Enriquecer y explorar un conjunto de datos de publicaciones científicas con técnicas de procesamiento del lenguaje natural

Jaumot García, Adrià

Curs 2016-2017

Director: Dr. HORACIO SAGGION

GRAU EN ENGINYERIA INFORMÀTICA

upf. Universitat Pompeu Fabra Barcelona
Escola Superior Politècnica

Treball de Fi de Grau

Enriquecer y explorar un conjunto de datos de publicaciones científicas con técnicas de procesamiento del lenguaje natural

Adrià Jaumot García

TREBALL FI DE GRAU

GRAU EN ENGINYERIA INFORMÀTICA

ESCOLA SUPERIOR POLITÈCNICA UPF

2017

DIRECTOR DEL TREBALL

Dr. Horacio Saggion

Dedicatoria

Quiero dedicar este proyecto especialmente a mis padres por todo el esfuerzo que han tenido que hacer durante años para que pudiera terminar cumpliendo mis objetivos. A mis abuelos, por ayudarles a ellos y por su preocupación e interés hacia mí.

A Alejandro, Dani, Fran, Alex, Alberto, Pauete, Arnau, La Peeta, a los Chicos, Ferran, Jaume, Carles y a las Guapis por el apoyo que me han mostrado a lo largo de este tiempo.

Sobre todo a Vicky, por toda la paciencia que ha tenido conmigo, por acompañarme a lo largo de todo este viaje y por haberme ayudado a superar todas las situaciones adversas con las que me he encontrado.

En general, a todas las aquellas personas que en algún momento u otro a lo largo de este tiempo me han mostrado su apoyo.

Finalmente, al Barça, por darme una vía de escape de mi rutina semanalmente y hacerme vibrar con aquello que más me gusta. Y como no, a Messi, por ser **D10S**.

Agradecimientos

En primer lugar, a Horacio por haberme dado la oportunidad de realizar un proyecto como este y haberme tutorizado a lo largo de todo el curso. También a Pablo por su ayuda y sus consejos en la etapa final.

También a toda la gente que se dedican a este campo de la tecnología y que comparten su conocimiento para que otras personas puedan hacer uso de él.

Finalmente a la Prable y a la Peeta por haberme aconsejado y ayudado a lo largo de todo el desarrollo del proyecto.

Resumen

Actualmente, la literatura científica se encuentra en auge. El número de artículos y documentos publicados ha experimentado un crecimiento exponencial a lo largo de las últimas décadas y han aparecido diferentes plataformas digitales que ofrecen un acceso abierto a estas publicaciones.

El objetivo de este TFG es utilizar técnicas de procesamiento de lenguaje natural para analizar el contenido de estas publicaciones científicas y poder crear diferentes tipos de representaciones acerca de la información obtenida.

El trabajo está basado en el análisis de la base de datos bibliográfica DBLP[1], que será investigada y analizada lingüísticamente para dar soporte a diferentes aplicaciones. Se realizará un análisis lingüístico, usando librerías disponibles, sobre los títulos, *abstracts* y la colaboración entre autores con el objetivo de enriquecer el contenido y la visualización de la información.

Abstract

Currently, the scientific literature is in a stage of great growth. The number of published articles and documents has grown exponentially over the last decades and different digital platforms have appeared that offer open access to these publications.

The purpose of this TFG is to use natural language processing techniques to analyze the content of these scientific publications and to be able to create different types of representations about the information obtained.

The work will be based on the analysis of the DBLP bibliographic database, which will be investigated and analyzed linguistically to support different applications. A linguistic analysis will be carried out, using available libraries, on the titles, abstracts and collaboration between authors with the aim of enriching the content and the visualization of the information.

[1] http://dblp.uni-trier.de/

ÍNDICE

Listado de figuras

Listado de tablas

1. INTRODUCCIÓN

1.1. Descripción del proyecto

El proyecto consiste en desarrollar una plataforma web que contenga información acerca de diferentes investigadores científicos y de publicaciones científicas con el fin de unificar en una sola plataforma toda la información que se puede encontrar en diferentes sitios web y la información que puede ser obtenida utilizando diferentes servicios que acceden a diferentes repositorios bibliográficos. A parte de unificar estos datos, también se pretende enriquecer la información utilizando diferentes técnicas de procesamiento de texto y crear diferentes representaciones que aporten valor y más información al usuario que utilice la plataforma desarrollada.

1.2. Objetivo

El objetivo de este TFG es combinar diferentes áreas de la Ingeniería como Big Data y Procesamiento del Lenguaje Natural para poder desarrollar el proyecto y cumplir con las especificaciones de este.
Se estudiarán diferentes plataformas y métodos para obtener grandes cantidades de datos para ser procesados posteriormente y se analizará cómo utilizar los métodos de forma eficiente para que el sistema pueda usar los datos lo antes posible.
Una vez los datos sean obtenidos se estudiarán y evaluarán diferentes métodos para que puedan ser procesados lingüísticamente de forma eficiente para el sistema.
También se evaluarán diferentes maneras de visualización de los datos para que el usuario pueda apreciar el valor aportado y que pueda visualizarlos de una forma intuitiva.

1.3. Visión general

El usuario puede acceder a una web y hacer uso de las diversas funcionalidades para poder profundizar en la información de los diferentes autores y publicaciones de estos y valorar cuál de esta tiene más relevancia para él.

La aplicación que se ha desarrollado permite que el usuario pueda:

- Buscar y comparar autores
- Ver información acerca de un autor y sus publicaciones
- Ver información acerca de una publicación
- Ver información acerca de las colaboraciones entre autores.

Las funcionalidades completas, junto con los métodos y herramientas que se han utilizado para su desarrollo, se pueden encontrar detalladas en las secciones 3.2, 5.1 y 6 de este documento. Sin embargo, en este apartado se pueden visualizar algunas imágenes acerca de las opciones que tiene el usuario.

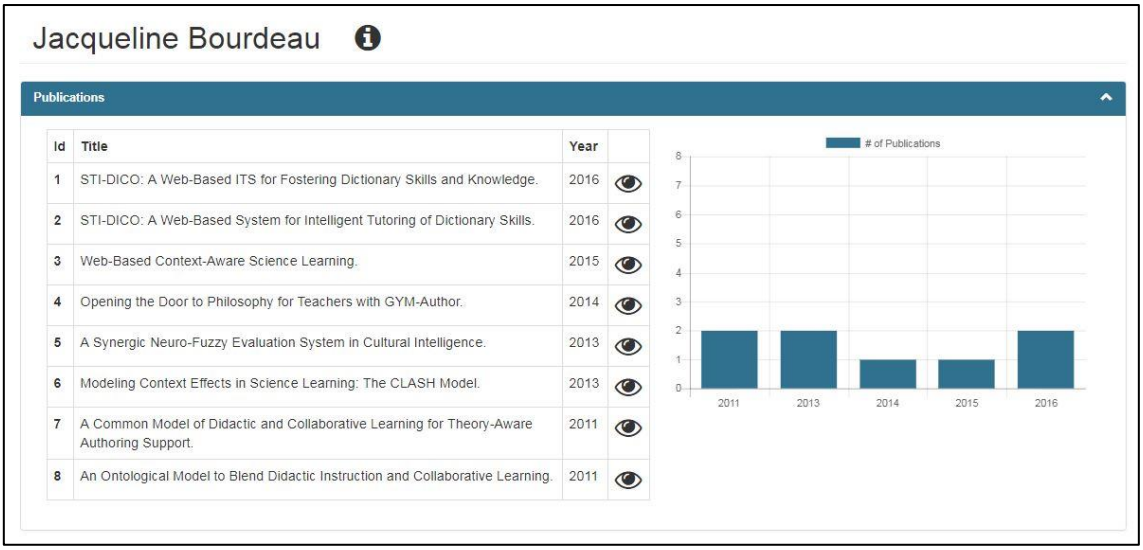


Figura 1 Ejemplo de funcionalidad: Listado de publicaciones. Elaboración propia

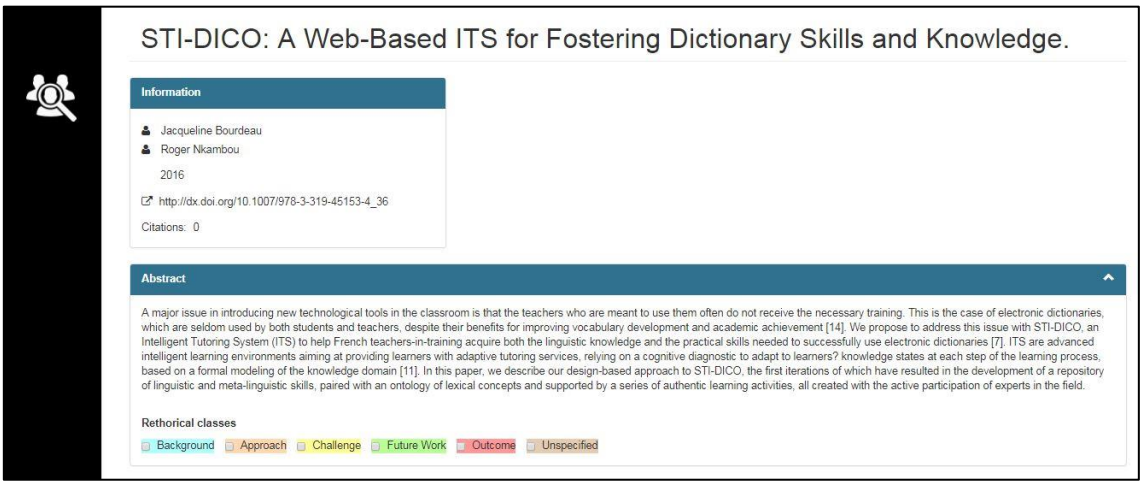


Figura 2 Ejemplo de funcionalidad: Detalle de una publicación. Elaboración propia

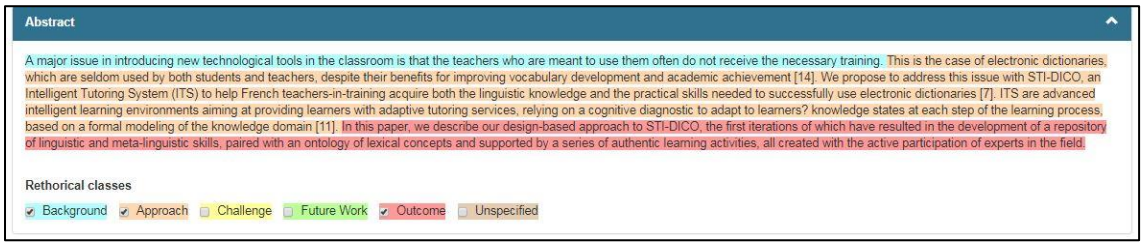


Figura 3 Ejemplo de funcionalidad: Categorización de oraciones. Elaboración propia

2. ESTADO DEL ARTE

2.1. Actualidad de la literatura científica

Según la ley bibliométrica de crecimiento exponencial formulada en 1976 por el británico Derek John de Solla Price, el crecimiento de la información científica ocurre a un ritmo muy alto, de tal forma que la información científica disponible puede duplicarse cada 10 o 15 años.
Esta situación ha producido que la literatura científica haya experimentado un crecimiento exponencial a lo largo de las últimas décadas, produciendo que el aumento del número de publicaciones sea proporcional al crecimiento en número de investigadores científicos en todo el mundo.
A continuación se puede observar un gráfico donde se muestra el aumento en cuanto a publicaciones científicas durante 1980 y 2012.

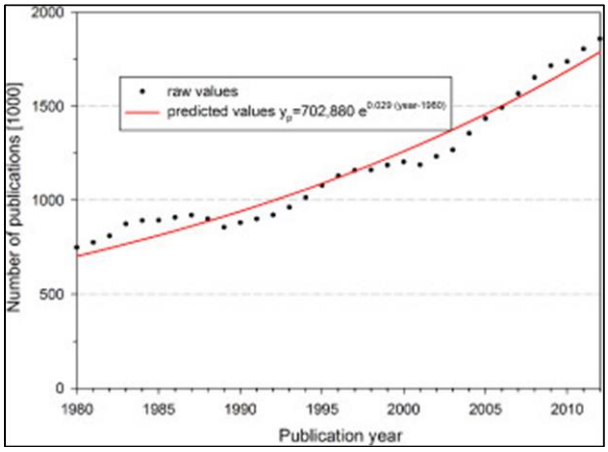


Figura 4 Crecimiento de las publicaciones científicas **(Ley bibliométrica de crecimiento exponencial)**

Este crecimiento exponencial de la literatura científica junto con la aparición de las nuevas tecnologías ha permitido que el acceso a las diferentes publicaciones también se encuentre en una dinámica creciente, ya que han aparecido diferentes plataformas digitales que permiten acceder al contenido de diferentes publicaciones y así ayudar a facilitar acceso a la información a diferentes investigadores o usuarios interesados en la ciencia.

Algunas ejemplos de estas plataformas digitales son SpringerLink[2], que proporciona acceso a más de 8.5 millones de documentos de investigación, o ScienceDirect[3], que actualmente ofrece acceso abierto a más de 250.000 publicaciones.

[2] https://link.springer.com/
[3] http://www.sciencedirect.com/

Los diferentes investigadores y usuarios interesados en utilizar estas plataformas también se han visto favorecidos de esta situación en la facilidad de acceso a aquellos documentos que se han utilizado para elaborar otras publicaciones.
Web of Science, por ejemplo, es un servicio que permite acceder a las publicaciones que han sido citadas en un documento en concreto.

Esta situación ha provocado que la importancia de los buscadores de literatura en línea sea más relevante, ya que pueden acceder a más contenido y a una mayor cantidad de documentos. Sin embargo, el tiempo de lectura de un documento ha decrecido. (Ronzano & Saggion, 2016)

2.2. Big Data y procesamiento de lenguaje natural en literatura científica

Debido a la expansión de Internet que se ha producido en las últimas décadas, la cantidad de datos a los que se pueden acceder en diferentes formatos ha aumentado considerablemente. De este hecho ha nacido el concepto de 'Big Data' (Arcila-Calderón, Barbosa-Caro, & Cabezuelo-Lorenzo, 2016), (Moreno & Redondo, 2016). Entre los datos que han experimentado este cambio, también se encuentran los que están relacionados con la literatura científica y que son de interés de análisis para los investigadores.

A raíz de esta situación, los campos de la ingeniería de procesamiento de lenguaje natural y análisis de texto han empezado a considerarse tecnologías imprescindibles para ayudar a los investigadores a trabajar con todo el volumen de información que tienen accesible. Gracias a ellas es posible analizar la estructura y la semántica del contenido de diferentes publicaciones para poder proporcionar información más concreta y crear resúmenes automáticos de textos que permitan reducir el tamaño del documento analizado y obtener el contenido más importante de este, de tal forma que un investigador científico pueda valorar más rápidamente la relevancia que tiene un documento para él.

Existen diferentes herramientas tecnológicas enfocadas a facilitar el proceso de análisis textual en publicaciones científicas como son Dr. Inventor[4], GATE[5] y SUMMA[6].

Dr. Inventor (Ronzano & Saggion, 2015) es un proyecto europeo en el que colaboran diferentes universidades y centros de investigación, como la Univesidad Pompeu Fabra (UPF), Universidad de Bedfordshire (BED) o Imagemetry (IME)[7] y se centra en actuar como un *"asistente de investigación personal para proporcionar ampliadas perspectivas acerca de la innovación científica"*.

[4] http://drinventor.eu/
[5] https://gate.ac.uk/
[6] http://www.taln.upf.edu/pages/summa.upf/documentation.html
[7] http://drinventor.eu/IME.html

GATE (Cunningham, et al., 2017) es una herramienta resultante de un proyecto de I+D iniciado en 1995, utilizada por una comunidad muy extensa de desarrolladores y usuarios y que permite realizar tareas de procesamiento de texto con el objetivo de utilizar la información obtenida en diferentes áreas de investigación.

Por su parte, SUMMA (Saggion, 2008), (Saggion, 2014) es una herramienta de resumen de texto que complementa las funcionalidades permitidas por GATE que permite añadir información necesaria para la creación de aplicaciones de resumen.

3. DISEÑO TÉCNICO

3.1. Arquitectura del proyecto

El proyecto entero se divide en dos proyectos Java con arquitectura Maven[8]. Este tipo de arquitectura permite añadir de forma externa las dependencias necesarias en un documento XML sin la necesidad de tener que descargarlas y añadirlas en el *build path*.

El primer proyecto es utilizado para implementar todas las tareas necesarias para poder obtener toda la información para su posterior visualización, como las tareas de *scraping*, procesamiento de texto con GATE y Dr. Inventor o peticiones a la biblioteca bibliográfica DBLP y el grafo de datos de Microsoft Academic Graph (MAG)[9].

El segundo proyecto está destinado a la visualización de los datos. En este caso, la arquitectura del proyecto es un Maven Web, lo que permite poder crear archivos de tipo JSP que contienen código en HTML, CSS, Javascript, etc. para la visualización de los datos vía web mezclando código Java para poder obtener y procesar los datos en el servidor. Para ello, es necesario crear un servidor web en el proyecto para poder compilar este tipo de archivos en su ejecución.

El motivo por el que se ha dividido el proyecto en dos subproyectos es que, aunque ambos comparten algunas clases y utilizan la misma base de datos, no comparten todas las dependencias y librerías, igual que un proyecto no necesita ser ejecutado mediante un servidor web. De esta forma, en cada proyecto se encuentran sólo esas dependencias, librerías y componentes que necesita para su funcionamiento.

3.1.1. Diagrama de arquitectura

En este apartado se muestra un diagrama con un diseño genérico de la arquitectura del sistema donde se refleja su comportamiento como un único proyecto, ya que, aunque internamente sean dos proyectos diferentes, el objetivo final es poder mostrar una visualización al usuario de los datos disponibles.

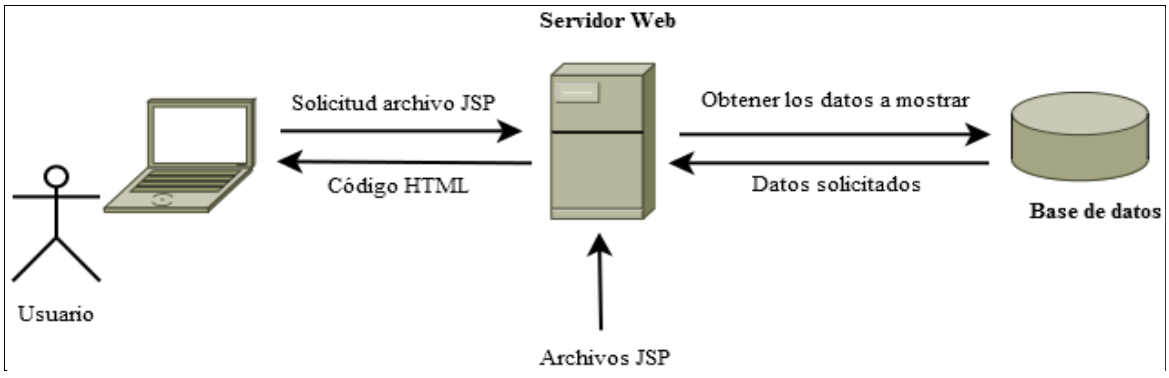


Figura 5. Arquitectura general del proyecto. Elaboración propia

[8] https://maven.apache.org/
[9] https://www.microsoft.com/en-us/research/project/microsoft-academic-graph/

El usuario hace una petición HTTP desde un navegador web a un Servidor Web solicitando un archivo JSP. A continuación, el Servidor detecta que la petición realizada es para un archivo JSP y se encarga de obtener el fichero y convertirlo en un Servlet[10], compilarlo y ejecutarlo. Durante la ejecución, el Servlet produce un fichero de salida en HTML, obteniendo los datos necesarios de la base de datos para poder mostrarlos.
Finalmente, el Servidor Web devuelve como respuesta a la petición realizada por el usuario el contenido HTML que ha sido creado

3.1.2. Diagrama de clases

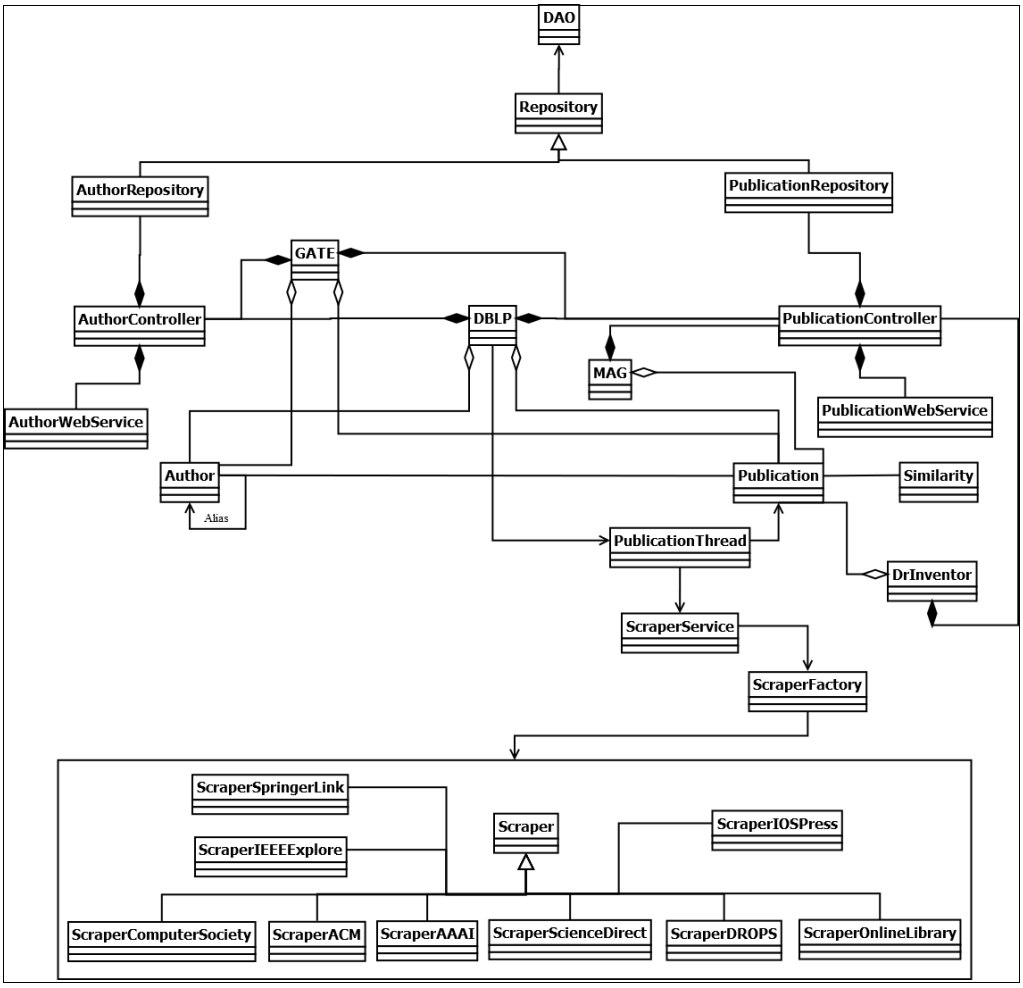


Figura 6 Diagrama de clases. Elaboración propia

La estructura formada por las clases que forman parte del proyecto ha sido diseñada siguiendo un modelo de capas, donde cada una de ellas se centra en una funcionalidad muy concreta y permite desarrollar las diferentes funcionalidades de forma organizada.

[10] https://users.dcc.uchile.cl/~jbarrios/servlets/general.html

Se pueden identificar cuatro capas diferentes:

- **Entidades**: son aquellas clases básicas para el funcionamiento del sistema y que contienen la información que es utilizada para implementar las diferentes funcionalidades y visualizar los datos. A este grupo pertenecen las clases *Author, Publication y Similarity*.

- **Controladores funcionales**: son aquellas clases que contienen todo el desarrollo principal de las funcionalidades del proyecto. *DBLP, Dr. Inventor, GATE y MAG* pertenecen a este grupo.

- **Controladores de entidades**: son aquellas clases que actúan como intermediarios entre los controladores funcionales y los repositorios o servicios. Cuando un controlador funcional quiere hacer uso de la base de datos o acceder a un servicio web, realiza una llamada al controlador de la entidad correspondiente y éste decide cuál es la siguiente llamada que ha de realizarse. Si el repositorio o servicio que será llamado necesita los datos de una forma concreta, el controlador se encarga de procesarlos para que éstos únicamente realicen las peticiones correspondientes a la base de datos o servicios y devolver los datos necesarios. *AuthorController y PublicationController* pertenecen a este grupo.

- **Repositorios/Servicios**: son aquellas clases dedicadas acceder a la base de datos, ya sea para insertar u obtener información y acceder a los servicios web necesarios para obtener datos. Cuando se accede a aquellos métodos relacionados con obtener información, estas clases se encargan de crear nuevas entidades, devolverlas al controlador de entidad y éste las devuelve al controlador funcional para que pueda trabajar con ellas.

A continuación se muestra una representación de este comportamiento con el siguiente ejemplo: “Obtener las publicaciones del servicio web DBLP e insertarlas en la base de datos”.

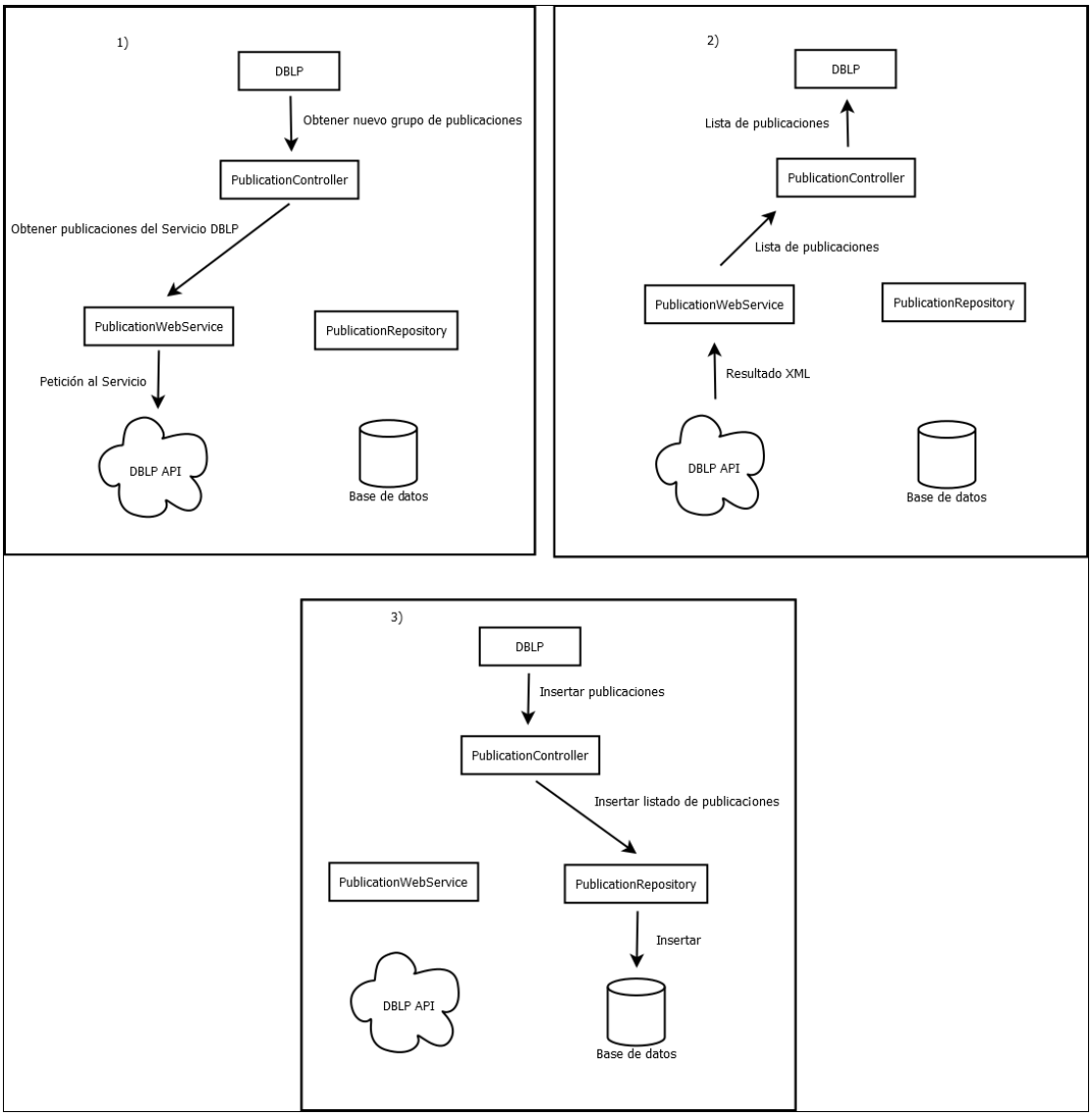


Figura 7 Comunicación entre clases. Elaboración propia

Finalmente, en la parte inferior del diagrama se muestra el diseño del sistema de *scraping*. Esta parte sigue el patrón de diseño *Factory*[11], explicado en el punto 4.4.5, y se accede a él mediante cada instancia de la clase *PublicationThread*. Las instancias de esta clase las realiza la clase DBLP con el objetivo de crear un *thread*[12] para cada dirección web a la que se quiere hacer *scraping*.

[11] https://www.tutorialspoint.com/design_pattern/factory_pattern.htm
[12] https://msdn.microsoft.com/es-es/communitydocs/win-dev/os/manejador-de-procesos-los-threads

3.1.3. Base de datos

Para poder trabajar con todos los datos disponibles y poder acceder a ellos de forma rápida una vez procesados es necesario poder almacenarlos en una base de datos ubicada en un servidor.

A continuación se muestra un diagrama de la base de datos y el propósito de cada una de las tablas existentes en ella.

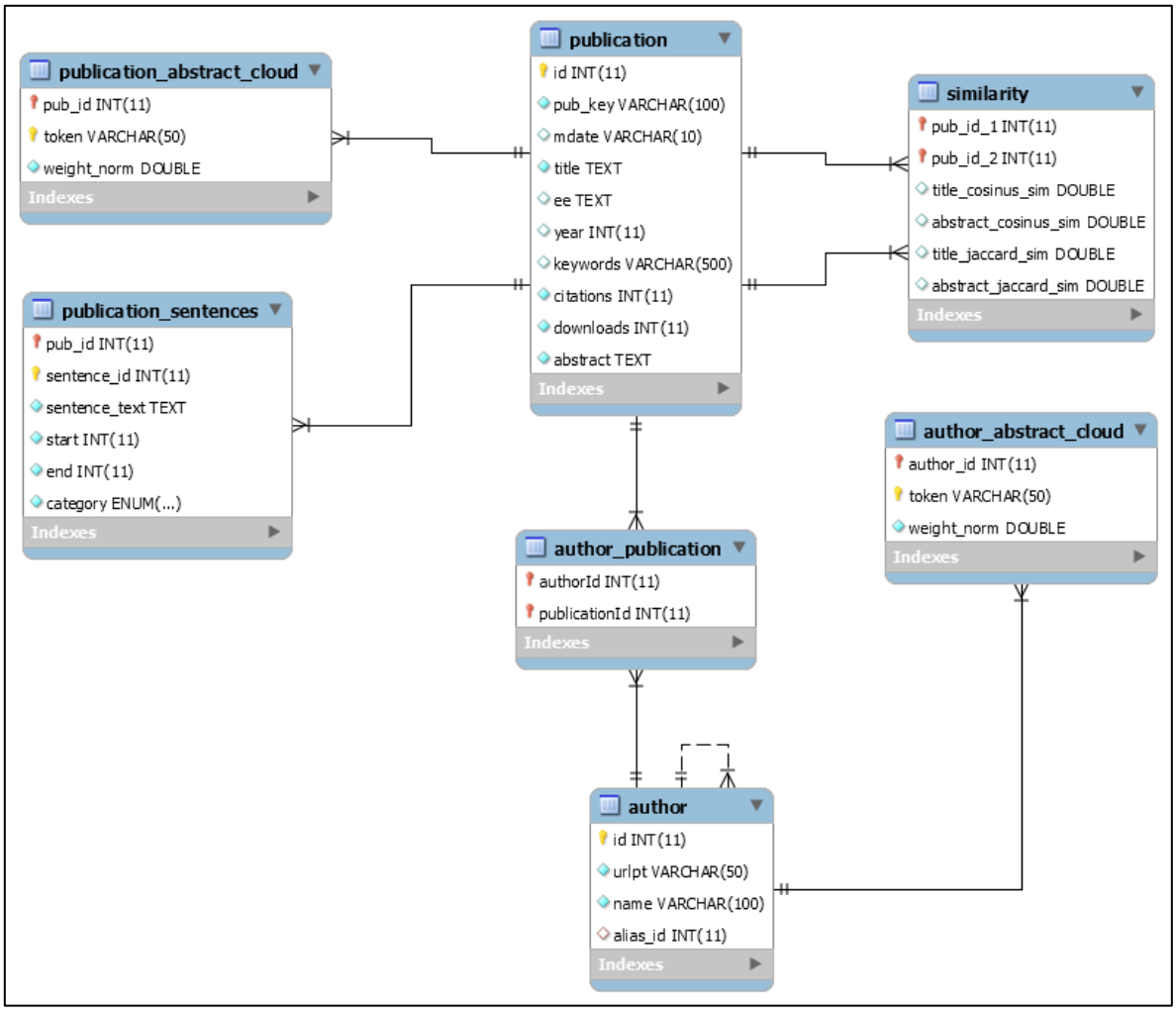


Figura 8 Base de datos. Elaboración propia

Nombre de la tabla	Propósito
author	Almacenar los datos de un autor
publication	Almacenar los datos de una publicación
author_publication	Almacenar los autores que han participado en una determinada publicación.
similarity	Almacenar la similitud entre dos publicaciones. Se almacena la similitud calculada por el método distancia Coseno y por el método *Jaccard* sobre el título y el *abstract* de las publicaciones.
publication_sentences	Almacenar la categoría retórica de las oraciones de una publicación
author_abstract_cloud	Almacenar las palabras más relevantes para un autor entre todos sus *abstracts* con su correspondiente peso.
publication_abstract_cloud	Almacenar las palabras más relevantes en el *abstract* de una publicación con su correspondiente peso.

Tabla 1 Tablas de la base de datos

3.1.4. Tecnologías utilizadas

En este punto se detallan las tecnologías utilizadas a más alto nivel. A medida que se detallen las diferentes partes del proyecto, se detallarán las tecnologías utilizadas para esa parte en concreto.

- **Java[13]:** lenguaje de programación principal con el que se ha desarrollado la parte de servidor del proyecto.

- **HTML[14], CSS[15] y jQuery[16]:** permiten crear la web para obtener una visualización de los datos.

- **Apache XAMPP[17]:** distribución de Apache que permite crear un servidor local para poder almacenar bases de datos y ejecutar funcionalidades que requieran disponer de un servidor.

[13] https://www.java.com/es/about/
[14] https://es.wikipedia.org/wiki/HTML
[15] http://www.w3c.es/Divulgacion/GuiasBreves/HojasEstilo
[16] https://jquery.com/
[17] https://www.apachefriends.org/es/index.html

- **Apache Tomcat**[18]**:** permite crear un entorno de ejecución web y compilar los archivos JSP creados.

- **MySQL**[19]**:** sistema de gestión de base de datos relacional que permite almacenar los datos y poder realizar diferentes consultas sobre ellos.

- **SQL**[20]**:** lenguaje con el que se realizan las consultas a la base de datos.

3.2. Diseño funcional

El proyecto ha sido diseñado con el objetivo de proporcionar las siguientes funcionalidades al usuario:

- **Ver detalle de un autor:** permite ver los datos relacionados con un autor, como sus alias, un gráfico de sus publicaciones ordenadas por años, un listado de colaboraciones con otros autores, un listado de sus publicaciones, un grafo de similitud entre sus publicaciones y una nube de términos con aquellos términos más relevantes para él.

- **Ver detalle de una publicación:** permite ver los datos de una publicación, como los autores que la han realizado, el año, el título, el número de citas, el *abstract*, las categorías retóricas de las oraciones y una nube de términos con aquellos términos más relevantes en el *abstract*.

- **Ver detalle de la colaboración entre autores:** permite ver datos acerca de la colaboración entre dos autores, como un listado con las publicaciones en las que han colaborado y un gráfico donde se muestra su colaboración a lo largo del tiempo.

- **Comparar autores:** permite comparar dos o más autores mostrando un grafo de similitud entre las publicaciones de cada uno de ellos y un grafo donde se muestra la similitud entre ellos utilizando una media de la similitud obtenida para sus publicaciones.

[18] http://tomcat.apache.org/
[19] https://www.mysql.com/
[20] https://www.w3schools.com/sql/sql_intro.asp

3.2.1. Diagrama de casos de uso

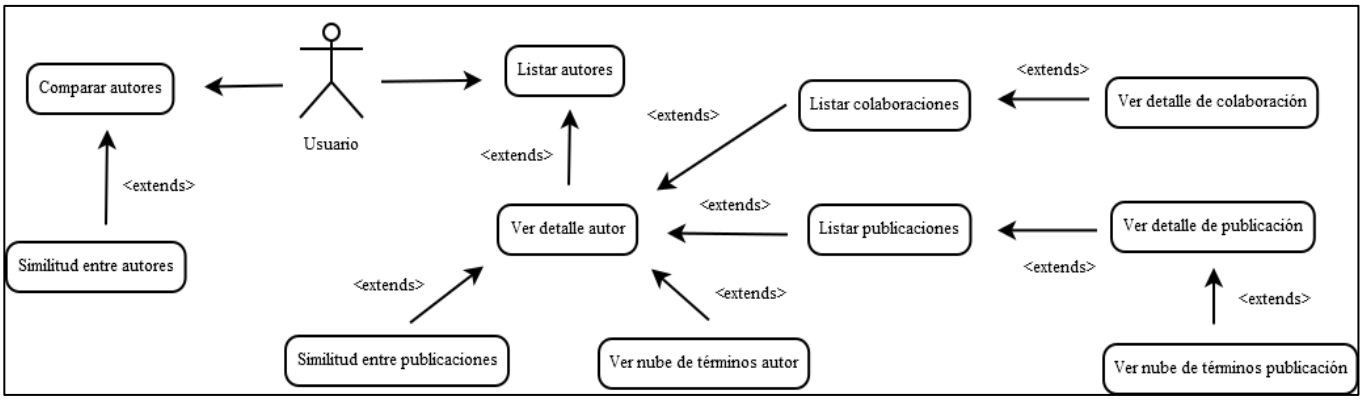


Figura 9 Casos de uso. Elaboración propia

En este diagrama se pueden observar las acciones que puede hacer el usuario, descritas anteriormente, junto con las relaciones de dependencias que existen entre ellas. Aquellas relaciones que están etiquetadas como "extends" significan que una acción complementa a la otra. Por ejemplo, la acción "Ver nube de términos publicación" complementa a la acción "Ver detalle de publicación". Otro tipo de relación existente es "include" y significa que una acción es necesaria para otra pueda existir. En este caso no hay ninguna relación de este tipo.

4. OBTENCIÓN DE DATOS

Para poder cumplir con los objetivos del proyecto es muy importante disponer de un conjunto de datos suficientemente grande y completo para poder analizar.
La obtención de este conjunto de datos ha sido una parte fundamental y la etapa más larga a lo largo del desarrollo del proyecto. Se han utilizado diferentes técnicas y herramientas para obtenerlo y la información necesaria para poder trabajar con ellos.

4.1. DBLP

Es un sitio web que contiene un repositorio bibliográfico en el que se puede encontrar información acerca de una gran cantidad de publicaciones científicas.

Es una herramienta muy popular para los diferentes investigadores ya que permite rastrear el trabajo realizado por otros compañeros de investigación y recuperar detalles bibliográficos para poder componer una lista de referencias en la creación de nuevos artículos, clasificar el perfil de otros investigadores y obtener información sobre revistas o conferencias.

Mediante DBLP (Ley, 2002) es posible consultar y obtener las publicaciones de diferentes autores, colaboraciones entre autores, publicaciones en determinadas conferencias y los autores que han participado en ellas, entre otros tipos de información.

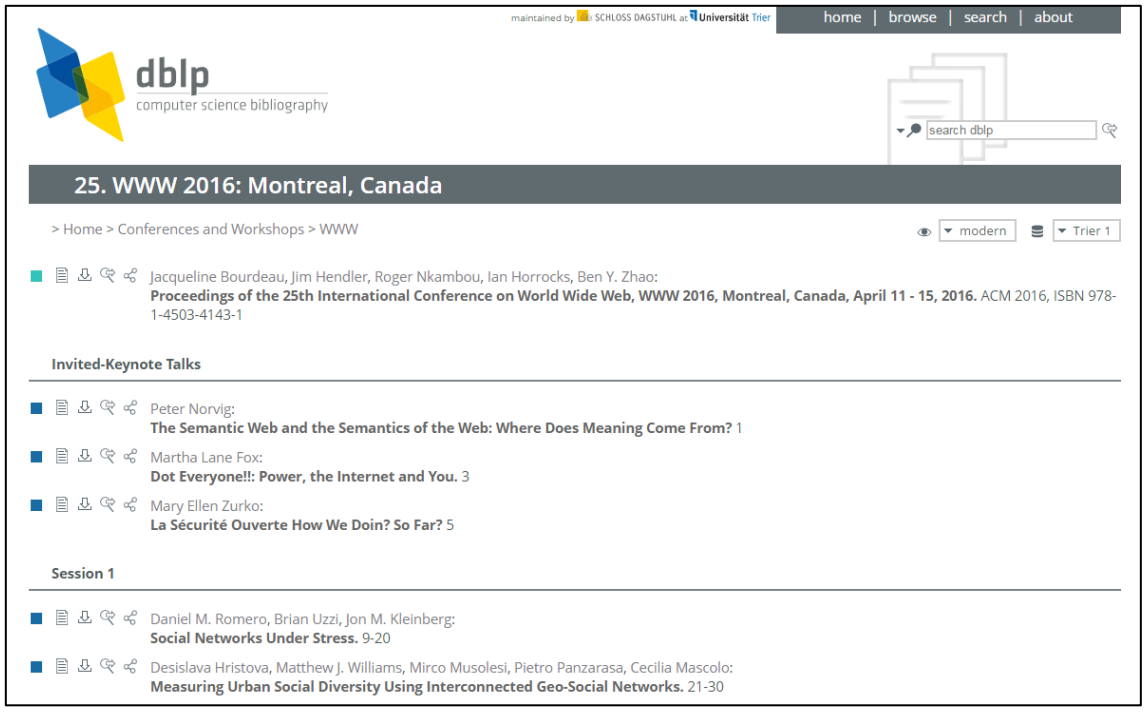


Figura 10 DBLP: Conferencia www2016 **(dblp: WWW 2016)**

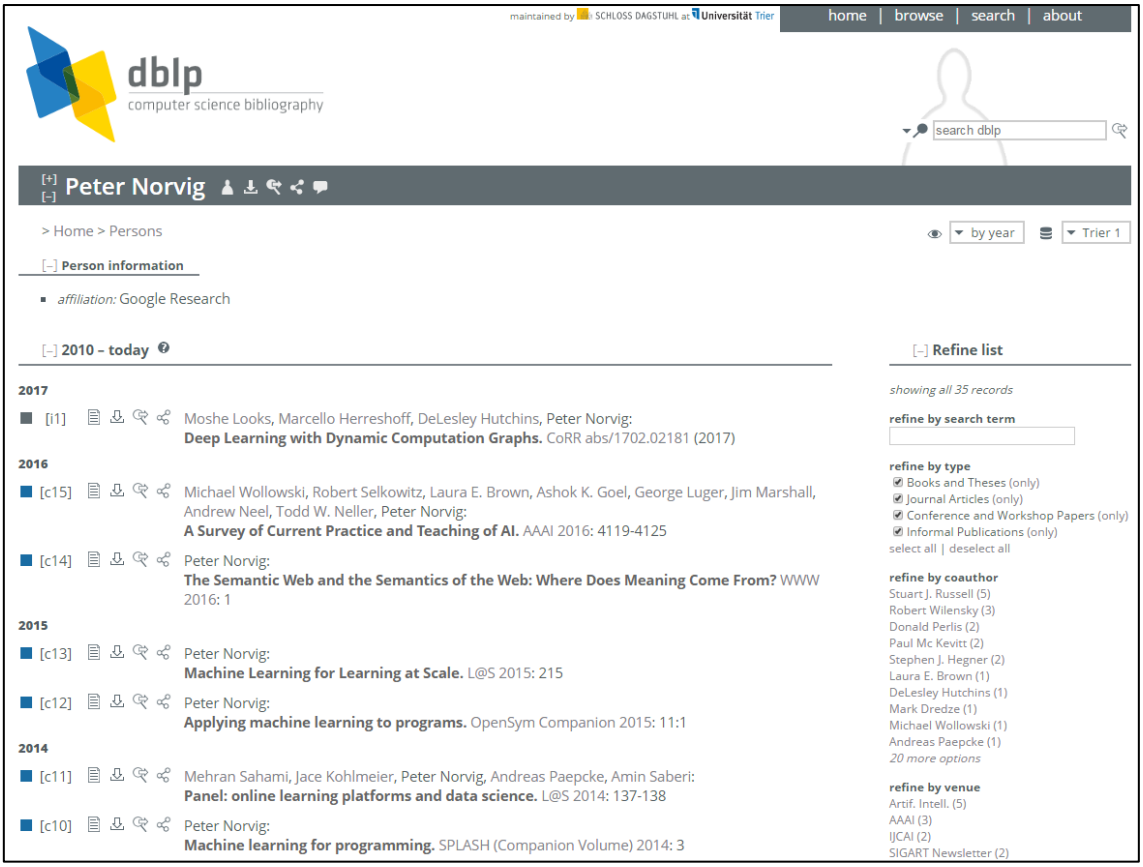


Figura 11 DBLP: Autor Peter Norvig **(dblp: Peter Norvig)**

A continuación se muestra un gráfico donde se observa una distribución de los diferentes tipos de publicaciones que es posible encontrar en el repositorio

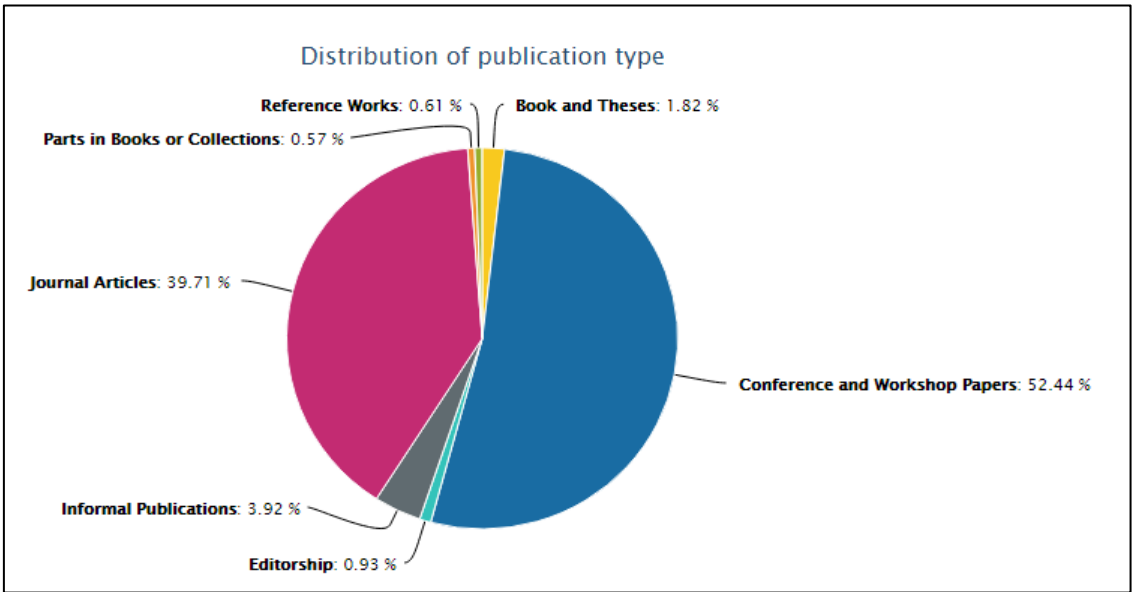


Figura 12 Tipos de publicaciones en DBLP **(dblp: Distribution of publication type)**

4.1.1. Extracción de datos en DBLP

Es posible acceder al contenido del repositorio de diferentes formas:

- **Descarga del contenido:** DBLP permite descargar todos los datos en un archivo XML. Para poder desarrollar un software que permita parsear los datos de este fichero, DBLP también tiene disponible un archivo *dtd* donde se detalla la estructura que debe cumplir el XML.

- **Peticiones XML:** DBLP dispone de una API la cual es posible acceder a ella mediante un determinado tipo de consultas y obtener un conjunto de datos más reducido y concreto. (Ley, 2009).

Sin embargo, también existen otras alternativas que permiten obtener la información almacenada en el repositorio y que proporciona una herramienta de búsqueda más potente que la que tiene DBLP.

- **FacetedDBLP[21]:** es una interfaz de búsqueda que permite encontrar datos de DBLP de una forma mucho más concreta, como por ejemplo, buscar por un conjunto de palabras clave. Esto permite a los usuarios poder obtener datos según un determinado tema de investigación y dividirlos en diferentes subtemas. Esta herramienta permite descargar todos los datos del repositorio DBLP en un archivo SQL para ser importado por un usuario directamente a una base de datos y poder trabajar localmente de la forma deseada sin necesidad de usar la interfaz de búsqueda.

4.1.2. DBLP en el proyecto

Para obtener las publicaciones y los datos se ha utilizado la opción de realizar peticiones a la API y obtener un resultado en XML, ya que no es necesario utilizar toda la información contenida en DBLP y es más eficiente procesar la respuesta obtenida de una petición concreta que procesar todo el contenido del fichero XML con todos los datos de DBLP cada vez que queremos obtener una parte determinada de esos datos.

Se pueden encontrar diferentes categorías de publicaciones y para cada una es posible encontrar diferente información. Para el proyecto se ha utilizado la categoría "*Conferences and Workshop Papers*"[22], en la que es posible encontrar aquellas publicaciones realizadas en conferencias o relacionadas con talleres de formación.

[21] http://dblp.l3s.de/dblp++.php
[22] http://dblp.uni-trier.de/faq/What+types+does+dblp+use+for+publication+entries.html

4.1.3. Arquitectura del sistema

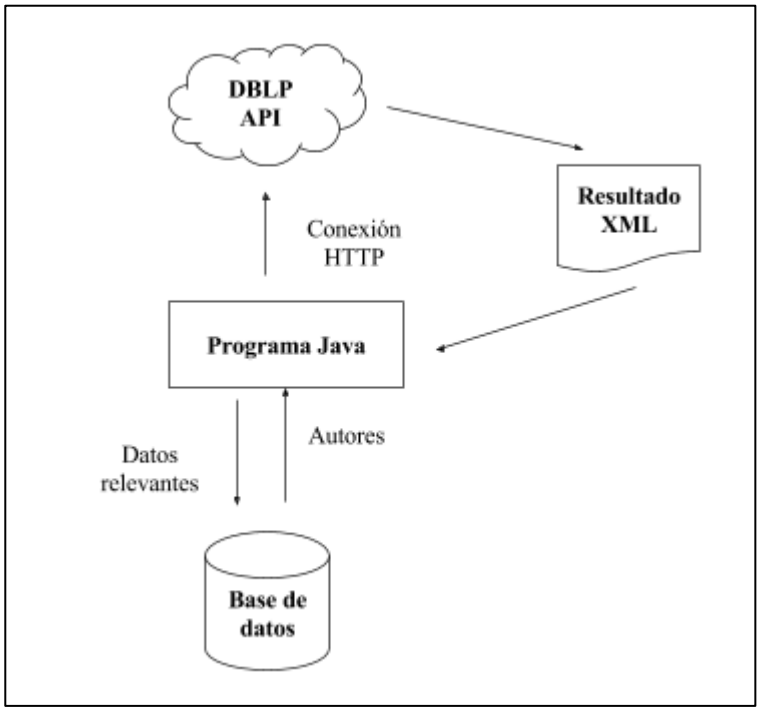


Figura 13 Obtención de datos mediante DBLP. Elaboración propia

Mediante un programa en Java se obtienen los datos de la base de datos que serán necesarios para solicitar la información a la API de DBLP. Esta devuelve una respuesta en formato XML que es parseada para obtener aquellos datos relevantes que, posteriormente, son almacenados en la base de datos.

4.1.4. Implementación del sistema

El servicio que ofrece DBLP es utilizado para obtener las publicaciones de un autor. Para ello, existe un término llamado "urlpt" que hace referencia a la clave con la que DBLP, internamente, almacena la información de un determinado autor. Para poder obtener las publicaciones de uno en concreto, se puede consultar el servicio de la siguiente manera: http://dblp.uni-trier.de/pers/xx/urlpt.xml

DBLP también proporciona un servicio donde mediante otro tipo de petición devuelve un listado de autores que coinciden con el criterio de búsqueda establecido en la petición. Sin embargo, en este caso se utilizan técnicas de *web scraping* (punto 4.4) para obtener un conjunto de autores con sus respectivas "urlpt".

Ejemplo:

- **Autor:** Jacqueline Bourdeau
- **Urlpt:** b/Bourdeau:Jacqueline
- **URL:** http://dblp.uni-trier.de/pers/xx/b/Bourdeau:Jacqueline.xml

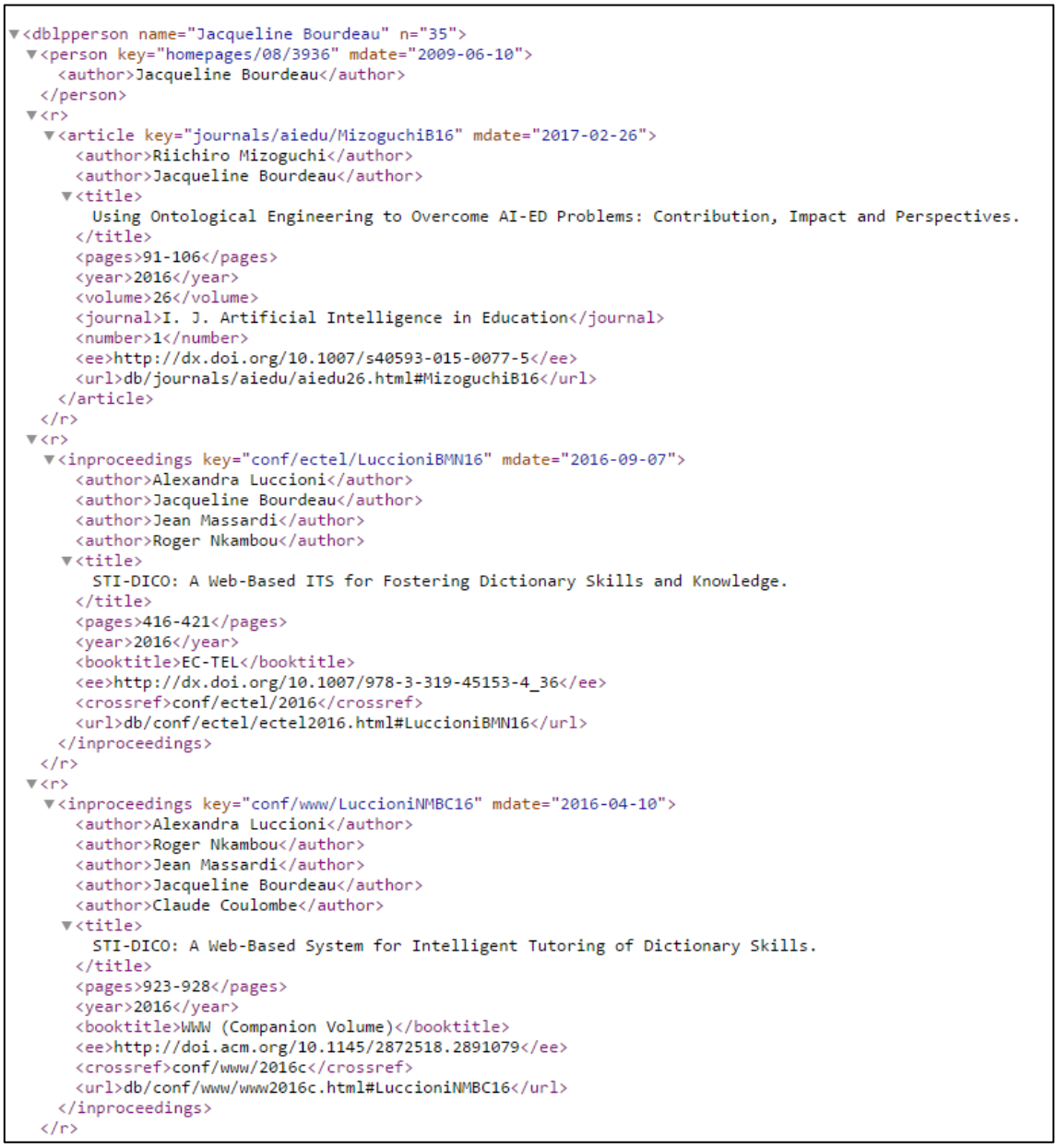


Figura 14 Resultado de una petición a DBLP

Se puede observar como con la anterior URL obtenemos un resultado en formato XML. En este documento tenemos un listado de todas las publicaciones del autor y la información de interés de cada una de ellas.

Los parámetros más importantes que se utilizan en el proyecto son los siguientes:

- **Key:** hace referencia al valor que con que se identifica la publicación en DBLP.
- **Title:** título de la publicación.
- **Year:** año de publicación
- **ee:** dirección de la plataforma digital donde se puede encontrar la publicación.

Una vez se obtiene el resultado, este es parseado para obtener los parámetros listados anteriormente y se almacenan los datos de cada publicación del tipo "*inproceedings*" en la base de datos. Las entradas etiquetadas como "*inproceedings*" hacen referencia a esas publicaciones relacionadas con la categoría "*Conferences and Workshop Papers*".

4.2. MAG

Microsoft Academic Graph (MAG) es un grafo que contiene información acerca de publicaciones científicas y autores, como el número de citas de una publicación, las afiliaciones de un autor y datos acerca de conferencias y diferentes áreas de estudio. Se puede acceder al contenido del MAG mediante el servicio "Microsoft Cognitive Services Academic Knowledge API[23]". Con este servicio, se pueden realizar diferentes consultas al MAG para recuperar el diferente contenido de interés. Los datos que se encuentran en el MAG son extraídos del índice web Bing y de su base de datos interna. Cada vez que Bing realiza una indexación, el contenido del MAG es actualizado.

Para poder utilizar el servicio "Microsoft Cognitive Services Academic Knowledge API" es necesario obtener una clave de suscripción para poder realizar las diferentes consultas.

4.2.1. MAG en el proyecto

Se ha utilizado en el proyecto para complementar la información de las publicaciones obtenidas mediante DBLP y *web scraping* con el número de citas para cada publicación.

[23] https://docs.microsoft.com/es-es/azure/cognitive-services/academic-knowledge/home

4.2.2. Arquitectura del sistema

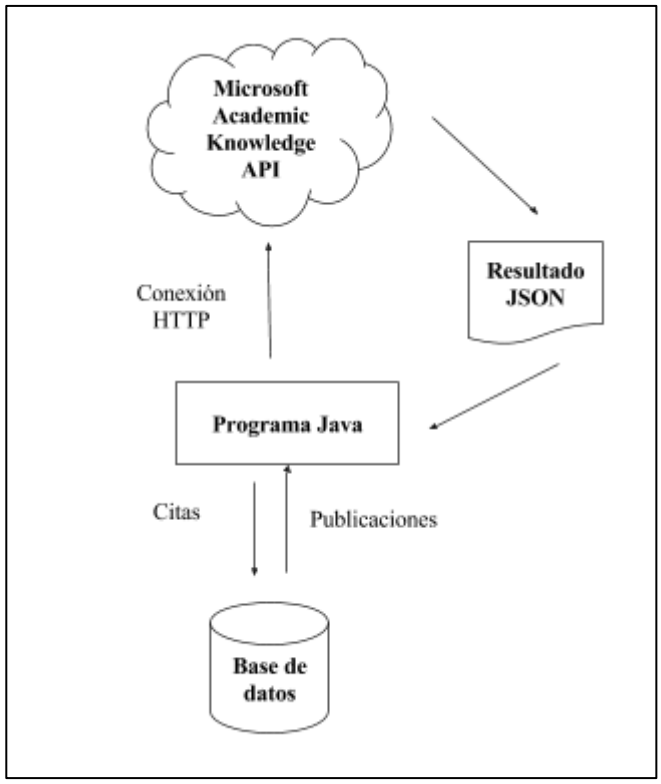


Figura 15 Obtención de datos mediante MAG. Elaboración propia

Mediante un programa en Java se obtienen los datos de la base de datos y se realiza una conexión HTTP con el servicio de Microsoft Academic Knowledge y se realiza una petición. Una vez el servicio la procesa, devuelve el resultado en formato JSON[24], que es parseado con el programa Java y almacenado en la base de datos.

4.2.3. Implementación del sistema

Lo primero que hay que tener en cuenta es que para poder realizar una solicitud a la API de Microsoft Academic Knowledge hay algunos datos que necesitan ser normalizados. En este caso, para obtener las citas relacionadas con una publicación, la petición se hace mediante el título y el año de publicación. El título puede contener una serie de caracteres que hagan que no coincida con la forma en como está almacenado en MAG. Por lo tanto, es necesario normalizar el título.

Para ello se ha utilizado la siguiente expresión regular: [^A-Za-zα-ωA-Ω0-9_]. Lo que se hace con ella es reemplazar todos aquellos caracteres en el título de la publicación que no se encuentren en los rangos definidos por estos valores por un espacio en blanco.

[24] http://www.json.org/json-es.html

Seguidamente, se tiene que crear la conexión HTTP y crear la petición especificando la URL a la que se quiere hacer la conexión y añadir los parámetros necesarios.

- **URL:** https://westus.api.cognitive.microsoft.com/academic/v1.0/evaluate
- **Parámetros**:
  - **expr:** consulta que se quiere realizar. En este caso es la siguiente: "And(Ti = [*título de la publicación normalizado*], Y = [*año de publicación*])". De esta forma se está solicitando aquellas publicaciones que coincidan con el título y el año.
  - **attributes:** hace referencia a aquella información que se quiere recuperar. En este caso se recuperan los atributos "Ti" y "CC", que hacen referencia al título y número de citas de la publicación.

Una vez los parámetros de la petición están configurados, se ejecuta y se obtiene un documento JSON con los resultados devueltos por la API. Este resultado es parseado para obtener el número de citas.

4.3. Web Scraping

"*Es una técnica utilizada mediante programas de software para extraer información de sitios web*". (Web Scraping)

Estos programas se utilizan para simular el comportamiento de un humano en la web de forma automatizada y es una técnica muy utilizada ya que permite obtener grandes cantidades de datos para poder ser tratados posteriormente.

4.3.1. Problemas de web scraping

Debido a que *web scraping* es un proceso automatizado, permite realizar muchas peticiones por segundo en un mismo sitio web, generando un aumento del tráfico que puede poner en riesgo el rendimiento óptimo del sistema que se está visitando.

Es por eso que muchos sitios web intentan evitar el uso de esta técnica y tienen procesos que controlan y detectan su posible uso. Algunas de las medidas que usan sus administradores para detener o disminuir peticiones que provengan mediante el uso de *web scraping* son las siguientes:

- **Añadir entradas al fichero robots.txt:** el fichero robots.txt es un fichero que se encuentra en el directorio raíz del sistema de un sitio web y que permite configurar qué robots tienen acceso a determinados archivos del sistema.

Si se añaden las siguientes líneas al fichero, todo robot queda excluido del acceso a los diferentes archivos del sistema.

User-agent: *
Disallow: /

- Monitorear el tráfico proveniente de una misma dirección IP y bloquearle el acceso al sistema si las peticiones realizadas siguen un comportamiento inusual.

- Añadir sistemas de verificación manual como, por ejemplo, *captcha*[25], lo que provoca que sea más complicado para los "*scrapers*"[26] realizar peticiones y obtener la información satisfactoriamente.

Cuando un determinado sitio web detecta el uso de esta técnica y bloquea el acceso a una cierta dirección IP, este bloqueo puede ser temporal (durar minutos, horas o días) o permanente.

El usuario que está haciendo *scraping* puede detectar o intuir si ha sido o no bloqueado de distintas formas. Es útil comprobar los tiempos de respuesta del sitio web para detectar si hay una cierta demora no habitual y si es continua. También se puede observar el código de respuesta obtenido después de realizar la petición. Cuando un sitio web detecta este tipo de proceso automatizado, suelen devolver ciertos códigos de respuesta para evitar que el usuario siga realizando peticiones.

Los códigos de respuesta más comunes suelen ser:

Código de respuesta	Significado
301	La página ha sido movida temporalmente
401	La dirección de origen no está autorizada a acceder a la página
403	Acceso denegado
404	Página no encontrada
408	Tiempo de espera agotado
429	Demasiadas peticiones desde la misma dirección de origen
503	La página no se encuentra disponible

Tabla 2 Códigos de respuesta en posible bloqueo de *scraping* **(How to prevent getting blacklisted while scraping)**

[25] https://support.google.com/a/answer/1217728?hl=es
[26] Usuarios que utilizan técnicas de *scraping*

4.3.2. Buenas prácticas en el uso de web scraping

Es complicado hacer *scraping* de forma eficiente y evitar que los diferentes sitios web que tienen protección contra este proceso lo detecten. Para ello, existen una serie de puntos que pueden ser de ayuda en la implementación de un proceso como este y que pueden dificultar más su detección y bloqueo. (How to prevent getting blacklisted while scraping)

1. **Realizar *scraping* de forma lenta:** programar el proceso para que su comportamiento sea más parecido al de un humano que al de un proceso automatizado. Realizar el proceso dividiendo todos los sitios web en grupos más reducidos y haciendo pausas entre las diferentes solicitudes.

2. **Realizar peticiones modificando la IP y usando servidores proxy:** utilizar diferentes direcciones IP para realizar las solicitudes para dificultar que el proceso sea detectado analizando únicamente la dirección IP de origen.

3. ***User-Agent Spoofing*:** cuando se realiza una solicitud mediante un navegador web, esta contiene una cabecera con información acerca de él. Rotar el tipo de navegador con el que se realiza la solicitud a medida que se ejecuta el proceso hace que sea más complicado detectarlo analizando las cabeceras acerca de la información del navegador.

4. **Vigilar con los *honeypots*:** algunos sitios web tienen instalado un sistema de *honeypots*, que consiste en crear un sistema falso para que se realice un ataque informático contra él y así poder obtener información acerca del atacante. Es complicado detectar este tipo de sistemas, pero es útil comprobar la visibilidad del enlace al que se quiere acceder analizando el código CSS. Algunos enlaces pueden tener la propiedad de estar ocultos o de tener un determinado color para mezclarse con el color de la página web y no ser visto. Analizar esta propiedad CSS puede ser de ayuda para detectar si el enlace al que se quiere acceder es un *honeypot* o no.

5. **No utilizar siempre el mismo patrón:** algunos sitios web tienen sistemas inteligentes que se dedican a analizar el comportamiento que tiene el usuario. Los robots usan una lógica muy específica y eso hace que sean más fáciles de detectar. Por ello es interesante añadir un comportamiento aleatorio en el proceso, como pueden ser diferentes movimientos de ratón o diferentes *clicks* en la página. Esta aleatoriedad puede hacer que el comportamiento del proceso automatizado sea más humano y que sea más complicado de detectar.

6. **Respetar el archivo robots.txt:** seguir las especificaciones que se encuentran en el archivo, ya que define las reglas que permiten tener un buen comportamiento sobre el sitio web. Entre estas reglas está la frecuencia con la que se permite a los procesos automatizados solicitar páginas, sobre qué páginas está permitido realizar *scraping* y sobre qué sitios de la página está permitido.

4.3.3. Web scraping en el proyecto

Mediante consultas a la API de DBLP podemos obtener mucha información acerca de los autores y de sus publicaciones. Sin embargo, por derechos de ley de protección de datos la API de DBLP no permite obtener los *abstracts* de las publicaciones[27].
Estos *abstracts* están publicados en diferentes plataformas digitales y DBLP únicamente proporciona las direcciones URL del sitio web en donde se encuentra una publicación.

En el proyecto se usa la técnica de *scraping* con dos objetivos concretos:

- **Obtener un conjunto de datos inicial de autores:** obtener un listado de autores con sus respectivas "urlpt" que forman parte de una determinada conferencia.

- **Obtener información acerca de las publicaciones:** obtener los *abstracts* de las publicaciones y otra información de interés que pueda ser útil.

[27] http://dblp.uni-trier.de/faq/Why+are+there+no+abstracts+in+dblp.html

4.3.4. Arquitectura del sistema

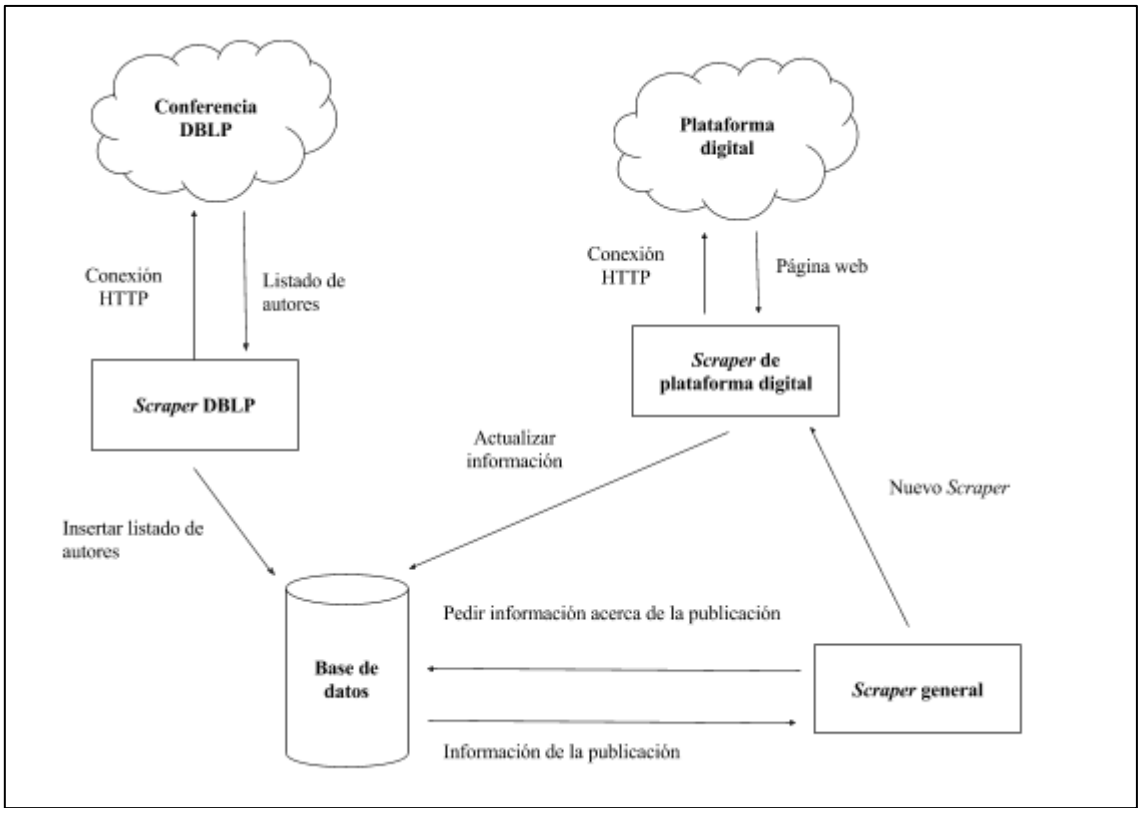


Figura 16 Arquitectura de scraping. Elaboración propia

En este diagrama se refleja el flujo de peticiones existentes entre el sistema y los diferentes sitios web a *scrapear*.

En primer lugar, se puede observar como existe un "DBLP Scraper" que realiza una petición HTTP a una URL de una conferencia almacenada en DBLP.
Mediante esta conexión, obtenemos el código fuente de la web para examinarlo y obtener un listado de todos los autores existentes para esa conferencia, para luego poder ser almacenado en la base de datos.

Justo al lado se muestra la arquitectura del sistema de *scraping* para obtener los datos de las plataformas digitales. Para ello, se puede diferenciar dos tipos de *scrapers*. Por un lado, un *scraper* general, que se encarga de obtener la información de una publicación y ver en qué plataforma digital está publicada e instanciar un nuevo *scraper* para ese tipo en concreto de plataforma. Por otro lado, el que se encarga de realizar una conexión HTTP al sitio web de destino y obtener el código fuente para poder realizar un parseo y obtener la información relevante necesaria para después actualizar la información de la publicación en la base de datos.

4.3.5. Implementación del sistema

Para implementar el sistema de *scraping* se han llevado a cabo algunas de las buenas prácticas detalladas en el punto 4.4.3. A continuación, se detallan los puntos más significativos acerca de la implementación.

1. ***Multithreading*:** debido a que se trata con una cantidad elevada de publicaciones y que el proceso de *scraping* es un proceso lento, se ha utilizado el concepto de trabajar con hilos (threads). Esto permite poder crear distintas unidades pequeñas de ejecución que se pueden ejecutar en paralelo.

2. **Obtener la información en pequeños grupos y esperar:** debido al control que tienen las plataformas ante estas técnicas, es importante no hacer excesivas conexiones en poco tiempo. Se crean grupos pequeños de 20 hilos para obtener los datos de las publicaciones, se ejecutan y cuando han finalizado, el proceso de *scraping* espera 20 segundos antes de ejecutar los siguientes 20 hilos. Si la técnica de *scraping* se utiliza para un conjunto de autores, para cada autor el proceso se ejecuta de forma secuencial pero después de obtener todos los datos de todas las publicaciones para 50 autores, se espera 30 segundos antes de volver a ejecutar el siguiente grupo.

3. **Usar diferentes *user-agent*:** utilizar un navegador aleatorio entre un conjunto de navegadores para dificultar que las plataformas digitales detecten el proceso.

4. **Uso del patrón *Factory*:** debido a que se utilizan diferentes plataformas digitales como objetivo de análisis, se ha diseñado el sistema con el patrón de diseño Factory. Este patrón de diseño permite crear fácilmente instancias de un objeto sin mostrar al cliente como este objeto es creado y referirnos a los diferentes objetos usando una única interfaz. Esto significa que el cliente puede crear un objeto referido a dos plataformas digitales diferentes y acceder a ellos de la misma manera, sin tener conocimiento de la lógica interna que existe.

4.3.6. Plataformas digitales

A continuación se muestra una tabla detallando qué plataformas digitales han sido analizadas a lo largo del proyecto para poder obtener la información acerca de las publicaciones y qué información puede ser extraída en cada una de ellas.

	ACM	Springer Link	Online Library	IEEE Explore
Abstract	Sí	Sí	Sí	Sí
Palabras clave	Sí	Sí	Sí	Sí
Etiquetas de autor	Sí	No	No	No
Número de citas	Sí	Sí	Sí	Sí
Número de descargas	No	Sí	No	No

Tabla 3 Plataformas digitales (I)

	Computer Society	AAIII	DROPS	IOS Press	Inder Science	Science Direct
Abstract	Sí	Sí	Sí	Sí	Sí	No
Palabras clave	Sí	Sí	Sí	Sí	No	No
Etiquetas de autor	No	No	No	No	No	No
Número de citas	No	No	No	No	No	Sí
Número de descargas	No	No	No	No	No	No

Tabla 4 Plataformas digitales (II)

Cómo se ha detallado anteriormente, cabe destacar que el *scraping* en las plataformas digitales sólo es utilizado para obtener los *abstracts*, ya que otro tipo de información como el número de citas ha sido obtenido mediante otros métodos.

4.3.7. Tecnologías utilizadas

- **Jsoup[28]:** es una librería desarrollada en Java que permite trabajar con el contenido de una página en HTML. Mediante una API permite conectarse a un sitio web, obtener el código HTML, parsearlo, manipular los datos de los diferentes elementos y extraer información utilizando DOM o selectores CSS.

Sin embargo, no todas las plataformas digitales pueden ser *scrapeadas* con facilidad mediante la librería Jsoup. Contenido web que, por ejemplo, esté sujeto a una rutina en Javascript es más complicado de obtener. Otro problema existente con esta librería es que no podemos acceder al contenido de interés si este está desarrollado con AngularJS[29]. Por ello, se ha utilizado PhantomJS.

- **PhantomJS[30]:** es un navegador sin interfaz gráfica desarrollado bajo el motor WebKit de Javascript y una herramienta muy usada en tareas de *testing* y automatización. Es posible configurar scripts que carguen páginas completas e interactuar totalmente con ellas. De esta forma, PhantomJS permite solucionar el problema existente con Jsoup.

El *scraping* se ha desarrollado con una combinación de ambas tecnologías, ya que cargar el contenido web en Jsoup es más rápido que cargarlo en PhantomJS, ya que este necesita esperar a que se cargue el Javascript existente y Jsoup no.

No todas las plataformas están desarrolladas bajo frameworks en Javascript o no en todas hay que esperar a que se cargue el Javascript para acceder totalmente al contenido, así que se usa Jsoup en aquellas en las que no es necesario esperar al Javascript y PhantomJS en las que sí.

Sin embargo, cabe destacar que PhantomJS es una herramienta mucho más potente que Jsoup.

[28] https://jsoup.org/
[29] https://angularjs.org/
[30] http://phantomjs.org/

4.3.8. Estadísticas

En este punto se puede observar un gráfico donde se muestra la cantidad de *abstracts* que se han recuperado de las publicaciones usando las técnicas de *web scraping* descritas en las diferentes plataformas digitales mencionadas en este capítulo.

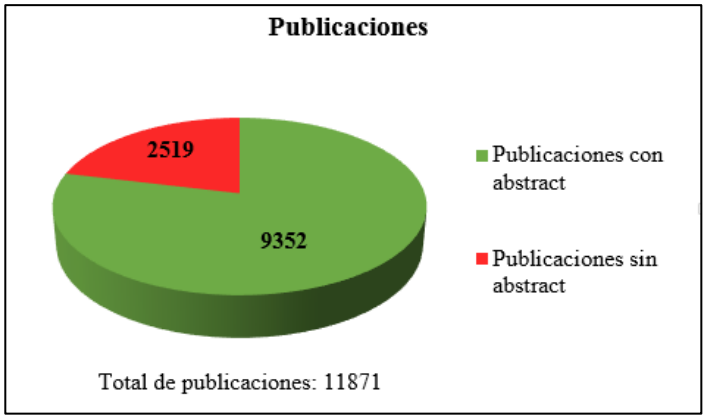


Figura 17 *Abstracts* obtenidos. Elaboración propia

La cantidad de documentos con la que se ha trabajado es de 11.871 y se han podido obtener un total de 9.352 *abstracts*, que equivale a un 78,78% del total de publicaciones. Por lo tanto, un 21,22% de publicaciones son las que no se han podido recuperar. Puede haber varios motivos que hayan producido que no se hayan podido recuperar, como por ejemplo que DBLP no proporcione la dirección de acceso donde se puede encontrar el *abstract* en la plataforma digital o que la plataforma digital no ofrezca información acerca de él. Sin embargo, el motivo más común entre todas las publicaciones que forman parte de este 21,22% es que la información acerca de los *abstracts* puede ser encontrada en una plataforma digital que no ha sido explorada y que no está contemplada en el sistema.

5. PROCESAMIENTO DE TEXTO

El procesamiento de texto (Witten, 2005) es una de las áreas pertenecientes al campo de la ingeniería de procesamiento de lenguaje natural y se caracteriza por recopilar información mediante la identificación de diferentes patrones lingüísticos acerca de un documento o varios documentos mediante un proceso de análisis de la estructura de estos.

Mediante herramientas enfocadas en este tipo de procesamiento es posible desarrollar sistemas que permitan recuperar determinada información contenida en un documento, comparar diferentes de ellos, crear resúmenes automáticos de su contenido o facilitar la tarea de realizar una bibliografía creando una red de citaciones bibliográficas.

5.1. Procesamiento de texto en el proyecto

Una vez se ha obtenido información acerca de las diferentes publicaciones, es necesario utilizar diferentes herramientas de procesamiento de texto para poder enriquecer la información y poder realizar una serie de representaciones de esta que sea útil para el usuario.

El análisis se ha basado en obtener características de interés acerca de los títulos y *abstracts* obtenidos para las publicaciones y se ha realizado para la implementación de los siguientes puntos:

- **Cálculo de similitudes entre publicaciones:** obtener la similitud entre dos documentos a partir del peso normalizado de las palabras del título y el *abstract* de cada una de las publicaciones y mediante distintos tipos de medidas de cálculo de similitudes.

- **Nube de términos para el *abstract* de las publicaciones:** obtener las palabras más relevantes de una publicación mediante el peso normalizado de las palabras contenidas en el *abstract*.

- **Nube de términos para un autor:** obtener las palabras más relevantes entre todas las publicaciones realizadas por el autor mediante el peso normalizado de todas las palabras de los diferentes *abstracts* publicados.

- **Obtener la categoría retórica de una oración:** categorizar cada una de las oraciones de un *abstract* y poder seleccionar aquellas frases que compartan la misma categoría.

Para poder realizar este tipo de análisis se han utilizado las siguientes herramientas:

- **GATE:** herramienta de procesamiento de lenguaje natural que permite obtener diferente información acerca del texto contenido en un documento.

- **Dr. Inventor:** herramienta desarrollada en Java y que forma parte de un proyecto Europeo que contiene diferentes módulos y servicios para el procesamiento de texto.

- **SUMMA**: herramienta de procesamiento de texto compatible con el entorno GATE que permite obtener estadísticas acerca de las diferentes palabras y oraciones contenidas en un documento y permite realizar resúmenes automáticos de texto.

Funcionalidad	Herramienta
Cálculo de similitudes	GATE + SUMMA
Nube de términos para publicaciones	GATE + SUMMA
Nube de términos para autores	GATE + SUMMA
Categorización de una frase	Dr. Inventor

Tabla 5 Funcionalidades y herramientas

5.2. GATE

Es la herramienta de procesamiento de lenguaje natural que más se ha utilizado en el proyecto.

GATE dispone de un conjunto de *plugins* que contienen diferentes recursos de procesamiento del lenguaje que pueden ser configurados para ajustar la información que puede ser extraída en cada documento.

Para poder obtener diferentes características acerca del texto, se han creado diferentes aplicaciones en GATE, donde cada una tiene un conjunto de recursos de procesamiento que permiten procesar y obtener diferentes características del texto.

Es posible guardar de forma externa el estado de estas aplicaciones, en formato *gapp*, para poder importarlas, ya sea en otra interfaz gráfica de GATE o utilizando las librerías de GATE que permiten que se pueda programar en lenguaje Java.

5.2.1. Uso en el proyecto

El uso de GATE tiene como objetivo aplicar los diferentes recursos lingüísticos disponibles para obtener información acerca de las palabras contenidas en un documento XML y poder usarlas en las diferentes implementaciones que se han descrito anteriormente.

Para ello se han utilizado los siguientes recursos de procesamiento para poder realizar los diferentes análisis lingüísticos.

Recurso	Descripción
Document Reset	Permite restablecer el documento al estado original antes de ser procesado, eliminando todo el conjunto de anotaciones que pueda y procesarlo como si fuera la primera vez.
English Tokeniser	Permite dividir el texto en unidades de texto más pequeñas como por ejemplo números, símbolos de puntuación, palabras o espacios. También tiene un "*JAPE transducer*", que permite reconocer expresiones regulares sobre las anotaciones de los documentos y permite adaptar los resultados genéricos del *Tokeniser* a los requisitos necesarios para poder realizar un etiquetado gramatical de las palabras.
Sentence Splitter	Permite dividir el texto en diferentes oraciones. Este componente es necesario para poder realizar un etiquetado gramatical.
POS Tagger	Permite etiquetar gramaticalmente cada una de las palabras o símbolos de un texto.
GATE Morphological Analyzer	Mediante un documento GATE tokenizado, obtiene el lema y un afijo de cada una de las unidades de texto obtenidas mediante English Tokeniser.
SUMMA NEs Statistics[31]	Añade a cada *Token* el número de veces que aparece en el documento, la frecuencia invertida y la relevancia del término.
Annotation Set Transfer	Permite copiar o mover anotaciones existentes en un nuevo grupo de anotaciones.
SUMMA Vector Computation[32]	Crea un vector para las anotaciones de un documento que contiene un mapa de términos y sus pesos.
SUMMA Normalize Vector[33]	Normaliza los datos del vector entre 0 y 1.

Tabla 6 Recursos utilizados

[31] http://www.taln.upf.edu/pages/summa.upf/SUMMA_NEs_Statistics.html
[32] http://www.taln.upf.edu/pages/summa.upf/SUMMA_Vector_Computation.html
[33] http://www.taln.upf.edu/pages/summa.upf/SUMMA_Normalize_Vector.html

5.2.2. Arquitectura del sistema

Se puede dividir la arquitectura del sistema en dos partes: crear el conjunto de documentos para cada publicación y ejecutar una aplicación gapp para obtener los datos acerca de los documentos.
A continuación se detalla cada una de estas arquitecturas.

1. **Crear conjunto inicial de documentos:** se basa en obtener un listado de publicaciones de la base de datos mediante un programa en Java. Seguidamente se obtiene el título y el *abstract* de cada una de ellas y se genera un documento XML (punto 5.3.4). Una vez el documento es creado se almacena a una carpeta local del sistema para que pueda ser procesado posteriormente.

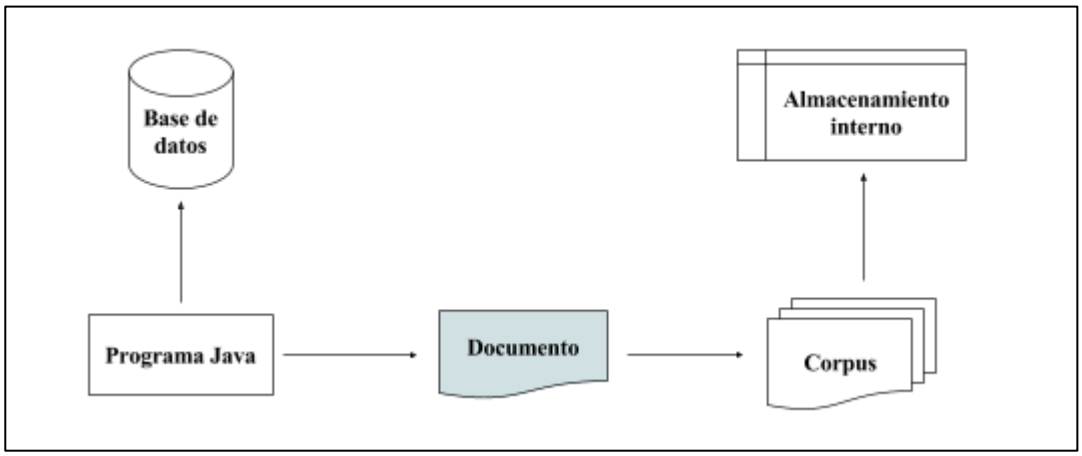


Figura 18 Arquitectura para crear un conjunto inicial de documentos. Elaboración propia

2. **Procesar documentos:** se basa en obtener un conjunto de documentos almacenados localmente para procesarlos mediante una gapp y obtener las características deseadas. Una vez se ha obtenido el resultado, este es almacenado en la base de datos y los nuevos documentos obtenidos después de ser procesados son almacenados localmente para poder ser usados más adelante.

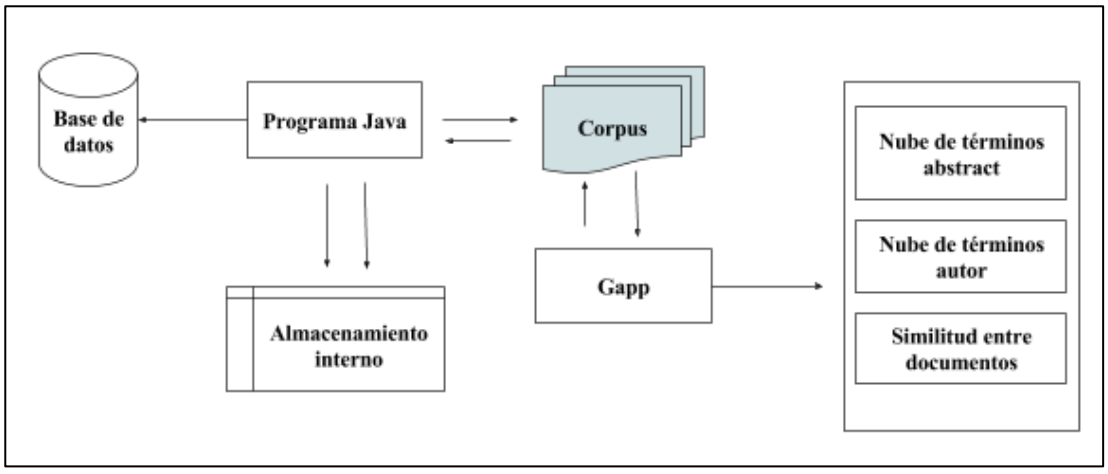


Figura 19 Arquitectura para procesar documentos. Elaboración propia

5.2.3. Estructura del Corpus

Los documentos contenidos en el Corpus están en formato XML y son creados mediante un programa en Java.

El programa en Java, como se muestra en el primer diagrama de arquitectura, se conecta a la base de datos, obtiene los datos de las publicaciones deseadas y crea un documento o conjunto de documentos para que su información sea almacenada forma local.

Se puede diferenciar entre dos estructuras diferentes:

1. **Documentos para publicaciones:** este tipo de documento se utiliza para almacenar individualmente el título y el *abstract* de cada una de las publicaciones que pertenecen al Corpus. La estructura XML del documento es la siguiente:

```
<?xml version='1.0' encoding='UTF-8'>
<doc>
    <title> </title>
    <abstract> </abstract>
</doc>
```

Figura 20 XML para publicaciones

2. **Documentos para autores:** este tipo de documentos se utiliza para almacenar todos los *abstracts* juntos de cada autor. La estructura XML del documento es la siguiente:

```
<?xml version='1.0' encoding='UTF-8'>
<doc>
    <abstract> </abstract>
</doc>
```

Figura 21 XML para autores

Hay que tener en cuenta que hay una serie de caracteres reservados para XML. En el momento de procesar los datos para generar los documentos para añadir al corpus, hay que mapear los caracteres reservados con su valor correcto para que sean interpretados como entidades XML[34].

[34] https://www.tutorialspoint.com/xml/xml_character_entities.htm

5.2.4. Aplicaciones GATE

Una aplicación GATE (gapp) es un conjunto de recursos lingüísticos ejecutados secuencialmente sobre un documento o un conjunto de documentos y que se encuentran contenidos dentro de un "Corpus Pipeline", que es el tipo de aplicación que permite ser ejecutada sobre un Corpus.
Es posible exportar en un archivo el estado de una determinada gapp para ser importada más adelante de nuevo, ya sea en una interfaz GATE o mediante un programa desarrollado en Java.

Se utilizan diferentes gapps para obtener la diferente información de interés en los documentos.

A continuación se detalla cada una de ellas y su estructura:

1. **TFG_IDFs:** esta gapp es usada para obtener una tabla de idf de todo el conjunto de publicaciones pertenecientes a un Corpus.

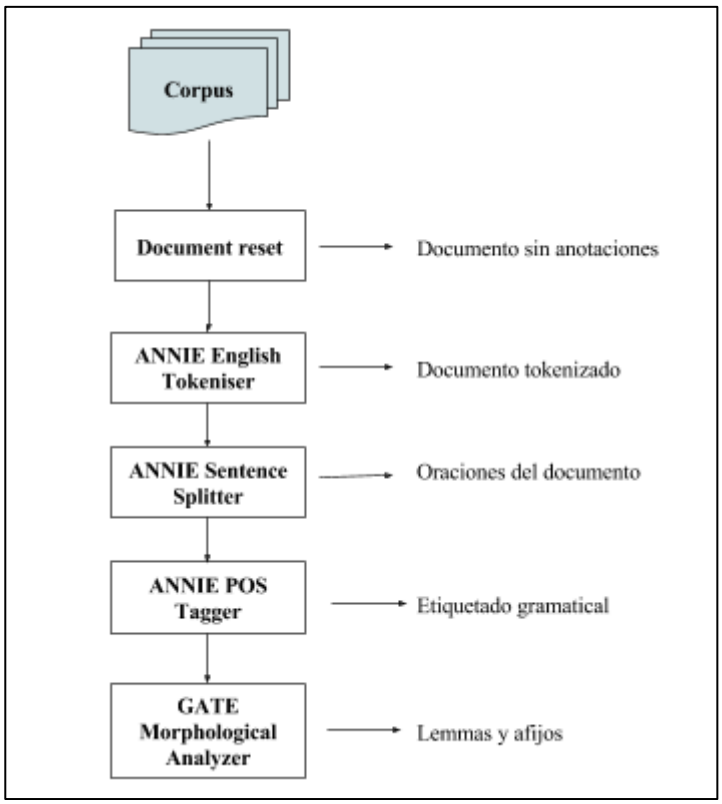


Figura 22 Componentes gapp 'TFG_IDFs'. Elaboración propia

2. **Transfer_and_Vecs:** esta gapp está contenida dentro de la "TFG_statistics" y es usada para crear una anotación para título y otra para *abstract* copiando el contenido que de estas dos anotaciones de la sección "*Original markups*". Los componentes para calcular el peso de las palabras del documento se encuentran dentro de esta gapp.

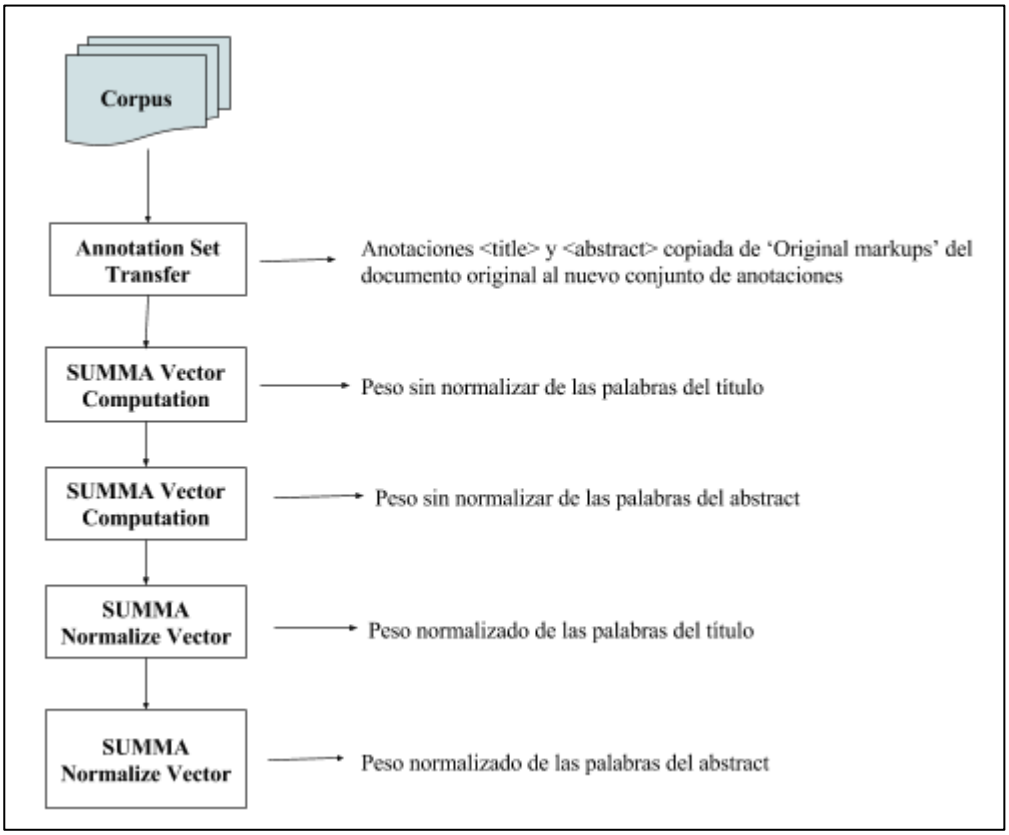


Figura 23 Componentes gapp 'Transfer_and_Vecs'. Elaboración propia

3. **TFG_statistics:** esta gapp es usada para obtener el peso que tiene cada una de las palabras en el título o en el *abstract* según donde se encuentren. Se obtiene el peso de la palabra de forma normalizada y sin normalizar.

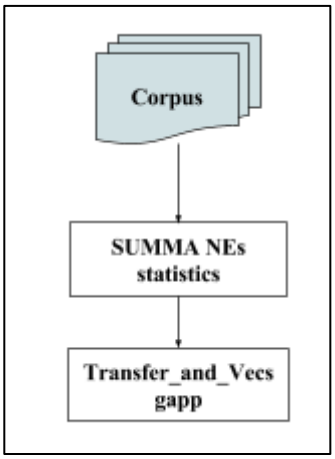


Figura 24 Componentes gapp 'TFG_statistics'. Elaboración propia

4. **Author_Abstracts:** esta gapp es usada para obtener el peso de una palabra dentro del conjunto de *abstracts* de un autor. Se obtiene el peso de la palabra de forma normalizada y sin normalizar.

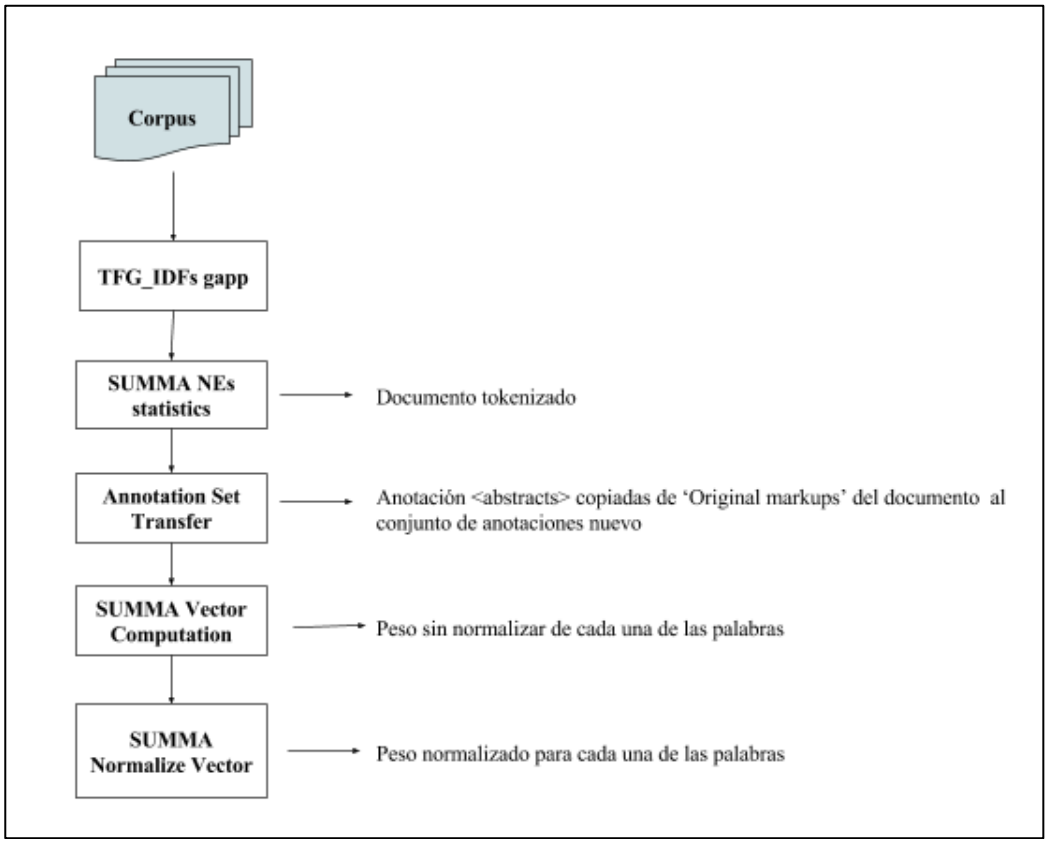


Figura 25 Componentes gapp 'Author_Abstracts'. Elaboración propia

5.2.5. Implementación de las funcionalidades

Las gapps descritas en el apartado anterior permiten obtener la información necesaria para poder realizar las diferentes implementaciones, pero, ¿cómo han sido configuradas? En este apartado se procede a explicar los puntos más importantes de la configuración de los diferentes recursos lingüísticos utilizados y las técnicas utilizadas para poder realizar las diferentes implementaciones.

5.2.5.1. Obtener tabla IDF

Para poder ejecutar las estadísticas y obtener el peso de las palabras, es necesario poder generar una tabla de IDF para poder ser usada en la ejecución de la gapp "TFG_statistics".

Una tabla de IDF es una tabla que contiene un documento con extensión *.idf*.
Este documento tiene la siguiente estructura:

```
11871
undermining 1
conformed   1
k-svd   2
unavailable 22
gps-equipped    4
gskeletonclu    1
multi-viewpoint 1
abrupt  3
descriptor  61
salary  1
pretend 1
cubically   1
conformer   1
snow-blower 1
dfaw    1
esair   1
down-sampled    1
dfat    1
```

Figura 26 Tabla IDF

Esta tabla es creada mediante la ejecución de la gapp 'TFG_IDFs'. Sin embargo, hay un problema importante en este punto, y es que se está trabajando con una cantidad elevada de documentos, tal y como se ha visto en el gráfico del punto 4.4.9. Para trabajar con una cantidad como esta de documentos es necesario que la máquina que se encarga de ejecutar las gapps tenga recursos necesarios para poder procesar todos los documentos. En este caso, se ha optado por ejecutar todos los documentos en bloques de 1000 publicaciones, lo que supone tener 11 documentos idf diferentes.

Para poder obtener un único documento, se ha creado un proceso en Java que permite leer todos los registros de los documentos y crear un documento nuevo con la unión de todos los datos.

Cada palabra leída es almacenada en un *HashMap*, donde la clave de este es la palabra y su valor es el número de veces que aparece en los documentos.

Para poder actualizar este *HashMap* correctamente se utiliza el siguiente criterio:

- Si la palabra leída existe como clave, obtenemos el valor de las veces que la palabra aparece en el documento que se está leyendo actualmente y se suma al valor que ya tiene en el *HashMap*.

- Si la palabra leída no existe como clave, añadimos una clave nueva con esta palabra y se almacena como valor el que se está leyendo actualmente.

Para obtener el total de documentos que se han procesado se va acumulando el valor obtenido en la primera línea de todos los documentos que son leídos.
Finalmente, se crea un documento nuevo donde en la primera línea se escribe el valor del total de documentos leídos, y a continuación se escribe en cada línea cada clave del *HashMap*, una tabulación, y su valor.

En las siguientes imágenes se muestra el código Java de lo explicado anteriormente.

```
Map< String , Integer >  words = new HashMap< String, Integer >();

//Recorrer todos los ficheros
for( int i=0; i < flist.length; i++)
{
    try (BufferedReader br = new BufferedReader( new FileReader( flist[i].getAbsolutePath() ) ) )
    {
        System.out.println( flist[i].getName() );
        int fileDocuments = Integer.parseInt( br.readLine() ); //Leer la primera linea para obtener el número de documentos analizados
        totalDocuments += fileDocuments;

        //Recorrer hasta el final del documento
        while ((sCurrentLine = br.readLine()) != null)
        {
            key = sCurrentLine.split("\\t")[0];
            value = Integer.parseInt( sCurrentLine.split("\\t")[1] );

            //Comprobar si la palabra ha sido añadida al map
            if( !words.containsKey( key ) )
                words.put( key , value);
            else
                words.put( key , words.get( key ) + value );
        }

    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Figura 27 Crear tabla IDF (I). Elaboración propia

```
//Guardar el contenido del map en un documento
try
{
    File fout = new File( documentsLocation + "\\tfg_idf_total.idf" );
    FileOutputStream fos = new FileOutputStream(fout);
    System.out.println( totalDocuments );
    BufferedWriter bw = new BufferedWriter(new OutputStreamWriter(fos));
    bw.write( String.valueOf( totalDocuments ) );
    bw.newLine();
    Object[] keys = words.keySet().toArray();
    for( Object objKey: keys)
    {
        bw.write( objKey + "\t" + words.get( objKey ) );
        bw.newLine();
    }

    bw.close();
}
catch(Exception e) { e.printStackTrace( ); System.out.println( "Error when the idf file has been created"); }
```

Figura 28 Crear tabla IDF (II). Elaboración propia

5.2.5.2. Procedimiento general

En cada una de las implementaciones es necesario realizar un procedimiento común. Este se basa en obtener los documentos internos almacenados con las anotaciones GATE correspondientes después de ejecutar las aplicaciones gapp necesarias desde el programa Java para poder realizar un parseo de estos documentos y obtener el peso normalizado de cada una de las palabras.

A continuación se muestra un tipo de documento GATE con las anotaciones obtenidas.

```
<Annotation Id="444" Type="Vector_Tit_Norm" StartNode="2" EndNode="74">
<Feature>
  <Name className="java.lang.String">fostering</Name>
  <Value className="java.lang.String">0.23940764117268193</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">dictionary</Name>
  <Value className="java.lang.String">0.4164563716752003</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">sti-dico</Name>
  <Value className="java.lang.String">0.6847068287539887</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">skill</Name>
  <Value className="java.lang.String">0.43174221131224</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">web-</Name>
  <Value className="java.lang.String">0.17737640202307192</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">base</Name>
  <Value className="java.lang.String">0.05922514758078182</Value>
</Feature>
<Feature>
  <Name className="java.lang.String">knowledge</Name>
  <Value className="java.lang.String">0.2811641644440437</Value>
</Feature>
</Annotation>
```

Figura 29 Documento GATE anotado

En esta imagen se muestra un ejemplo de una anotación "*Vector_Tit_Norm*" y se observa la relación de cada una de las palabras del *abstract* con su peso normalizado. Si se compara el resultado que se ha almacenado en el documento con el resultado que se obtiene utilizando la interfaz de GATE, podemos observar como se obtienen los mismos valores.

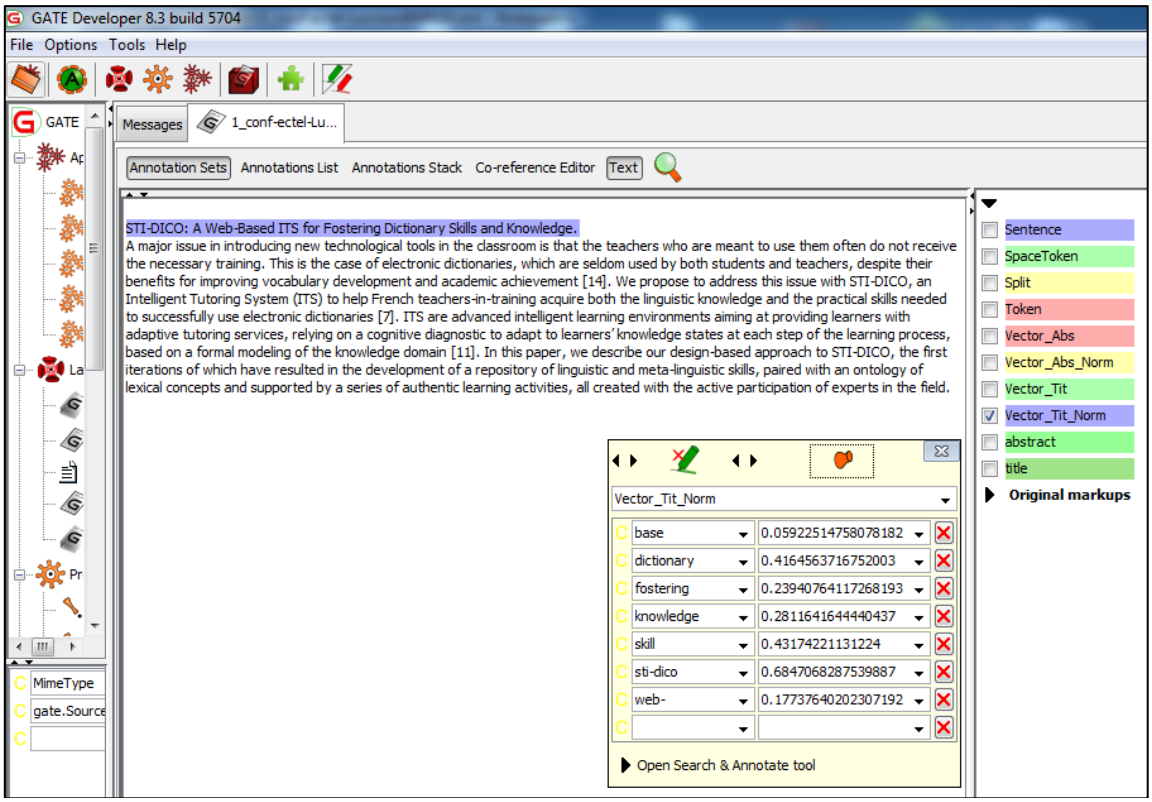


Figura 30 Anotaciones en interfaz GATE

5.2.5.3. Cálculo del peso de una palabra

Para obtener el peso que tiene una palabra en un conjunto de documentos se usan los recursos lingüísticos "*SUMMA Vector Computation*" y "*SUMMA Normalize Vector*" para poder normalizar estos peso.

Gracias a la tabla IDF, estos recursos pueden calcular rápidamente el peso para cada una de las palabras.

Uno de los métodos más usados para calcular la relevancia de un término es el sistema TF-IDF (Frecuencia de término, frecuencia inversa de documento)[35].

Así pues, podemos definir la relevancia de un término "t" como: $relevance(t) = tf * idf$, dónde:

- **tf:** distribución del término en el documento, es decir, cuántas veces aparece la palabra en el documento.

[35] https://www.upf.edu/hipertextnet/en/numero-5/pln.html

- **idf:** distribución del término en el Corpus, es decir, cuántas veces aparece la palabra en el conjunto de documentos. Esta medida indica si el término es muy común o no y se puede calcular como:

$$idf(t) = \log\left(\frac{\text{NUMDOC}}{\text{NUMDOC(t)}}\right)$$

$\boldsymbol{NUMDOC}$: *número de documentos*
$\boldsymbol{NUMDOC(t)}$: *número de documentos en los que el término aparece*

5.2.5.4. Nube de términos

Una nube de términos es una representación gráfica en forma de nube de diferentes palabras en la que se pretende destacar la importancia de estas según el tamaño con el que están representadas dentro de la nube.
A continuación se puede observar un ejemplo sobre cómo es una nube de términos.

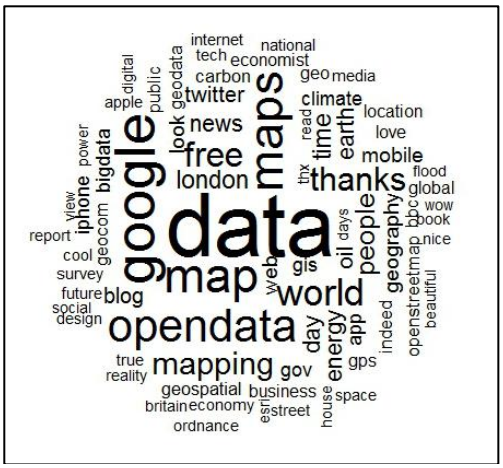


Figura 31 Ejemplo de nube de términos[36]

Para calcular la nube de términos es necesario filtrar sólo por aquellos "*Tokens*" que tengan una categoría gramatical de interés para ver si contextualmente aporta suficiente información para que en la nube de términos aparezca como una de las palabras más relevantes. Por ejemplo, un adverbio puede aparecer como una de las palabras más relevantes en un documento pero en este caso, esta categoría gramatical no aporta suficiente información en la nube de términos, ya que no proporciona información acerca de, por ejemplo, sobre qué tópicos puede hacer referencia un documento.

[36] https://georeferenced.wordpress.com/2013/01/15/rwordcloud/

Para generar la nube de términos se ha decidido filtrar por las categorías gramaticales que hacen referencia a sustantivos, ya que se refiere a términos más concretos y pueden encajar mejor para intuir sobre a qué tópico hace referencia un documento. (Part-of-Speech Tags used in the Hepple Tagger)

- **NNP:** Nombre propio en singular
- **NN:** nombre en singular
- **NNS:** nombre en plural

El procedimiento para poder calcular esta nube es el siguiente:

1. Obtener las diferentes entradas etiquetadas como "*Feature"* de la anotación que interesa en el documento GATE. Por ejemplo, para los *abstracts* de un documento, existe una anotación llamada "*Vector_Abs_Norm*".

2. Seguidamente se recorren todas las *features* obtenidas y se comprueba si la categoría gramatical para cada una pertenece a una de las anteriores. Si es así, se obtiene el atributo "*root*" y el atributo "*weight_norm*" que hacen referencia al lema de la palabra y al peso normalizado de esta.

5.2.5.5. Cálculo de similitudes entre documentos

Para calcular la similitud entre dos documentos se utilizan dos técnicas diferentes:

- **Distancia coseno:** esta medida de similitud trata de representar las palabras en un espacio vectorial. Cada conjunto de palabras es un vector y se trata de calcular el valor del coseno del ángulo entre ellos.

  Sean I y J dos vectores de palabras, la distancia coseno entre ellos se calcula de la siguiente manera:

$$Sim(I,J) = \frac{\sum(w_i * w'_j)}{\sqrt{\sum {w_i}^2} * \sqrt{\sum {w_j}^2}}$$

  El numerador es la suma de la multiplicación de los pesos de aquellas palabras que están contenidas en los dos vectores y el denominador es la raíz cuadrada de la suma del peso elevado al cuadrado de cada una de las palabras de los vectores.

- **Similitud *Jaccard*:** mide el grado de similitud entre dos conjuntos en función de la unión y la intersección de estos dos. Se calcula de la diferente manera:

$$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$

$|A \cap B|$ es el tamaño de la intersección, es decir, cuantas palabras comparten los dos conjuntos y $|A \cup B|$ es el tamaño de la unión, es decir, el total de palabras diferentes que hay en los dos conjuntos. (Jaccard Index / Similarity Coefficient)

Por lo tanto, se puede observar que en esta medida, como más palabras comparten los dos conjuntos, más parecidos son.

5.3. Dr. Inventor

Es una librería realizada en Java que permite el análisis textual de diferentes publicaciones científicas y permite ayudar a investigadores o a cualquier usuario interesado a extraer y estudiar la diferente información contenida en las publicaciones, como por ejemplo su estructura o la desambiguación lingüística.

Dr. Inventor permite extraer y caracterizar diferentes aspectos de un documento, como por ejemplo:

- **Elementos estructurales**: permite obtener diferentes secciones del documento, como el título, *abstract*, entradas bibliográficas o las oraciones que pertenecen a una sección concreta.

- **Bibliografía:** permite obtener las entradas bibliográficas accediendo a servicios web externos, como por ejemplo, Bibsonomy y Google Scholar.

- **Categorización de oraciones:** permite clasificar las oraciones contenidas en el documento en diferentes categorías.

- **Selección de las oraciones más relevantes:** permite seleccionar las oraciones más relevantes de un documento mediante la extracción de información contenida en resúmenes de este realizados de forma automática.

Para poder utilizar correctamente esta librería, es necesario configurar un documento de propiedades y se necesita tener una cuenta en BabelNet[37], que es un recurso multilingüe que puede ser utilizado como un diccionario enciclopédico o como una gran red semántica que relaciona diferentes conceptos y entidades lingüísticas.

[37] http://babelnet.org/about

5.3.1. Dr. Inventor en el proyecto

Esta librería ha sido utilizada para obtener la categoría retórica de las oraciones de un documento, con el objetivo de obtener el conjunto de frases que pertenecen a una determinada categoría y de esta forma poder segmentar más el documento y obtener una visualización únicamente de las partes del texto que sean de más interés. Sin embargo, segmentar las oraciones del texto de esta forma podría servir para, por ejemplo, obtener la similitud de documentos únicamente comparando las oraciones que pertenecen a un cierto tipo de categoría.

Es posible categorizar una oración de la siguiente forma:

- **Background:** información necesaria para ayudar a contextualizar el documento.
- **Approach:** introducción acerca del tema en el que se va a tratar a lo largo del documento.
- **Challenge:** información acerca del objetivo o motivación sobre el tema que trata el documento.
- **Future Work:** información acerca de las pretensiones de futuro que se explican en el documento.
- **Outcome:** información acerca de los resultados obtenidos descritos en el documento.
- **Sentence:** oraciones que no es posible usar para generar anotaciones debido a que no están correctamente segmentadas.
- **Unspecified:** información acerca de la estructura del documento.

5.3.2. Implementación

En este caso, este apartado se divide en dos partes: inicialización de la librería e implementación de la categorización retórica de las frases.

5.3.2.1. Inicialización de la librería

Primero es necesario configurar correctamente el entorno de trabajo para poder implementar y ejecutar satisfactoriamente la categorización de frases. Para ello, hay que añadir la librería a un proyecto Java y configurar el archivo de propiedades añadiendo los siguientes parámetros:

Parámetro	Significado
resourceFolder.fullPath	Ubicación del directorio que contiene los recursos de Dr.Inventor
connector.bibsonomy.userid	Id de usuario de Bibsonomy[38]
connector.bibsonomy.apkey	Clave utilizada en Bibsonomy
Babelnet.APIkey	Clave para el acceso a BabelNet

Tabla 7 Archivo de propiedades Dr. Inventor

A continuación, hay que especificar un argumento Java VM indicando que se habilita se puede usar un espacio de memoria igual o superior a 5GB.

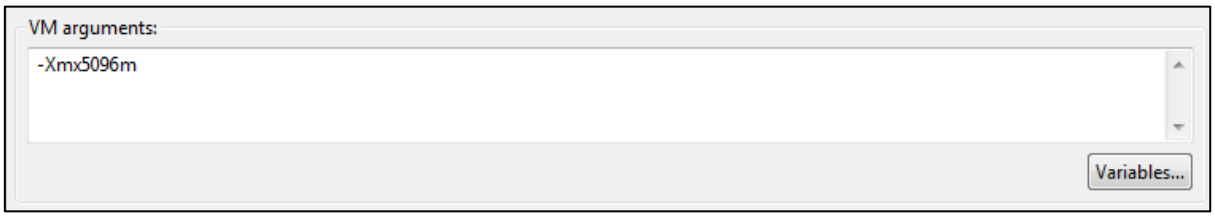


Figura 32 Argumentos VM para el uso de Dr. Inventor en Java

Llegados a este punto, solo falta indicar en qué directorio se encuentra el archivo de propiedades configurado previamente.

```
Factory.setDRIPropertyFilePath("C:\\DrInventor\\drinventor.properties");
```

Figura 33 Configurar el archivo de propiedades. Elaboración propia

5.3.2.2. Categorización retórica de las oraciones

Dr. Inventor acepta como entrada diferentes tipos de documento que pueden ser procesados, como documentos en formato PDF, JATS XML o TXT.

El formato que se ha utilizado para crear los diferentes documentos a procesar es el formato *txt*, ya que el tipo de documentos que se quiere procesar es muy simple porque únicamente contienen el *abstract* de la publicación. Por lo tanto, el primer paso es crear un documento con formato *txt* para cada una.

[38] https://www.bibsonomy.org/

Seguidamente, es necesario inicializar el módulo de Dr Inventor que se encarga de obtener las categorías de las frases de la siguiente manera:

```
ModuleConfig modConfigurationObj = new ModuleConfig();
modConfigurationObj.setEnableRhetoricalClassification(true);
Factory.setModuleConfig(modConfigurationObj);
```

Figura 34 Configuración de módulos. Elaboración propia

Una vez el documento es creado, se crea un nuevo documento en Dr Inventor y se extraen todas las frases que contiene y para cada una de ellas se extrae la siguiente información para ser almacenada en la base de datos:

- **Id:** identificador que la librería proporciona a la frase extraída
- **Inicio:** posición de inicio de la frase en el documento.
- **Fin:** posición en donde termina la frase en el documento.
- **Texto:** texto que contiene la frase.
- **Categoría:** categoría retórica de la frase.

Estos datos son almacenados en la base de datos porque el proceso de parseo de un documento es costoso en cuanto a recursos del sistema y así, con hacerlo una sola vez es suficiente para, posteriormente, poder obtener los datos de forma muy rápida. Si no se hiciera de esta forma, siempre que el usuario desee consultar la categorización de las frases de un documento debería esperar todo el tiempo que el sistema necesite para procesar el documento.

6. VISUALIZACIÓN

Como se ha descrito previamente, la visualización de los datos se ha realizado mediante archivos JSP combinando código HTML, CSS y jQuery con código en Java. De esta forma, podemos tratar código en el servidor y utilizar el cliente para mostrarlo.

6.1. Tecnologías utilizadas

Para la visualización de los datos, parte de las tecnologías detalladas en el punto 3.1.5 son utilizadas. Sin embargo, se han utilizado también otras más específicas para enriquecer la visualización de los datos.

- **D3.js**[39]**:** librería Javascript que permite visualizar datos utilizando HTML, SVG y CSS. Permite interactuar y animar datos que pertenecen a conjuntos de datos muy grandes de forma muy rápida y eficiente. Ha sido utilizada para crear los diferentes grafos de similitudes entre publicaciones y autores mostrando el grafo de la mejor forma posible en función de los datos[40].

- **Chart.js**[41]**:** librería Javascript que permite visualizar los datos en diferentes tipos de gráficos y cargarlos en un objeto *<canvas>* de HTML5. Ha sido utilizada para crear los gráficos que muestran las publicaciones y las colaboraciones en función de los años.

- **Zingchart**[42]**:** librería Javascript que permite visualizar los datos en diferentes tipos de gráficos. Ha sido utilizada para visualizar los términos más relevantes con su respectivo peso tanto para los autores como para las publicaciones en un formato de nube.

[39] https://d3js.org/
[40] https://bl.ocks.org/mbostock/4062045
[41] http://www.chartjs.org/
[42] http://www.zingchart.com

6.2. Resultado de visualización

En este apartado se muestran algunas imágenes acerca de los resultados obtenidos utilizando las tecnologías detalladas en el punto anterior y de los datos procesados con Dr. Inventor.

Listado y comparación de autores: esta funcionalidad se puede encontrar en la página principal, en la que se muestra un listado de los autores y el usuario tiene la opción de seleccionar uno y acceder a la página donde se muestra su información o seleccionar varios autores con el fin de obtener datos acerca de la similitud de sus publicaciones y de la similitud entre ellos.

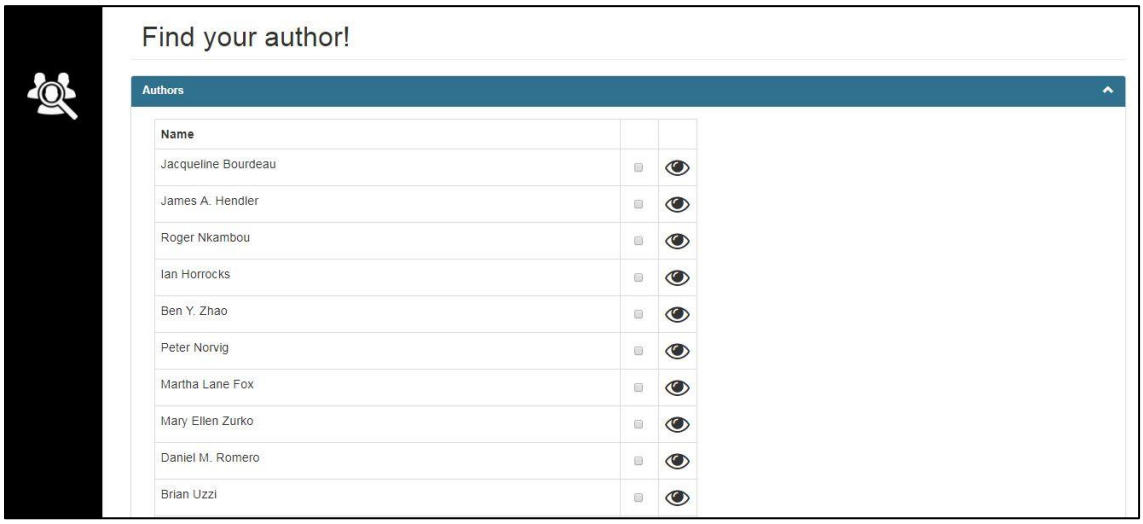


Figura 35 Listado de autores. Elaboración propia

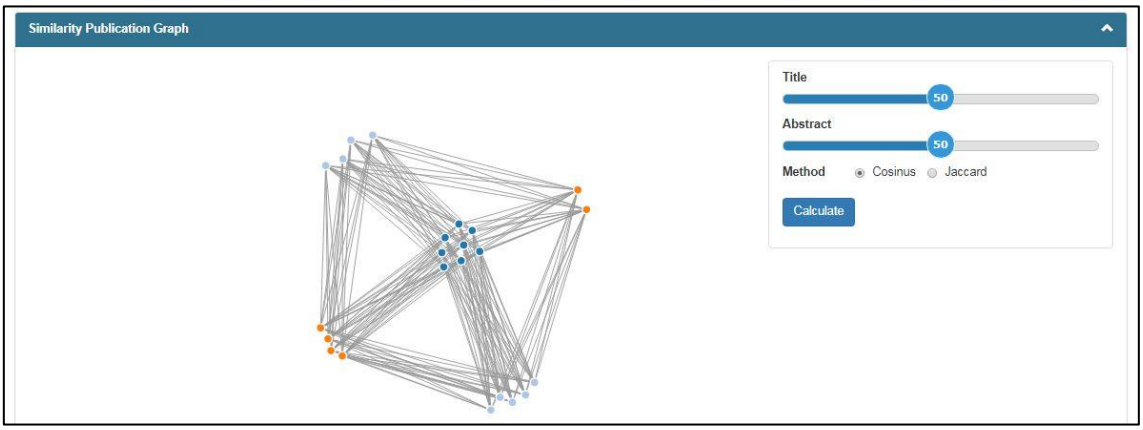
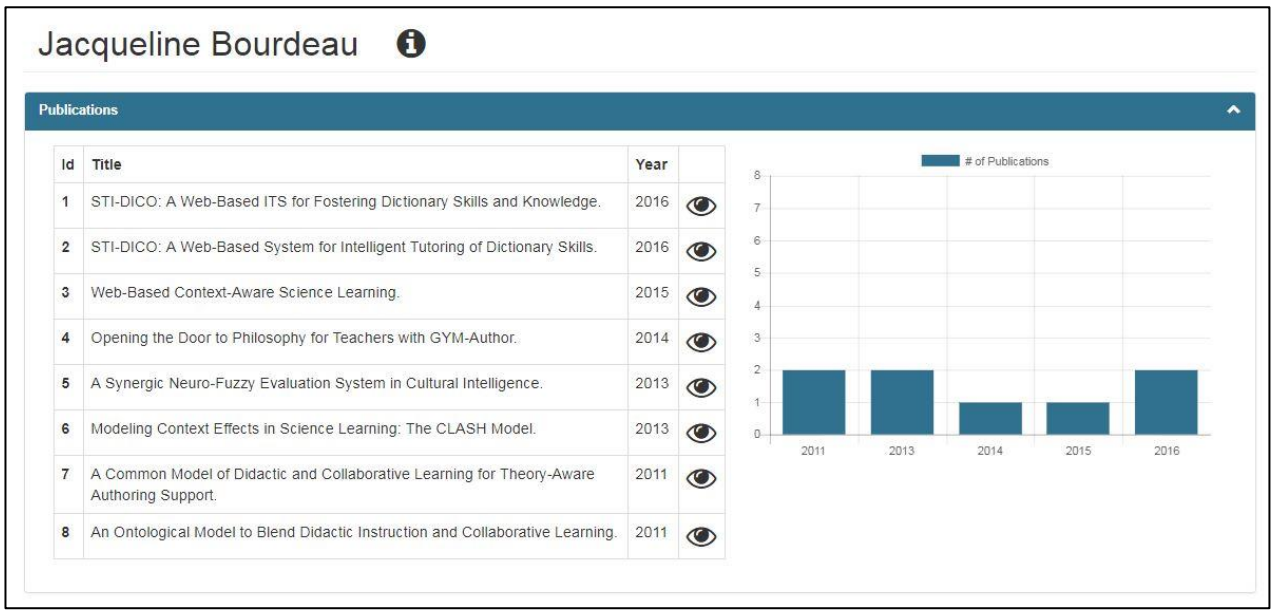


Figura 36 Similitud entre las publicaciones de autores. Elaboración propia

Detalle de un autor: el usuario puede acceder a varios datos acerca de un autor determinado, como sus publicaciones, sus colaboraciones, un grafo de similitud entre sus publicaciones, una nube de término e información básica sobre el autor, como por ejemplo, los alias que utiliza.

Figura 37 Detalle de un autor. Elaboración propia

Publicaciones de un autor con una distribución gráfica por años: esta funcionalidad se puede encontrar en la página del detalle de un autor. El usuario observa un listado con todas las publicaciones del autor, junto con el año de publicación, una opción para ver el detalle de la publicación y un gráfico donde se observa cuántas publicaciones ha realizado el autor cada año.



Id	Title	Year
1	STI-DICO: A Web-Based ITS for Fostering Dictionary Skills and Knowledge.	2016
2	STI-DICO: A Web-Based System for Intelligent Tutoring of Dictionary Skills.	2016
3	Web-Based Context-Aware Science Learning.	2015
4	Opening the Door to Philosophy for Teachers with GYM-Author.	2014
5	A Synergic Neuro-Fuzzy Evaluation System in Cultural Intelligence.	2013
6	Modeling Context Effects in Science Learning: The CLASH Model.	2013
7	A Common Model of Didactic and Collaborative Learning for Theory-Aware Authoring Support.	2011
8	An Ontological Model to Blend Didactic Instruction and Collaborative Learning.	2011

Figura 38 Publicaciones de un autor. Elaboración propia

Listado de colaboraciones de un autor: esta funcionalidad puede encontrarse en la página del detalle de un autor y permite obtener un listado de aquellos autores que han colaborado con el autor al que el usuario está visitando. En el listado aparece el nombre del autor con el que ha colaborado seguido del número de publicaciones que han realizado juntos, a parte de una opción para poder visualizar el detalle de la colaboración.

Figura 39 Listado de colaboraciones. Elaboración propia

Colaboraciones de un autor con una distribución gráfica por años: esta funcionalidad se puede encontrar en la página del detalle de la colaboración ente autores. Se asemeja a la funcionalidad de observar las publicaciones de un autor con una distribución gráfica por años. En este caso, el usuario observa un listado de aquellas publicaciones en las que han colaborado dos determinados autores junto con el año de publicación y una opción para poder ver el detalle de cada una de ellas. Al lado de este listado aparece un gráfico donde se puede observar la evolución de la colaboración de los dos autores durante los años.

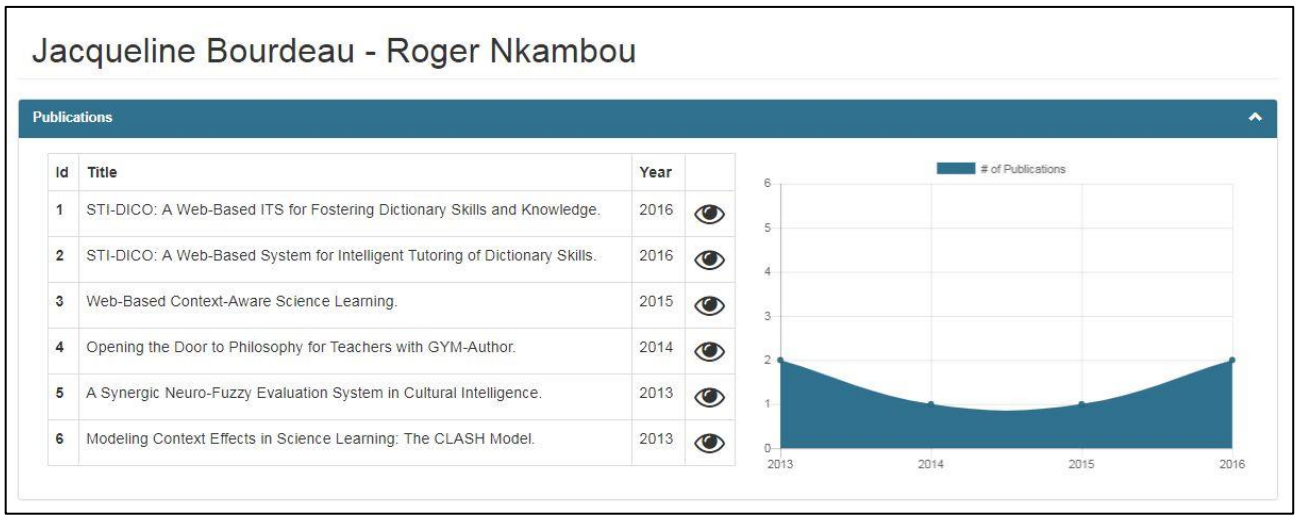


Figura 40 Colaboraciones de un autor. Elaboración propia

Grafo de similitud: esta funcionalidad se puede encontrar en la página del detalle de un autor y permite observar un grafo donde se simula la distancia a la que se encuentran las publicaciones del autor entre ellas. Cada uno de los nodos del grafo hace referencia a una publicación y la rama que los une hace referencia a la similitud entre ellas. Como más cerca se encuentran, más se parecen y como más lejos, menos. Es posible clicar encima de un nodo para observar una tabla con valores donde se muestra el valor numérico de la distancia entre las publicaciones en porcentaje. El usuario también puede ajustar el método de comparación que desea realizar y cuanto peso desea darle al título y al *abstract* de la publicación en el cálculo de similitud.

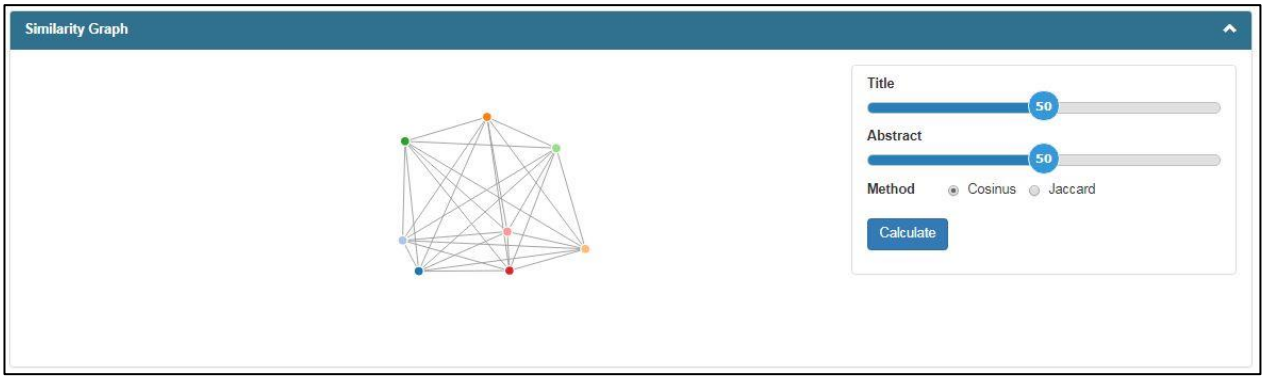


Figura 41 Grafo de similitud método Coseno. Elaboración propia

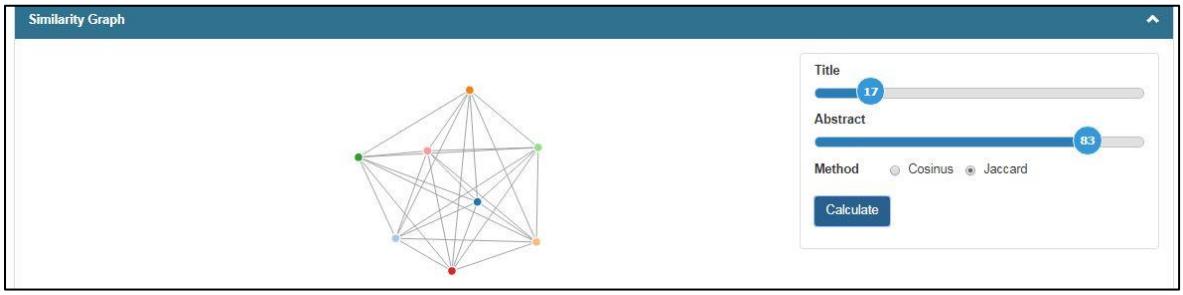
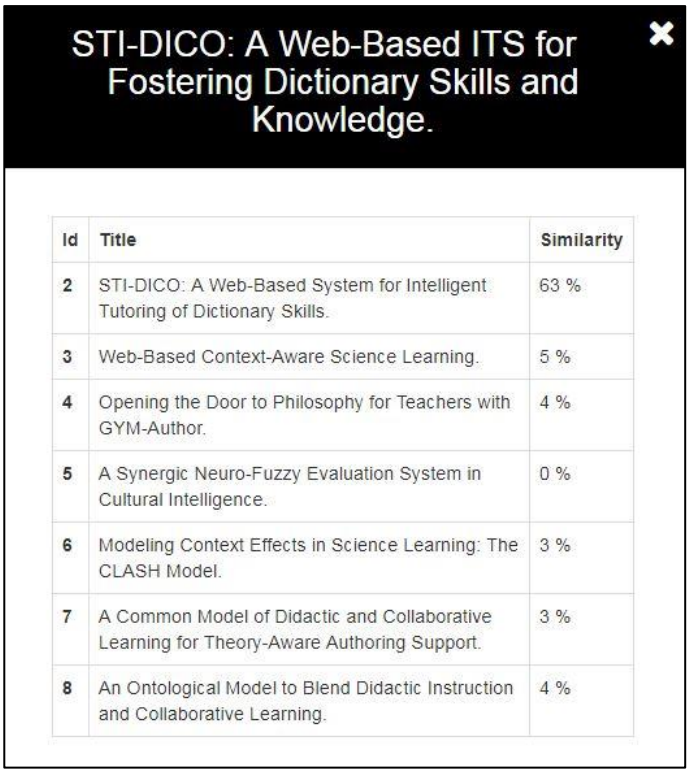


Figura 42 Grafo de similitud método *Jaccard*. Elaboración propia



Id	Title	Similarity
2	STI-DICO: A Web-Based System for Intelligent Tutoring of Dictionary Skills.	63 %
3	Web-Based Context-Aware Science Learning.	5 %
4	Opening the Door to Philosophy for Teachers with GYM-Author.	4 %
5	A Synergic Neuro-Fuzzy Evaluation System in Cultural Intelligence.	0 %
6	Modeling Context Effects in Science Learning: The CLASH Model.	3 %
7	A Common Model of Didactic and Collaborative Learning for Theory-Aware Authoring Support.	3 %
8	An Ontological Model to Blend Didactic Instruction and Collaborative Learning.	4 %

Figura 43 Similitud entre publicaciones. Elaboración propia

Nube de términos de un autor: esta funcionalidad se puede encontrar en la página detalle de un autor y permite observar una nube de términos de las 100 palabras más relevantes entre todos los *abstracts* de todas las publicaciones del autor. Estas palabras tienen en común que su categoría gramatical es 'Sustantivo'. Como mayor es el tamaño de la palabra, más relevante es.

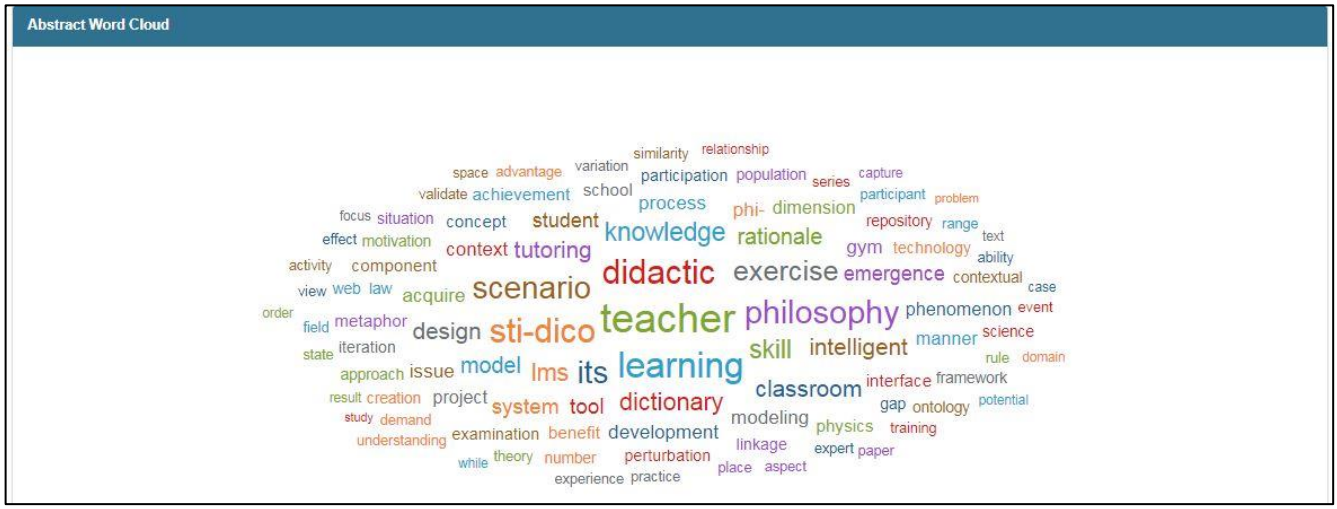


Figura 44 Nube de términos. Elaboración propia

Detalle de una publicación: en esta funcionalidad se puede acceder al detalle de una publicación y observar la información acerca de esta, como el título, el *abstract*, los autores que la han realizado, el año de publicación o el número de citas. Mediante esta funcionalidad es posible acceder a la opción de categorizar oraciones u observar una nube de términos de la publicación.

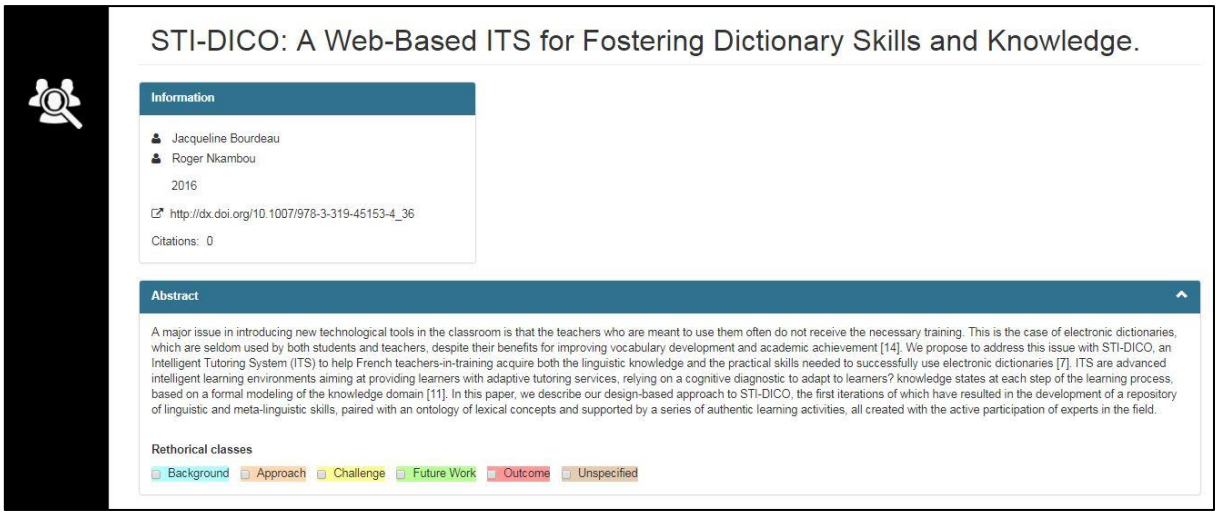


Figura 45 Detalle de una publicación. Elaboración propia

Categorización de oraciones: esta funcionalidad puede encontrarse en la página de detalle de una publicación y permite al usuario seleccionar un o varios tipos de categoría y visualizar qué oraciones del *abstract* de la publicación forman parte de ellas. Si para esa publicación no se ha podido obtener el *abstract*, no aparece la opción de poder seleccionar las categorías.

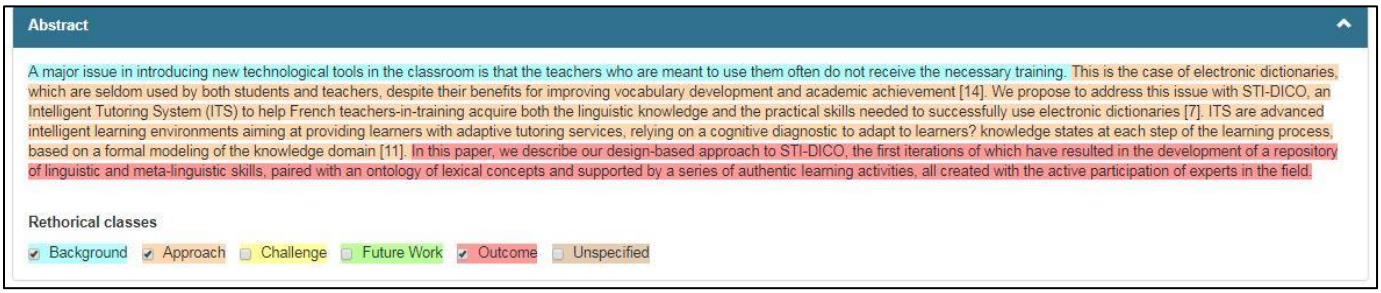


Figura 46 Categorización retórica. Elaboración propia

Nube de términos de una publicación: esta funcionalidad puede encontrarse en la página de detalle de una publicación y es una funcionalidad prácticamente idéntica a visualizar la nube de términos de un autor, con la diferencia de que en este caso solo se visualizan aquellas palabras pertenecientes al *abstract* de la publicación y el tamaño de la nube no está sujeto a 100, sino que se muestran todas aquellas que cumplan que su categoría gramatical sea 'Sustantivo'.

Figura 47 Nube de términos de una publicación

7. MEJORAS

A continuación se exponen algunas mejoras que podrían aplicarse al proyecto para mejorar su eficiencia referente a la obtención de la información y al procesamiento de texto.

- **Jenkins[43]:** es un sistema de integración continua alojado en un servidor que permite obtener diferentes versiones de un proyecto mediante un controlador de versiones, como por ejemplo Git, y desplegar la versión deseada en el servidor. También permite crear nuevas tareas y configurar su ejecución. Una mejora interesante sería montar un sistema como este para desplegar de forma fácil y rápida la versión del proyecto deseada y crear tareas que se ejecuten a una determinada hora para realizar esas funciones que requieren mucho tiempo, como por ejemplo, obtener las publicaciones de nuevos autores, utilizar la técnica de *scraping* para obtener los *abstracts*, procesar los nuevos documentos en GATE y calcular las similitudes entre las publicaciones nuevas.

- **Servidor:** disponer de un servidor con unas características de hardware más potentes para poder realizar todas esas tareas computacionalmente costosas de forma más rápida y eficaz.

- **Uso de *multithreading* en el cálculo de similitudes:** mejorar el proceso de cálculo de similitudes entre publicaciones utilizando la técnica de *multithreading*, tal y como se utiliza para obtener los *abstracts* de las publicaciones. De esta forma se reduciría el tiempo de ejecución de este proceso.

- **Rotar la dirección IP en la técnica de *scraping*:** aplicar una de las buenas prácticas definidas anteriormente para realizar *scraping* correctamente. Modificar la dirección IP de origen con la que se realiza la conexión a una plataforma digital para que este proceso automatizado sea más difícil de detectar.

[43] https://jenkins.io/

8. CONCLUSIONES

A nivel personal, este proyecto se inició con la intención de poder trabajar con una cantidad grande de datos y poder procesarlos con el objetivo de aportar un cierto valor a un usuario y poder trabajar más a fondo en los campos de la ingeniería relacionados con Big Data y Procesamiento del Lenguaje Natural para poder afrontar nuevos retos en el futuro relacionados en estos campos.

Después de varios meses de desarrollo y de haber encontrado muchos problemas por el camino, puedo dar por cumplido tanto mi objetivo personal como el objetivo para el que se ha desarrollado el proyecto.

El proyecto cumple con las especificaciones que han sido mencionadas a lo largo de este documento, las funcionalidades han sido implementadas y han permitido obtener una unificación de la información que puede ser extraída de diferentes plataformas y repositorios bibliográficos en una sola plataforma y se han utilizado diferentes técnicas y herramientas de procesamiento de lenguaje natural para obtener información extra que permite proporcionar un valor añadido al usuario.

Si bien es cierto que la cantidad de datos con la que se ha trabajado es un conjunto muy reducido de todos los datos disponibles, tratar con un total de 11.871 publicaciones científicas ha sido un conjunto de datos suficientemente grande para poder explorar la dificultad de obtener la información necesaria y poder procesarla.

Cabe destacar que este proyecto tiene mucho margen para seguir su desarrollo, ya que se puede complementar investigando datos acerca de otro tipo de publicaciones u otras plataformas digitales y usando otras técnicas de procesamiento de lenguaje natural para crear nuevas implementaciones que permitan obtener otro tipo de información de interés para poder visualizarla.

Personalmente puedo valorar toda esta etapa como una experiencia positiva, ya que no sólo me siento muy satisfecho con el trabajo realizado y con el resultado de este, sino que el haberme encontrado con problemas que no sabía cómo resolver y con etapas donde no veía del todo claro poder cumplir con los objetivos en el plazo de tiempo disponible, me ha ayudado a tener la paciencia y la calma necesaria para poder superar estas situaciones y valorar los resultados obtenidos y todo el esfuerzo realizado para llegar a ellos.

BIBLIOGRAFÍA

Arcila-Calderón, C., Barbosa-Caro, E., & Cabezuelo-Lorenzo, F. (2016). Técnicas big data: análisis de textos a gran escala para la investigación científica y periodística. *El profesional de la información, 25*(4), 623-631.

Cunningham, H., Maynard, D., Bontcheva, K., Tablan, V., Aswani, N., Roberts, I., y otros. (2017). *Developing Language Processing Components with GATE Version 8 (a User Guide).*

dblp: Distribution of publication type. (s.f.). Recuperado el 10 de 05 de 2017, de Dblp.uni-trier.de: http://dblp.uni-trier.de/statistics/distributionofpublicationtype

dblp: Frequently Asked Questions. (s.f.). Recuperado el 10 de 05 de 2017, de Dblp.uni-trier-de: http://dblp.uni-trier.de/faq/

dblp: Peter Norvig. (s.f.). Recuperado el 10 de 05 de 2017, de Dblp.uni-trier.de: http://dblp.uni-trier.de/pers/hd/n/Norvig:Peter

dblp: WWW 2016. (s.f.). Recuperado el 10 de 05 de 2017, de Dblp.uni-trier.de: http://dblp.uni-trier.de/db/conf/www/www2016.html

Dr. Inventor Text Mining Framework Documentation. (s.f.). Recuperado el 06 de 06 de 2017, de Driframework.readthedocs.io: http://driframework.readthedocs.io/en/latest/

Estándar de exclusión de robots. (s.f.). Recuperado el 15 de 05 de 2017, de es.wikipedia.org: https://es.wikipedia.org/wiki/Est%C3%A1ndar_de_exclusi%C3%B3n_de_robots

How to prevent getting blacklisted while scraping. (s.f.). Recuperado el 15 de 05 de 2017, de Scrapehero.com: https://www.scrapehero.com/how-to-prevent-getting-blacklisted-while-scraping/

Jaccard Index / Similarity Coefficient. (s.f.). Recuperado el 27 de 05 de 2017, de http://www.statisticshowto.com/jaccard-index/

Ley bibliométrica de crecimiento exponencial. (s.f.). Recuperado el 08 de 06 de 2017, de Es.wikipedia.org: https://es.wikipedia.org/wiki/Ley_bibliom%C3%A9trica_de_crecimiento_exponencial

Ley, M. (2002). The DBLP Computer Science Bibliography: Evolution, Research Issues, Perspectives. En H. Alberto, F. LaenderArlindo, & L. Oliveira, *String Processing and Information Retrieval* (págs. 1-10).

Ley, M. (2009). *DBLP XML Requests.* Techincal report, Universitat Trier, Informatik.

Ley, M. (2009). DBLP: Some Lessons Learned. *Proceedings of the VLDB Endowment, 2*(2), 1493-1500.

Moreno, A., & Redondo, T. (2016). Text Analytics: the convergence of Big Data and Artificial Intelligence. *International Journal of Interactive Multimedia and Artificial Intelligence, 3*(6), 57-64.

Part-of-Speech Tags used in the Hepple Tagger. (s.f.). Recuperado el 25 de 05 de 2017, de https://gate.ac.uk/sale/tao/splitap7.html

Qué es el Web scraping. (s.f.). Recuperado el 15 de 05 de 2017, de Sitelabs: https://sitelabs.es/web-scraping-introduccion-y-herramientas/

Ronzano, F., & Saggion, H. (2015). Dr. Inventor Framework: Extracting Structured Information from Scientific Publications. En N. Japkowicz, & S. Matwin, *Discovery Science* (págs. 209-220). Canada: Springer.

Ronzano, F., & Saggion, H. (2016). Natural Language Processing for Intelligent Access to Scientific Information. En *COLING (Tutorials)* (págs. 9-13). ACL.

Saggion, H. (2008). SUMMA. A Robust and Adaptable Summarization Tool. *TAL, 49*(2), 103-125.

Saggion, H. (2014). Creating Summarization Systems with SUMMA. En *LREC* (págs. 4157-4163).

SUMMA - A software library for development of text summarization systems and applications. (s.f.). Recuperado el 29 de 05 de 2017, de Knowledge.upf.edu: http://knowledge.upf.edu/technologies-software/summa-software-library-development-text-summarization-systems-and-applications

Testear Javascript con PhantomJS, QUnit y MSBuild | Koalite. (s.f.). Recuperado el 17 de 05 de 2017, de Blog.koalite.com: http://blog.koalite.com/2012/06/testear-javascript-con-phantomjs-qunit-y-msbuild/

Web Scraping. (s.f.). Recuperado el 15 de 05 de 2017, de Es.wikipedia.org: https://es.wikipedia.org/wiki/Web_scraping

Witten, I. H. (2005). Text Mining. En *The Practical Handbook of Internet Computing.*