Semantic Web
Proyecto Etapa 1
Grupo 2

Nicolás Klopstock, Alejandro Salgado, Miguel Ángel Zapata

Descripción Fuentes

Se nos entregó a través del correo la fuente en un archivo **JSON** con más de 150.000 objetos. La información disponible son Papers, que después de revisar algunos casos se puede entender que son de tipo académicos enfocados en estudios computacionales.

Estructura

Es importante resaltar que todos los objetos mantienen las mismas variables y orden. Así mismo, existe **homogeneidad en que se usan las mismas llaves y los mismos tipos de valores**, pero el valor asociado a la llave varía dependiendo del objeto que se esté revisando.

Llave	Descripción	Ejemplo
id	Llave del archivo fuente JSON. No tiene relación con los papers.	2376401
paperid	Id del paper dentro del JSON. A partir de este no es posible obtener el archivo PDF del artículo.	10.1.1.29.3209
title	Título del paper.	A Nonrigid-Body Approach to Matching Mammograms
abstract	Resumen del paper.	Breast cancer is the most frequently diagnosed form of cancer and the most common…
year	Año de publicación del paper.	1999
venue	Evento en el que se enseñó el contenido del paper.	In Proc. of the IEEE Image Processing and its Applications
venueType	Tipo del evento.	CONFERENCE
authors	Colección de nombres de los autores. Es la única llave y valor que no entrega un valor único. Si solo se tiene un nombre igualmente se almacena como una lista.	["m wirth", "c choi", "a jennings"]

Table 1 Descripción Llaves Archivo Fuente

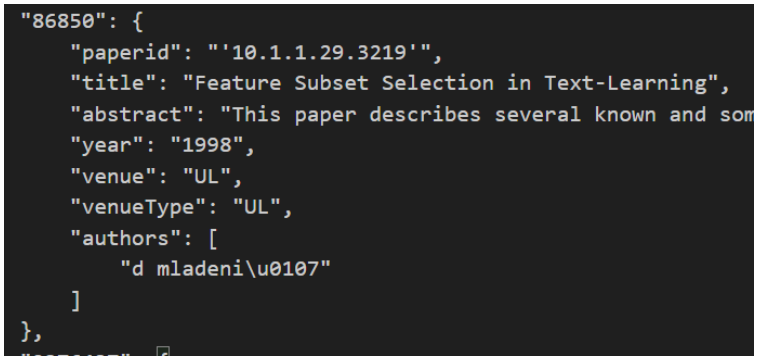

Figure 2 Ejemplo Objeto Archivo Fuente

La segunda fuente de información es la **API de semanticscholar.org**. De la cual usamos dos endpoints: "Academic Graph: Paper" y "Academic Graph: Author". La primera nos brinda información sobre el título, las referencias, citaciones y si tiene acceso público al PDF del paper, entre otros (ver figura 2). Mientras que la segunda nos brinda la lista de papers que el autor consultado ha escrito (ver figura 3).

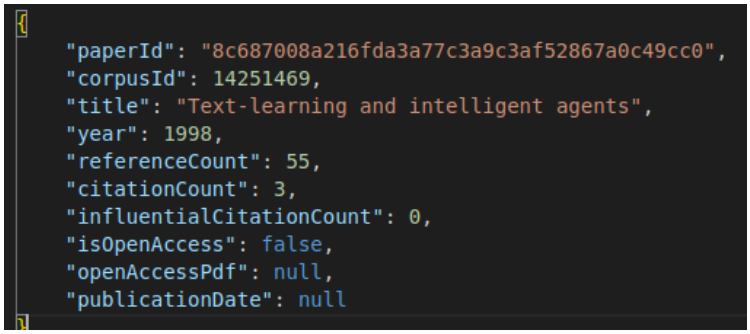

Figure 3 Ejemplo API Academic Graph: Paper

Figure 4 Ejemplo API Academic Graph: Author

A partir de un proceso de comprobación e investigación para obtener los PDFs usando las variables del JSON principal se llegó a la conclusión de que la llave *paperid* no tiene relación a la API de semanticscholar. Por lo tanto, se decidió hacer uso del título y los autores para buscar los PDF.

Proceso Extracción PDF

El primer proceso consiste en, usando una de las APIs, obtener el PDF de cada paper. Para esto es importante detallar el proceso, por lo que en esta sección se explica un paso a paso.

Tecnologías Usadas

- El lenguaje de programación es **Python**, usando sus librerías para archivos JSON.
- Se uso la API de **semanticscholar.org** para obtener la información de básica de los papers, incluyendo el ID que permite obtener el PDF.
- Se uso la librería **request** de Python para obtener los PDFs con solicitudes HTTP.

- Contenedor Docker para servidor GROBID.
- Repositorio https://github.com/titipata/scipdf_parser para analizar los PDFs.

Descripción Proceso

Para el proceso de extracción de PDFs se desarrollaron 3 clases en Python que se encargan de la lectura del JSON, revisión información en las APIs y extracción final del documento para cada uno de los objetos de papers que se nos entregaron. Lo primero que se hace es una lectura del archivo JSON, luego se empiezan a hacer los siguientes pasos para cada uno de los papers que se leyeron:

1. Usando los autores, se hace una búsqueda con la API de si se tiene algún registro de esos artículos. En el caso de que se tengan se guarda el id, fecha y nombre del paper. *Si se tienen más de un autor se ingresa como una tupla.*
    a. Se verifica que el código de respuesta sea 200, y si no se genera error.
2. Se compara el título del paper obtenido en la consulta con el del JSON original. Si coincidan, se continúa con el proceso usando el id obtenido.
3. Ahora, usando el ID, nuevamente se accede a la API pero en este caso se obtienen más datos del paper como su fecha de publicación, cantidad de referencias, etc. Pero lo más importante es que se obtiene si el archivo es de acceso libre y la URL es accesible.
    a. Se verifica que el código de respuesta sea 200, y si no se genera error.
4. Finalmente, usando la URL se obtiene el PDF de la Web. Si se encuentra se almacena en modo binario en una carpeta dentro del computador.

Como se explicó antes, este proceso es repetitivo y al final se obtienen en una misma carpeta todos los PDFs que se logren encontrar siguiendo los pasos.

Extracción y Limpieza Metadatos

Para la extracción de los metadatos se utilizaron las librerías **SciPDF.** Durante este proceso, se hizo revisión de integridad de los datos, enfocado en evitar duplicador. Usando **Grobid** se obtuvo la metadata de los documentos al parsear cad PDF a XML. Por último, se almacena la información en un archivo JSON. Por cada registro de documento se maneja un título, detalles de autor y detalles de las referencias. Creando un archivo JSON con los siguientes campos:

Relacionados con el título:

Idno, title, publication_year, abstract,

Relacionados con el autor:

authorForename, authorSurname, authorEmail, authorAffiliation, authorAddressLine, authorPostCode, authorSettlement, authorCountry,

Relacionados con referencias:

referenceTitle, referencePublicationDate, referenceMeeting, referenceCity, referenceCountry, referenceNote, referenceAuthorForename, referenceAuthorSurname

Al final se obtuvo la metadata de aproximadamente 4400 papers y su información (las razones por las que no se llegó a los 5000 se explican más adelante).

El proceso de limpieza de datos, crucial para asegurar la integridad y evitar la duplicación de los datos recolectados, fue un esfuerzo continuo durante toda la fase de extracción de PDFs y metadatos. Este proceso implicó una serie de revisiones al extraer los datos, incluyendo una comprobación de coincidencia de autores y títulos, el manejo cuidadoso de caracteres especiales, y la identificación de registros duplicados (lo cual se comprobó durante y posterior a la extracción). Además, se hizo una normalización de títulos de artículos, con la misma intención de evitar duplicados y tener datos más correctos.

Para tener total certeza de la calidad de datos con los que nos íbamos a quedar, hicimos una validación de relaciones autor-artículo. Esto quiere decir que, según los artículos que encontrábamos, revisamos si realmente existe un autor asociado a ese paper y si el autor tiene ese paper bajo su nombre. Notamos que, por problemas en la forma en la que se escribe el título de un artículo o el nombre de un autor, puede haber duplicados. Este post-procesamiento logra una forma efectiva de comprobar la realidad de los datos, verificando las relaciones correctas; además de ser otra una forma de cerciorar la no duplicidad de artículos.

La limpieza de datos se diseñó no solo para mejorar la calidad y fiabilidad de la información recolectada sino también para facilitar su análisis y almacenamiento posterior, garantizando que cada pieza de información fuera única y estuviera correctamente formateada de acuerdo con los estándares establecidos para el proyecto.

Retos y Problemas del Proceso

El desafío central consistió en adoptar tecnología externa a la que normalmente utilizamos. Inicialmente, enfrentamos el proceso colaborativo de seleccionar la API para la extracción de PDFs y metadatos. Para ello, fue necesario idear una forma de localizar los papers, considerando que no teníamos un identificador único en el JSON para obtener los PDFs. La integración de tecnología externa para la extracción de PDFs y metadatos marcó un reto en el proceso, el cual se amplificó con la necesidad de implementar la limpieza de datos. Parte esencial de este esfuerzo fue asegurar la precisión en la revisión de autores y títulos, así como el manejo de caracteres especiales y la normalización de mayúsculas y minúsculas. Este paso fue crucial para mantener la integridad de la información recolectada, enfocándose especialmente en evitar la duplicación de datos durante la extracción.

Una vez que logramos idear el enfoque adecuado, el proceso no fue tanto un desafío conceptual como un desafío de tiempo. Con cada nueva implementación de los pasos, realizamos pruebas individuales para asegurar el correcto funcionamiento del proyecto. Finalmente, otro desafío consistió en utilizar máquinas virtuales para extraer los datos. Aunque no resultó complicado, fue necesario dejar el código ejecutándose durante un tiempo considerable y utilizar una capacidad significativa de memoria.

Es importante destacar que se obtuvieron únicamente 4400 datos finales para los JSON. Después del proceso de la extracción de los PDFs se obtuvieron 6366 artículos que se pudieron descargar. Sin embargo, al usar Grobid, algunos de los documentos no se lograron parsear de PDF a XML, por lo que no se pudieron obtener los 5000 datos completos.