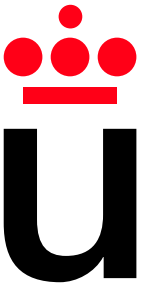

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2023-2024

Trabajo Fin de Grado

APLICACIÓN DE MEDIDAS DE CENTRALIDAD EN LA BASE DE DATOS BIBLIOGRÁFICA SCOPUS

Autor: Sergio Hernández Sandoval
Tutor: Clara Simón De Blas
Co-tutor: Regino Criado Herrero

Agradecimientos

Expreso mi sincero agradecimiento a mis tutores del TFG, Clara Simón de Blas y Regino Criado Herrero, por su vital apoyo y orientación a lo largo de todo el trabajo. La disponibilidad, dedicación y facilidades que me han proporcionado continuamente han sido fundamentales para alcanzar los resultados obtenidos. Sin ellos, todo hubiera sido mucho más difícil.

Resumen

Este Trabajo de Fin de Grado aborda la aplicación de medidas de centralidad en la base de datos bibliográfica *Scopus*. Los hipergrafos, utilizados para modelar las complejas redes de conexiones existentes en esta extensa plataforma de nivel mundial, son la herramienta principal de estudio. La exploración de la centralidad de vector propio en estas redes permite evaluar la importancia de sus elementos, con el objetivo final de identificar el elemento más influyente dentro de los diversos entramados de *Scopus*.

En las secciones iniciales, se proporciona el contexto del trabajo y se exponen los fundamentos del estudio, acompañados de una definición clara de los objetivos. La tercera sección se dedica a brindar una comprensión completa y detallada de la metodología empleada, abarcando todas las fases del proceso. Finalmente, al presentar los resultados obtenidos tras el uso del marco metodológico, se destacará la relevancia de esta herramienta para la comprensión y el éxito en la identificación del elemento más influyente en las redes de *Scopus*.

Palabras clave:

- Python
- Web Scraping
- Scopus
- MySQL
- Hipergrafo
- Centralidad

Índice de contenidos

Índice de tablas

Índice de figuras

Índice de algoritmos

1 Introducción

En la actualidad, nos encontramos en la era de la información y los datos, en la cual la acumulación de conocimiento se produce a una velocidad inigualable. Este extenso océano de datos se encuentra en diversos ámbitos, y en esta perspectiva, las plataformas bibliográficas se presentan como elementos fundamentales. Así, bases de datos bibliográficas como *Scopus*, *IEEE Xplore*, *Web of Science* o *ScienceDirect* no solo actúan como tesoros, ofreciendo una gran cantidad de información académica proveniente de diversas disciplinas, sino que también reflejan la necesidad imperiosa de almacenar y gestionar de manera eficiente la gran cantidad de datos generados. En este contexto, estas plataformas se convierten en pilares fundamentales para la investigación y el intercambio de conocimientos en una amplia variedad de ramas, lo que define nuestra era impulsada por la información.

Las plataformas bibliográficas trascienden la noción de repositorios simples. Se establecen como dinámicas redes que interconectan una gran cantidad de entidades, entre las que se pueden encontrar por ejemplo autores, instituciones o publicaciones científicas. Esta interconexión genera una compleja red de relaciones, una estructura que se modela de manera eficaz a través del uso de los llamados grafos. Aquí es donde entra en juego la Teoría de Grafos.

La Teoría de Grafos nos brinda un sólido marco conceptual para analizar y comprender la intrincada red de relaciones que se encuentra en una base de datos bibliográfica mundial. En los términos más básicos, un grafo puede ser definido como un conjunto de puntos denominados nodos, que representan entidades, y un conjunto de aristas, que representan las conexiones entre pares de estas entidades. Cada arista une dos nodos, lo cual refleja la naturaleza de estas relaciones.

Esta representación gráfica, además de permitir visualizar de manera intuitiva la estructura de estas conexiones, también posibilita aplicar herramientas analíticas para extraer patrones significativos y comprender la dinámica subyacente de la red académica.

A medida que profundizamos en las intrincadas relaciones entre los elementos de las redes académicas, se hace evidente la necesidad de trascender las representaciones binarias. Aunque los grafos son eficientes para captar relaciones sencillas, las complejidades que surgen cuando intervienen más de dos elementos no pueden modelarse adecuadamente con los grafos tradicionales. Es en este punto donde surge el concepto de hipergrafo. Mientras que los grafos convencionales representan conexiones entre pares de nodos, los hipergrafos permiten modelizar relaciones más enrevesadas. Estas implican conjuntos formados por cualquier número finito de nodos, por lo que proporcionan una representación más completa de la complejidad en el tejido académico. De este modo, los hipergrafos resultan ser una herramienta valiosa para esclarecer y descifrar las variadas y detalladas relaciones que caracterizan a las redes académicas modernas.

En cualquier red que incluya elementos conectados, ya sea un callejero, Internet, una base de datos o una red social, no todos los elementos tienen la misma importancia. En el contexto de las redes académicas, este principio se aplica en que algunos nodos son más relevantes que otros para comprender la dinámica y la propagación de la información. Aquí es donde entra en juego la noción de centralidad.

La centralidad de un nodo en un grafo o hipergrafo mide su importancia relativa en la red. Existen diversas formas de conceptualizarla, y tres de las más destacadas son la centralidad de grado, la centralidad de intermediación y la centralidad de vector propio.

La centralidad de grado asocia la importancia cuantitativa de un nodo a su número total de conexiones. En una red social, por ejemplo, la identificación de nodos con una alta centralidad de grado podría resultar crucial para comprender la propagación de un virus informático, ya que estos nodos, al contar con múltiples conexiones, pueden ser un elemento esencial en la transmisión.

La centralidad de intermediación se enfoca en la capacidad de un nodo para actuar como intermediario o puente entre otros nodos, facilitando la comunicación. En la red de metro de una ciudad, por ejemplo, esta medida se convierte en un aspecto fundamental, donde ciertas estaciones, aunque no tengan muchas conexiones directas, son cruciales para el flujo eficiente de personas de un punto a otro.

Por último, la centralidad de vector propio evalúa tanto la cantidad como la calidad de las conexiones de un nodo. En una red social, por ejemplo, sería crucial para la correcta identificación de los usuarios más influyentes, considerando no

solo la cantidad de conexiones, sino también la influencia de estas.

En el contexto de las bases de datos bibliográficas y la exploración de redes académicas, pondremos el foco en la centralidad de vector propio. Esta medida nos proporciona las herramientas necesarias para el cálculo de la centralidad en redes modeladas por hipergrafos, siendo especialmente relevante al valorar tanto la cantidad como la calidad de las conexiones de un nodo. Esto nos permitirá identificar nodos críticos en la propagación de información y la cohesión de la red académica.

Asimismo, es imprescindible destacar que mi Trabajo de Fin de Grado en Matemáticas, titulado *Estudio de la centralidad en hipergrafos a través de sus grafos lineales asociados* [1], es un complemento esencial al presente análisis. Al profundizar en él sobre todas las bases teóricas y matemáticas asociadas a los grafos e hipergrafos y, de manera específica, abordar en detalle el cálculo de la centralidad de vector propio de estos últimos, ambos trabajos se entrelazan de manera coherente. Juntos, proporcionan una sólida base conceptual para la exploración de la dinámica en redes modeladas por hipergrafos, enriqueciendo así la comprensión de la centralidad de vector propio y su aplicación en este contexto específico a bases de datos bibliográficas.

En este marco, la combinación de la teoría de grafos e hipergrafos y la aplicación práctica en bases de datos bibliográficas ofrece una perspectiva innovadora para abordar los desafíos actuales en la gestión y análisis de la información académica en nuestra época marcada por la información.

2

Objetivos

Este Trabajo de Fin de Grado se sumerge en el desarrollo de una metodología integral capaz de efectuar el cálculo de la centralidad de vector propio en hipergrafos, utilizando los datos extraídos de la plataforma bibliográfica *Scopus*.

La extensa cantidad de datos disponible en la red de *Scopus* proporciona un terreno muy propicio para explorar las complejas redes generadas en ella, con un enfoque preciso en la identificación de los autores más influyentes en base a consultas aplicadas a todos sus registros usando filtros acotadores, tales como el tipo de documento, el año de publicación, el idioma o palabras clave.

Esta metodología abordará la extracción de los datos de *Scopus* y su posterior almacenamiento en una base de datos, terminando con su procesamiento para el cálculo de la centralidad de vector propio del hipergrafo que modele toda la información. Esta metodología destaca por su excepcional flexibilidad, ofreciendo una guía adaptable para extender el estudio de la centralidad a otras redes construidas a partir de datos de *Scopus*. Esta flexibilidad permite ajustes simples en el código, facilitando el análisis en diferentes contextos y siendo una herramienta práctica para investigadores que deseen aplicar este análisis a otras redes sin reconstruir completamente el marco metodológico. Además, se convierte en una valiosa herramienta para mi Trabajo de Fin de Grado en Matemáticas, donde se abordan todas las bases teóricas y matemáticas asociadas a los grafos, hipergrafos y sus centralidades de vector propio.

3

Metodología

En este trabajo se ha desarrollado una metodología con el objetivo de servir como herramienta para el cálculo de la centralidad de vector propio de los hipergrafos que modelan las redes de la plataforma bibliográfica *Scopus*. Para lograr este propósito, se ha dividido en tres fases bien diferenciadas:

1. Recopilación de los datos procedentes de la página web de *Scopus* a través de la técnica de web scraping.

2. Transferencia de la información extraída mediante web scraping hacia una base de datos.

3. Desarrollo de algoritmos para calcular la centralidad de la red utilizando la información almacenada en la base de datos.

Aunque el objetivo principal de esta metodología es determinar el autor más influyente de una red científica, el desarrollo de cada una de las fases se ejemplificará con una misma red concreta para facilitar su seguimiento y comprensión. En cada fase, se proporcionará un detalle exhaustivo de las herramientas utilizadas, los requerimientos necesarios y los distintos pasos seguidos para alcanzar nuestro objetivo. Este enfoque no solo facilitará la replicación del trabajo y asegurará la comprensión integral de la metodología empleada, sino que también fortalecerá la base para futuras investigaciones y desarrollos en este campo específico. Respecto al código, para su realización se tendrá en cuenta las referencias [2], [3] y [4].

3.1. Web Scraping

El web scraping, también conocido como raspado web o extracción de datos web, representa una técnica fundamental para la obtención automatizada de información diversa de páginas web. Esta herramienta versátil encuentra aplicación en una amplia variedad de contextos, abarcando desde investigaciones académicas hasta el análisis de datos, ofreciendo un medio eficaz para acceder a información valiosa en el entorno digital.

Con el propósito de alcanzar las metas establecidas en este trabajo, se realizará la extracción de datos desde la página web conocida como *Scopus*.

Scopus es una de las principales bases de datos bibliográficas a nivel mundial y una potente herramienta de análisis de citas. La empresa editorial líder en el ámbito académico, *Elservier*, decidió lanzarla en el año 2004, y desde ese momento se encarga de gestionarla.

Scopus proporciona actualmente información detallada sobre una cifra alrededor de 91 millones de registros, incluyendo artículos de revistas, libros, actas de congresos, capítulos de libros, tesis doctorales y otros tipos de publicaciones académicas. Debido a que ofrece una visión general completa de la producción de investigación mundial en los campos de ciencia, tecnología, medicina, ciencias sociales, artes y humanidades, *Scopus* se ha convertido en una herramienta muy valiosa para investigadores, estudiantes y profesionales de cualquier disciplina, permitiéndoles encontrar y analizar la información que necesitan. Todo esto constituye la justificación de la elección de esta base de datos bibliográfica.

Para acceder al portal web de *Scopus*, véase reflejado en La Figura 3.1, puede utilizar el enlace proporcionado en [5].

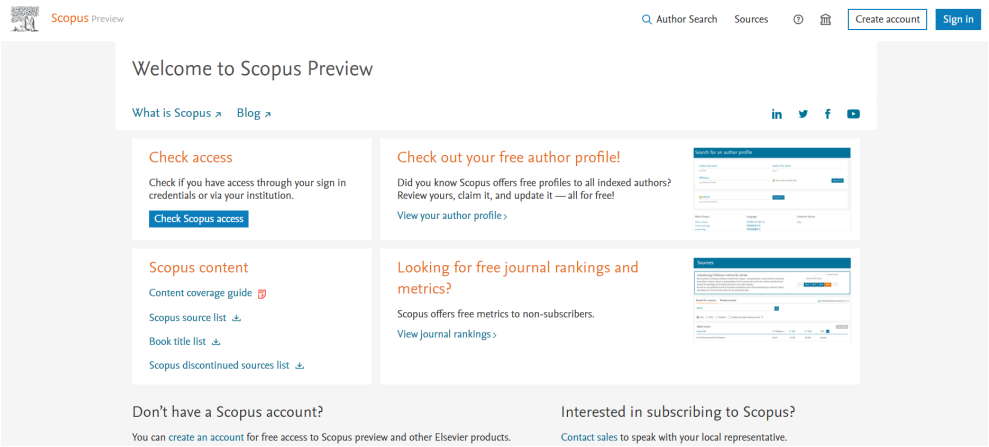

Figura 3.1: Ilustración que captura la pantalla de inicio del portal de *Scopus* al acceder a su web.

Para explorar la base de datos de *Scopus* y realizar búsquedas, suele ser necesario contar con una suscripción o acceder a través de una institución educativa o biblioteca que disponga de dicha suscripción. Esto se debe a que *Scopus* no ofrece acceso directo a usuarios individuales sin afiliación institucional o sin suscripción.

Para acceder a la página de inicio de sesión de *Scopus*, véase reflejada en La Figura 3.2, puede utilizar el enlace proporcionado en [6].

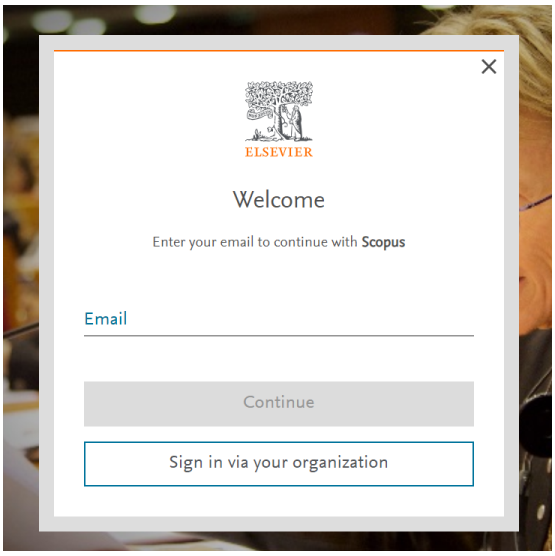

Figura 3.2: Ilustración que captura la pantalla de inicio de sesión del portal de *Scopus*.

Como estudiante de la Universidad Rey Juan Carlos, aprovecho el acceso a *Scopus* mediante la suscripción universitaria. Para llevar a cabo mi inicio de sesión he seguido los siguientes pasos:

- Visito la página de inicio de sesión de *Scopus* visible en La Figura 3.2 a través del enlace proporcionado o desde la interfaz de inicio pulsando en el botón de *Sign in* ubicado en la esquina superior derecha que se visualiza en La Figura 3.1.
- Selecciono la opción *Sign in via your organization*.
- Ingreso mi correo electrónico universitario.
- Confirmo mi organización, siendo redirigido a la pantalla de inicio de sesión y autenticación de mi cuenta universitaria.
- Completado este proceso, me redirigen a la página del explorador de *Scopus* reflejada en La Figura 3.3, desde la cual ya se pueden realizar búsquedas y otras funciones disponibles. También accesible desde el enlace proporcionado en [7] tras haber iniciado sesión.

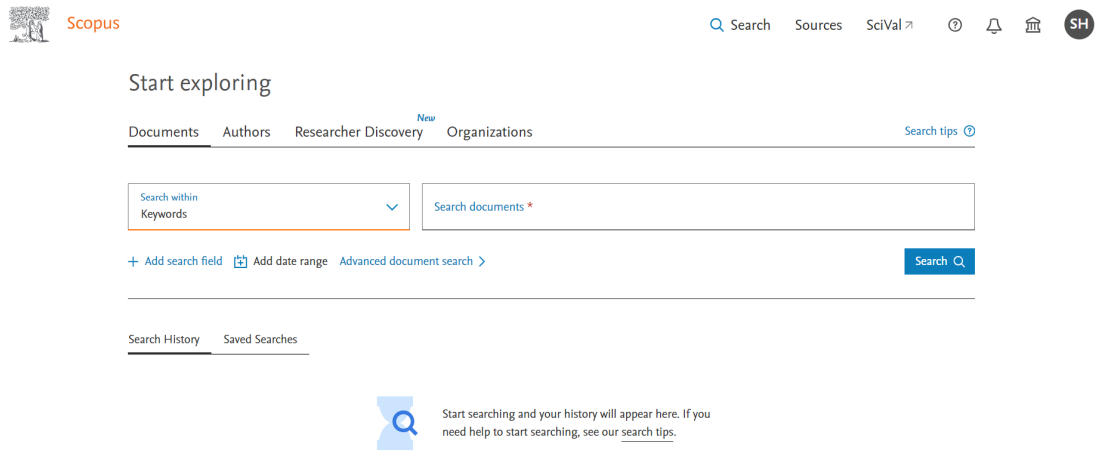

Figura 3.3: Ilustración que captura la interfaz del explorador de *Scopus* una vez se haya iniciado sesión.

Para facilitar el posterior proceso de web scraping, *Scopus* ofrece una interfaz de programación de aplicaciones (API) que permite a sus suscriptores acceder a sus datos. Sin embargo, para utilizarla, es necesario solicitar una *API key* o clave API. La *API key* de *Scopus* es una cadena alfanumérica que autentica a los usuarios suscritos al utilizar la API.

Los pasos para solicitarla son los siguientes:

- Inicie sesión en su cuenta y acceda al enlace proporcionado en [8].
- En esa pantalla, reflejada en La Figura 3.4, podrá consultar en cualquier momento sus *API keys*. Si aún no dispone de ninguna, haga clic en *Create API key*, recuadrado en rojo en la imagen.
- Acepte todas las condiciones y haga clic en *Submit*.
- Su *API key* habrá sido creada y podrá consultarla en la misma página anterior de La Figura 3.4.

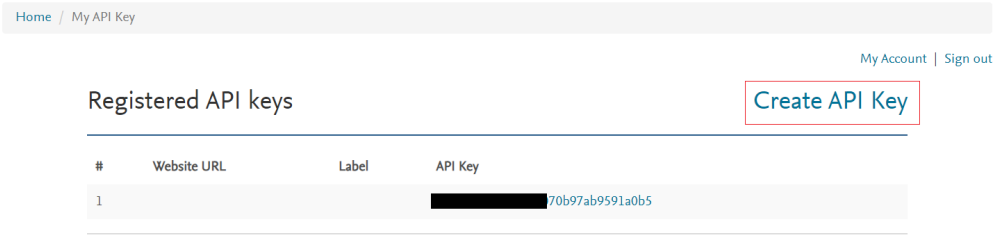

Figura 3.4: Ilustración que captura la pantalla para consultar tus *API keys* de *Scopus*, señalando con un cuadrado rojo el botón para crear una nueva *API key*.

Habiendo definido todos los conceptos y requisitos vinculados a la parte web de *Scopus*, avanzamos ahora hacia los aspectos relacionados con el código del script.

El código, tanto de esta fase como del resto, se va a desarrollar en el lenguaje de programación *Python*, principalmente como causa de su versatilidad, amplia comunidad de desarrollo y robustas bibliotecas especializadas en análisis de datos y grafos. Esto nos garantiza una implementación eficiente y escalable para realizar el análisis de centralidad en la red bibliográfica de *Scopus*. Por ello, es esencial contar con su instalación en el sistema. En mi caso, la versión de *Python* usada es la *3.11.4*, en el sistema operativo de *Windows 10*. Si aún no está instalado, puede descargarlo y posteriormente instalarlo accediendo al enlace proporcionado en [9]. En mi caso, utilizaré el entorno de desarrollo *Visual Studio Code*, aunque puede usarse cualquier otro según las preferencias individuales.

El objetivo primordial de esta fase de web scraping va a consistir en obtener de manera sistemática y eficiente el listado completo de todos los autores vinculados a documentos específicos en un archivo CSV tras una exhaustiva consulta en la base de datos de Scopus mediante el uso de filtros, como por ejemplo la existencia de palabras clave, el tipo de documento o su año de publicación, entre otros.

Para alcanzar este objetivo, se utiliza una biblioteca de Python llamada *pybliometrics*, diseñada para acceder, analizar y visualizar datos bibliométricos. Esta biblioteca se basa en la API de *Scopus* y ofrece diversas funciones para realizar búsquedas, recuperar datos, analizar resultados y visualizar información específica de *Scopus*. En particular, utilizaremos *ScopusSearch*, una funcionalidad destacada de esta biblioteca. *ScopusSearch* constituye una opción clave para llevar a cabo búsquedas avanzadas, recuperar datos precisos y realizar análisis detallados en el contexto de la investigación bibliométrica.

Es crucial tener en cuenta un detalle importante respecto a esta biblioteca. Para acceder a *Scopus* mediante su API, la identificación de la institución del usuario se realiza a través del rango de direcciones IP. Con el fin de prevenir posibles fallos de autenticación, es necesario cumplir con una de las siguientes condiciones:

1. Estar conectado a la red de la institución, como la red wifi de algún campus de la Universidad Rey Juan Carlos en mi caso.
2. Utilizar la VPN de la institución, posiblemente con el requerimiento de la configuración de un proxy, como conectarme a la VPN de la Universidad Rey Juan Carlos en mi caso.
3. Emplear un InstToken proporcionado por *Scopus*.

En mi situación particular, la alternativa de utilizar un InstToken se descartó al momento debido a su complejidad. Dado que los campus de la Universidad Rey Juan Carlos están lejos de mi hogar, la posibilidad de conectarme a la red Wi-Fi de la universidad se contemplaría como último recurso. Utilizar la VPN de la Universidad Rey Juan Carlos para trabajar desde casa fue la primera alternativa considerada debido a la conveniencia que ofrece esta opción. No obstante, los estudiantes no tenemos los permisos ni accesos necesarios para conectarnos a la VPN de la universidad. Ante esta limitación, tras informar al Centro de Atención al Usuario (CAU) sobre mi situación, me sugirieron que mi tutora del Trabajo Fin de Grado realizara una solicitud pidiendo los permisos. Mi tutora, Clara Simón de Blas, gestionó la solicitud y en apenas un par de días se me otorgó acceso a la VPN, permitiéndome conectarme con éxito. A pesar de estar conectado a la VPN de mi universidad, persistían problemas de autenticación. Como consecuencia de esto, la opción finalmente elegida fue la de conectarme a la red Wi-Fi de la universidad desde algún campus. Esta decisión se basó en la eficacia comprobada de esta conexión para evitar errores de autenticación.

Otra biblioteca valiosa presente en Python, especialmente diseñada para manipular datos de manera eficiente, es *pandas*. Esta biblioteca proporciona estructuras de datos flexibles y herramientas de análisis que son esenciales para trabajar con conjuntos de datos, incluyendo la capacidad de filtrar, transformar y visualizar información de manera efectiva. La usaremos para poder organizar la información en el archivo CSV que queremos obtener.

Para incorporar tanto *pybliometrics* como *pandas* en el entorno de desarrollo, simplemente hay que ejecutar el siguiente comando en la terminal:

```
pip install pybliometrics pandas
```

Para utilizar eficazmente estas bibliotecas es necesario importarlas al inicio del script de la siguiente forma:

```
from pybliometrics.scopus import ScopusSearch
import pandas as pd
```

Para obtener resultados precisos de *Scopus* usando *ScopusSearch*, es esencial realizar consultas utilizando una combinación estratégica de filtros, dada la enorme cantidad de registros en esta plataforma. La Tabla 3.1 presenta algunos de los filtros principales, incluyendo su descripción y ejemplos de consulta.

Para examinar la documentación completa acerca de todos los códigos de consulta, el formato y la forma de usarlos en *ScopusSearch* de *pybliometrics*, puede acceder al enlace proporcionado en [10]. En dicha página aparecen en la columna de la derecha todos los códigos de consultas existentes, pudiendo ver tanto su descripción como ejemplos de uso seleccionándolos. Véase reflejado esto en La Figura 3.5.

Estos filtros pueden combinarse con los operadores lógicos `OR`, `AND` y `AND NOT` (enumerados por orden de preferencia) para construir consultas más complejas, ofreciendo flexibilidad y precisión en las búsquedas en *Scopus*.

Tabla 3.1: Principales filtros de *ScopusSearch* que se pueden emplear para llevar a cabo las consultas en la API de *Scopus*.

Código	**Ejemplo de Consulta**	**Descripción**
KEY	`KEY(intelligence)`	Documentos que contienen la palabra *intelligence*.
	`KEY(NOT intelligence)`	Documentos que no contienen la palabra *intelligence*.
	`KEY(intelligence AND dynamics)`	Documentos con ambas palabras *intelligence* y *dynamics*.
	`KEY(intelligence OR dynamics)`	Documentos con al menos una de las palabras *intelligence* o *dynamics*.
PUBYEAR	`PUBYEAR IS 2020`	Documentos publicados en el año 2020.
	`PUBYEAR AFT 2020`	Documentos publicados después de 2020.
	`PUBYEAR BEF 2020`	Documentos publicados antes de 2020.
DOCTYPE	`DOCTYPE(ar)`	Solo artículos de investigación.
	`DOCTYPE(ch)`	Solo capítulos de libros.
LANGUAGE	`LANGUAGE(english)`	Solo documentos en inglés.
	`LANGUAGE(spanish)`	Solo documentos en español.
SUBJAREA	`SUBJAREA(MEDI)`	Solo documentos relacionados con Medicina.
	`SUBJAREA(ARTS)`	Solo documentos relacionados con Artes y Humanidades.
AUTHOR	`AUTHOR(Smith)`	Solo documentos con *Smith* en el nombre o apellido de los autores.
	`AUTHOR(jk)`	Solo documentos con *jk* en el nombre o apellido de los autores.

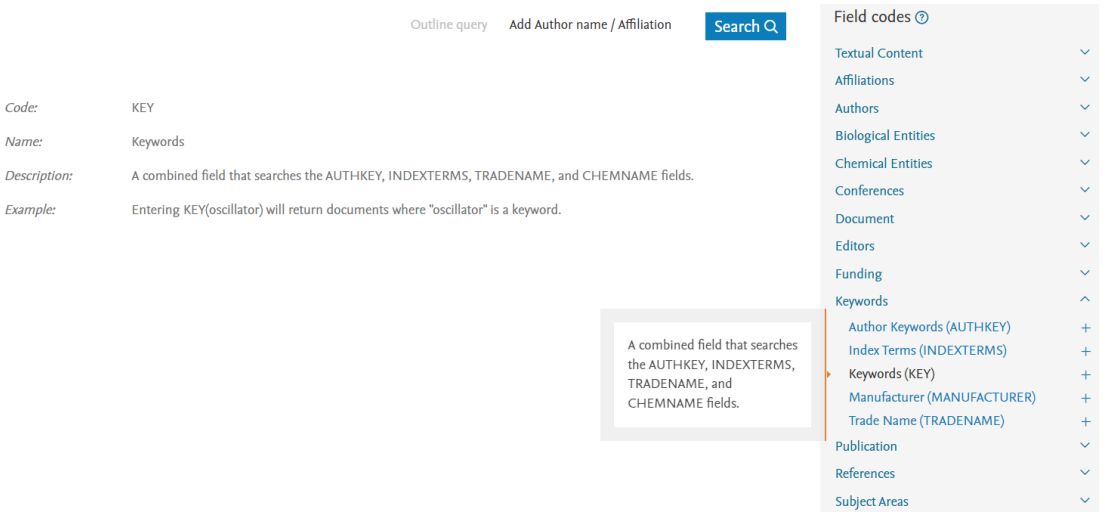

Figura 3.5: Ilustración que captura la página que documenta todos los posibles códigos de consulta.

La consulta que utilizaremos como ejemplo para guiar esta metodología y las siguientes fases se centra en obtener todos los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contengan las palabras clave *complex*, *networks* y *dynamics*. El formato de la consulta específica es:

```
KEY(complex AND networks AND dynamics) AND PUBYEAR AFT 2020
AND PUBYEAR BEF 2024
```

Este ejemplo servirá como referencia para ilustrar cada fase del proceso, proporcionando un seguimiento y una comprensión detallada de la metodología empleada en este trabajo.

Utilizando la *API key* obtenida previamente y la consulta deseada usando los filtros anteriores, se inicia el proceso creando una instancia de la clase *ScopusSearch*. Esta instancia actúa como un enlace esencial entre el código y la API de *Scopus*, permitiendo realizar búsquedas y obtener resultados específicos según los parámetros establecidos mediante los filtros.

Posteriormente, se emplea la biblioteca *pandas* de *Python* para crear un *DataFrame*, una estructura tabular que organiza la información de manera eficiente, facilitando su manipulación y análisis. Es importante destacar que no se pueden copiar directamente los resultados obtenidos de la consulta de *ScopusSearch* en el archivo CSV, ya que pueden presentar formatos inusuales. Por ello, se utiliza el *DataFrame* como una herramienta intermedia que permite estructurar y organizar la información de manera clara y eficiente.

Durante la iteración sobre los resultados de la búsqueda, se extraen los nombres de los autores de cada documento, los cuales se ubican en los campos

`author_names`. Si en lugar de los autores nos interesara otro campo, se sustituiría este por el correspondiente. Esto solo se hará para aquellos documentos en los cuales los autores estén correctamente definidos, por lo que los documentos sin autores no se tendrán en cuenta. Para cada documento válido, se copian todos los autores en un *DataFrame* organizados en una columna. Posteriormente, se transpone el *DataFrame*, obteniendo así los autores organizados en una sola fila, la cual será copiada al *DataFrame* original. Este método, repetido para cada documento, garantiza un procesamiento eficiente de la información y una estructura clara en el *DataFrame* resultante.

Finalmente, la información organizada en el *DataFrame* se guarda en un archivo CSV. Este archivo constituye una lista completa de autores relacionados con la búsqueda específica realizada en *Scopus*, en el que en cada fila estarán todos los nombres de los autores que colaboran en un mismo documento, habiendo tantas filas como documentos. De esta forma, se presenta la información de manera accesible y lista para la siguiente fase.

3.2. Almacenaje en base de datos

Concluida la fase de web scraping y obtenido el archivo CSV que almacena los nombres de los autores organizados por los documentos correspondientes a la consulta realizada, se procede a la segunda etapa. En esta fase, se efectúa la transferencia de toda esta información a una base de datos, facilitando así su manipulación en la fase final del trabajo.

La base de datos seleccionada para este proyecto es *MySQL*, una opción familiar y sencilla que he utilizado previamente en la universidad. Concretamente usaremos *MySQL Workbench*, una herramienta de administración y desarrollo que te permite conectarte y gestionar bases de datos *MySQL*. Actualmente, estoy utilizando la versión *8.0.32*. Se puede descargar *MySQL Server* en el enlace proporcionado en [11], el cual también incluye la opción de *MySQL Workbench*.

Una vez instalado tanto *MySQL Server* como *MySQL Workbench* siguiendo las instrucciones del asistente de instalación, es necesario crear una nueva conexión en *MySQL Workbench*. Desde la página de bienvenida que se muestra en La Figura 3.6 tras iniciar la aplicación, debes pulsar en el botón `+` situado al lado de *MySQL Connections* (recuadrado en rojo). Después, aparecerá una ventana emergente como la de La Figura 3.7, donde deberás introducir los siguientes detalles, dejando el resto por defecto:

- `Connection Name`: asigna un nombre descriptivo a tu conexión.
- `Username`: ingresa el nombre de usuario *root* o el nombre de otro usuario que hayas configurado durante la instalación.

- `Password`: ingresa la contraseña asociada al usuario (*Store in Vault* para almacenarla).

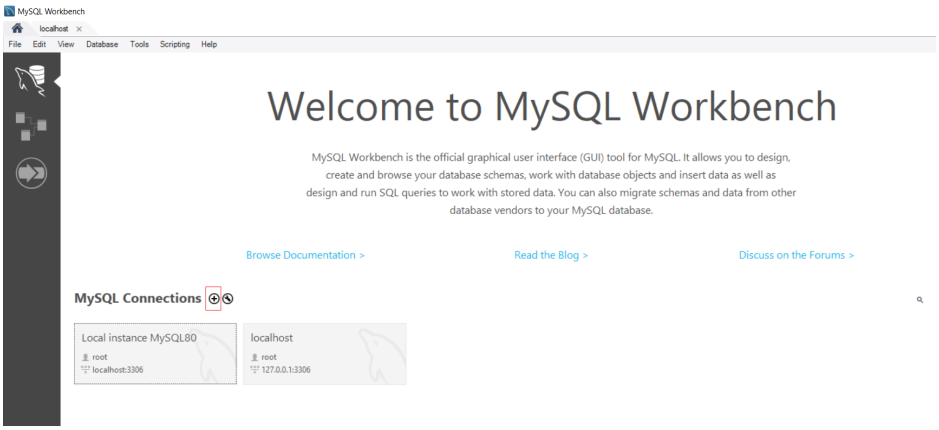

Figura 3.6: Ilustración que captura la pantalla de bienvenida de *MySQL Workbench* señalando con un recuadro en rojo el icono que se debe pulsar para crear una nueva conexión.

Tras pulsar en *OK*, la conexión a la base de datos estará creada. Es importante recordar que el usuario y la contraseña son específicos del servidor *MySQL* y están vinculados a la configuración que se realizó en la instalación de *MySQL Server*.

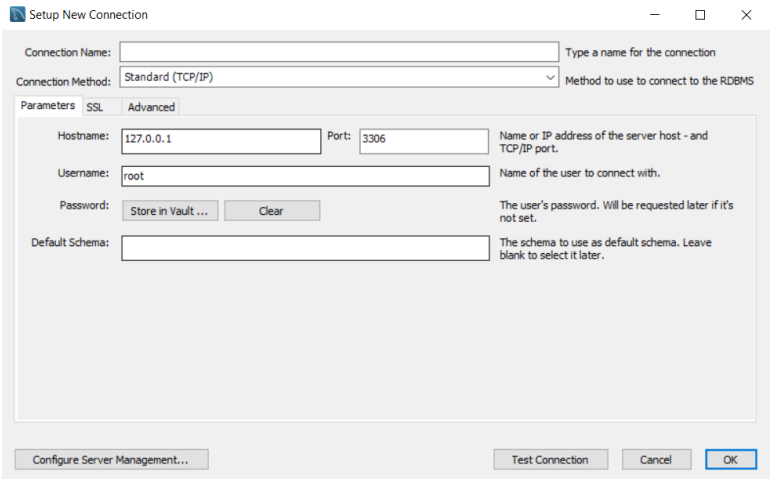

Figura 3.7: Ilustración que captura la ventana emergente en la que hay que introducir los datos de la nueva conexión que se desea crear.

Una vez dentro de la conexión creada, nos encontraremos con una pantalla similar a la ilustrada en La Figura 3.8. En este momento, se nos brinda la oportunidad de crear un nuevo esquema, lo que implica establecer la estructura lógica y la disposición de los elementos de la base de datos que estamos creando.

Desde la pantalla de la recién creada conexión, visualizada en la Figura 3.8, para generar un esquema nuevo, simplemente selecciona el icono recuadrado en rojo. A continuación, introduce un nombre adecuado para el nuevo esquema y confirma la acción pulsando *Apply*. Este proceso te permitirá crear eficientemente un esquema, organizando la estructura lógica de la base de datos de manera personalizada.

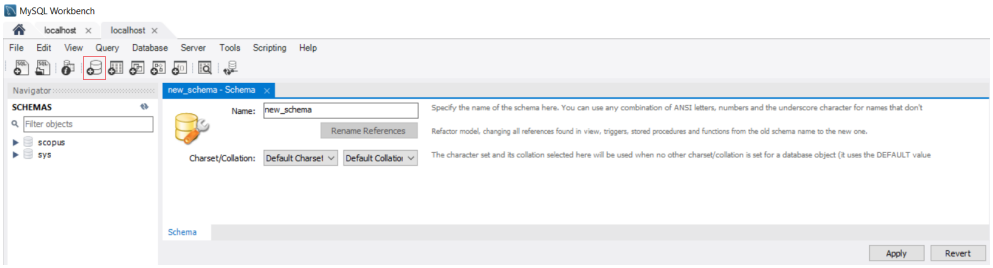

Figura 3.8: Ilustración que captura la manera de crear un nuevo esquema en la conexión creada señalando con un recuadro en rojo el icono que se debe pulsar.

Una vez seguidos todos estos pasos, tenemos todo listo para introducirnos en el código del script.

Existen bibliotecas en *Python* diseñadas específicamente para interactuar eficientemente con bases de datos *MySQL*, y una de las opciones más utilizadas, y la que vamos a usar, es *mysql-connector-python*. Esta biblioteca proporciona un conector *MySQL* para *Python*, permitiendo gestionar conexiones a bases de datos *MySQL* y ejecutar consultas de manera efectiva. Para incorporar *mysql-connector-python* en el entorno de desarrollo, simplemente hay que ejecutar el siguiente comando en la terminal:

```
pip install mysql-connector-python
```

Además, *Python* incluye una biblioteca integral (no es necesario instalarla) llamada *csv* que facilita la manipulación de archivos en formato CSV. Esta biblioteca ofrece funciones y herramientas esenciales para la lectura y escritura eficientes de datos en archivos CSV. A través de ella, vamos a tener acceso a métodos que facilitan la manipulación de filas y columnas en archivos CSV, permitiendo operaciones como la lectura de líneas, la división de datos y la realización de tareas relacionadas con la manipulación de datos tabulares.

Para utilizar eficazmente estas bibliotecas en nuestro script es necesario importarlas al inicio de la siguiente forma:

```
import mysql.connector
import csv
```

A continuación, es necesario establecer una conexión con nuestra base de datos *MySQL* mediante la utilización del recién importado módulo *mysql.connector*. La configuración de la conexión involucra parámetros esenciales como el `host`, que especifica la ubicación de la base de datos (en mi caso, `localhost`, indicando que se encuentra en el mismo servidor), el `user`, que determina el nombre de usuario (en mi caso utilizo el usuario `root`), la `password`, que determina la contraseña asociada a dicho usuario, y la `database`, que hace referencia al nombre del esquema de la base de datos en el cual vamos a almacenar la información (en mi caso lo he denominado `scopus`). Es fundamental gestionar con diligencia las credenciales de acceso para asegurar la integridad y seguridad de la conexión durante la ejecución del proyecto.

Una vez establecida la conexión con la base de datos, necesitamos crear un `cursor` para interactuar con ella desde nuestro script en *Python*. Un `cursor` actúa como una herramienta para ejecutar comandos *MySQL* y recuperar resultados de la base de datos. Imagina la conexión como el acceso a una biblioteca y el `cursor` como un explorador de libros dentro de esa biblioteca. Necesitamos el `cursor` para buscar, añadir, o modificar información en la base de datos. Es esencial porque nos permite realizar operaciones específicas de manera dinámica y eficiente.

En la esencia de cualquier base de datos reside la estructura organizativa de sus tablas. Estas tablas desempeñan un papel fundamental al actuar como elementos clave para la organización y almacenamiento coherente de datos. Cada tabla se convierte en un contenedor lógico que representa una entidad específica y sus relaciones, lo que, a su vez, permite una gestión eficaz de la información y la ejecución de consultas optimizadas.

En este mismo momento, contamos con los datos en un archivo CSV que contiene información sobre los autores de diversos documentos. La estrategia para almacenar estos datos de manera eficiente implica la creación de tres tablas distintas: una para los autores, otra para los documentos y una tercera que establezca las relaciones entre ellos. De esta manera, cada tabla cumple un propósito claro y contribuye a la cohesión y eficiencia del sistema.

Esta organización en tablas, reflejada en La Figura 3.9, facilita la gestión de la información, permitiendo un acceso rápido y estructurado a los datos, un aspecto crucial en la implementación de bases de datos efectivas.

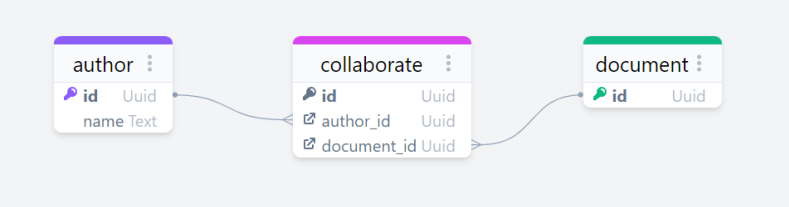

Figura 3.9: Esquema de la base de datos con las tablas `author`, `document` y `collaborate`.

En la figura 3.9 se presenta el esquema antes mencionado que debe tener la base de datos, que consta de tres tablas esenciales: `author`, `document` y `collaborate`. Cada tabla desempeña un papel crucial en la representación y organización de la información. A continuación, se detallan los campos clave de cada tabla:

- Tabla `author`
  - `id`: Identificador único y clave primaria (PK) de la tabla, utilizado como identificación del autor.
  - `name`: Campo de texto que almacena el nombre del autor.
- Tabla `document`
  - `id`: Identificador único y clave primaria (PK) de la tabla, utilizado como identificación del documento.
- Tabla `collaborate`
  - `id`: Identificador único y clave primaria (PK) de la tabla, utilizado como identificación de la relación de colaboración.
  - `author_id`: Clave foránea (FK) que referencia al `id` en la tabla `author`, estableciendo una relación con los autores.
  - `document_id`: Clave foránea (FK) que referencia al `id` en la tabla `document`, estableciendo una relación con los documentos.

Con esta estructura, la tabla `collaborate` opera como una tabla de unión, permitiendo la relación muchos a muchos entre autores y documentos. Este tipo de relación hace referencia a cómo varias instancias de una entidad (en este caso, autores y documentos) pueden estar relacionadas entre sí a través de una tabla de unión (en este caso, collaborate). La presencia de múltiples filas en `collaborate` con diferentes combinaciones de `author_id` y `document_id` permite representar las diversas colaboraciones, proporcionando así flexibilidad en el modelado de las relaciones. Desglosemos estas relaciones:

- Autores (Tabla `author`)
  - Un autor puede colaborar en varios documentos (relación uno a muchos).
  - Pueden existir varios autores diferentes que colaboran en el mismo documento.

- Documentos (Tabla `document`)
  - Un documento puede tener la colaboración de varios autores (relación uno a muchos).
  - Pueden existir varios documentos diferentes en los que un mismo autor ha colaborado.
- Autores colaborando en documentos (Tabla `collaborate`)
  - Esta tabla actúa como una tabla de unión que permite establecer las relaciones muchos a muchos.
  - Cada entrada en la tabla `collaborate` representa una colaboración específica entre un autor y un documento.
  - La presencia de múltiples filas con diferentes combinaciones de `author_id` y `document_id` permite representar las diversas colaboraciones.

Por ejemplo, si se tienen autores A1, A2 y A3, y documentos D1, D2 y D3, la tabla `collaborate` podría contener entradas como (A1, D1), (A2, D2), (A2, D3), (A3, D3), indicando quiénes colaboran en qué documentos. Cada entrada en la tabla `collaborate` representa una colaboración específica entre un autor y un documento, identificada por su propio `id` único.

Aunque la creación de tablas desde el entorno visual del *MySQL Workbench* es una opción válida, optamos por una aproximación más conveniente y flexible: la creación de tablas mediante código. Este enfoque ofrece un mayor control sobre la estructura de las tablas y facilita la reproducción y compartición del código.

Para mantener la integridad y coherencia de la base de datos, hemos adoptado la práctica de eliminar todas las tablas existentes antes de crear las nuevas cada vez que se ejecuta el script. Para evitar errores de eliminación de tablas debido a la existencia de claves foráneas y relaciones entre ellas, es necesario desactivarlas previamente. Este procedimiento ayuda a sortear posibles problemas con tablas preexistentes y garantiza que cada ejecución proporcione un conjunto limpio y actualizado de tablas.

Las consultas *SQL* que se han empleado tanto para la eliminación y creación de tablas han sido las siguientes, usando la función *execute()* en la conexión creada:

- Eliminación de una tabla (repetir con todas las existentes): `DROP TABLE table_name`.
- Creación de la tabla `author`: `CREATE TABLE author (id INT NOT NULL AUTO_INCREMENT, name VARCHAR(100) NOT NULL, PRIMARY KEY (id))`.

- Creación de la tabla `document`: `CREATE TABLE document (id INT NOT NULL AUTO_INCREMENT, PRIMARY KEY (id))`.
- Creación de la tabla `collaborate`: `CREATE TABLE collaborate (id INT NOT NULL AUTO_INCREMENT, author_id INT NOT NULL, document_id INT NOT NULL, PRIMARY KEY (id), FOREIGN KEY (author_id) REFERENCES author(id), FOREIGN KEY (document_id) REFERENCES document(id))`.

Después de completar este proceso, hemos logrado con éxito crear nuestras tablas en nuestra base de datos. Este funcionamiento adecuado se refleja claramente en La Figura 3.10, que muestra las tablas creadas.

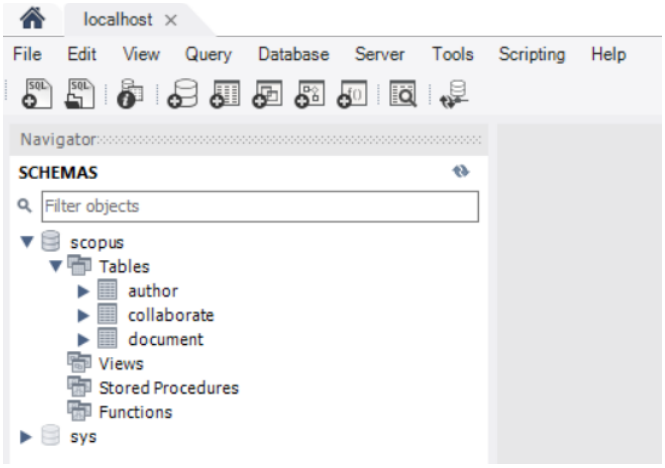

Figura 3.10: Ilustración que captura la correcta creación de las tablas en el esquema de la base de datos.

Para procesar la información almacenada en el archivo CSV y posteriormente insertarla en las tablas de nuestra base de datos relacional *MySQL*, el código sigue una secuencia de pasos. Estos pasos están diseñados para gestionar los datos de manera efectiva, asegurando la integridad y consistencia en la base de datos.

En una primera etapa, se procede a la apertura y lectura del archivo CSV mediante el uso del módulo *csv*, facilitando así la manipulación de los datos tabulares contenidos en dicho archivo. La especificación de la codificación UTF-8 es crucial para garantizar el manejo correcto de posibles caracteres especiales presentes en los nombres de los autores.

Después de abrir y leer el archivo CSV, procedemos a iterar sobre cada fila, que representa un documento específico.

Durante cada iteración, nos centramos en extraer y limpiar los nombres de los autores asociados a ese documento. Este proceso de limpieza nos lleva a crear

una lista que contiene los nombres depurados de los autores, eliminando espacios en blanco innecesarios. Esta lista eficiente y concisa de nombres se utiliza para evitar la inclusión de valores nulos o vacíos.

Luego, introducimos el nuevo documento en la tabla `document` de la base de datos, asociándole un ID ascendente, el cual guardamos.

Posteriormente, por cada nombre en la lista, realizamos una verificación en nuestra base de datos mediante consultas para confirmar que dicho nombre no está ya almacenado, evitando duplicados. Si el autor no existe aún, procedemos con su inserción en la tabla `author` de la base de datos y guardamos su nuevo ID asignado. Si, por el contrario, el autor ya está presente, guardamos su correspondiente ID asignado con su primera inserción.

Tras comprobar la presencia del autor y guardar su ID asignado, procedemos a insertar en la tabla `collaborate` la relación entre el ID del documento recién creado y el ID del autor en cuestión.

Al concluir este proceso para todos los nombres de la lista, tanto el autor, como todos los autores de ese documento y sus respectivas relaciones han sido insertadas correctamente en la base de datos, asegurando la integridad y evitando duplicaciones. Por ejemplo, si tenemos `ID_Author_1`, `ID_Author_2`, `ID_Author_3` y `ID_Document_1`, se introducirán las relaciones (`ID_Author_1`, `ID_Document_1`), (`ID_Author_2`, `ID_Document_1`) y (`ID_Author_3`, `ID_Document_1`) en la tabla `collaborate`.

Las consultas *SQL* que se han empleado han sido las siguientes:

- Comprobación de la existencia en nuestra base de datos del nombre de un autor: `SELECT id FROM author WHERE name =%s",(author_name))`.
- Introducir en la tabla `author` el nombre de un autor: `INSERT INTO author (name) VALUES (%s)`.
- Introducir en la tabla `document` un documento: `INSERT INTO document () VALUES ()`.
- Introducir en la tabla `collaborate` una relación autor-documento: `INSERT INTO collaborate (author_id, document_id) VALUES (%s,%s)`.

Este flujo de operaciones se replica para cada fila del archivo CSV, es decir, para cada documento existente en el conjunto de datos. De esta manera, si el CSV contiene `x` número de filas, generaremos `x` documentos, realizando `x` iteraciones de este proceso de inserción y relación en la base de datos.

Finalmente, para asegurar la persistencia de todas las inserciones y relaciones realizadas, confirmamos todos los cambios en la base de datos mediante la

ejecución de la función *commit()* en la conexión creada. Este paso es esencial para garantizar que los datos se integren de manera coherente en la base de datos *MySQL*, asegurando la consistencia y fiabilidad de la información almacenada.

Después de completar este proceso, hemos logrado con éxito transferir toda la información almacenada en el archivo CSV a nuestra base de datos, lista para su posterior procesamiento en la siguiente fase.

3.3. Cálculo de centralidad

Llegados a esta tercera y última fase, nos encontramos con que tenemos en nuestra base de datos *MySQL* toda la información que queríamos. Por un lado, todos los documentos resultantes de la consulta. Por otro, los diferentes nombres de todos los autores que colaboran en al menos uno de estos documentos. Por último, todas las relaciones de colaboración de cada autor con cada uno de los documentos.

En total, disponemos de dos entidades bien diferenciadas: autores y documentos. Las relaciones existentes entre ambas genera una compleja red interconectada, una estructura que se modela de manera eficaz a través del uso de los llamados grafos. Como nuestro objetivo final es determinar el autor más influyente, definiremos los autores como los nodos del grafo. Diferentes autores se conectarán entre ellos tomando como referencia los documentos. La idea clara es que si varios autores colaboran en un mismo documento, estén relacionados, convirtiendo a los documentos en aristas. No obstante, surge un problema. Las aristas en los grafos solo son capaces de unir dos nodos, y en nuestra información, tenemos que en un solo documento pueden participar desde un autor hasta un número finito. Es por ello que es necesario modelarlo con los llamados hipergrafos, una extensión de los grafos simples. Los hipergrafos presentan nodos al igual que los grafos, pero con una diferencia clave: las aristas, llamadas hiperaristas, permiten conectar más de dos nodos, adquiriendo una importancia vital en nuestro contexto.

De esta forma, toda nuestra información se modelaría con un hipergrafo, en el cual los autores constituirían los nodos y los documentos constituirían las hiperaristas que conectarían a los diversos nodos. Un ejemplo visual de esta estructura queda reflejado en La Figura 3.11.

En La Figura 3.11 se puede apreciar como los autores del documento 1 son el autor 1, 2 y 3; del documento 2 son el autor 2 y 5; del documento 3 son el autor 4 y 6; y del documento 4 son el autor 5, 6 y 7. Se trata de un ejemplo simple y concreto con pocos autores y documentos para visualizar y entender la estructura. Los hipergrafos que obtengamos en los casos reales con redes de *Scopus* estarán constituidos por una inmensa cantidad de nodos e hiperaristas.

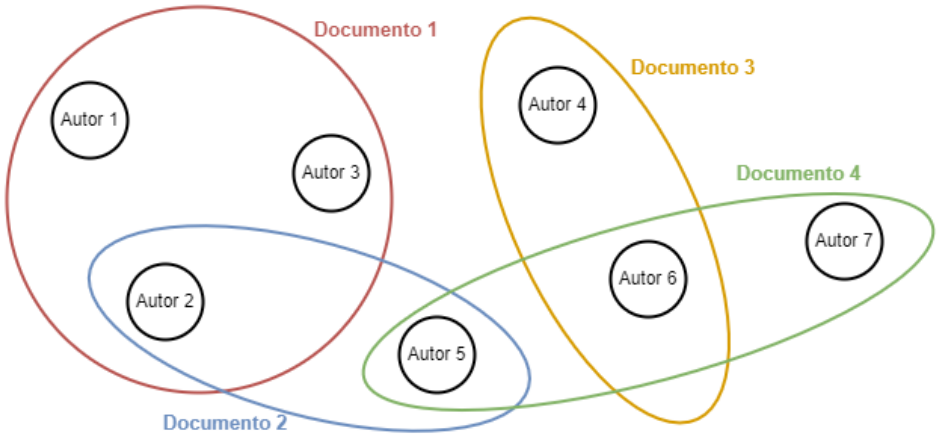

Figura 3.11: Ejemplo simple del tipo de hipergrafo que modela nuestro problema del cálculo del autor más influyente en la red creada.

El objetivo de esta fase coincide con el propósito final de esta metodología, que es determinar el autor más influyente de la red creada. Esto se traduce al cálculo de la centralidad del hipergrafo que modela la red. Para llevar a cabo esta fase se va a tomar como referencia mi Trabajo de Fin de Grado en Matemáticas titulado *Estudio de la centralidad en hipergrafos a través de sus grafos lineales asociados* [1]. En dicho documento, ya mencionado al comienzo del trabajo, se trata teóricamente el estudio de la centralidad en hipergrafos. Además, todas las explicaciones concretas sobre cualquier aspecto de esta fase pueden consultarse en él.

Para el cálculo de la centralidad se van a usar una serie de bibliotecas de *Python*. Necesitamos *mysql-connector-python*, ya instalada y usada en la fase anterior de almacenaje en base de datos, para establecer la conexión con nuestra base de datos e interaccionar con los datos que contiene; *numpy* para realizar cálculos numéricos y trabajar con matrices; *scipy* para calcular los autovalores y autovectores de una matriz; *networkx* para trabajar y realizar operaciones con grafos; y por último, *matplotlib* para representar y visualizar grafos. Para incorporar las nuevas bibliotecas en el entorno de desarrollo, simplemente hay que ejecutar el siguiente comando en la terminal:

```
pip install numpy scipy networkx matplotlib
```

Para utilizar eficazmente estas bibliotecas en nuestro script es necesario importarlas al inicio de la siguiente forma:

```
import mysql.connector
import numpy as np
from scipy.linalg import eig
import networkx as nx
import matplotlib.pyplot as plt
```

Una vez establecida la conexión con nuestra base de datos y creado el cursor para poder interaccionar con ella, el primer paso es recuperar todas las filas de nuestra tabla `collaborate`, puesto que es la que guarda las relaciones entre los autores y los documentos. A partir de ellas necesitamos crear la matriz de adyacencia del grafo lineal asociado al hipergrafo que modela la red.

De ser los autores los nodos y los documentos las hiperaristas en el hipergrafo como se ilustra en La Figura 3.11, en el grafo lineal asociado los nodos pasan a ser los documentos, y dos documentos van a estar unidos por una arista si en ambos colabora un mismo autor común, como se ilustra en La Figura 3.12.

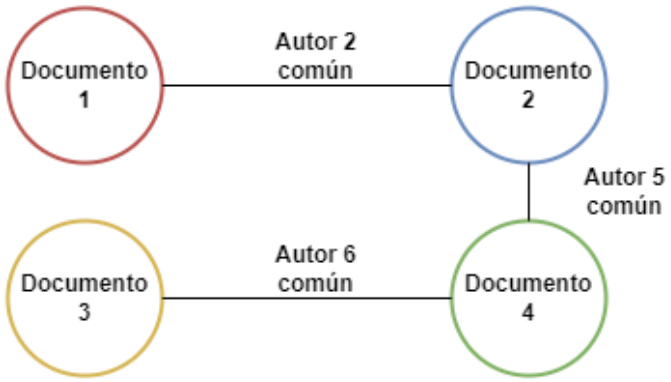

Figura 3.12: Ejemplo del grafo lineal asociado al hipergrafo de La Figura 3.11, para reflejar el tipo de grafo lineal con el que se va a trabajar.

Como se acaba de mencionar previamente, la estructura que modela los datos es un hipergrafo. No obstante, para determinar su centralidad, debemos pasar por su grafo lineal asociado. Es aquí donde contamos con una ventaja. Gracias a la forma en la cual tenemos almacenada la información en la base de datos, no es necesario graficar ni construir el hipergrafo completo, sino que directamente es posible construir el grafo lineal y trabajar a partir de él, ahorrando una costosa labor.

A partir de las relaciones recuperadas de la tabla `collaborate`, se va a crear una lista que va a almacenar en cada posición los IDs de los autores que colaboran en un mismo documento. De esta forma, cada conjunto de la lista puede relacionarse directamente con un documento único, definiendo nuestros nodos del grafo

lineal asociado al hipergrafo. Esto es, el conjunto en la primera posición de la lista hará referencia a los autores que colaboran en el documento con ID 1, el segundo conjunto con los del documento con ID 2, y así sucesivamente. La longitud de dicha lista otorga el número de nodos de dicho grafo lineal, equivalentemente el número de documentos. Posteriormente, con estos dos datos, creamos la matriz de adyacencia del grafo lineal asociado a nuestro hipergrafo.

En este momento, con la matriz de adyacencia creada, interviene una herramienta muy valiosa, el Teorema de Perron-Frobenius. Dicho Teorema asegura la existencia y unicidad en matrices cuadradas, no negativas e irreducibles, de un autovector asociado al autovalor dominante, el cual corresponde con el vector de centralidad del grafo. Por ello, necesitamos verificar que la matriz de adyacencia del grafo lineal cumple tanto la no negatividad como la irreducibilidad.

No obstante, al aplicar este Teorema sobre las matrices de adyacencia de los grafos lineales asociados a los hipergrafos que modelan nuestros datos de *Scopus*, surge un problema. Al aplicar los diversos filtros para obtener los documentos que deseamos, los autores participantes en ellos, en la mayor parte de los casos, están poco relacionados. La cantidad de autores y documentos es tan extensa, que muchos autores apenas participan en un solo documento. Este hecho afecta a nuestro hipergrafo y, por consiguiente, a su grafo lineal asociado, puesto que se crean islas y nodos aislados que no se conectan con ningún otro. Esto hace que el grafo dirigido creado a partir de su matriz de adyacencia no sea fuertemente conexo, y por tanto, la matriz de adyacencia del grafo lineal no sea irreducible. Por ello, hay que tomar medidas para conseguir esta irreducibilidad sin alterar los resultados. La solución más conveniente es la inserción, en caso de no cumplirse las condiciones de dicho Teorema, de un nodo ficticio, también denominado nodo *dummy*. Este nodo ficticio, concretamente un autor ficticio, será introducido como colaborador en cada uno de los documentos. De esta forma, cada autor podrá relacionarse con el resto a través de él, consiguiendo de esta manera la irreducibilidad y el cumplimiento de las condiciones del Teorema de Perron-Frobenius.

Una vez introducido el nuevo nodo en la base de datos junto con todas las relaciones de colaboración, se repite el proceso anterior con los nuevos datos. Tras ello, se llega al mismo punto, la matriz de adyacencia del grafo lineal asociado, en este caso, actualizado con el nuevo nodo, asegurándonos de que es tanto no negativa como irreducible. Una vez cumplidas las condiciones del Teorema, se hallan los autovalores de dicha matriz. De todos ellos, buscamos el autovalor dominante (mayor autovalor en valor absoluto), del cual hallaremos su autovector asociado, normalizado en norma 1.

Este vector obtenido, llamado vector de Perron, nos proporciona la importancia de cada nodo del grafo lineal, lo que implica que en este momento conocemos la importancia de cada hiperarista en el hipergrafo, puesto que los nodos en el grafo lineal equivalen a las hiperaristas en el hipergrafo. Es decir, conocemos la importancia de cada documento. Ahora solo nos falta calcular la de cada autor a

partir de la de los documentos.

Llegados aquí, tenemos que aplicar la fórmula fundamental de extensión de centralidad a hipergrafos para cada nodo, estudiada en [1], y así construir el vector de centralidad que buscamos. Dicha fórmula queda reflejada en pseudocódigo en El Algoritmo 1.

En este algoritmo, se recibe como entradas el vector de Perron recién calculado (*perron*), el número total de nodos del grafo lineal (*lg_nodes*), el tamaño máximo de las hiperaristas del hipergrafo (*max_size_he*), la lista obtenida al principio con los conjuntos de IDs de los autores que participan en cada documento (*hiperedges*) y el número total de nodos del hipergrafo (*hg_nodes*).

Sea n el número total de nodos del hipergrafo, equivalentemente el número total de autores (*hg_nodes*); m el tamaño máximo de las hiperaristas del hipergrafo, equivalentemente el número máximo de autores que colaboran en un solo documento (*max_size_he*); y p el número total de nodos de su grafo lineal asociado, equivalentemente el número de documentos (*lg_nodes*). Tras esta definición, dicho algoritmo tiene una complejidad algorítmica de $o(n \cdot m \cdot p)$.

Algoritmo 1 Algorithm to calculate the centrality vector of the hipergraph

Input: $hg\_nodes, max\_size\_he, lg\_nodes, hiperedges, perron$
Output: *centrality*
Algorithm complexity: $O(n \cdot m \cdot p)$
$centrality \leftarrow [0] * hg\_nodes$
for cada valor i desde 1 hasta $hg\_nodes + 1$ **do**
  $sum \leftarrow [0] * (max\_size\_he - 1)$
  for cada valor j desde 2 hasta $(max\_size\_he + 1)$ **do**
    for cada valor k de 0 hasta $(lg\_nodes - 1)$ **do**
      if longitud de $hiperedges[k] = j$ y el valor de i está en $hiperedges[k]$ **then**
        $sum[j-2] \leftarrow sum[j-2] + perron[k]$
      end if
    end for
    $sum[j-2] \leftarrow sum[j-2]/j$
  end for
  for cada l en sum **do**
    $centrality[i-1] \leftarrow centrality[i-1] + l$
  end for
end for
return centrality

Una vez hallado este vector de centralidad del hipergrafo en el cual cada componente hace referencia a la importancia de cada nodo, lo único que falta es determinar a qué autor corresponde la coordenada con el valor más alto. Sin embargo, es aquí donde tenemos que tener en cuenta la inserción de nuestro nodo o autor ficticio. Como colabora en todos los documentos, es trivial que va a ser el autor cuya componente del vector de centralidad sea la mayor de todas. Por ello, si se trata de un caso en el que se ha insertado el nodo ficticio, no podemos buscar la coordenada más alta, sino que debemos encontrar la segunda componente del vector más alta, y a partir de ella obtener su autor vinculado con una consulta a la base de datos. Si no es necesario introducir el nodo ficticio, se escoge la más alta. De esta forma, estaremos determinando el autor más influyente según la consulta que hayamos realizado en la primera etapa de web scraping.

Tras llevar a cabo todo lo anterior, habremos obtenido el nombre del autor más influyente de todos los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*. Con ello hemos alcanzado nuestro objetivo con éxito y se da por finalizada la metodología empleada.

4

Resultados

Una vez explicada la metodología aplicada con todo tipo de detalles, a continuación se presentan los resultados obtenidos en cada una de las fases.

4.1. Web scraping

El objetivo de esta fase era obtener en un archivo CSV todos los nombres de los autores que han colaborado en cada uno de los documentos obtenidos de la consulta realizada, correspondiente a todos los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*.

En la Figura 4.2, se presenta el formato del archivo CSV resultante del web scraping con la consulta definida previamente, donde se registran los nombres de todos los autores. Cada fila representa un documento único, y todos los autores en una misma fila colaboran en este mismo documento.

La Figura 4.1 muestra la coincidencia exacta entre los resultados obtenidos mediante el web scraping y aquellos obtenidos al realizar la búsqueda directamente desde la interfaz web de *Scopus*, lo que valida la eficacia del proceso. En ella, se aprecia que se han obtenido 2.496 documentos en ambos casos, y si se revisan en ambas fuentes coinciden todos los resultados. En este caso el número coincide debido a que todos los documentos que cumplen los parámetros de la consulta presentan autores. No obstante, es probable dependiendo de la consulta

obtener un número mínimamente menor mediante el web scraping que mediante la búsqueda desde la interfaz de *Scopus* debido a que durante el web scraping solo se tienen en cuenta aquellos con el campo de autor no vacío como ya se ha explicado anteriormente, puesto que es la entidad de estudio de este trabajo.

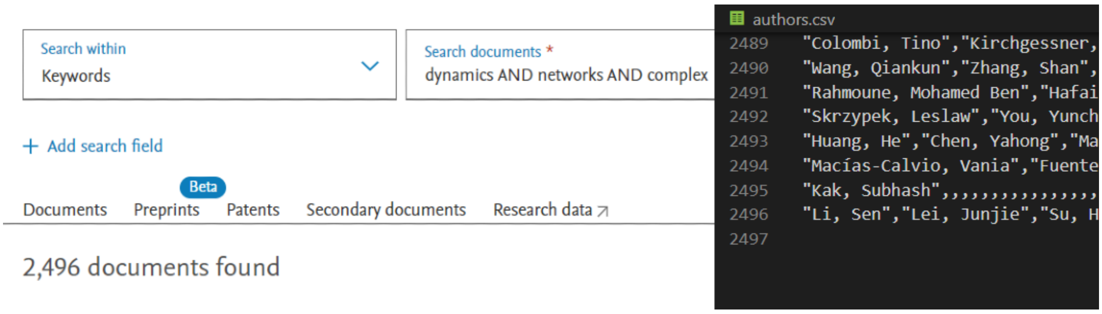

Figura 4.1: Comparación de los resultados obtenidos mediante el web scraping y la búsqueda directa en la página web de *Scopus*.

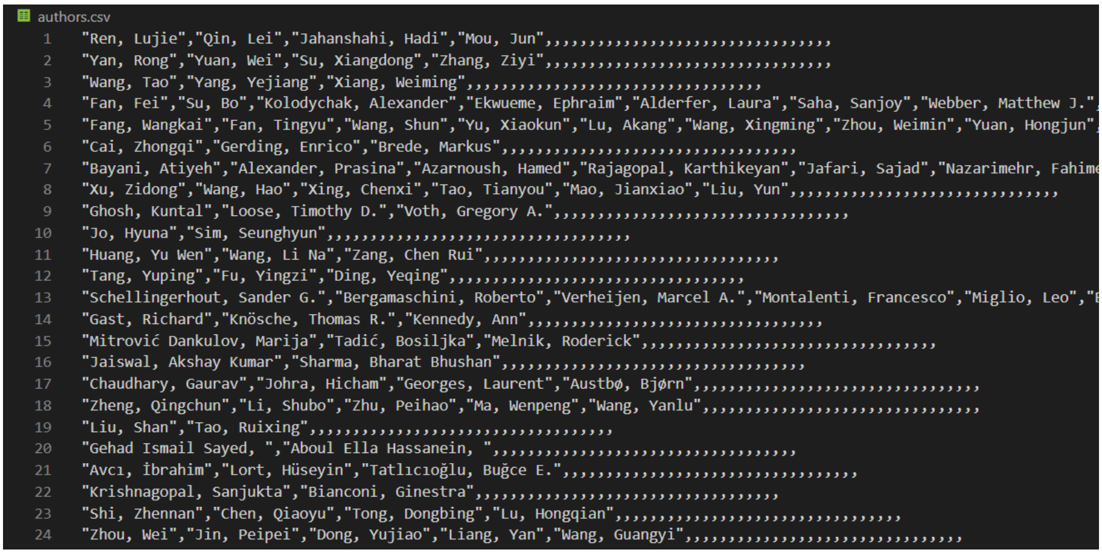

Figura 4.2: Ilustración del archivo CSV con el formato en el que se guardan todos los nombres de los autores clasificados por documentos.

4.2. Almacenaje en base de datos

El objetivo de esta fase era transferir y almacenar toda la información del archivo CSV obtenido en la fase anterior a la base de datos *mySQL*.

El funcionamiento correcto de este proceso se refleja claramente en La Figura 4.3, donde se muestran todas las tablas creadas con los datos almacenados. Debido a la inmensa cantidad de entradas en cada una de las tablas, solo se muestran las primeras ejemplificándolo.

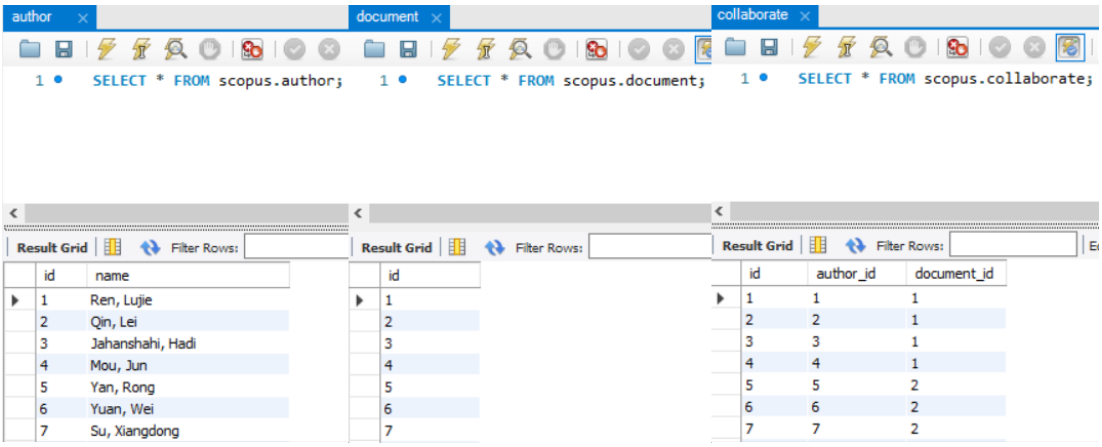

Figura 4.3: Ilustración que captura la correcta inserción de los datos en las tablas de la base de datos.

4.3. Cálculo de la centralidad

El objetivo de esta fase era determinar el autor más influyente a partir del cálculo de la centralidad de vector propio del hipergrafo que modela los datos almacenados en la base de datos en la fase anterior.

Como paso intermedio, se ha generado el grafo lineal asociado al hipergrafo, cuya representación se muestra en La Figura 4.4. Aunque la representación gráfica no es indispensable, se ha considerado útil incluirla en el código por si se requiere. Gracias a la manera en que hemos almacenado la información y capturado las relaciones en la base de datos, no ha sido necesario construir el hipergrafo que modela este problema, ya que hemos creado directamente su grafo lineal asociado, lo que ha ahorrado tiempo y complejidad adicionales. Como se observa en la figura, debido a la gran cantidad de documentos, el número de nodos es apenas discernible.

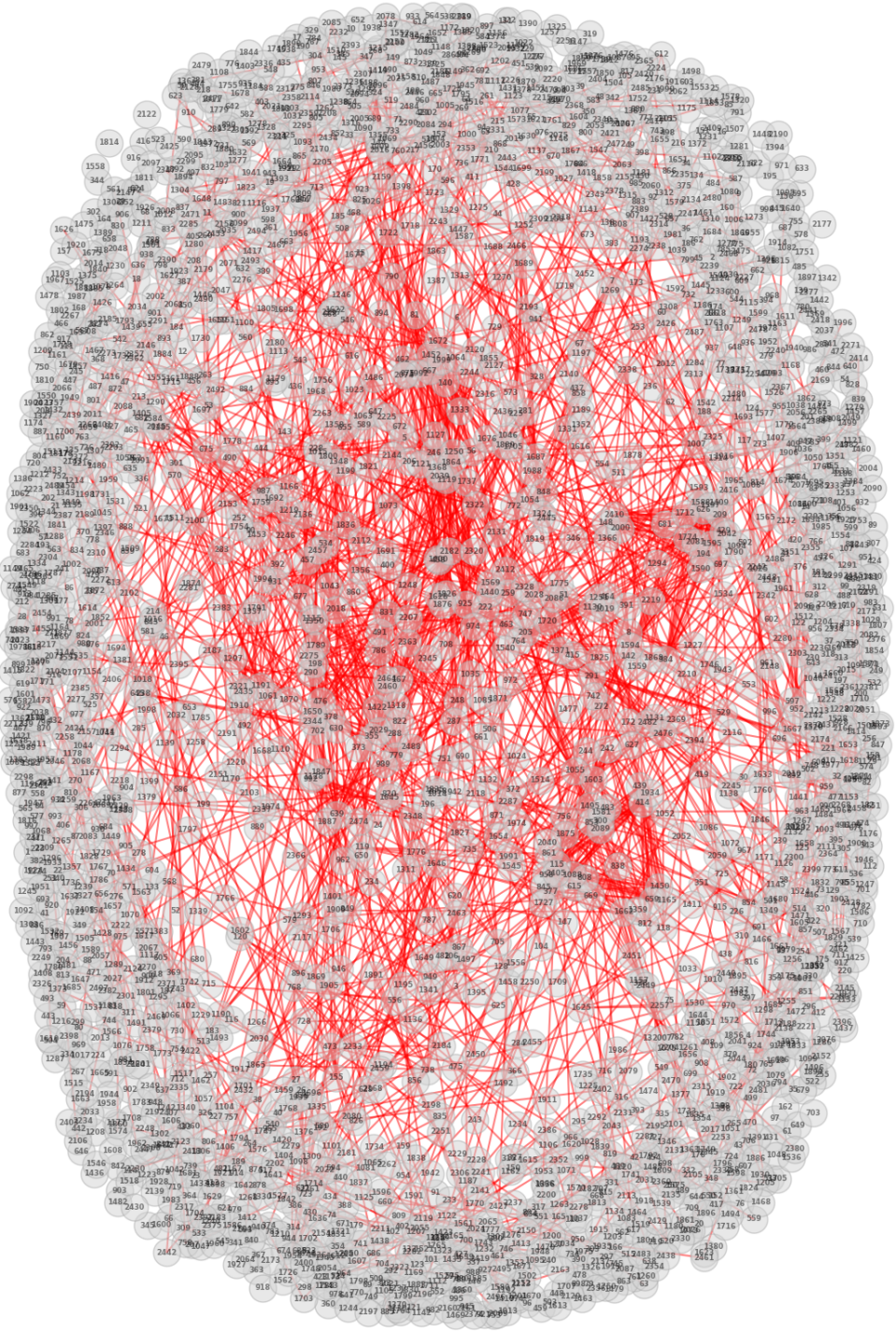

Figura 4.4: Ilustración que refleja el grafo lineal asociado al hipergrafo que modela los datos obtenidos de la consulta de los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*.

Como se puede apreciar en La Figura 4.5, algunos nodos no están conectados a ningún otro, lo que genera que su matriz de adyacencia no sea irreducible y no se cumplan las condiciones del Teorema de Perron-Frobenius, como se ha explicado anteriormente en la metodología. Para resolver esta situación, se introduce un nodo ficticio, lo que resulta en la generación del nuevo grafo lineal representado en La Figura 4.5. Aunque se muestra para propósitos ilustrativos, los grafos lineales que se crean tras la adición de este nodo no aportan significativamente debido a la dificultad para poder distinguir algo (las representación de todas las aristas genera prácticamente un fondo de color sólido debido a la cantidad), por lo que no se ha incluido en el código y se presenta de manera complementaria.

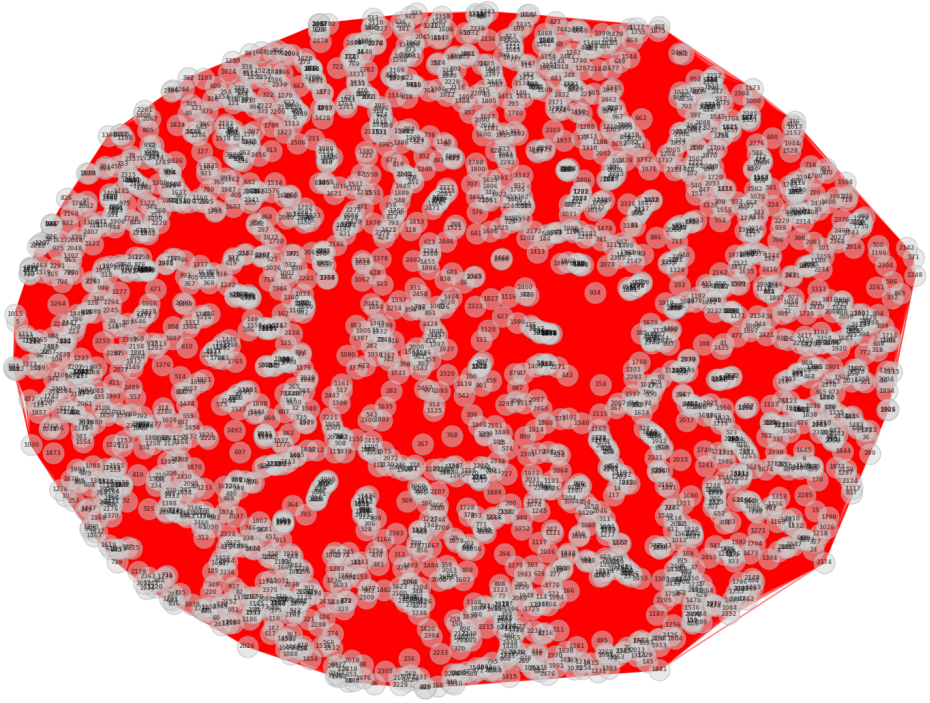

Figura 4.5: Ilustración que refleja el grafo lineal asociado al hipergrafo tras la inserción del nodo ficticio que modela los datos obtenidos de la consulta de los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*.

La Figura 4.6 demuestra y refleja el resultado final obtenido. El autor más influyente entre los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*, es *Kun Guo*.

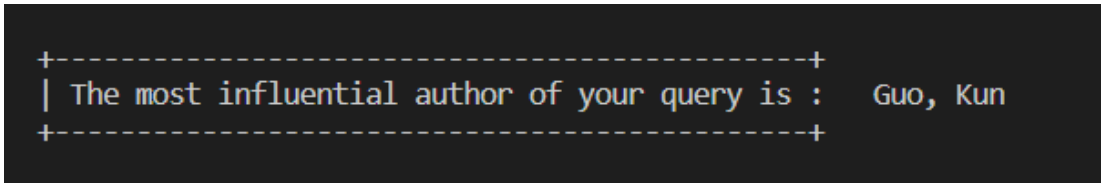

Figura 4.6: Ilustración que captura el nombre del autor más influyente respecto a todos los documentos publicados entre los años 2021 y 2023, ambos incluidos, que contienen las palabras clave *complex*, *networks* y *dynamics*.

4.4. Tiempos de ejecución

En el contexto de este proyecto, resulta fundamental comprender y evaluar detenidamente el tiempo requerido de ejecución para cada una de las fases clave: web scraping, almacenamiento en base de datos y cálculo de centralidad. Con este propósito, se ha llevado a cabo un registro de los tiempos de ejecución, abarcando una diversidad de consultas que incluyen un amplio rango de datos. Todo esto queda reflejado en La Tabla 4.1.

Este enfoque nos permite no solo comprender el desempeño actual del sistema, sino también proyectar su escalabilidad futura. Al medir y analizar los tiempos de ejecución en función de la cantidad de registros involucrados en cada consulta, obtenemos información valiosa sobre cómo nuestro sistema responde a diferentes cargas de trabajo y volúmenes de datos.

Es esencial comprender que los tiempos presentados son estimaciones basadas en el rendimiento de mi portátil personal. En el proceso de ejecución, multitud de variables pueden afectar a los tiempos registrados. Estos factores pueden ir desde la capacidad del hardware hasta la complejidad de las consultas realizadas. Además, el entorno de ejecución, el estado del sistema operativo y la carga de procesamiento en el momento de la prueba también pueden influir significativamente en el tiempo de ejecución. Por lo tanto, aunque los tiempos proporcionados ofrecen una valiosa referencia, es importante interpretarlos con cautela y considerarlos como indicativos del rendimiento potencial, sujeto a variabilidad en diferentes entornos y condiciones.

Inicialmente, se consideró almacenar los tiempos con números más elevados de registros, como 100 mil, 200 mil o 500 mil. Sin embargo, el proceso de web scraping solo puede realizarse en el campus de mi universidad, utilizando su red

wifi. Dado el tiempo que llevaría, especialmente teniendo en cuenta los registros más bajos, esta opción no resulta factible.

Dado que estoy desarrollando el proyecto en un portátil estándar, la cantidad masiva de datos excede a partir de cierta cifra la capacidad de memoria de mi equipo. Como resultado, algunas fases del cálculo de la centralidad no pueden completarse en mi portátil (marcadas con N/A en la celda correspondiente en La Tabla 4.1). Estas tareas podrían ejecutarse con éxito y en menos tiempo en un dispositivo o máquina con mayores capacidades y potencia.

Tabla 4.1: Tiempos de ejecución de las tres fases de la metodología empleada en función de diferentes consultas en *Scopus*. Los tiempos marcados con N/A no están disponibles debido a la falta de capacidad del dispositivo de ejecución para manejar tales cifras de datos.

Consulta	Nº registros	Tiempo WS	Tiempo DB	Tiempo Centralidad
`KEY(supermarket) AND PUBYEAR IS 2010`	98	3,49 s	0,54 s	0,99 s
`KEY(supermarket) AND PUBYEAR IS 2020`	267	10,9 s	1,67 s	1,92 s
`KEY(chloroplast) AND PUBYEAR IS 2013`	1.176	1,32 min	13,18 s	25,3 s
`KEY(chloroplast) AND PUBYEAR AFT 2020 AND PUBYEAR BEF 2023`	4.183	3,99 min	2,18 min	10,73 min
`KEY(student) AND PUBYEAR IS 2008 AND DOCTYPE(ar)`	8.693	6,62 min	3,20 min	39,82 min
`KEY(student) AND PUBYEAR IS 2009`	16.113	11,97 min	7,45 min	1 h 43 min
`KEY(knowledge) AND PUBYEAR IS 2017`	31.762	24,12 min	40,84 min	N/A
`KEY(knowledge) AND PUBYEAR AFT 2016 AND PUBYEAR BEF 2019`	66.224	1 h 7 min	2 h 47 min	N/A

5

Conclusiones y trabajos futuros

5.1. Conclusiones

Este Trabajo de Fin de Grado se integra perfectamente con mi investigación en el Trabajo de Fin de Grado en Matemáticas. La teoría sin aplicaciones prácticas puede carecer de relevancia en determinados contextos. En la actualidad, las bases de datos de información están experimentando un notable auge, y la capacidad de almacenar grandes volúmenes de registros genera la necesidad imperiosa de desarrollar algoritmos para analizarlos y extraer resultados y conclusiones de ellas. Precisamente, este documento aborda esa necesidad. La elección de *Scopus* debido a la vasta cantidad de registros mundiales que almacena ha permitido poder aplicar las medidas de centralidad fiel y exitosamente a los hipergrafos creados a partir de sus datos.

Dada la enorme diversidad de autores y documentos disponibles, resulta de gran utilidad en cualquier ámbito poder determinar quién es el autor más influyente en una temática específica o en un año determinado. Esto permite a los investigadores, académicos y profesionales acceder de manera eficiente a la información relevante y estar al tanto de las contribuciones de los autores más significativos en sus áreas de interés, enriqueciendo significativamente el conocimiento y fomentando su intercambio. En este sentido, he logrado el objetivo de desarrollar una herramienta que permita todo esto.

Como línea de mejora en este trabajo se propone la creación de grafos interactivos, permitiendo el poder moverse por la red aumentando o disminuyendo el

zoom para su correcta visualización, además de resaltar aquellos más influyentes una vez calculada sus centralidades. Esto supondría apenas una funciones en el código debido a la sencillez, pero como dicha visualización no era el objetivo final del trabajo, no se ha implementado. Otra posible línea de mejora es crear una aplicación software amigable que permita a cualquier usuario llevar a cabo este análisis de la centralidad en redes científicas.

5.2. Trabajos futuros

En el ámbito de futuras investigaciones, se abre una amplia gama de opciones para llevar a cabo a raíz de la realización de este Trabajo de Fin de Grado.

Surge la necesidad y la posibilidad de aplicar otros tipos de centralidades, como la centralidad de grado o la intermediación, a las redes creadas a partir de los datos almacenados en la plataforma bibliográfica *Scopus*. Además, en *Scopus*, en lugar de limitarse a determinar los autores más influyentes a través de consultas con filtros, aparece la necesidad de explorar la importancia de otras entidades presentes en la base de datos, creando hipergrafos con otros tipo de entidades distintas.

Asimismo, es posible la aplicación de la centralidad de vector propio, así como de otros tipos de métricas de centralidad como la de grado o de intermediación ya mencionadas con anterioridad, a diferentes bases de datos que almacenan otro tipo de información no académica. Por ejemplo, se podría determinar la influencia de usuarios influyentes en Facebook, la influencia de entidades financieras en la base de datos del Banco de España, y los nodos más importantes en redes de comunicación en bases de datos de telecomunicaciones.

Bibliografía y referencias

[1] S. Hernández Sandoval (2024), *Estudio de la centralidad en hipergrafos a través de sus grafos lineales asociados.* Grado en Matemáticas. Universidad Rey Juan Carlos.

[2] M. J. Johnson, *A concise introduction to programming in Python*, 2012.

[3] J. Goerzen, *Foundations of Python network programming*, 2004.

[4] J. V. Guttag, *Introduction to computation and programming using Python : with application to understanding data*, 2021.

[5] "Interfaz de la pantalla de inicio scopus," https://www.scopus.com/ [Accedido en: 22/12/2023].

[6] "Interfaz de la pantalla de inicio de sesión de scopus," https://id.elsevier.com/as/authorization.oauth2?platSite=SC%2Fscopus&ui_locales=en-US&scope=openid+profile+email+els_auth_info+els_analytics_info+urn%3Acom%3Aelsevier%3Aidp%3Apolicy%3Aproduct%3Aindv_identity&els_policy=idp_policy_indv_identity_plus&response_type=code&redirect_uri=https%3A%2F%2Fwww.scopus.com%2Fauthredirect.uri%3FtxGid%3D17673fc380f0243ee5d7a1c5486b31fb&state=userLogin%7CtxId%3D7B68B0E25B0B45856C9C6CAF184B2650.i-07309c838e5f98617%3A6&authType=SINGLE_SIGN_IN&prompt=login&client_id=SCOPUS [Accedido en: 22/12/2023].

[7] "Interfaz del explorador de scopus," https://www.scopus.com/search/form.uri?display=basic&zone=header&origin=#basic [Accedido en: 22/12/2023].

[8] "Consultar api key de scopus," https://dev.elsevier.com/apikey/manage [Accedido en: 23/12/2023].

[9] "Descargar python," https://www.python.org/downloads/ [Accedido en: 26/12/2023].

[10] "Documentación de consultas de pybliometrics," https://www.scopus.com/search/form.uri?display=advanced [Accedido en: 27/12/2023].

[11] "Descargar mysql server," https://dev.mysql.com/downloads/installer/ [Accedido en: 28/12/2023].