructura y características distintas, esto quiere decir que cada vídeo tiene, como es lógico, una afluencia de diálogos distinta, lo que provoca que aumente o disminuya el tiempo de procesado según la cantidad de diálogos presentes en el vídeo.
Para demostrar que los fragmentos con poco o ningún diálogo (silencios) requerían menos tiempo de procesado, se hizo una prueba con un fragmento de audio de 50 segundos de duración el cual se procesó con las dos herramientas de reconocimiento de voz (*IBM* y *CMU Sphinx*). Los resultados fueron un tiempo de procesado entorno a los dos segundos en ambas herramientas, tiempo empleado por las herramientas para inicializar el proceso de reconocimiento de voz. Por lo tanto, se llegó a la conclusión de que los fragmentos con silencios o sin diálogos no son procesado por ninguna de las dos herramientas. Esta afirmación se puede observar nuevamente en la figura 11, comprobando que ambas herramientas tienen una tendencia de tiempo de procesado análoga.

Finalmente se ha seleccionado la herramienta de reconocimiento de voz de *IBM* debido a que, como ya se ha visto en los resultados expuestos en esta sección, tanto su tasa de aciertos como el tiempo de procesamiento que ofrece es claramente mejor al de la otra alternativa (*CMU Sphinx*).

5.2 Estimación de los costes de computo

Para la evaluación del sistema de etiquetado se ha puesto en marcha el sistema con las herramientas finales elegidas para cada etapa, que se pueden observar en la figura 2, y se ha introducido en el sistema la colección de vídeos utilizada en la sección 5.1 (de tres minutos de duración cada vídeo). En este caso los vídeos han atravesado todo el sistema de etiquetado y se han almacenado los tiempos de procesamiento de cada etapa y los tiempos de lectura y escritura en los *buffers*.
Una vez procesada toda la colección de vídeos por el sistema de etiquetado, los tiempos de procesado de cada etapa se pueden observar en la tabla 7. Viendo los resultados de la tabla se puede observar que en la primera etapa (*YoutubeDL*) algunos valores sufren desviaciones importantes sobre la media, esto es debido a que esta herramienta, al realizar la operación de descarga de vídeos, depende de la sobrecarga de la red y la conexión a Internet.

Vídeo	YoutubeDL (s)	Jave (s)	IBM (s)	Jate (s)	Pybossa(s)
1	8.07	1.074	165.031	1.136	0.015
2	12.12	0.908	128.201	0.768	0.024
3	14.12	0.852	124.631	1.009	0.017
4	15.35	0.898	158.149	0.635	0.013
5	23.07	0.811	157.257	0.648	0.012
6	11.52	0.922	107.89	0.602	0.011
7	9.26	0.888	155.542	0.632	0.014
8	16.21	0.756	164.042	0.587	0.017
9	12.46	0.784	158.485	0.629	0.021
10	13.18	0.872	116.353	0.590	0.017
11	9.14	0.83	157.182	0.611	0.013
12	12.55	0.842	170.951	0.698	0.015
13	10.59	0.826	147.697	0.592	0.016
14	11.03	0.684	160.359	0.809	0.012
15	12.32	0.904	138.885	0.738	0.017
16	9.22	0.843	180.193	0.626	0.013
17	10.74	0.684	138.222	0.563	0.013
18	13.67	0.708	106.149	0.583	0.013
19	9.45	0.808	113.155	0.838	0.015
20	10.89	0.743	119.013	0.584	0.013
Media	12.25	0.831	143.369	0.694	0.015

Tabla 7: Tiempo de procesamiento para cada etapa del sistema de etiquetado para una colección de veinte vídeos.

En la tabla 7 se puede apreciar los distintos valores de las medias del tiempo de procesamiento de cada etapa. Viendo estos valores se puede determinar qué etapas son más o menos lentas, y cuáles suponen un cuello de botella para el sistema de etiquetado, ya que esta etapa más lenta penalizará en tiempo a todo el sistema por las dependencias de unas etapas con otras. La etapa más lenta es, como se puede observar en la tabla, la etapa de reconocimiento de voz de *IBM*.

La evolución de estos costes de computo para vídeos de una duración superior para cada etapa se comportaría de la siguiente forma. La etapa de *YoutubeDL* tendría un comportamiento lineal con respecto a la duración del vídeo dependiendo siempre del estado de la conexión a Internet. Las etapas de *Jave*, *IBM* y *Jate* también tendrían un comportamiento lineal con respecto a la duración del vídeo que entra al sistema de etiquetado, ya que cuanto mayor es la duración de los vídeos mayor cantidad de datos tiene que decodificar la herramienta *Jave*, mayor audio tiene que reconocer la herramienta de *IBM*, y mayor cantidad de texto tiene que procesar la herramienta *Jate*. Por último la herramienta *Pybossa* siempre debería tener un coste de computo independiente de la duración del vídeo que entra al sistema de etiquetado, ya que su tarea es crear una nueva tarea para la validación de 20 términos clave.
Como se puede observar en la tabla no aparecen los resultados de la última etapa del sistema *Pisixde*. Esto se debe a que al generar el grafo de conceptos relacionando los 20 términos clave obtenidos en etapas anteriores no llega a finalizar, se queda bloquedado. Sin embargo, al generar grafos de conceptos relacionando un menor número de términos si que llega a finalizar. Ajenos al sistema de etiquetado, se ejecuto el *script* de *pisixde* por separado en una terminal sel sistema operativo (distribución *Linux*) con 20 términos clave obtenidos del sistema de etiquetado para comprobar si funcionaba estando fuera del sistema de etiquetado. El *script* finalizaba dando unos costes de computo de 12 minutos para un vídeo y 38 minutos para otro distinto y generando correctamente los grafos de conceptos. Un ejemplo de grafo generado se puede visualizar en la figura 17 del anexo B.

Para terminar con la evaluación del sistema de etiquetado, los tiempos de lectura y escritura de los *buffers* del sistema han dado resultados muy cercanos a cero (0.001 segundos). Estos tiempos de lectura y escritura se han obtenido midiendo el tiempo empleado en leer y escribir de la estructura FIFO interna del *buffer*. Dado que la lectura y escritura se realiza sobre objetos tipo *File* de *JAVA*, estas lecturas y escrituras se realizan de forma tan rápida ya que solo se lee y escribe la dirección de memoria donde se ha creado el objeto fichero (*File*) y por otro lado los ficheros que se han utilizado en la prueba no han sido de gran tamaño (no superior a 5MB).

6 Gestión del proyecto

En este apartado se va a describir la metodología organizativa para la realización del trabajo así como los esfuerzos empleados para llevarlo a cabo.

6.1 Organización

La metodología utilizada para realizar el proyecto ha sido un desarrollo en cascada. Esta metodología divide el desarrollo en distintas etapas ordenadas, de tal forma que cada etapa debe esperar a la finalización de la etapa anterior. Las principales etapas han sido: planificación, análisis, diseño, implementación, pruebas y memoria.
Al inicio del proyecto se hizo un planteamiento del problema y un posterior análisis de herramientas o recursos a integrar en el proyecto. A continuación se llevo a cabo una etapa de diseño de la arquitectura del sistema y la implementación de la misma. Finalmente se realizaron pruebas de integración (pruebas al conjunto de componentes del sistema) A y evaluación del sistema y por último se llevó a cabo la redacción de este documento.
Periódicamente se han realizado reuniones de control y gestión del proyecto para revisar los progresos y analizar resultados.
El desarrollo real del proyecto se puede observar en la figura 13 donde se puede apreciar el desarrollo en cascada en un diagrama de Gantt. La planificación inicial del proyecto se puede observar en la figura 12. Los motivos por los cuales derivó esta planificación en el desarrollo que se puede observar en la figura 13 fueron: dificultades en la implementación y falta de experiencia a la hora de preveer esfuerzos en la realización de un proyecto de estas características.

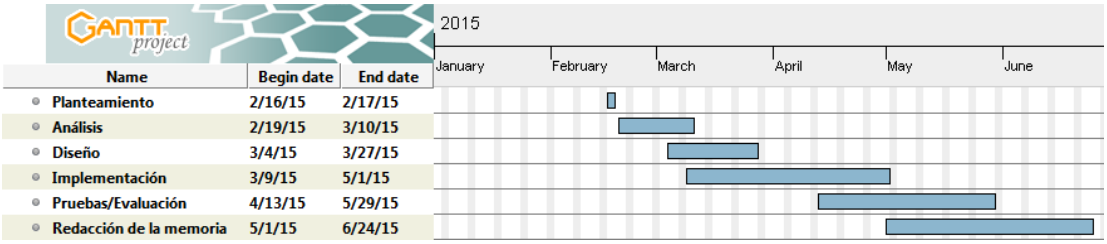

Figura 12: Diagrama de Gantt de la planificación inicial del proyecto.

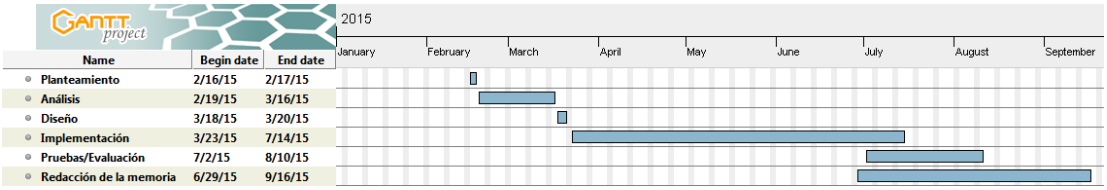

Figura 13: Diagrama de Gantt del desarrollo real del proyecto.

6.2 Esfuerzos

Para la finalización del trabajo se han llevado a cabo distintas fases y cada una de ellas han supuesto unos esfuerzos u horas de dedicación. En la tabla 8 se detallan los esfuerzos para cada una de esas etapas.

Etapa	Horas
Planteamiento	10
Análisis	44.5
Diseño	12
Implementación	143.5
Pruebas	42.5
Evaluación	28
Memoria	85
Total	365.5

Tabla 8: Relación de las fases del proyecto y los esfuerzos dedicados.

7 Conclusiones y trabajo futuro

En este último apartado se va a finalizar este documento con las conclusiones finales obtenidas del proyecto y el trabajo futuro que se puede realizar a partir de este.

7.1 Conclusiones

Finalmente se ha alcanzado el objetivo principal: desarrollar un sistema de etiquetado de vídeos de forma automática, utilizando para ello el contenido presente en el audio de los vídeos.
Se ha realizado un análisis previo del sistema para determinar posibles herramientas a utilizar. Tras el análisis realizado se han seleccionado las siguientes herramientas: *YoutubeDL* (Descarga de vídeo), *Jave* (Extracción de audio), *IBM - SpeechToText* (Reconocimiento de voz), *Jate* (Extracción de términos clave), *Pybossa* (Validación de términos clave) y *Pisixde* (Obtención de nuevo metadatos).
Se ha conseguido desarrollar una aplicación web que permita a usuarios, sin necesidad de conocimientos del sistema, utilizar el sistema de etiquetado, pudiendo etiquetar grandes colecciones de vídeos.
Se ha desplegado y desarrollado otra aplicación web con el objetivo de que los usuarios puedan validar los resultados obtenidos por el sistema de etiquetado y así detectar posibles vídeos con etiquetas que no corresponden a su contenido.
Se ha llevado a cabo una evaluación del sistema a partir de pruebas de integración de todas los componentes que intervienen en el sistema de etiquetado.
Los problemas detectados durante el transcurso del desarrollo del proyecto han sido relacionados con la herramienta *Pisixde*. Al principio el servidor remoto no contestaba a las peticiones ya que se debia a un cambio en el nombre de la dirección utilizada. Por otro lado, al integrar la herramienta al sistema, no llegaba a finalizar con éxito al pasarle un conjunto de 20 términos clave para generar los grafos de conceptos.

Por otro lado, con este proyecto se han asentado conocimientos relacionados con la ingeniería del *software* y las tecnologías web. También se ha logrado adquirir experiencia al embarcarse en un proyecto de esta magnitud.

7.2 Trabajo futuro

Algunas de las opciones para dar continuidad al trabajo desarrollado son las que se exponen a continuación:

- Almacenar toda la información referente y disponible de los vídeos procesados (términos clave, título, descripción, fecha, etc) en ficheros XML (eXtensible Markup Language) para una mejor representación y posterior procesamiento. Al procesar grandes colecciones de vídeos se obtendría una colección de ficheros XML con la que poder gestionar y procesar toda esa información. La idea principal que se propone es conseguir tener toda esta información relacionada (o enlazada) para poder llevar a cabo un buscador semántico de la colección de vídeos procesados por el sistema de etiquetado.

- Completar las tareas de validación generadas por el sistema de etiquetado a partir de todos los vídeos procesados por dicho sistema.

- Utilizar los resultados generados por el sistema de etiquetado por herramientas de aprendizaje electrónico (*e-learning*). Utilizar el sistema de etiquetado para procesar vídeos de las plataformas de *e-learning* para ofrecer a sus usuarios mejor información referente al contenido de los recursos multimedia que disponga la plataforma.

Anexos

A Prueba de integración

En este anexo se va a describir el proceso llevado a cabo para realizar una prueba de integración del sistema de etiquetado.

Para comenzar la prueba el usuario deberá acceder a la interfaz web que permite interactuar con el sistema de etiquetado. Esta interfaz web permite al usuario enviar los vídeos que quiere etiquetar al sistema de etiquetado indicando su URL. Como se puede observar en la figura 14, la interfaz web tiene dos opciones para enviar las URLs de los vídeos. Por un lado, se permite enviar una única URL de un único vídeo (1) y por otro lado, se permite enviar al sistema de etiquetado un fichero de texto (2) que contenga un listado de las URLs de una colección de vídeos.

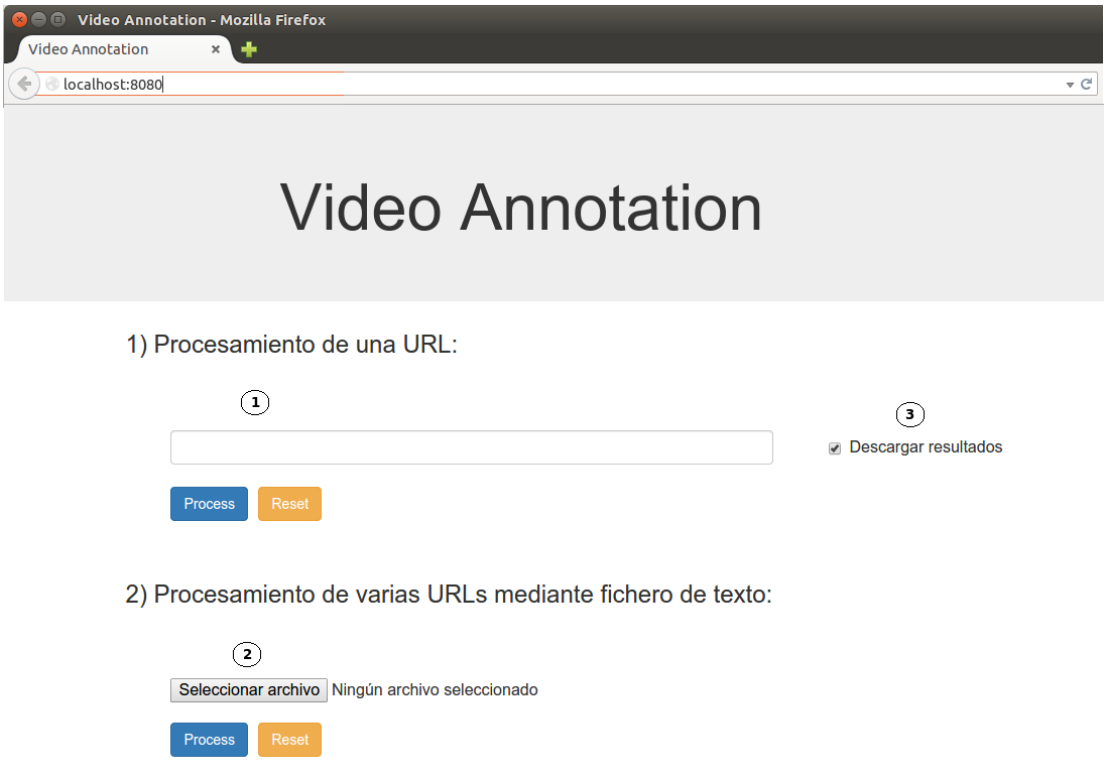

Figura 14: Opciones presentas en la interfaz web del sistema de etiquetado.

Para la primera opción se puede elegir si visualizar los resultados (las etiquetas generadas) en la propia interfaz al finalizar o descargar los resultados en un fichero comprimido (.ZIP) como se puede ver en la figura 15 (1).

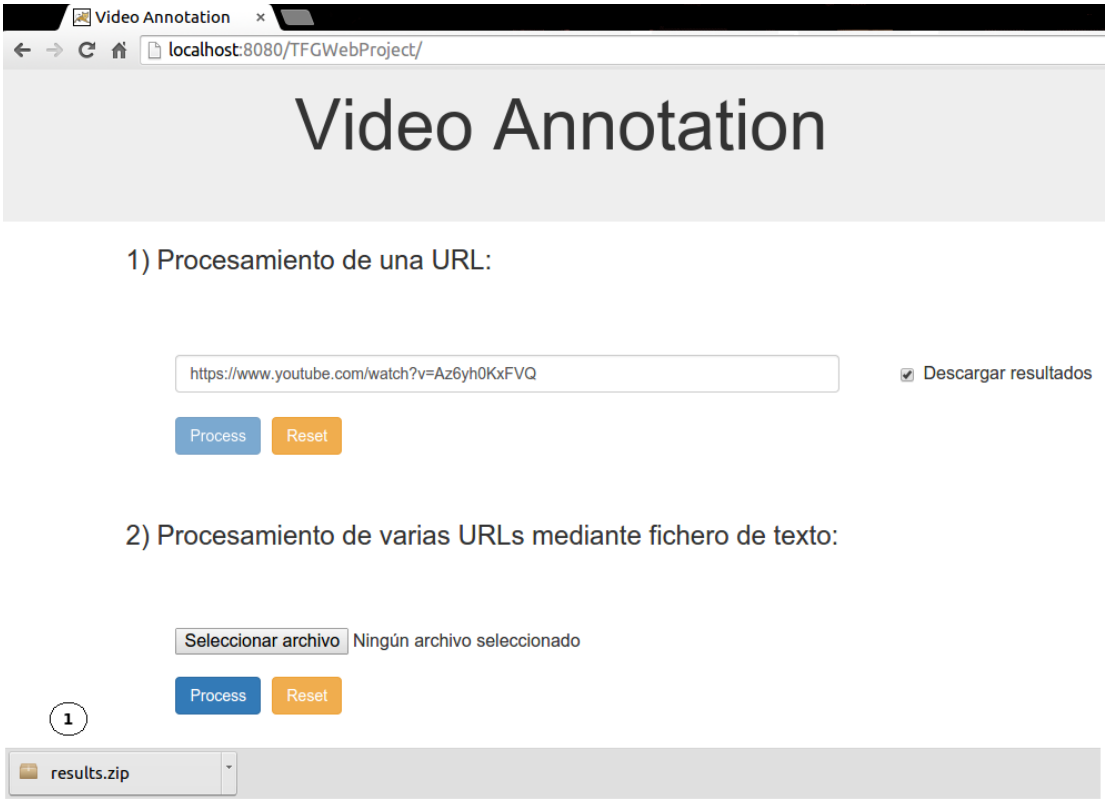

Figura 15: Descarga de los resultados a través de la interfaz web del sistema de etiquetado.

Internamente, una vez se han enviado los vídeos, el sistema de etiquetado empieza a procesar lo vídeos y los datos van pasando por las distintas etapas. En la figura 16 se puede observar un ejemplo de ejecución y de los eventos que ocurren dentro del sistema de etiquetado. Una vez finaliza la ejecución de toda la petición de vídeos se contesta al usuario enviándoles los resultados.

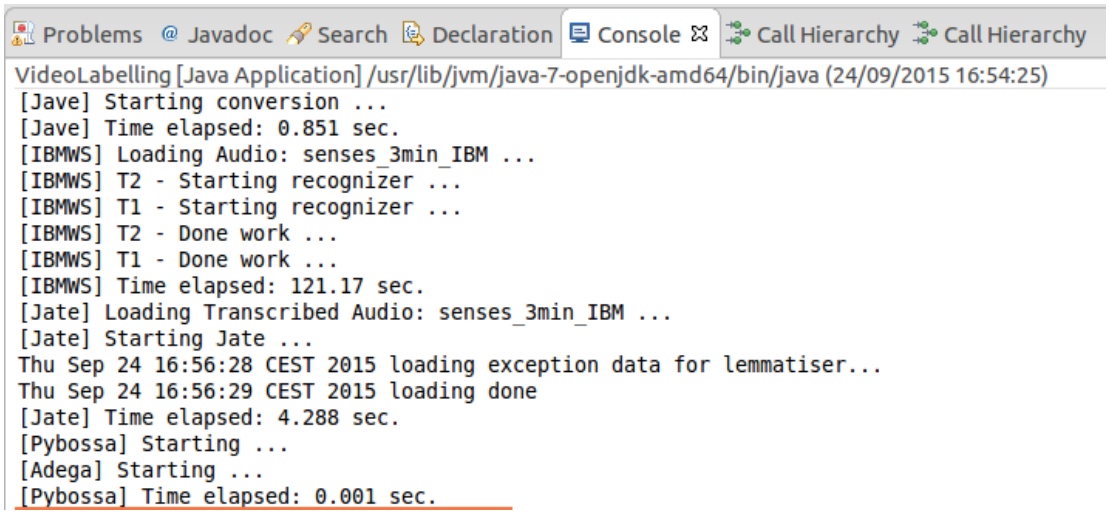

Figura 16: Sucesión de eventos en la ejecución interna del sistema de etiquetado.

B Grafo generado por la herramienta *Pisixde*

En este anexo se van a mostrar dos ejemplos de grafos generados por la herramienta *Pisixde* a partir de los términos clave generados por el sistema de etiquetado para dos vídeos. Se van a mostrar los grafos de dos términos distintos, siendo cada término perteneciente a un vídeo distinto. Como los grafos están definidos como ficheros XML se va a hacer uso de una herramienta externa llamada *Gephi* [23] para la visualización de los mismos. Los grafos generados por esta herramienta presentan nodos y aristas. Los nodos representan cada uno de los conceptos recuperados, mientras que las aristas entre ellos representan las relaciones existentes.

En primer lugar se va a mostrar el grafo que pertenece al término *Byte* del vídeo *mars* de la colección de vídeos utilizada en este documento. El grafo se puede ver en la figura 17 junto a los datos de cada nodo que se pueden observar en la figura 18 que muestra una tabla de los datos del grafo.

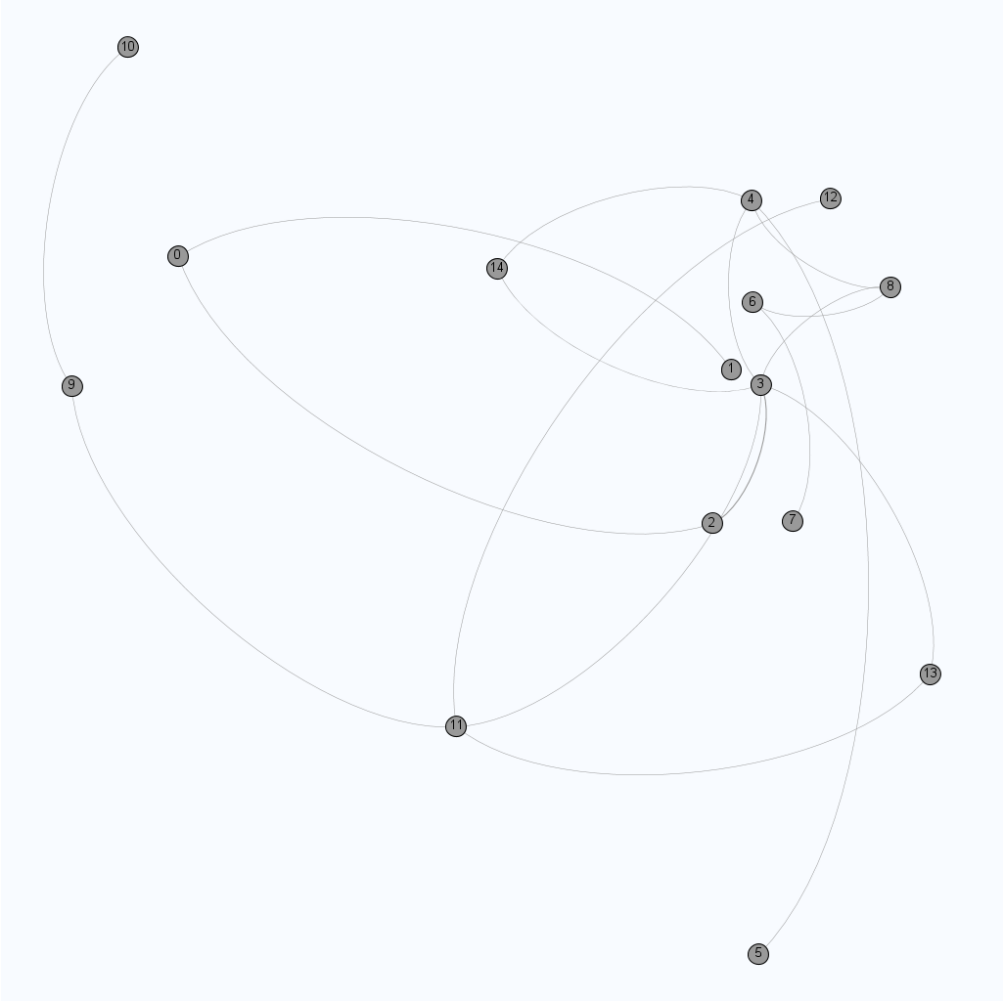

Figura 17: Grafo de conceptos del término clave *Byte* del vídeo *mars*.

Nodes	Id	Label	name	shortname
0	0	0	http://dbpedia.org/resource/Category:Numeral_systems	http://dbpedia.org/resource/Category:Numeral_systems
1	1	1	Numeral systems@en	Numeral systems@en
2	2	2	http://dbpedia.org/resource/Binary_prefix	http://dbpedia.org/resource/Binary_prefix
3	3	3	http://dbpedia.org/resource/Byte	http://dbpedia.org/resource/Byte
4	4	4	http://dbpedia.org/resource/Category:Data_unit	http://dbpedia.org/resource/Category:Data_unit
5	5	5	Data unit@en	Data unit@en
6	6	6	http://dbpedia.org/resource/Category:Computer_data	http://dbpedia.org/resource/Category:Computer_data
7	7	7	Computer data@en	Computer data@en
8	8	8	http://dbpedia.org/resource/Category:Computer_memory	http://dbpedia.org/resource/Category:Computer_memory
9	9	9	http://dbpedia.org/resource/Category:Type_systems	http://dbpedia.org/resource/Category:Type_systems
10	10	10	Type systems@en	Type systems@en
11	11	11	http://dbpedia.org/resource/Category:Data_types	http://dbpedia.org/resource/Category:Data_types
12	12	12	Data types@en	Data types@en
13	13	13	http://dbpedia.org/resource/Category:Primitive_types	http://dbpedia.org/resource/Category:Primitive_types
14	14	14	http://dbpedia.org/resource/Category:Units_of_information	http://dbpedia.org/resource/Category:Units_of_information

Figura 18: Tabla con los datos del grafo del término *Byte* del vídeo *mars*.

Como se puede ver en el grafo y en la tabla de datos, el término clave *Byte* se encuentra en el nodo 3 y esta relacionado con términos o conceptos como *Computer memory* (8), *Data* (4) y *Binary prefix* (2).

El segundo ejemplo que se presenta es el término *Animal* del vídeo *senses*. En la figura 19 se puede ver el grafo que conforman todos los conceptos relacionados de alguna manera con este término. Mientras que en la figura 20 se pueden apreciar los datos de cada nodo del grafo.

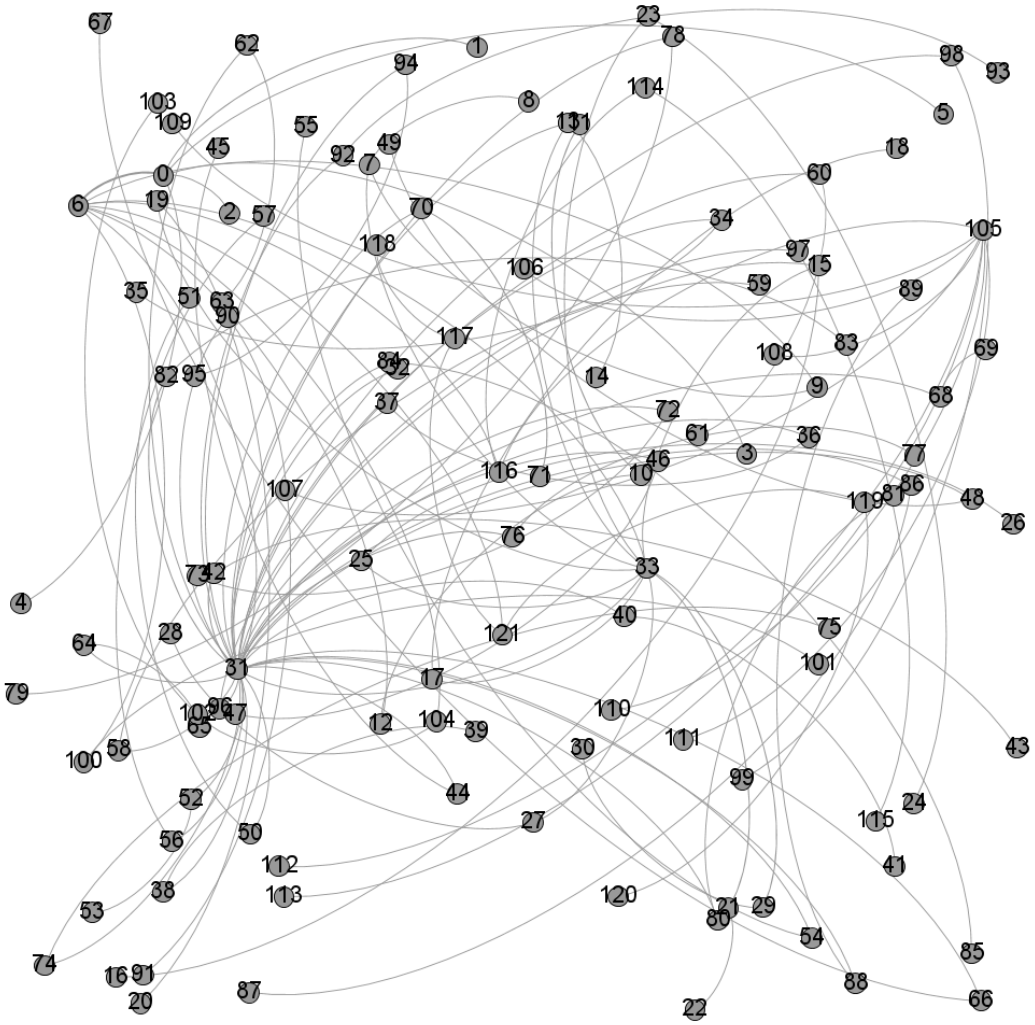

Figura 19: Grafo de conceptos del término clave *Animal* del vídeo *senses*.

En este caso podemos ver como del nodo 32 (el concepto *Animal*) salen un gran número de aristas que representan relaciones con otros nodos o conceptos, como por ejemplo el nodo 33 que representa el concepto *Zoology*.

No...	Id	La...	name
24	24	24	Animal genetics@en
25	25	25	http://dbpedia.org/resource/Category:Animal_metabolism
26	26	26	Animal metabolism@en
27	27	27	http://dbpedia.org/resource/Category:Animal_proteins
28	28	28	Animal proteins@en
29	29	29	http://dbpedia.org/resource/Category:Branches_of_biology
30	30	30	Branches of biology@en
31	31	31	http://dbpedia.org/resource/Category:Animals
32	32	32	Animals@en
33	33	33	http://dbpedia.org/resource/Category:Zoology
34	34	34	http://dbpedia.org/resource/Category:Animal_monuments
35	35	35	Animal monuments@en
36	36	36	http://dbpedia.org/resource/Category:Animals_with_only_t
37	37	37	Animals with only two limbs@en
38	38	38	http://dbpedia.org/resource/Category:Carnivorous_animals
39	39	39	Carnivorous animals@en
40	40	40	http://dbpedia.org/resource/Category:Shelters_built_or_us
41	41	41	Shelters built or used by animals@en
42	42	42	http://dbpedia.org/resource/Category:Wayward_animals
43	43	43	Wayward animals@en
44	44	44	http://dbpedia.org/resource/Category:Animals_by_location
45	45	45	Animals by location@en
46	46	46	http://dbpedia.org/resource/Category:Animal_products
47	47	47	Animal products@en
48	48	48	http://dbpedia.org/resource/Category:Animals_in_media
49	49	49	Animals in media@en
50	50	50	http://dbpedia.org/resource/Category:Animals_that_can_ch
51	51	51	Animals that can change color@en
52	52	52	http://dbpedia.org/resource/Category:Blind_animals
53	53	53	Blind animals@en

Figura 20: Tabla con los datos del grafo del término *Animal* del vídeo *senses*.

Bibliografía

[1] *Guía Breve de la Web Semántica - W3C (World Wide Web Consortium)* [en línea]. Disponible en: `http://www.w3c.es/Divulgacion/GuiasBreves/WebSemantica` [Consulta: Agosto 2015]

[2] Lunaweb Ltd. *ClipConverter* [en línea]. Disponible en: `http://www.clipconverter.cc/es/` [Consulta: Septiembre 2015]

[3] Diego Uscanga. *aTube Catcher* [en línea]. Disponible en: `http://www.atube.me/video/` [Consulta: Septiembre 2015]

[4] Alexey Kuznetsov. *VGET - GitHub* [en línea]. Disponible en: `https://github.com/axet/vget` [Consulta: Agosto 2015]

[5] Ricardo García. *Youtube-dl - GitHub* [en línea]. Disponible en: `https://github.com/rg3/youtube-dl` [Consulta: Agosto 2015]

[6] Carlo Pelliccia. *JAVE - Java Audio Video Encoder* [en línea]. Disponible en: `http://www.sauronsoftware.it/projects/jave/index.php` [Consulta: Agosto 2015]

[7] Crowd-Sourced Community. *FFmpeg.* [en línea]. Disponible en: `https://ffmpeg.org/` [Consulta: Agosto 2015]

[8] ConnectSolutions, LLC. *Xuggler* [en línea]. Disponible en: `http://www.xuggle.com/xuggler` [Consulta: Agosto 2015]

[9] Alphabet Inc. *Google Speech API* [en línea]. Disponible en: `https://www.google.com/speech-api/v2/recognize` [Consulta: Agosto 2015]

[10] *CMU Sphinx.* [en línea]. Disponible en: `http://cmusphinx.sourceforge.net/` [Consulta: Agosto 2015]

[11] *IBM - Wikipedia, la enciclopedia libre.* [en línea]. Disponible en: `https://es.wikipedia.org/wiki/IBM` [Consulta: Septiembre 2015]

[12] IBM Corp. *IBM Speech to text.* [en línea]. Disponible en: `http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/speech-to-text.html` [Consulta: Agosto 2015]

[13] IBM Corp. *IBM Watson Developer Cloud.* [en línea]. Disponible en: `http://www.ibm.com/smarterplanet/us/en/ibmwatson/developercloud/` [Consulta: Agosto 2015]

[14] *List of speech recognition software - Wikipedia, la enciclopedia libre* [en línea]. Disponible en: `https://en.wikipedia.org/wiki/List_of_speech_recognition_software` [Consulta: Agosto 2015]

[15] Ziqi Zhang. *JATE - Java Automatic Term Extraction* [en línea]. Disponible en: `http://code.google.com/p/jatetoolkit/` [Consulta: Agosto 2015]

[16] Daniel Lombraña. *Pybossa* [en línea]. Disponible en: `http://pybossa.com/` [Consulta: Agosto 2015]

[17] M. Lama, J. C. Vidal, E. Otero-García, A. Bugarín, S. Barro, *"Semantic linking of learning object repositories to dbpedia"*, Educational Technology & Society, vol. 15 (4) (2012), páginas 47–61.

[18] Crowd-Sourced Community. *DBpedia* [en línea]. Disponible en: `http://wiki.dbpedia.org/` [Consulta: Agosto 2015]

[19] Crowd-Sourced Community. *Wikipedia, la enciclopedia libre* [en línea]. Disponible en: `https://www.wikipedia.org/` [Consulta: Septiembre 2015]

[20] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad and Michael Stal. *Pattern-oriented Software Architecture: A System of Patterns* (VOLUME 1) (pp. 53-70). New York, NY, USA: John Wiley & Sons, Inc, 1996.

[21] *TED* [en línea]. Disponible en: `https://www.ted.com/talks` [Consulta: Septiembre 2015]

[22] *Chromim-dev* [en línea]. Disponible en: `https://groups.google.com/a/chromium.org/forum/#!forum/chromium-dev` [Consulta: Septiembre 2015]

[23] Crowd-Sourced Community. *Gephi - makes graphs handy* [en línea]. Disponible en: `http://gephi.github.io/` [Consulta: Septiembre 2015]