

Zotero, OpenAlex y modelos de lenguaje

Manual técnico para gestionar y ampliar una biblioteca de investigación con Python

Danny Murillo González
Investigador
Universidad Tecnológica de Panamá - CIDITIC
Universidad de Granada, España - Doctorando

CC BY-NC-SA 4.0

Atribución/Reconocimiento-NoComercial-CompartirIgual 4.0 Internacional

05/11/2026

Contenido

	Introducción
1	Las herramientas: Zotero, OpenAlex y OpenRouter
2	Preparación del entorno
3	Biblioteca y colecciones en Zotero
4	Estructura de los ítems: campos, creadores y tipos
5	Documentos, adjuntos y PDF
6	Metadatos: cobertura, etiquetas, duplicados y resúmenes
7	Notas y organización de la biblioteca
8	Búsquedas en OpenAlex
9	De metadatos a contexto para el LLM
10	Consultas a un LLM con OpenRouter
11	Filtro de relevancia y memoria local
12	Taxonomía y palabras clave generadas por el LLM
13	Escribir en Zotero: notas e ítems
14	El agente completo

Manual técnico con Python para consultar colecciones, documentos, metadatos, PDF y notas de Zotero, y ampliar la biblioteca con OpenAlex y modelos de lenguaje mediante OpenRouter.

El manual está pensado para investigadores que mantienen su bibliografía en Zotero. Enseña paso a paso, a identificar colecciones y documentos, comprobar la disponibilidad de los PDF, medir la cobertura de los metadatos, leer el texto completo, completar resúmenes ausentes y registrar notas. Las partes finales conectan la biblioteca con OpenAlex y con un modelo de lenguaje, y terminan con un agente básico de vigilancia bibliográfica.
Se utiliza como ejemplo, una la colección en Zotero de su computador, en este ejemplo la colección *Bibliometria_Forense* (`6APD6L29`), pero se explica como seleccionar otra colección para las pruebas.

Mapa del manual

Parte	Contenido	Herramienta
Introducción	Finalidad, enfoque y alcance	
1	Zotero, OpenAlex y OpenRouter: qué aporta cada servicio	Conceptos
2	Entorno, credenciales, configuración y prueba de conexiones	Preparación
3	Biblioteca y colecciones: identificadores, jerarquía, colección de trabajo y documentos	Zotero
4	Estructura de los ítems: campos, acceso seguro, creadores y tipos	Zotero
5	Documentos, adjuntos y PDF: disponibilidad, texto completo y búsquedas	Zotero
6	Metadatos: cobertura, etiquetas, duplicados, exportación y resúmenes ausentes	Zotero
7	Notas y organización: lectura, creación y pertenencia a colecciones	Zotero
8	Búsquedas en OpenAlex	OpenAlex
9	De metadatos a contexto para un modelo de lenguaje	OpenRouter
10	Consultas a un LLM con OpenRouter	OpenRouter
11	Filtro de pertinencia y memoria local	Agente básico
12	Taxonomía y palabras clave generadas por el LLM	Agente básico
13	Escribir los resultados en Zotero	Agente básico
14	El agente completo	Agente básico

Las partes siguen el orden de las herramientas y, dentro de cada una, avanzan de la lectura a la escritura. Las partes 3 a 7 solo necesitan las credenciales de Zotero; OpenAlex se incorpora en la parte 8 y OpenRouter en las partes 9 y 10. Cada parte usa variables y funciones definidas en las anteriores, por lo que conviene ejecutarlas en orden. Las operaciones que

modifican la biblioteca dependen del interruptor `ESCRIBIR_EN_ZOTERO`, definido en la sección 6.7: con `False` solo se simulan y con `True` escriben en Zotero.

> **Sobre las salidas.** Las celdas que llaman a Zotero, OpenRouter u OpenAlex indican *Salida registrada* cuando conservan la salida de una ejecución real con la colección *Bibliometria_Forense*; las de las partes 2 a 10 proceden de la ejecución del 30 de octubre de 2026. Las celdas que no usan la red indican *Salida generada al ejecutar*. Las celdas sin salida deben ejecutarse con tu biblioteca para ver los resultados.

Introducción

Una biblioteca de Zotero reúne, a lo largo de una tesis o de un proyecto, cientos de referencias con sus PDF, notas y etiquetas. La interfaz gráfica resulta cómoda para leer y citar, pero responde mal a preguntas sobre el conjunto que condicionan cualquier análisis posterior: qué identificador tiene cada colección, qué documentos contiene, cuántos tienen el PDF adjunto, cuántos carecen de resumen o de DOI y dónde quedaron las notas de lectura. Estas preguntas pertenecen al control de calidad del corpus, y conviene resolverlas antes de usar la colección como base de una revisión o de un estudio.

Este manual enseña a responderlas mediante la API web de Zotero y la librería `pyzotero`. El recorrido avanza por pasos: localizar las colecciones y sus identificadores, listar los documentos de una colección, separar los que tienen PDF de los que no, medir la cobertura de los metadatos, revisar etiquetas y posibles duplicados, leer el texto completo de un PDF, completar un resumen ausente a partir del propio PDF y crear notas, tanto vinculadas a un documento como independientes. Cada paso combina una explicación de su uso, el código necesario, la salida obtenida y una pauta para interpretarla.

Las partes siguientes amplían la biblioteca hacia fuentes y herramientas externas. OpenAlex, un índice abierto de la producción científica, permite buscar publicaciones relacionadas con la colección; OpenRouter da acceso a modelos de lenguaje (LLM) a los que se envían los metadatos como contexto. Las últimas partes reúnen estas piezas en un agente básico que propone lecturas nuevas y las registra en Zotero. Ese agente es deliberadamente sencillo y funciona como prueba de integración de las tres API: su interés está en mostrar cómo circulan los datos entre ellas, y no en sustituir una búsqueda bibliográfica sistemática.

Este material se concibió inicialmente como la construcción de un agente bibliográfico, antes de conocer las conexiones entre Zotero y los LLM mediante el Protocolo de Contexto de Modelos (MCP, *Model Context Protocol*). Con un servidor MCP para Zotero, desarrollado por la comunidad, un asistente basado en un LLM puede buscar en la biblioteca, leer metadatos y texto completo o crear notas a partir de instrucciones en lenguaje natural, sin programar cada llamada a la API. Para muchas de las tareas de este manual, esa vía resulta hoy más sencilla y probablemente más adecuada. El trabajo realizado conserva, sin embargo, su valor como práctica y como experiencia: construir cada consulta obliga a conocer la estructura de los datos de Zotero, los permisos de la API, el funcionamiento de OpenAlex y los fallos que aparecen al integrar varios servicios. Ese conocimiento sigue siendo necesario para configurar, evaluar y depurar cualquier conexión automatizada, incluidas las que ofrece MCP.

Los procedimientos tienen límites que el lector debe tener presentes. Un recuento de metadatos indica si un campo tiene valor, no si ese valor es correcto. Un resumen extraído del PDF es una propuesta que requiere verificación, por lo que el manual lo marca con una etiqueta, y las respuestas de un LLM son interpretaciones que deben contrastarse con los documentos. Para seguir el material bastan conocimientos básicos de Python y Jupyter y una cuenta de Zotero; OpenAlex y OpenRouter solo se necesitan desde la parte 8. Las claves se escriben en una celda del notebook (sección 2.5) porque esta versión está pensada para practicar en local.

1

Las herramientas: Zotero, OpenAlex y OpenRouter

El manual trabaja con tres servicios. Cada uno cumple un papel distinto: Zotero guarda lo que ya tienes, OpenAlex ofrece lo que se publica y OpenRouter da acceso al modelo de lenguaje que relaciona ambas cosas.

¿Qué es Zotero?

Zotero es un gestor de referencias bibliográficas libre y de código abierto, desarrollado por la Corporation for Digital Scholarship. Guarda referencias, PDF, notas y etiquetas, y las organiza en **colecciones** (agrupaciones jerárquicas; un mismo ítem puede pertenecer a varias). La aplicación de escritorio sincroniza la biblioteca con tu cuenta de zotero.org.

- **Qué usamos:** la **API web v3** (`https://api.zotero.org`), que permite leer y modificar desde un programa la biblioteca sincronizada. En Python la usamos mediante la librería `pyzotero` .
- **Qué necesitas:** tu **User ID** y una **clave de API** privada (se explica cómo obtenerlos en la parte 2.2).
- **A tener en cuenta:** la API ve la copia sincronizada en zotero.org. Si añades algo en Zotero de escritorio, el programa no lo verá hasta que se sincronice.

¿Qué es OpenAlex?

OpenAlex es un índice abierto de la producción científica mundial, mantenido por la organización sin ánimo de lucro OurResearch. Se lanzó en 2022 como sucesor de Microsoft Academic Graph y reúne registros de obras (artículos, libros, preprints, datos), con sus autores, instituciones, revistas, temas y citas. Sus datos se publican con licencia CC0.

- **Qué usamos:** el punto de acceso `https://api.openalex.org/works` para buscar publicaciones por texto y filtrarlas por año, tipo o número de citas.
- **Qué necesitas:** una clave de API gratuita (parte 2.4). Identifica tus peticiones y da límites de uso más amplios; tu correo (`mailto`) es un complemento opcional.
- **A tener en cuenta:** devuelve el resumen en un formato particular (índice invertido) que hay que reconstruir (parte 8.4).

¿Qué es OpenRouter?

OpenRouter es un servicio que da acceso a modelos de lenguaje (LLM) de muchos proveedores, como Anthropic, DeepSeek, OpenAI, Google o Meta, con **una sola clave** y una API compatible con la de OpenAI. Un LLM es un modelo entrenado con grandes cantidades de texto, capaz de resumir, clasificar y generar texto a partir de instrucciones.

- **Qué usamos:** el endpoint `https://openrouter.ai/api/v1` a través de la librería `openai`. Cambiar de modelo es cambiar una cadena, por ejemplo `deepseek/deepseek-chat-v3.1` o `mistralai/mistral-small-3.2-24b-instruct`.
- **Qué necesitas:** una clave de API (`sk-or-v1-…`) y créditos en la cuenta (parte 2.3). Como alternativa gratuita, el manual permite usar un modelo abierto en tu propio ordenador con Ollama (sección 2.9). Los modelos de pago se cobran por uso, según los tokens enviados y recibidos.
- **A tener en cuenta:** el modelo solo sabe lo que le envías en el mensaje. Por eso la calidad de los metadatos de Zotero condiciona la calidad de la respuesta.

Cómo encajan

Paso	Servicio	Qué aporta
Conocer la colección	Zotero	Colecciones, documentos, PDF, metadatos y notas (partes 3 a 7)
Entender el tema	OpenRouter (LLM)	Perfil temático, taxonomía y frases de búsqueda
Buscar novedades	OpenAlex	Publicaciones recientes relacionadas
Filtrar	OpenRouter (LLM)	Decisión de pertinencia sobre cada candidato
Guardar	Zotero	Resúmenes completados y notas con los resultados

La biblioteca personal delimita el corpus de partida; el índice externo permite ampliar la recuperación; y el modelo ayuda a interpretar el texto disponible. Mantener claras estas funciones permite identificar el origen de un error: un registro incompleto, una búsqueda poco específica o una inferencia inadecuada requieren soluciones distintas. La integración se estudia aquí como intercambio entre sistemas con responsabilidades explícitas.

2

Preparación del entorno

En este apartado se instalan las librerías, se obtienen las claves de Zotero, OpenRouter y OpenAlex y se escriben en una única celda del notebook (parte 2.5), pensada para practicar en local. Todas las partes siguientes dependen de estas variables; si falta alguna, `verificar_config()` lo indica. Al terminar quedan listos los clientes `zot` (Zotero) y `llm` (OpenRouter) y las utilidades comunes.

2.1 Instalación de librerías

Librería	Para qué se usa
`pyzotero`	Cliente de la API web de Zotero (lectura y escritura)
`openai`	Cliente compatible con OpenRouter (mismo formato de mensajes)
`requests`	Llamadas HTTP a la API de OpenAlex
`pypdf`	Extrae el texto de los PDF cuando Zotero no lo tiene indexado (sección 5.3)

Ejecuta la celda una sola vez por entorno.

Un procedimiento académico reutilizable necesita separar las dependencias del programa, las credenciales de acceso y las decisiones de investigación. Esa separación permite adaptar el ejemplo a otra biblioteca y documentar las condiciones de ejecución. Conviene registrar las versiones de las librerías y la fecha de las pruebas para interpretar cambios posteriores en el comportamiento de las API.

```
# %pip install -q pyzotero openai requests pypdf
# %pip install pypdf
```

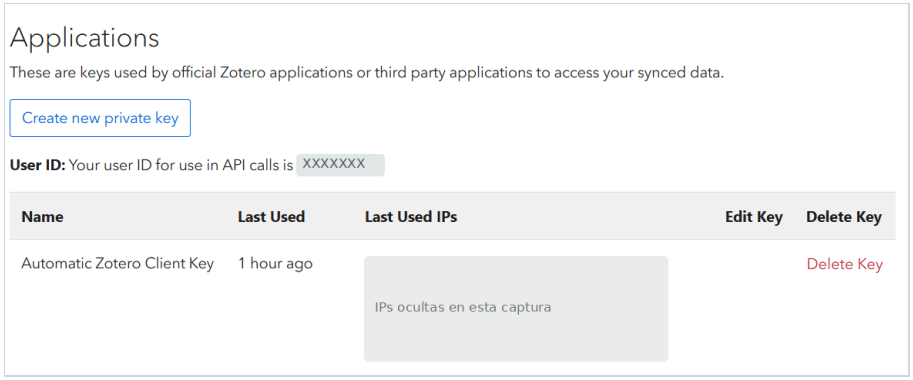
2.2 Credenciales de Zotero: User ID y clave de API

La API de Zotero necesita dos datos: tu **User ID** (número que identifica tu biblioteca) y una **clave privada** que autoriza al programa. Ambos se gestionan en la misma página de zotero.org.

Ver tu User ID

1. Inicia sesión en https://www.zotero.org.

2. Haz clic en tu **nombre de usuario**, en la parte superior derecha, y elige **Settings**.
3. Abre la pestaña **Security**.
4. En el bloque **Applications** aparece la línea *User ID: Your user ID for use in API calls is …*. Ese número va en `ZOTERO_USER_ID`.



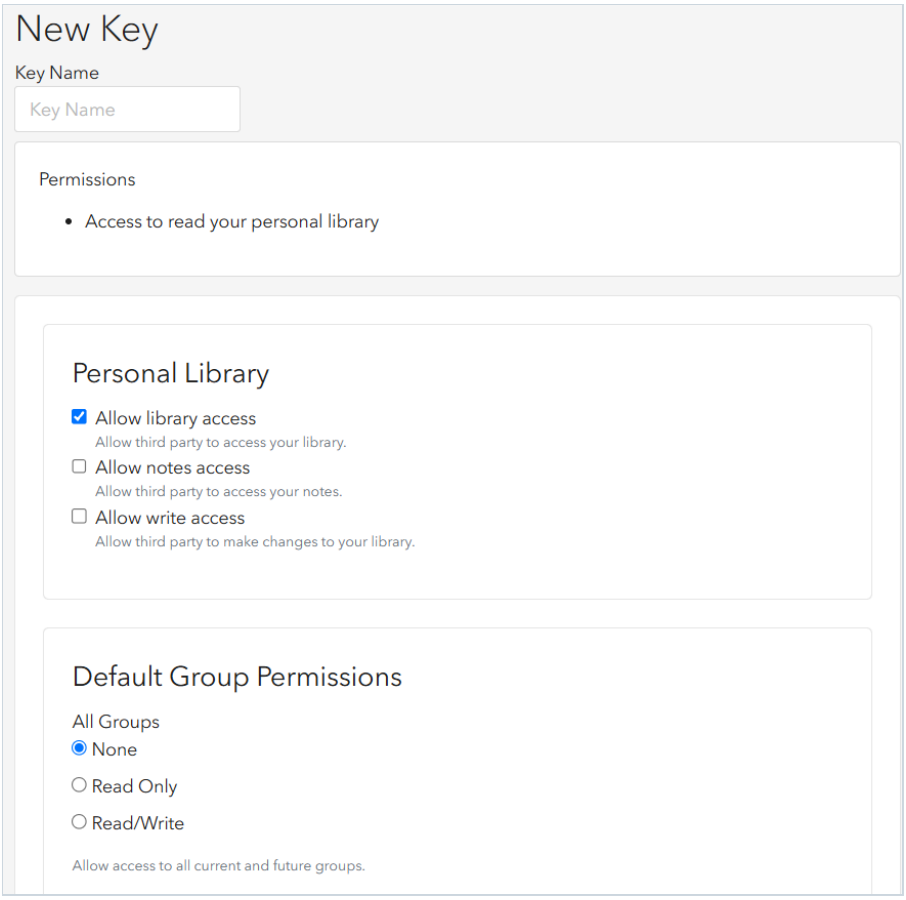
Bloque Applications. El User ID y las IPs se han ocultado en la captura.

Crear una clave de API

1. En el mismo bloque **Applications**, pulsa el botón **Create new private key**.
2. En **Key Name** escribe un nombre que reconozcas, por ejemplo `agente-zotero-openalex`.
3. Marca los permisos según lo que vaya a hacer el agente:

Permiso	Marcar	Para qué se necesita
Allow library access	Siempre	Leer colecciones, documentos y metadatos (partes 3 a 14)
Allow notes access	Necesario para leer notas	Leer las notas de los documentos y de las colecciones (partes 5 y 7)
Allow write access	Solo si vas a escribir	Completar resúmenes, crear notas e ítems (partes 6, 7 y 13)
Default Group Permissions	*None*, salvo que uses grupos	*Read Only* o *Read/Write* da acceso a las bibliotecas de grupo

4. Baja hasta el final de la página y pulsa **Save Key**.
5. Copia la clave que muestra Zotero y pégala en `ZOTERO_API_KEY`, dentro de la celda de claves (parte 2.5).



Formulario New Key con el permiso de lectura marcado. Para las partes 6, 7 y 13, marca también Allow write access.

Desde la misma lista puedes cambiar los permisos (**Edit Key**, sección 7.1 con captura) o eliminar una clave (**Delete Key**). Elimina cualquier clave que haya quedado escrita en un notebook o compartida. No borres *Automatic Zotero Client Key*: es la que usa Zotero de escritorio para sincronizar.

El identificador determina qué biblioteca se consulta y la clave establece qué operaciones están autorizadas. Esta distinción ayuda a comprender que acceder a los metadatos, leer notas y modificar registros son capacidades diferentes. En proyectos colectivos también permite definir quién prepara el corpus y quién puede incorporar cambios, de acuerdo con la organización del trabajo.

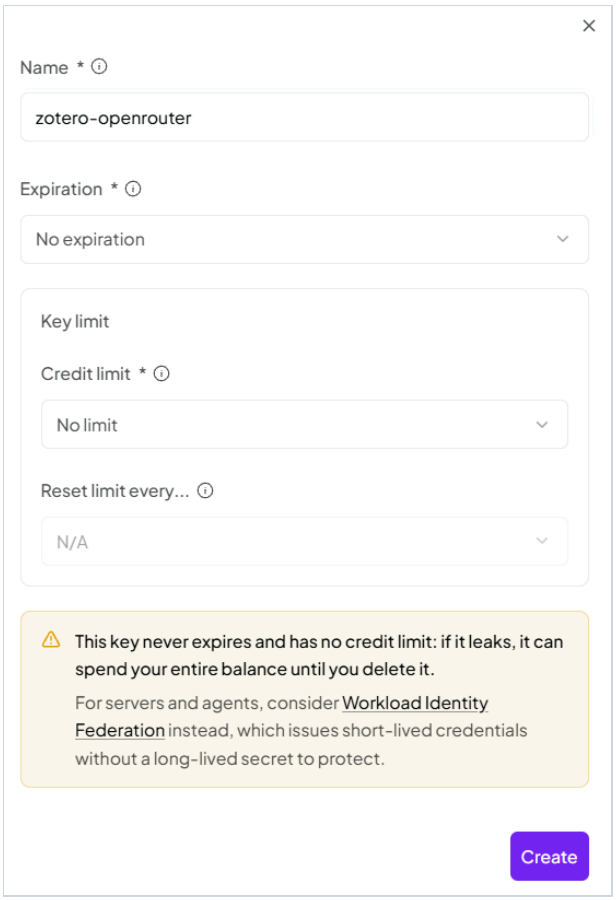
2.3 OpenRouter: clave de API y créditos

Una vez tengas cuenta en https://openrouter.ai, necesitas una **clave de API** para usar sus modelos desde Python. La clave es un código aleatorio de letras, números y caracteres especiales que empieza por `sk-or-v1-`. La usarán este notebook y cualquier plugin o herramienta que conectes a OpenRouter.

Crear la clave

1. Ve a https://openrouter.ai/workspaces/default/keys y pulsa el botón **Create API key**.
2. En **Name** escribe un nombre que identifique su uso, por ejemplo `zotero-openrouter`.

3. En **Expiration** elige si la clave caduca. *No expiration* es cómodo; una fecha de caducidad reduce el riesgo si la clave se filtra.
4. En **Credit limit** fija el gasto máximo de la clave y, en **Reset limit every…**, cada cuánto se reinicia. Con *No limit*, una clave filtrada puede gastar todo tu saldo (es lo que advierte el aviso amarillo). Para un agente que se ejecuta solo, conviene poner un límite.
5. Pulsa **Create**. Copia en ese momento la clave generada y pégala en `OPENROUTER_API_KEY`, dentro de la celda de claves (parte 2.5).



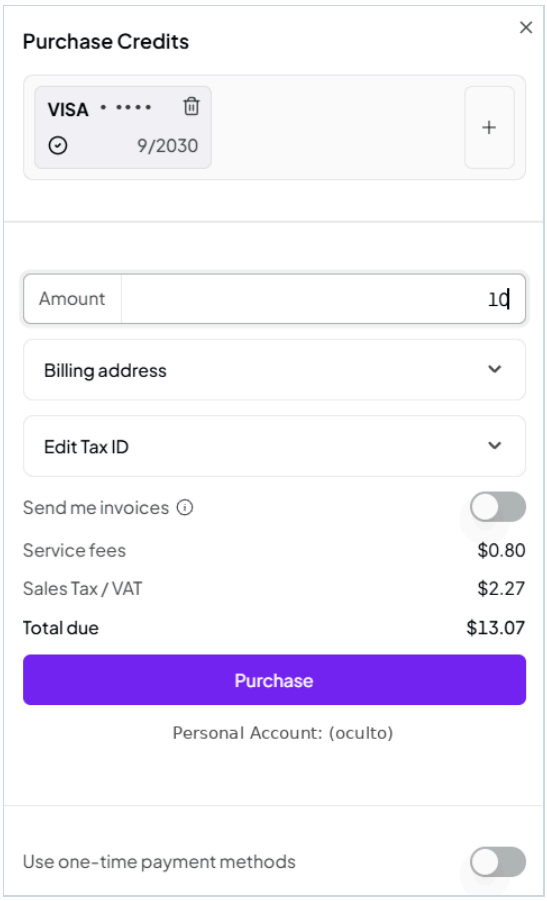
Formulario de creación de clave. Sin caducidad ni límite de crédito, la clave puede gastar todo el saldo hasta que la elimines.

Comprar créditos

Los modelos LLM de pago descuentan su coste del saldo de la cuenta. Sin créditos solo puedes usar los modelos gratuitos, que tienen límites de peticiones y de velocidad.

1. Ve a https://openrouter.ai/settings/credits y abre el formulario **Purchase Credits**.
2. Elige un método de pago o añade uno con +.
3. Escribe la cantidad en **Amount**. El total suma una comisión del servicio y el IVA: en la captura, 10 USD de créditos suponen 13,07 USD (0,80 USD de comisión y 2,27 USD de IVA).
4. Si necesitas factura (por ejemplo, para justificar el gasto en un proyecto), completa **Billing address** y **Edit Tax ID** y activa **Send me invoices**.

5. Pulsa **Purchase**. El saldo aparece en la misma página de créditos.



Formulario Purchase Credits. Los datos de la tarjeta y de la cuenta se han ocultado.

El acceso mediante API permite integrar las consultas al modelo en un procedimiento documentable. Para comparar resultados deben conservarse el identificador del modelo, las instrucciones, los parámetros y la fecha de ejecución. El coste influye en cuántos documentos pueden evaluarse y debe considerarse al diseñar una prueba, junto con la calidad de las respuestas.

2.4 OpenAlex: clave de API

La API de OpenAlex responde sin clave a peticiones de prueba, pero con una clave tus peticiones quedan identificadas y disfrutas de límites de uso más amplios. La clave es gratuita.

1. Crea una cuenta o inicia sesión en https://openalex.org.
2. Ve a https://openalex.org/settings/api-key.
3. Copia la clave y pégala en `OPENALEX_API_KEY`, dentro de la celda de claves (parte 2.5). Si quieres, añade también tu correo en `OPENALEX_MAILTO`.

La función `buscar_openalex()` (parte 8) añade la clave a cada petición automáticamente.

Una clave facilita administrar el uso de la fuente cuando el ejemplo pasa de unas pocas consultas a ejecuciones periódicas. La documentación vigente sitúa la gestión de claves en Settings → API; la ruta de la interfaz puede cambiar. Los límites y presupuestos deben consultarse en la documentación oficial al adaptar el procedimiento.

2.5 Tus claves de API

En esta versión de práctica, pensada para trabajar **solo en tu ordenador**, las claves se escriben directamente en la celda siguiente. Sustituye cada texto `PEGA_AQUI_…` por el valor correspondiente, manteniendo las comillas.

Variable	Servicio	Dónde se obtiene
`ZOTERO_USER_ID`, `ZOTERO_API_KEY`	Zotero	Parte 2.2
`OPENROUTER_API_KEY`	OpenRouter	Parte 2.3 (no hace falta si usas un modelo local)
`PROVEEDOR_LLM`, `MODELO_LOCAL`	OpenRouter u Ollama	Sección 2.9: `"openrouter"` o `"local"` y el modelo descargado
`OPENALEX_API_KEY`	OpenAlex	Parte 2.4 (recomendada)
`OPENALEX_MAILTO`	OpenAlex	Tu correo (opcional)

> **Importante.** Con las claves escritas, este notebook pasa a ser un archivo privado. No lo envíes, no lo subas a GitHub ni a la nube y no lo compartas sin borrar antes estos valores. Si una clave se difunde por error, revócala en el panel del servicio y genera una nueva. Más adelante, la versión con archivo `.env` sacará las claves del notebook.

La centralización de credenciales facilita comprender qué cliente depende de cada servicio. Para reutilizar el cuaderno en docencia o colaboración, la configuración debe distinguirse del material que se distribuye. Las celdas originales se conservan como ejemplo local; esta edición no sustituye su mecanismo de credenciales.

```
# ═══════════════ TUS CLAVES DE API (solo uso local, práctica) ═══════════════
# Sustituye cada PEGA_AQUI_... por tu valor, manteniendo las comillas.

ZOTERO_USER_ID      = "PEGA_AQUI_TU_USER_ID"          # número, sección 2.2
ZOTERO_API_KEY      = "PEGA_AQUI_TU_CLAVE_ZOTERO"     # sección 2.2
ZOTERO_LIBRARY_TYPE = "user"                          # "user" o "group"

OPENROUTER_API_KEY  = "PEGA_AQUI_TU_CLAVE_OPENROUTER" # empieza por sk-or-v1-, sección 2.3
MODELO_LLM          = "deepseek/deepseek-chat-v3.1"   # o "mistralai/mistral-small-3.2-24b-instruct"

# Proveedor del modelo de lenguaje (sección 2.9): "openrouter" (en la nube, de pago) o "local"
(Ollama, gratuito)
PROVEEDOR_LLM       = "openrouter"
MODELO_LOCAL        = "llama3.1:8b"                # descargado con Ollama; o "qwen3:8b", "gpt-
oss:20b"
```

```
OPENALEX_API_KEY    = "PEGA_AQUI_TU_CLAVE_OPENALEX"   # recomendada, sección 2.4
OPENALEX_MAILTO     = ""                              # opcional: tu correo
```

2.6 Librerías y configuración

Esta celda reúne las librerías y el resto de **variables de configuración**: direcciones de los servicios y rutas de archivos. Al final comprueba que las claves de la parte 2.5 estén escritas. Las variables de trabajo, como la colección elegida o el número de frases de búsqueda, no están aquí: se definen en el código de cada parte, en el punto donde se deciden y se usan.

Las direcciones de servicio y las rutas de salida describen dónde se intercambian y conservan los datos. Las variables de investigación describen qué colección y qué tema se estudian. Distinguir ambas clases de decisiones facilita reproducir el procedimiento en otro entorno sin confundir un cambio técnico con una modificación del diseño de búsqueda.

```
# ── Librerías ──────────────────────────────────────────
import os
import re
import json
import html
import sqlite3
import logging
import datetime
import time
from pathlib import Path
from collections import Counter

import requests
from pyzotero import zotero
from openai import OpenAI

# ── Servicios (las claves están en la celda de la parte 2.5) ─
OPENROUTER_BASE_URL = "https://openrouter.ai/api/v1"
OPENALEX_URL        = "https://api.openalex.org/works"
OLLAMA_BASE_URL     = "http://localhost:11434/v1"      # servidor local de Ollama (sección 2.9)

# Proveedor del modelo: si la celda de claves es de una versión anterior, se usa OpenRouter
PROVEEDOR_LLM = globals().get("PROVEEDOR_LLM", "openrouter")
MODELO_LOCAL  = globals().get("MODELO_LOCAL", "llama3.1:8b")
MODELO_OPENROUTER = MODELO_LLM                       # se conserva para volver a OpenRouter (sección
10.1)
if PROVEEDOR_LLM == "local":
    MODELO_LLM = MODELO_LOCAL

# Los campos opcionales vacíos se tratan como "sin valor"
OPENALEX_API_KEY = OPENALEX_API_KEY if OPENALEX_API_KEY and not
OPENALEX_API_KEY.startswith("PEGA_AQUI") else None
OPENALEX_MAILTO  = OPENALEX_MAILTO or None

# ── Archivos locales del agente ────────────────────────
BASE_DIR           = Path.cwd()
ARCHIVO_ESTRUCTURA = BASE_DIR / "estructura_tematica.txt"
ARCHIVO_KEYWORDS   = BASE_DIR / "keywords_relevantes.txt"
ARCHIVO_HISTORIAL  = BASE_DIR / "historial_detallado.json"
ARCHIVO_LOG        = BASE_DIR / "agente_academico.log"

# ── Registro (log): UTF-8 y sin claves ─────────────────
```

```python
class OcultarClaves(logging.Filter):
    """Sustituye claves y correos por *** antes de escribir cada línea en el log."""
    PATRON = re.compile(r"(api_key=|mailto=|sk-or-v1-)[^&\s'\"]+")

    def filter(self, record):
        mensaje = self.PATRON.sub(r"\1***", record.getMessage())
        for secreto in (ZOTERO_API_KEY, OPENROUTER_API_KEY, OPENALEX_API_KEY):
            if secreto and len(secreto) > 8:
                mensaje = mensaje.replace(secreto, "***")
        record.msg, record.args = mensaje, ()
        return True


logging.basicConfig(filename=ARCHIVO_LOG, level=logging.INFO, encoding="utf-8", force=True,
                    format="%(asctime)s - %(levelname)s - %(message)s",
                    datefmt="%Y-%m-%d %H:%M:%S")
for manejador in logging.getLogger().handlers:
    manejador.addFilter(OcultarClaves())
for libreria in ("httpx", "httpcore", "urllib3", "openai"):      # sus mensajes incluyen URL completas
    logging.getLogger(libreria).setLevel(logging.WARNING)


def _enmascarar(valor):
    """Muestra solo el inicio de una clave para comprobar que se ha escrito."""
    if not valor or str(valor).startswith("PEGA_AQUI"):
        return "FALTA"
    return f"{valor[:4]}… ({len(valor)} caracteres)"


def verificar_config():
    obligatorias = {"ZOTERO_USER_ID": ZOTERO_USER_ID,
                    "ZOTERO_API_KEY": ZOTERO_API_KEY}
    if PROVEEDOR_LLM != "local":                # con un modelo local no hace falta clave de OpenRouter
        obligatorias["OPENROUTER_API_KEY"] = OPENROUTER_API_KEY
    opcionales = {"OPENALEX_API_KEY": OPENALEX_API_KEY,   # recomendada
                  "OPENALEX_MAILTO": OPENALEX_MAILTO}
    print("Configuración cargada")
    for nombre, valor in {**obligatorias, **opcionales}.items():
        print(f"  {nombre:<20} {_enmascarar(valor)}")
    if PROVEEDOR_LLM == "local":
        print(f"  {'Proveedor LLM':<20} local (Ollama)")
    print(f"  {'Modelo LLM':<20} {MODELO_LLM}")
    faltan = [k for k, v in obligatorias.items() if not v or str(v).startswith("PEGA_AQUI")]
    if faltan:
        raise RuntimeError(f"Faltan claves en la celda de la parte 2.5: {', '.join(faltan)}")


verificar_config()
```

Salida generada al ejecutar

```
Configuración cargada
  ZOTERO_USER_ID       1762… (8 caracteres)
  ZOTERO_API_KEY       Ew3B… (24 caracteres)
  OPENROUTER_API_KEY   sk-o… (73 caracteres)
  OPENALEX_API_KEY     4chz… (22 caracteres)
  OPENALEX_MAILTO      dann… (20 caracteres)
  Modelo LLM           deepseek/deepseek-chat-v3.1
```

Salida generada con valores ficticios en la celda de claves, para mostrar el formato. Las claves nunca se imprimen completas.

2.7 Clientes y utilidades comunes

Se crean una sola vez y se reutilizan: `zot` para Zotero y `llm` para el modelo de lenguaje. `llm` apunta a OpenRouter o, si `PROVEEDOR_LLM = "local"`, al servidor de Ollama de tu ordenador (sección 2.9). Ambos servicios usan el mismo formato de la API de OpenAI, de modo que el resto del notebook no cambia. Las utilidades evitan repetir el mismo código en cada sección.

Un cliente encapsula la comunicación con un servicio y evita reconstruirla en cada operación. Esto favorece la coherencia del procedimiento: las consultas a una misma biblioteca comparten la configuración de acceso y las llamadas al modelo utilizan una interfaz común. No implica que los servicios mantengan idéntico estado entre ejecuciones.

```
zot = zotero.Zotero(ZOTERO_USER_ID, ZOTERO_LIBRARY_TYPE, ZOTERO_API_KEY)

if PROVEEDOR_LLM == "local":
    llm = OpenAI(base_url=OLLAMA_BASE_URL, api_key="ollama")   # Ollama ignora la clave, pero el
cliente exige una
    print(f"Clientes listos: Zotero y Ollama (modelo local {MODELO_LLM})")
else:
    llm = OpenAI(base_url=OPENROUTER_BASE_URL, api_key=OPENROUTER_API_KEY)
    print("Clientes listos: Zotero y OpenRouter")
```

Salida registrada

```
Clientes listos: Zotero y OpenRouter
```

Las **utilidades comunes** resuelven tareas que se repiten en todo el notebook: leer un campo de Zotero sin errores (`campo`, `primer_autor`, `etiquetas`), comparar títulos ignorando mayúsculas y signos (`normalizar_titulo`) y guardar archivos de texto o JSON de forma segura (`guardar_texto`, `cargar_json`, `guardar_json`).

Las funciones comunes hacen explícitas las reglas con las que se interpretan registros heterogéneos. Un valor ausente debe distinguirse de un error de acceso, y normalizar un título no equivale a establecer identidad documental. Documentar estas reglas permite discutir qué información se conserva, qué se simplifica y qué debe revisarse manualmente.

```
# ── Lectura segura de metadatos de Zotero ──────────────────
def campo(item, nombre, defecto=""):
    """Devuelve item['data'][nombre] o `defecto` si el campo no existe o está vacío."""
    valor = item.get("data", {}).get(nombre)
    return valor if valor else defecto


def primer_autor(item):
    creadores = campo(item, "creators", [])
    if not creadores:
        return "Anónimo"
    c = creadores[0]
    return c.get("lastName") or c.get("name") or "Anónimo"


def etiquetas(item):
    return [t["tag"] for t in campo(item, "tags", [])]
```

```
def normalizar_titulo(titulo):
    """Minúsculas y sin signos: sirve para detectar duplicados."""
    return re.sub(r"\s+", " ", re.sub(r"[^\w\s]", " ", (titulo or "").lower())).strip()


# ── Archivos ──────────────────────────────────────────────
def guardar_texto(ruta, contenido):
    """Escribe y fuerza el volcado a disco (evita archivos vacíos si el proceso se corta)."""
    with open(ruta, "w", encoding="utf-8") as f:
        f.write(contenido)
        f.flush()
        os.fsync(f.fileno())


def cargar_json(ruta, defecto):
    if Path(ruta).exists():
        with open(ruta, encoding="utf-8") as f:
            return json.load(f)
    return defecto


def guardar_json(ruta, datos):
    guardar_texto(ruta, json.dumps(datos, indent=4, ensure_ascii=False))
```

2.8 Comprobar las conexiones

Antes de trabajar con la biblioteca conviene comprobar que las tres credenciales funcionan. `probar_conexiones()` hace una petición mínima a cada servicio: pide a Zotero el número de ítems de la biblioteca, busca en OpenAlex una palabra y consulta en OpenRouter el estado de la clave, una petición que no genera texto ni consume créditos. Si `PROVEEDOR_LLM = "local"`, en lugar de OpenRouter comprueba que Ollama esté en marcha y que el modelo elegido esté descargado. Si un servicio falla, la función indica qué clave revisar y muestra solo el código de respuesta, nunca la URL de la petición, que contiene la clave de OpenAlex.

Si OpenAlex u OpenRouter fallan, las partes 3 a 7 pueden seguirse igualmente, porque solo usan Zotero.

```
def probar_conexiones():
    """Petición mínima a Zotero, OpenAlex y OpenRouter. No muestra claves ni URL."""
    try:
        print(f"{'Zotero':<11} correcto   {zot.num_items()} ítems de primer nivel en la biblioteca")
    except Exception as e:
        print(f"{'Zotero':<11} error      {type(e).__name__}: revisa ZOTERO_USER_ID y ZOTERO_API_KEY
(sección 2.2)")

    params = {"search": "zotero", "per_page": 1}
    if OPENALEX_API_KEY:
        params["api_key"] = OPENALEX_API_KEY
    if OPENALEX_MAILTO:
        params["mailto"] = OPENALEX_MAILTO
    try:
        r = requests.get(OPENALEX_URL, params=params, timeout=30)
        if r.ok:
            modo = "con clave" if OPENALEX_API_KEY else "sin clave"
```

```
            print(f"{'OpenAlex':<11} correcto   {r.json()['meta']['count']:,} publicaciones para
'zotero' ({modo})")
        else:
            print(f"{'OpenAlex':<11} error      respuesta {r.status_code}: revisa OPENALEX_API_KEY
(sección 2.4)")
    except requests.RequestException as e:
        print(f"{'OpenAlex':<11} error      sin conexión ({type(e).__name__})")

    if PROVEEDOR_LLM == "local":
        try:
            r = requests.get(OLLAMA_BASE_URL.removesuffix("/v1") + "/api/tags", timeout=10)
            modelos = [m["name"] for m in r.json().get("models", [])]
            estado = "correcto  " if (MODELO_LOCAL in modelos or f"{MODELO_LOCAL}:latest" in modelos)
else "falta modelo"
            print(f"{'Ollama':<11} {estado} {len(modelos)} modelo(s) descargado(s); en uso:
{MODELO_LOCAL}")
        except requests.RequestException:
            print(f"{'Ollama':<11} error      no responde en localhost:11434; abre Ollama (sección
2.9)")
        return

    try:
        r = requests.get("https://openrouter.ai/api/v1/key", timeout=30,
                         headers={"Authorization": f"Bearer {OPENROUTER_API_KEY}"})
        if r.ok:
            datos = r.json().get("data", {})
            limite = datos.get("limit")
            print(f"{'OpenRouter':<11} correcto   uso acumulado {datos.get('usage', 0):.4f} USD; "
                  f"límite de la clave: {limite if limite is not None else 'sin límite'}")
        else:
            print(f"{'OpenRouter':<11} error      respuesta {r.status_code}: revisa OPENROUTER_API_KEY
(sección 2.3)")
    except requests.RequestException as e:
        print(f"{'OpenRouter':<11} error      sin conexión ({type(e).__name__})")


probar_conexiones()
```

Salida registrada

```
Zotero      correcto   2484 ítems de primer nivel en la biblioteca
OpenAlex    correcto   122,536 publicaciones para 'zotero' (con clave)
OpenRouter  correcto   uso acumulado 23.1929 USD; límite de la clave: sin límite
```

Ejecuta la celda para ver los resultados de tu biblioteca.

2.9 Modelo local y gratuito con Ollama

En lugar de OpenRouter, las partes que usan un LLM (10 a 14) pueden trabajar con un modelo que se ejecuta en tu propio ordenador. La opción más sencilla es **Ollama**, un programa gratuito y de código abierto que descarga modelos abiertos y los sirve en `http://localhost:11434` con el mismo formato de API que OpenAI. Por eso basta con cambiar la dirección del cliente `llm` (sección 2.7): el código de las demás partes no cambia.

Un modelo local no tiene coste por consulta, funciona sin conexión una vez descargado y los textos de tu biblioteca no salen de tu ordenador. A cambio, depende de la memoria y del procesador de tu equipo, responde más despacio que un servicio en la nube, sobre todo sin tarjeta gráfica, y los modelos que caben en un portátil son menos precisos que los grandes modelos

comerciales. Para practicar y para tareas acotadas, como el filtro de pertinencia de la parte 11, suele bastar; para la taxonomía de la parte 12 conviene comparar sus resultados con los de un modelo de OpenRouter.

Instalar Ollama y descargar un modelo

1. Descarga el instalador desde https://ollama.com/download (Windows, macOS o Linux) y ejecútalo. Ollama queda en marcha en segundo plano; en Windows aparece su icono junto al reloj.
2. Abre una terminal (en Windows, *PowerShell*) y comprueba la instalación con `ollama --version`.
3. Descarga un modelo con `ollama pull`, seguido de su nombre. Por ejemplo, `ollama pull llama3.1:8b`.
4. Pruébalo en la terminal con `ollama run llama3.1:8b "¿Qué es Zotero?"`. El comando `ollama list` muestra los modelos descargados y `ollama rm <modelo>` borra uno para liberar espacio.

Modelos abiertos adecuados para empezar (los tamaños son aproximados; el catálogo completo, con sus variantes, está en https://ollama.com/library):

Familia	Nombre en Ollama	Descarga	Memoria recomendada	Observaciones
Llama (Meta)	`llama3.2:3b`	unos 2 GB	8 GB	Rápido incluso sin tarjeta gráfica; respuestas más simples
Llama (Meta)	`llama3.1:8b`	unos 5 GB	16 GB	Opción por defecto del manual: buen equilibrio y respuestas directas
Qwen (Alibaba)	`qwen3:8b`	unos 5 GB	16 GB	Buen rendimiento en español; razona antes de responder, lo que lo hace más lento
GPT (OpenAI)	`gpt-oss:20b`	unos 14 GB	16 GB o más	Modelo abierto de OpenAI, el más capaz de la tabla; también razona antes de responder

Los modelos GPT que usa ChatGPT no pueden descargarse; `gpt-oss` es la familia de pesos abiertos que OpenAI publicó para uso local.

Activar el modelo local en el notebook

1. En la celda de claves (sección 2.5), cambia `PROVEEDOR_LLM = "openrouter"` por `PROVEEDOR_LLM = "local"` y escribe en `MODELO_LOCAL` el nombre exacto del modelo descargado.
2. Ejecuta de nuevo las celdas de las secciones 2.5 a 2.8. `verificar_config()` ya no exige la clave de OpenRouter, y la prueba de conexiones comprueba Ollama en su lugar.
3. Ejecuta la celda siguiente, que lista los modelos descargados y hace una consulta de prueba.

Para volver a OpenRouter basta con escribir de nuevo `"openrouter"` y repetir el paso 2.

Tamaño del contexto

Ollama trabaja por defecto con una ventana de contexto reducida, y lo que no cabe en ella se descarta sin aviso. Con colecciones grandes, el contexto de la sección 9.2 puede superarla. La API compatible con OpenAI no permite cambiar ese tamaño en cada petición; la documentación de Ollama indica crear una variante del modelo con un archivo de texto llamado `Modelfile`, con estas dos líneas:

`FROM llama3.1:8b`

`PARAMETER num_ctx 16384`

Después se crea la variante con `ollama create llama3.1-16k -f Modelfile` y se escribe `MODELO_LOCAL = "llama3.1-16k"`. Un contexto mayor necesita más memoria.

Una alternativa con interfaz gráfica es **LM Studio** (https://lmstudio.ai), que también ofrece un servidor compatible con OpenAI, en `http://localhost:1234/v1`; para usarlo, basta con cambiar `OLLAMA_BASE_URL` en la sección 2.6.

```
def probar_modelo_local(pregunta="Responde en una frase: ¿para qué sirve Zotero?"):
    """Comprueba que Ollama está en marcha, lista los modelos descargados y hace una consulta de
prueba."""
    try:
        r = requests.get(OLLAMA_BASE_URL.removesuffix("/v1") + "/api/tags", timeout=10)
        r.raise_for_status()
    except requests.RequestException:
        print("Ollama no responde en localhost:11434. Abre la aplicación Ollama (o ejecuta 'ollama
serve').")
        return
    modelos = [m["name"] for m in r.json().get("models", [])]
    print("Modelos descargados:", ", ".join(modelos) or "ninguno")
    if MODELO_LOCAL not in modelos and f"{MODELO_LOCAL}:latest" not in modelos:
        print(f"El modelo {MODELO_LOCAL} no está descargado. En una terminal: ollama pull
{MODELO_LOCAL}")
        return

    cliente = OpenAI(base_url=OLLAMA_BASE_URL, api_key="ollama")
    inicio = time.time()
    respuesta = cliente.chat.completions.create(
        model=MODELO_LOCAL, temperature=0.2, max_tokens=1024,
        messages=[{"role": "user", "content": pregunta}])
    texto = re.sub(r"<think>.*?</think>", "", respuesta.choices[0].message.content or "",
flags=re.S).strip()
    print(f"\n{MODELO_LOCAL} respondió en {time.time() - inicio:.1f} s:\n{texto}")
    print(f"\nProveedor activo en el notebook: {PROVEEDOR_LLM}")


probar_modelo_local()
```

Salida registrada

```
Modelos descargados: qwen3.6:latest, bge-m3:latest, qwen2.5:3b-instruct, llama3:latest, gpt-oss:120b-cloud
El modelo llama3.1:8b no está descargado. En una terminal: ollama pull llama3.1:8b
```

Ollama está en marcha y lista los modelos descargados en tu ordenador. El modelo indicado en `MODELO_LOCAL` *(*`llama3.1:8b`*, el valor por defecto) no estaba entre ellos, por lo que la celda no hizo la consulta de prueba. Basta con escribir en* `MODELO_LOCAL` *uno de los modelos de la lista, por ejemplo* `"llama3:latest"` *o* `"qwen2.5:3b-instruct"`*, o descargar el que falta con* `ollama pull`*. Dos modelos de la lista no sirven para este uso:* `bge-m3` *es un modelo de embeddings, que convierte textos en vectores y no genera respuestas, y* `gpt-oss:120b-cloud` *se ejecuta en los servidores de Ollama, no en tu ordenador, y requiere una cuenta de Ollama.*

3

Biblioteca y colecciones en Zotero

En este apartado se establece la conexión con la biblioteca, se obtiene una visión de conjunto de su contenido, se listan las colecciones con su identificador y su posición en la jerarquía, se comprueba si una colección existe y se elige la colección de trabajo. Al final se obtienen los identificadores de sus documentos. Todas las partes siguientes trabajan sobre esa colección, guardada en la variable `COLECCION_ID`, y sobre la lista `documentos`. Basta con haber ejecutado la parte 2 y disponer de una clave con *Allow library access*.

En Zotero, cada objeto tiene una **clave** de ocho caracteres (`key`) que lo identifica de forma única dentro de la biblioteca: colecciones, documentos, adjuntos y notas. Esa clave es el identificador que usan todas las llamadas a la API. Zotero devuelve cada objeto como un diccionario: la clave está en `item['key']` y los metadatos en `item['data']`.

3.1 Conectar y leer los últimos ítems

`zot.top()` devuelve solo ítems de primer nivel (artículos, libros…), sin los PDF ni las notas que cuelgan de ellos. Las notas independientes sí son ítems de primer nivel, y las más recientes (por ejemplo, las que crea el agente) pueden ocupar toda la lista; por eso la celda las excluye con `itemType="-note"`. El signo `-` delante de un tipo lo excluye de la consulta.

Examinar pocos ítems permite reconocer el formato de respuesta antes de procesar una colección completa. Esta inspección inicial ayuda a comprobar que se consulta la biblioteca prevista y que los campos relevantes están disponibles. La muestra sirve para aprender la estructura; no permite describir estadísticamente toda la biblioteca.

```
items = zot.top(limit=10, itemType="-note")      # sin notas independientes

for item in items:
    print(f"[{campo(item, 'itemType', 'N/A').upper()}] - {campo(item, 'title', 'Sin título')}")
```

Salida registrada

```
[JOURNALARTICLE] - Companies selling authorship slots thrive in a culture that equates success with a strong publication record. By Christine Ro & Jack Leeming
[JOURNALARTICLE] - Bibliometric evaluation of Forensic Science International as a scholarly journal within the subject category legal medicine
[JOURNALARTICLE] - Reference accuracy of primary studies about reference accuracy: A descriptive study
[JOURNALARTICLE] - Sneaked references: Fabricated reference metadata distort citation counts
[JOURNALARTICLE] - Detecting anomalous referencing patterns in PubMed papers suggestive of author-centric reference list manipulation
[JOURNALARTICLE] - Unveiling Deception: Establishing a Taxonomic Framework for Disinformation within Scientific Discourse
[JOURNALARTICLE] - From Louvain to Leiden: guaranteeing well-connected communities
[JOURNALARTICLE] - Altmetrics at institutional level: Visibility on the web of the scientific production of Spanish universities from Altmetric.com
[JOURNALARTICLE] - A large-scale comparison of coverage and mentions captured by the two altmetric
```

```
aggregators: Altmetric.com and PlumX
[JOURNALARTICLE] - Assessment of scientific research: improvement of science policies in Latin America
```

3.2 Panorama de la biblioteca

`zot.num_items()` devuelve el número de ítems de primer nivel, es decir, los documentos y las notas independientes, sin los adjuntos ni las notas que dependen de un documento. `zot.everything(zot.top())` recorre todas las páginas de resultados (la API entrega como máximo 100 por llamada) y permite contar los ítems por tipo.

El recuento por tipo muestra la composición de la biblioteca antes de trabajar con una parte de ella. Una biblioteca con muchos `webpage` o `document` suele indicar referencias importadas desde el navegador con metadatos incompletos. En bibliotecas grandes la celda puede tardar, porque descarga todos los registros de primer nivel.

```python
total_primer_nivel = zot.num_items()
biblioteca = zot.everything(zot.top())
por_tipo = Counter(campo(it, "itemType") for it in biblioteca)

print(f"Ítems de primer nivel en la biblioteca: {total_primer_nivel}\n")
print(f"{'Tipo':<22} {'Ítems':>6}")
for tipo, n in por_tipo.most_common():
    print(f"{tipo:<22} {n:>6}")
```

Salida registrada

```
Ítems de primer nivel en la biblioteca: 2484

Tipo                   Ítems
journalArticle          1955
attachment               156
book                     108
conferencePaper           69
note                      52
report                    45
preprint                  39
document                  33
bookSection               13
thesis                     8
blogPost                   3
webpage                    3
```

La biblioteca tiene 2484 ítems de primer nivel, de los cuales 1955 son artículos de revista. Los 52 ítems de tipo `note` son notas independientes, muchas creadas por el propio manual y por el agente, y los 156 de tipo `attachment` son archivos añadidos sin documento padre.

3.3 Colecciones y sus identificadores

La primera celda lista el nombre y la clave (ID) de cada colección y define `clave_de_coleccion()`, que devuelve la clave a partir del nombre. Las colecciones pueden anidarse: cada una guarda la clave de su colección madre en `parentCollection`, que vale `False` en las de primer nivel.

```python
colecciones = zot.everything(zot.collections())

# La salida muestra solo las colecciones que usa el manual, para no exponer el resto
```

```
# de la biblioteca. Con colecciones_visibles = None se muestran todas.
colecciones_visibles = {"6APD6L29", "UDMXJXJS", "2XNUVRA7"}    # Bibliometria_Forense, revision-
sistematica, ia-educacion

print(f"{len(colecciones)} colecciones en la biblioteca\n")
for col in colecciones:
    if colecciones_visibles is None or col["key"] in colecciones_visibles:
        print(f"Nombre: {col['data']['name']} | ID: {col['key']}")
if colecciones_visibles is not None:
    print(f"... y {sum(c['key'] not in colecciones_visibles for c in colecciones)} colecciones más sin
mostrar")


def clave_de_coleccion(nombre):
    """Devuelve la clave (ID) de una colección a partir de su nombre."""
    for col in zot.everything(zot.collections()):
        if col["data"]["name"].lower() == nombre.lower():
            return col["key"]
    raise ValueError(f"No existe ninguna colección llamada '{nombre}'")
```

Salida registrada

```
20 colecciones en la biblioteca

Nombre: revision-sistematica | ID: UDMXJXJS
Nombre: Bibliometria_Forense | ID: 6APD6L29
Nombre: ia-educacion | ID: 2XNUVRA7
... y 17 colecciones más sin mostrar
```

Para no exponer el resto de la biblioteca, la salida muestra solo las tres colecciones que se usan en el manual; con `colecciones_visibles = None` *la celda las lista todas.*

Jerarquía y tamaño. La celda siguiente reconstruye la ruta completa de cada colección (madre / hija / nieta) y muestra cuántos ítems y subcolecciones contiene, según los campos `meta.numItems` y `meta.numCollections` que devuelve la API junto con cada colección. Los adjuntos y las notas hijas pertenecen a su documento y no se asignan a colecciones, de modo que normalmente no forman parte de ese recuento; la sección 3.4 permite comprobarlo en tu biblioteca.

```
por_clave = {col["key"]: col for col in colecciones}


def ruta_coleccion(clave):
    """Ruta completa de una colección: Madre / Hija / Nieta."""
    partes = []
    while clave:
        col = por_clave.get(clave)
        if col is None:                # colección no recuperada
            partes.append(f"[{clave}]")
            break
        partes.append(col["data"]["name"])
        clave = col["data"].get("parentCollection") or None
    return " / ".join(reversed(partes))


print(f"{'ID':<10} {'Ítems':>6} {'Subcol.':>8}  Colección")
for col in sorted(colecciones, key=lambda c: ruta_coleccion(c["key"]).lower()):
    if colecciones_visibles is not None and col["key"] not in colecciones_visibles:
        continue
```

```
    meta = col.get("meta", {})
    print(f"{col['key']:<10} {meta.get('numItems', 0):>6} {meta.get('numCollections', 0):>8}  "
          f"{ruta_coleccion(col['key'])}")
if colecciones_visibles is not None:
    print(f"... y {sum(c['key'] not in colecciones_visibles for c in colecciones)} colecciones más sin
mostrar")
```

Salida registrada

```
ID          Ítems  Subcol.  Colección
6APD6L29       96        0  Bibliometria_Forense
2XNUVRA7       25        0  ia-educacion
UDMXJXJS       26        0  revision-sistematica
... y 17 colecciones más sin mostrar
```

Bibliometria_Forense tiene 96 ítems asignados, mientras que la sección 3.5 encuentra 63 documentos. La diferencia corresponde a ítems de primer nivel que no son documentos, sobre todo las notas independientes creadas en las secciones 7.3 y 13.2 y por el agente (sección 7.1). La ruta completa (madre / hija) aparece en las colecciones anidadas, que esta salida no muestra.

3.4 Comprobar una colección y elegir la colección de trabajo

La **colección de trabajo** se fija copiando su clave (columna `ID`) de la salida de la sección 3.3 en la variable `COLECCION_ID`. También puede obtenerse por su nombre con `clave_de_coleccion()`. Esta variable se usa en el resto del notebook; para trabajar con otro tema basta con cambiarla aquí.

Después, `collection_items()` devuelve **todo** lo que contiene la colección: artículos, PDF adjuntos y notas. `Counter` resume cuántos hay de cada tipo.

Contar por tipo permite diferenciar publicaciones de recursos auxiliares. Para análisis temáticos interesa normalmente el ítem bibliográfico principal; para evaluar disponibilidad documental interesan sus adjuntos. Estos recuentos responden a preguntas distintas y requieren denominadores distintos. Un mismo ítem puede pertenecer a varias colecciones sin constituir por ello un duplicado bibliográfico.

Antes de fijarla conviene comprobar que existe. `buscar_coleccion()` acepta la clave o el nombre de una colección (sin distinguir mayúsculas) y devuelve su clave, o `None` si no existe. Trabaja sobre la lista `colecciones` de la sección 3.3, sin nuevas peticiones a la API; si acabas de crear la colección en Zotero, vuelve a ejecutar antes la sección 3.3 para actualizar la lista.

```
def buscar_coleccion(clave_o_nombre):
    """Clave de la colección indicada por su clave o su nombre, o None si no existe."""
    for col in colecciones:
        if clave_o_nombre == col["key"] or str(clave_o_nombre).lower() == col["data"]["name"].lower():
            return col["key"]
    return None


for prueba in ("6APD6L29", "Bibliometria_Forense", "coleccion-que-no-existe"):
    clave = buscar_coleccion(prueba)
    estado = f"existe: {clave} | {ruta_coleccion(clave)}" if clave else "no existe en la biblioteca"
    print(f"{prueba:<26} {estado}")
```

Salida registrada

```
6APD6L29                   existe: 6APD6L29 | Bibliometria_Forense
Bibliometria_Forense       existe: 6APD6L29 | Bibliometria_Forense
```

```
coleccion-que-no-existe    no existe en la biblioteca
```

Ejecuta la celda para comprobar las colecciones de tu biblioteca. Las dos primeras pruebas localizan la misma colección, por clave y por nombre; la tercera devuelve `None`*. La celda siguiente fija la colección de trabajo con su clave y cuenta su contenido por tipo.*

```python
# Colección de trabajo: clave copiada de la salida de 3.3
COLECCION_ID = "6APD6L29"                    # Bibliometria_Forense
# Alternativa equivalente, por nombre:
# COLECCION_ID = clave_de_coleccion("Bibliometria_Forense")

items_col = zot.collection_items(COLECCION_ID, limit=100)

conteo = Counter(campo(it, "itemType") for it in items_col)
print(conteo)
print(f"Total de items: {len(items_col)}")
print("-" * 20)
for tipo, cantidad in conteo.most_common():
    print(f"{tipo}: {cantidad}")
```

Salida registrada

```
Counter({'attachment': 40, 'journalArticle': 35, 'note': 20, 'preprint': 4, 'report': 1})
Total de items: 100
--------------------
attachment: 40
journalArticle: 35
note: 20
preprint: 4
report: 1
```

La mitad de los ítems son adjuntos: cada artículo trae su PDF. Para trabajar solo con los documentos usa `zot.collection_items_top(COLECCION_ID)`*.*

Llamada	Devuelve
`zot.collection_items(key)`	Todo lo que hay en la colección
`zot.collection_items_top(key)`	Solo ítems de primer nivel
`zot.collections_sub(key)`	Subcolecciones
`zot.children(item_key)`	Adjuntos y notas de un ítem
`zot.item(item_key)`	Un ítem concreto

3.5 Identificadores de los documentos de una colección

`collection_items_top()` devuelve solo los ítems de primer nivel de la colección, sin adjuntos ni notas hijas. La celda descarta además las notas independientes y guarda el resultado en la lista `documentos`, que usan las partes 5, 6 y 7. Para cada documento muestra su clave, tipo, año, primer autor y título, y exporta la tabla a un archivo CSV que puede abrirse con una hoja de cálculo.

La lista de claves es el inventario de la colección. Permite citar un documento sin ambigüedad en el código (por ejemplo, para leer su PDF o crearle una nota), comparar la colección en dos momentos y documentar qué corpus se utilizó en un análisis. La fecha de Zotero admite formatos diversos (`2024-03-01` , `March 2024` , `2024`), por lo que el año se extrae con una expresión regular.

```
import csv


def anio(item):
    """Año de publicación a partir del campo date, sea cual sea su formato."""
    m = re.search(r"\b(1[5-9]|20)\d{2}\b", campo(item, "date"))
    return m.group(0) if m else "s.f."


documentos = [it for it in zot.everything(zot.collection_items_top(COLECCION_ID))
              if campo(it, "itemType") not in ("note", "attachment", "annotation")]

print(f"Colección {COLECCION_ID}: {len(documentos)} documentos\n")
print(f"{'ID':<10} {'Tipo':<16} {'Año':<5} {'Primer autor':<18} Título")
filas_visibles = 5                      # filas en pantalla; la lista completa va al CSV
for it in documentos[:filas_visibles]:
    print(f"{it['key']:<10} {campo(it, 'itemType'):<16} {anio(it):<5} "
          f"{primer_autor(it)[:17]:<18} {campo(it, 'title', '(sin título)')[:60]}")
if len(documentos) > filas_visibles:
    print(f"... y {len(documentos) - filas_visibles} documentos más (lista completa en el archivo
CSV)")

archivo_csv = BASE_DIR / f"documentos_{COLECCION_ID}.csv"
with open(archivo_csv, "w", newline="", encoding="utf-8-sig") as f:
    escritor = csv.writer(f, delimiter=";")
    escritor.writerow(["clave", "tipo", "anio", "primer_autor", "titulo", "doi"])
    for it in documentos:
        escritor.writerow([it["key"], campo(it, "itemType"), anio(it), primer_autor(it),
                           campo(it, "title"), campo(it, "DOI")])
print(f"\nTabla guardada en {archivo_csv.name}")
```

Salida registrada

```
Colección 6APD6L29: 63 documentos

ID         Tipo             Año   Primer autor       Título
AVTE4IA3   journalArticle   s.f.  Thibault           Companies selling authorship slots thrive in a culture
that
DISXDZNA   journalArticle   2023  Jones              Bibliometric evaluation of Forensic Science
International as
A2WC95B8   journalArticle   s.f.  Logan              Reference accuracy of primary studies about reference
accura
BGR79MAI   journalArticle   s.f.  Besançon          Sneaked references: Fabricated reference metadata
distort ci
DABLKWGS   journalArticle   2024  Wren               Detecting anomalous referencing patterns in PubMed
papers su
... y 58 documentos más (lista completa en el archivo CSV)

Tabla guardada en documentos_6APD6L29.csv
```

La colección tiene 63 documentos; la celda muestra los cinco primeros y guarda la tabla completa en `documentos_6APD6L29.csv` *, que puede abrirse con una hoja de cálculo.* `s.f.` *indica que el campo de fecha está vacío; la sección 6.1 cuenta 19 documentos en esa situación. Para ver más filas, cambia* `filas_visibles` *.*

3.6 Colecciones a las que pertenece cada documento

Un documento puede asignarse a varias colecciones sin duplicarse: Zotero guarda un solo registro, cuyo campo `collections` contiene las claves de todas sus colecciones. La celda muestra las colecciones de los primeros documentos con su ruta completa, mediante `ruta_coleccion()` (sección 3.3), y cuenta cuántos documentos pertenecen a más de una. Una colección tampoco incluye automáticamente los documentos de sus subcolecciones: cada pertenencia es explícita.

```
for it in documentos[:8]:
    print(f"{it['key']}  {campo(it, 'title', '(sin título)')[:70]}")
    for clave_col in campo(it, "collections", []):
        print(f"     en: {ruta_coleccion(clave_col)}")

varias = [it for it in documentos if len(campo(it, "collections", [])) > 1]
print(f"\n{len(varias)} de {len(documentos)} documentos pertenecen a más de una colección.")
```

Salida registrada

```
AVTE4IA3  Companies selling authorship slots thrive in a culture that equates su
     en: Bibliometria_Forense
DISXDZNA  Bibliometric evaluation of Forensic Science International as a scholar
     en: Bibliometria_Forense
A2WC95B8  Reference accuracy of primary studies about reference accuracy: A desc
     en: Bibliometria_Forense
BGR79MAI  Sneaked references: Fabricated reference metadata distort citation cou
     en: Bibliometria_Forense
DABLKWGS  Detecting anomalous referencing patterns in PubMed papers suggestive o
     en: Bibliometria_Forense
Y2JEG29H  Unveiling Deception: Establishing a Taxonomic Framework for Disinforma
     en: Bibliometria_Forense
2HX4YAQB  Project Rachel: Can an AI Become a Scholarly Author?
     en: Bibliometria_Forense
84SRPNGD  Who is Actually Harmed by Predatory Publishers?
     en: Bibliometria_Forense

1 de 63 documentos pertenecen a más de una colección.
```

Ejecuta la celda para ver los resultados de tu biblioteca.

4

Estructura de los ítems: campos, creadores y tipos

En este apartado se examinan los campos de un ítem, se leen de forma segura, se obtienen todos sus creadores y se consultan los campos que admite cada tipo de documento. Esos campos (título, resumen, autores, etiquetas) son el material con el que trabajan las partes siguientes. Se apoya en los ítems leídos en la sección 3.1 y en la colección de trabajo elegida en la sección 3.4.

4.1 Anatomía de un ítem

Campo	Qué contiene
`itemType`	Tipo: `journalArticle`, `book`, `preprint`, `report`, `attachment`, `note`…
`title`	Título del documento
`abstractNote`	Resumen
`date`	Fecha de publicación (texto libre)
`creators`	Lista de autores: `{'creatorType', 'firstName', 'lastName'}` o `{'name'}`
`tags`	Etiquetas: `[{'tag': 'texto'}]`
`collections`	Claves de las colecciones a las que pertenece
`DOI`, `url`, `publicationTitle`	Identificadores y revista
`parentItem`	En adjuntos y notas: clave del ítem padre

Para ver todos los campos disponibles de un artículo:

El tipo documental determina qué campos son pertinentes, mientras que la clave identifica un registro dentro de la biblioteca. `parentItem` representa una relación entre un documento y sus recursos asociados; no debe interpretarse como una nueva publicación. Esta distinción evita inflar el tamaño del corpus contando artículos, PDF y notas como unidades científicas independientes.

```
ejemplo = zot.top(limit=1, itemType="-note")[0]["data"]
print(ejemplo["itemType"], "|", ejemplo.get("title", "(sin título)"))
```

```
print(ejemplo.keys())
```

Salida registrada

```
journalArticle | Companies selling authorship slots thrive in a culture that equates success with a strong
publication record. By Christine Ro & Jack Leeming
dict_keys(['key', 'version', 'itemType', 'title', 'creators', 'abstractNote', 'publicationTitle',
'publisher', 'place', 'date', 'volume', 'issue', 'section', 'partNumber', 'partTitle', 'pages', 'series',
'seriesTitle', 'seriesText', 'journalAbbreviation', 'DOI', 'citationKey', 'url', 'accessDate', 'PMID',
'PMCID', 'ISSN', 'archive', 'archiveLocation', 'shortTitle', 'language', 'libraryCatalog', 'callNumber',
'rights', 'extra', 'tags', 'collections', 'relations', 'dateAdded', 'dateModified'])
```

4.2 Acceso seguro a los campos

Cada tipo de ítem tiene campos distintos. Un artículo tiene título y autores; su PDF adjunto tiene título (el nombre del archivo) pero no autores; una nota no tiene título ni autores. Un bucle que lea `item['data']['creators']` directamente se detiene con un error en el primer ítem que no tenga ese campo, y los ítems siguientes no se procesan.

La celda toma una muestra real: el primer documento de una colección que tenga adjuntos o notas, junto con sus ítems hijos. Después lee la muestra de dos formas: con acceso directo y con las utilidades `campo()` y `primer_autor()` de la sección 2.7. La muestra procede de la colección de trabajo (`COLECCION_ID`, sección 3.4); para usar otra, sustituye `coleccion_muestra` por cualquier clave de la sección 3.3.

`primer_autor()` devuelve `Anónimo` cuando un registro no tiene creadores. En un documento, eso señala un metadato ausente; en un adjunto o una nota es lo esperado, porque esos ítems nunca tienen autores. Por eso la celda muestra `(no aplica)` para adjuntos y notas, en lugar de `Anónimo`.

```
coleccion_muestra = COLECCION_ID        # colección de trabajo (sección 3.4); o cualquier clave de la
sección 3.3

# Primer documento de la colección que tenga hijos (PDF o notas), junto con esos hijos
muestra = []
for doc in zot.collection_items_top(coleccion_muestra, limit=25):
    hijos = zot.children(doc["key"])
    if hijos:
        muestra = [doc] + hijos
        break
if not muestra:
    raise ValueError("Ningún documento de los 25 primeros tiene adjuntos o notas; prueba otra
colección.")
print(f"Muestra: documento {muestra[0]['key']} y {len(muestra) - 1} ítem(s) hijo(s)\n")

print("1) Acceso directo: item['data']['title'] e item['data']['creators']")
for i, it in enumerate(muestra, 1):
    d = it["data"]
    try:
        print(f"   {i}. {d['itemType']:<15} {d['title'][:45]:<47} {d['creators'][0]['lastName']}")
    except (KeyError, IndexError) as e:
        print(f"   {i}. {d['itemType']:<15} {type(e).__name__}: {e}. "
              f"El bucle se detiene; quedan {len(muestra) - i} ítem(s) sin procesar")
        break

print("\n2) Acceso seguro: campo() y primer_autor()")
for i, it in enumerate(muestra, 1):
    tipo = campo(it, "itemType")
```

```
    autor = "(no aplica)" if tipo in ("attachment", "note") else primer_autor(it)
    print(f"   {i}. {tipo:<15} {campo(it, 'title', '(sin título)')[:45]:<47} {autor}")
```

Salida registrada

```
Muestra: documento AVTE4IA3 y 3 ítem(s) hijo(s)

1) Acceso directo: item['data']['title'] e item['data']['creators']
   1. journalArticle  Companies selling authorship slots thrive in    Thibault
   2. note            KeyError: 'title'. El bucle se detiene; quedan 2 ítem(s) sin procesar

2) Acceso seguro: campo() y primer_autor()
   1. journalArticle  Companies selling authorship slots thrive in    Thibault
   2. note            (sin título)                                    (no aplica)
   3. note            (sin título)                                    (no aplica)
   4. attachment      PDF                                             (no aplica)
```

El documento de la muestra, `AVTE4IA3`, tiene dos notas y un PDF. Con acceso directo, el bucle falla en el primer hijo, una nota, que no tiene `title`, y los otros dos ítems no se procesan. Con las utilidades seguras se procesan los cuatro.

4.3 Todos los creadores de un documento

`primer_autor()` simplifica la presentación, pero omite a los coautores, a los editores y a los autores institucionales. Para examinar la autoría completa se lee el ítem con `zot.item(clave)` y se recorre su lista `creators`. Cada creador tiene un rol (`creatorType`: `author`, `editor`, `contributor`…) y un nombre, que puede estar separado en `firstName` y `lastName` o reunido en un solo campo `name`, como ocurre con las instituciones.

La celda usa el documento de la muestra de la sección 4.2; puede sustituirse `clave_item` por la clave de cualquier documento. La versión del ítem (`version`) aumenta con cada modificación y la usa Zotero para detectar cambios simultáneos al escribir (sección 7.4). Dos creadores con el mismo nombre no son necesariamente la misma persona: la API no desambigua identidades.

```
clave_item = muestra[0]["key"]          # o la clave de cualquier documento
item = zot.item(clave_item)

print(f"Clave: {item['key']} | versión: {item.get('version')}")
print(f"Título: {campo(item, 'title', '(sin título)')}\n")
for creador in campo(item, "creators", []):
    nombre = creador.get("name") or " ".join(
        x for x in (creador.get("firstName", ""), creador.get("lastName", "")) if x)
    print(f"  {creador.get('creatorType', 'sin rol'):<12} {nombre}")
```

Salida registrada

```
Clave: AVTE4IA3 | versión: 10961
Título: Companies selling authorship slots thrive in a culture that equates success with a strong
publication record. By Christine Ro & Jack Leeming

  author       Sébastien Thibault
```

Ejecuta la celda para ver los resultados de tu biblioteca.

4.4 Campos admitidos por tipo de documento

Artículos, libros y tesis no comparten el mismo esquema. `zot.item_type_fields(tipo)` devuelve los campos que admite un tipo, en el orden en que aparecen en Zotero. Conviene consultarlo antes de preparar una importación o una actualización: un campo que el tipo no admite se rechaza al escribir. La lista indica qué campos pueden tener valor, no cuáles están completos; la cobertura se mide en la sección 6.1.

```
for tipo in ("journalArticle", "book", "thesis"):
    campos_tipo = zot.item_type_fields(tipo)
    print("\nTipo:", tipo)
    print(", ".join(c["field"] for c in campos_tipo))
```

Salida registrada

```
Tipo: journalArticle
title, abstractNote, publicationTitle, publisher, place, date, volume, issue, section, partNumber,
partTitle, pages, series, seriesTitle, seriesText, journalAbbreviation, DOI, citationKey, url, accessDate,
PMID, PMCID, ISSN, archive, archiveLocation, shortTitle, language, libraryCatalog, callNumber, rights, extra

Tipo: book
title, abstractNote, series, seriesNumber, volume, numberOfVolumes, edition, date, publisher, place,
originalDate, originalPublisher, originalPlace, format, numPages, ISBN, DOI, citationKey, url, accessDate,
ISSN, archive, archiveLocation, shortTitle, language, libraryCatalog, callNumber, rights, extra

Tipo: thesis
title, abstractNote, thesisType, university, place, date, series, seriesNumber, numPages, DOI, ISBN,
citationKey, url, accessDate, ISSN, archive, archiveLocation, shortTitle, language, libraryCatalog,
callNumber, rights, extra
```

Un artículo admite `DOI` e `ISSN`; un libro, `ISBN` y `edition`; una tesis, `thesisType` y `university`. Por eso la sección 6.1 cuenta la cobertura de cada campo solo entre los documentos cuyo tipo lo admite.

5

Documentos, adjuntos y PDF

En este apartado se examina qué acompaña a cada documento de la colección: PDF, otros archivos y notas. Se calcula cuántos documentos tienen PDF y cuántos no, se lista cuáles son, se lee el texto completo de un PDF y se buscan términos dentro de él. Se apoya en la lista `documentos` de la sección 3.5 y deja definidas `pdf_por_doc`, `con_pdf`, `sin_pdf` y `texto_completo()`.

Un documento de Zotero es un ítem de primer nivel; sus PDF, otros archivos y notas son **ítems hijos** que guardan la clave del documento en el campo `parentItem`. Un adjunto PDF se reconoce por `itemType = "attachment"` y `contentType = "application/pdf"`.

5.1 Adjuntos y notas de cada documento

`zot.children(clave)` devuelve los hijos de un documento. La celda recorre los primeros 20 ítems de la biblioteca y, para cada documento, indica si tiene un PDF, otro archivo o una nota. Tener un PDF asociado amplía las posibilidades de lectura y extracción, mientras que una nota puede conservar interpretaciones y decisiones del investigador. Sin embargo, disponer del archivo no garantiza que contenga texto extraíble, ni que una nota represente fielmente el documento. La relación entre ítem y recursos permite comprobar qué evidencia está realmente disponible.

```
items = zot.top(limit=20, itemType="-note")      # sin notas independientes

for item in items:
    tipo = campo(item, "itemType")
    if tipo in ("journalArticle", "book", "preprint"):
        print(f"ARTÍCULO: {campo(item, 'title', 'Sin título')} [ID: {item['key']}]")
        for hijo in zot.children(item["key"]):
            t = campo(hijo, "itemType")
            if t == "attachment" and campo(hijo, "contentType") == "application/pdf":
                print("   └─ - Tiene un PDF adjunto")
            elif t == "attachment":
                print("   └─ - Tiene otro archivo adjunto")
            elif t == "note":
                print("   └─ ° Tiene una nota adjunta")
    elif tipo == "note":
        print(f" NOTA SUELTA: {re.sub('<[^>]+>', '', campo(item, 'note'))[:50]}...")
```

Salida registrada

```
ARTÍCULO: Companies selling authorship slots thrive in a culture that equates success with a strong
publication record. By Christine Ro & Jack Leeming [ID: AVTE4IA3]
   └─ ° Tiene una nota adjunta
   └─ ° Tiene una nota adjunta
```

```
   └─ - Tiene un PDF adjunto
ARTÍCULO: Bibliometric evaluation of Forensic Science International as a scholarly journal within the
subject category legal medicine [ID: DISXDZNA]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Reference accuracy of primary studies about reference accuracy: A descriptive study [ID: A2WC95B8]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Sneaked references: Fabricated reference metadata distort citation counts [ID: BGR79MAI]
   └─ ° Tiene una nota adjunta
   └─ - Tiene un PDF adjunto
ARTÍCULO: Detecting anomalous referencing patterns in PubMed papers suggestive of author-centric reference
list manipulation [ID: DABLKWGS]
   └─ ° Tiene una nota adjunta
   └─ ° Tiene una nota adjunta
   └─ - Tiene un PDF adjunto
ARTÍCULO: Unveiling Deception: Establishing a Taxonomic Framework for Disinformation within Scientific
Discourse [ID: Y2JEG29H]
   └─ - Tiene un PDF adjunto
ARTÍCULO: From Louvain to Leiden: guaranteeing well-connected communities [ID: QM2P4FQF]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Altmetrics at institutional level: Visibility on the web of the scientific production of Spanish
universities from Altmetric.com [ID: QG7Q3ML8]
   └─ - Tiene un PDF adjunto
ARTÍCULO: A large-scale comparison of coverage and mentions captured by the two altmetric aggregators:
Altmetric.com and PlumX [ID: YNDN4CLM]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Assessment of scientific research: improvement of science policies in Latin America [ID: P9R9DSWJ]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Tipos de estudio y métodos de investigación [ID: F568RB9M]
ARTÍCULO: Reglas y consejos sobre la investigación biológica . Los tónicos de la voluntad Santiago F . Ramón
y Cajal [ID: K6RD4J69]
ARTÍCULO: يشهوژپ هلاقم لـيصا [ID: ZGBB6SFA]
ARTÍCULO: Assessing the dissemination of COVID-19 articles across social media with altmetric and PlumX
metrics: Correlational study [ID: 6WT2GPWT]
ARTÍCULO: La ciencia abierta Y [ID: Z7YTRZ6V]
ARTÍCULO: Alfabetización en Comunicación Científica: Acreditación, OA, redes sociales, altmetrics,
bibliotecarios incrustados y gestión de la identidad digital [ID: KB3MAFSY]
ARTÍCULO: The impact of altmetrics on the visibility of open acess Brazilian nursing articles: A case study
[ID: HMGYWQT2]
   └─ - Tiene un PDF adjunto
ARTÍCULO: Altmetrics y alfabetización científica [ID: ZJFK4DAN]
ARTÍCULO: Analysis of the scientific production in politics and educational management published in SCIELO
2012-2015 [ID: 6PJGUQ3Y]
   └─ - Tiene un PDF adjunto
ARTÍCULO: PlumX as a potential tool to assess the macroscopic multidimensional impact of books [ID:
5GP7M9BC]
   └─ - Tiene un PDF adjunto
```

5.2 Cuántos documentos tienen PDF y cuáles

Consultar `children()` documento a documento exige una petición por cada uno. Es más eficiente descargar una sola vez todo el contenido de la colección con `collection_items()`, que incluye los adjuntos, y agrupar los PDF por su `parentItem`. El diccionario `pdf_por_doc` asocia la clave de cada documento con la lista de sus PDF.

El campo `linkMode` indica dónde está el archivo, y condiciona la forma de leerlo en la sección 5.3:

`linkMode`	**Significado**
`imported_file`	Archivo copiado en el almacenamiento de Zotero (carpeta `storage`)
`imported_url`	Archivo descargado de una página web y guardado en `storage`
`linked_file`	Archivo vinculado: permanece en otra carpeta de tu ordenador
`linked_url`	Enlace a una URL, sin archivo local

La proporción de documentos con PDF indica qué parte del corpus puede leerse o analizarse a texto completo. La lista de documentos sin PDF es, en la práctica, una lista de tareas: textos que hay que localizar, descargar o solicitar.

```
todo_col = zot.everything(zot.collection_items(COLECCION_ID))     # documentos, adjuntos y notas
claves_doc = {it["key"] for it in documentos}

pdf_por_doc = {}                                  # clave del documento -> lista de sus PDF
for it in todo_col:
    d = it["data"]
    if (d.get("itemType") == "attachment" and d.get("contentType") == "application/pdf"
            and d.get("parentItem") in claves_doc):
        pdf_por_doc.setdefault(d["parentItem"], []).append(it)

con_pdf = [it for it in documentos if it["key"] in pdf_por_doc]
sin_pdf = [it for it in documentos if it["key"] not in pdf_por_doc]
n = len(documentos) or 1

print(f"Documentos en la colección: {len(documentos)}")
print(f"  Con PDF: {len(con_pdf):>4} ({len(con_pdf) / n:.0%})")
print(f"  Sin PDF: {len(sin_pdf):>4} ({len(sin_pdf) / n:.0%})")
print(f"  Documentos con más de un PDF: {sum(len(v) > 1 for v in pdf_por_doc.values())}")

print("\nDocumentos CON PDF  (documento -> PDF | almacenamiento | archivo)")
filas_visibles = 5                        # documentos por lista en pantalla
for it in con_pdf[:filas_visibles]:
    print(f"  {it['key']}  {campo(it, 'title', '(sin título)')[:70]}")
    for pdf in pdf_por_doc[it["key"]]:
        print(f"     -> {pdf['key']} | {campo(pdf, 'linkMode'):<13} | {campo(pdf, 'filename', '(sin
nombre)')[:50]}")

if len(con_pdf) > filas_visibles:
    print(f"  ... y {len(con_pdf) - filas_visibles} documentos más con PDF")

print("\nDocumentos SIN PDF")
for it in sin_pdf[:filas_visibles]:
    print(f"  {it['key']}  {anio(it):<5} {campo(it, 'title', '(sin título)')[:70]}")
if len(sin_pdf) > filas_visibles:
    print(f"  ... y {len(sin_pdf) - filas_visibles} documentos más sin PDF")
```

Salida registrada

```
Documentos en la colección: 63
  Con PDF:   63 (100%)
  Sin PDF:    0 (0%)
  Documentos con más de un PDF: 0

Documentos CON PDF  (documento -> PDF | almacenamiento | archivo)
  AVTE4IA3  Companies selling authorship slots thrive in a culture that equates su
```

```
   -> KD55JQGH | imported_file | Thibault - Companies selling authorship slots thri
DISXDZNA  Bibliometric evaluation of Forensic Science International as a scholar
   -> SZMLDRXM | imported_file | Jones - 2023 - Bibliometric evaluation of Forensic
A2WC95B8  Reference accuracy of primary studies about reference accuracy: A desc
   -> GB4YP8X6 | imported_file | Logan y Hussong-Christian - Reference accuracy of
BGR79MAI  Sneaked references: Fabricated reference metadata distort citation cou
   -> SDR6LBT3 | imported_file | Besançon et al. - Sneaked references Fabricated re
DABLKWGS  Detecting anomalous referencing patterns in PubMed papers suggestive o
   -> FCTWKX6H | imported_file | Wren y Georgescu - 2024 - Detecting anomalous refe
... y 58 documentos más con PDF

Documentos SIN PDF
```

Los 63 documentos de la colección tienen PDF, de modo que todo el corpus puede consultarse a texto completo. La celda muestra los cinco primeros de cada lista; las variables `con_pdf` *y* `sin_pdf` *contienen las listas completas.*

5.3 Consultar el texto completo de un PDF

Para pasar al LLM algo más que el resumen se necesita el texto de los PDF. La función `texto_completo()` acepta la clave de un artículo o la de su PDF y prueba tres vías, porque cada una falla por motivos distintos:

Vía	Cuándo funciona	Por qué puede fallar
Índice de texto completo de Zotero	Zotero de escritorio ha indexado el PDF y lo ha sincronizado	El PDF no se ha indexado, o la sincronización de texto completo está desactivada
Archivo en tu carpeta local de Zotero	Trabajas en el mismo ordenador donde está Zotero	La ruta de tu carpeta de Zotero es distinta (parámetro `carpeta_zotero`)
Descarga desde Zotero Storage	El PDF se sincroniza con el almacenamiento de zotero.org	Usas WebDAV o archivos vinculados, o la clave no permite acceder a los archivos

Si ninguna vía funciona, la función explica el motivo en lugar de devolver un texto vacío sin aviso. Un PDF escaneado (solo imagen) no contiene texto extraíble.

El título y el resumen orientan la selección inicial, pero el texto completo permite contrastar métodos, resultados y limitaciones. La extracción requiere inspección porque puede alterar el orden de lectura, omitir tablas o introducir ruido. Cuando se utilice para sustentar una afirmación, conviene conservar la referencia del documento y verificar la ubicación de la evidencia en el original.

```python
def texto_completo(item_key, max_caracteres=8000, carpeta_zotero=Path.home() / "Zotero"):
    """Devuelve el texto del primer PDF de un ítem. Acepta la clave del artículo o la del PDF.

    Prueba tres vías, en este orden:
      1. el índice de texto completo de Zotero (API web);
      2. el archivo PDF en tu carpeta local de Zotero (storage/<clave>/archivo.pdf);
      3. la descarga del PDF desde la API (solo si el archivo está en Zotero Storage).
    """
    from pypdf import PdfReader
    import io

    def texto_de_pdf(origen):
        lector = PdfReader(origen)
        return "\n".join((p.extract_text() or "") for p in lector.pages[:40]).strip()
```

```
    item = zot.item(item_key)
    if campo(item, "itemType") == "attachment":
        adjuntos = [item]
    else:
        adjuntos = [h for h in zot.children(item_key) if campo(h, "itemType") == "attachment"]
    pdfs = [a for a in adjuntos if campo(a, "contentType") == "application/pdf"]
    if not pdfs:
        print(f"{item_key}: no tiene ningún PDF adjunto ({len(adjuntos)} adjunto(s) de otro tipo).")
        return ""
    pdf = pdfs[0]
    clave, archivo = pdf["key"], campo(pdf, "filename")

    # 1. Índice de texto completo de Zotero
    try:
        contenido = zot.fulltext_item(clave).get("content", "")
        if contenido:
            print(f"{clave}: texto obtenido del índice de Zotero.")
            return contenido[:max_caracteres]
    except Exception:
        pass    # el PDF no está indexado en el servidor: se prueba la siguiente vía

    # 2. Archivo local (adjunto almacenado o archivo vinculado)
    if campo(pdf, "linkMode") == "linked_file":
        ruta_local = Path(campo(pdf, "path"))
    else:
        ruta_local = carpeta_zotero / "storage" / clave / archivo
    if archivo and ruta_local.exists():
        texto = texto_de_pdf(str(ruta_local))
        if texto:
            print(f"{clave}: texto extraído del archivo local {ruta_local.name}.")
            return texto[:max_caracteres]

    # 3. Descarga desde Zotero Storage
    try:
        texto = texto_de_pdf(io.BytesIO(zot.file(clave)))
        if texto:
            print(f"{clave}: texto extraído del PDF descargado de Zotero Storage.")
            return texto[:max_caracteres]
    except Exception:
        pass

    print(f"{clave}: sin texto disponible. El PDF no está indexado en el servidor, no está en "
          f"{ruta_local.parent} y no se pudo descargar; si es un PDF escaneado (imagen), no contiene
texto.")
    return ""


texto = texto_completo("S2DECHFS")      # Modificar clave de un artículo o de su PDF
print(texto[:500])
```

Salida registrada

```
S2DECHFS: texto obtenido del índice de Zotero.
Quantitative Methods
in Research Evaluation
Citation indicators, altmetrics,
and artificial intelligence
Mike Thelwall
```

```
 Quantitative Methods
in Research Evaluation
Citation Indicators, Altmetrics, and Artificial Intelligence
Mike Thelwall


 Published by White Rose University Press (Universities of Leeds, Sheffield and York) University of York,
Heslington, York, UK, YO10 5DD https://universitypress.whiterose.ac.uk
Text © Mike Thelwall 2026
First published 2026
Cover designed by Kate Petherbrid
```

Consultar el contenido de un PDF. Con el texto disponible, una búsqueda con expresiones regulares localiza cada aparición de un término y muestra su contexto. La celda aplica la búsqueda al primer documento con PDF de la colección; cambia `termino` o la clave para consultar otro.

```python
termino = "citation"                       # término o expresión regular
doc = con_pdf[0]                             # o: zot.item("CLAVE_DEL_DOCUMENTO")

texto = texto_completo(doc["key"], max_caracteres=300_000)
apariciones = [m.start() for m in re.finditer(termino, texto, flags=re.IGNORECASE)]

print(f"\n{campo(doc, 'title')[:80]}")
print(f"Caracteres de texto: {len(texto):,} | apariciones de '{termino}': {len(apariciones)}\n")
for pos in apariciones[:5]:
    fragmento = texto[max(0, pos - 90): pos + 90].replace("\n", " ")
    print(f"  ...{fragmento}...")
```

Salida registrada

```
KD55JQGH: texto obtenido del índice de Zotero.

Companies selling authorship slots thrive in a culture that equates success with
Caracteres de texto: 23,283 | apariciones de 'citation': 6

  ...ast ten more papers, in journals in the top half of those ranked by the Web of Science, a citation
index. And to meet this demanding requirement, he’d turned to paper mills: fraudu...
  ...uplicated images, suspicious e-mail addresses, implausible collaborations1 and irrelevant citations.
Another red flag is “tortured phrases” resulting from attempts to hide plagiari...
  ...ication with the journal. They might already have published a paper and want to boost its citations;
paper mills often create citation networks to inflate the impact of their paper...
  ... already have published a paper and want to boost its citations; paper mills often create citation
networks to inflate the impact of their papers. And journal editors willing to ac...
  .... Otherwise paper mills can, and often do, manipulate reviews, revisions, authorships and citations.
Svetlana Shakirova, head of science at Kurmangazy Kazakh National Conservatory ...
```

El texto del primer documento, `AVTE4IA3`, suma 23.283 caracteres, con 6 apariciones de citation.

5.4 Búsquedas por texto, etiqueta y tipo

La API admite búsqueda y filtros en la propia petición, sin descargar toda la biblioteca. Con `qmode="everything"` la búsqueda incluye el texto completo indexado de los PDF, lo que la convierte en la forma más rápida de localizar documentos que mencionan un término.

Parámetro	Ejemplo	Efecto
`q`	`q="citation cartels"`	Búsqueda rápida en título y campos de creadores
`qmode`	`qmode="everything"`	Incluye el texto completo indexado; no equivale a filtrar cualquier campo
`tag`	`tag="integridad"`	Filtra por etiqueta (`"a || b"` = una u otra)
`itemType`	`itemType="-attachment"`	Filtra por tipo (`-` excluye)
`sort`, `direction`	`sort="dateAdded"`, `direction="desc"`	Ordena
`since`	`since=1520`	Solo lo modificado desde una versión de la biblioteca

Los filtros recuperan registros según condiciones explícitas y ayudan a construir subconjuntos de trabajo. Una coincidencia de texto o etiqueta no equivale a relevancia científica ni a recuperación semántica. Tampoco constituye por sí sola un sistema RAG: para hablar de esa arquitectura habría que definir cómo se recuperan evidencias y cómo se incorporan a la generación de respuestas.

```
def mostrar(resultados, titulo):
    print(f"\n{titulo} ({len(resultados)})")
    for it in resultados:
        print(f"  [{it['key']}] {campo(it, 'date', 's.f.')[:4]} · {campo(it, 'title', 'Sin título')
[:80]}")


mostrar(zot.items(q="citation cartels", limit=10), "Título/autor contiene 'citation cartels'")
mostrar(zot.items(q="paper mills", qmode="everything", limit=10), "Cualquier campo o texto completo
con 'paper mills'")
mostrar(zot.collection_items(COLECCION_ID, itemType="-attachment", sort="dateAdded",
                             direction="desc", limit=5), "Últimos 5 ítems añadidos a la colección (sin
adjuntos)")
```

Salida registrada

```
Título/autor contiene 'citation cartels' (7)
  [NMBP4FIP] 2016 · Toward the Discovery of Citation Cartels in Citation Networks
  [3A5B8JWR] s.f. · Citation cartels-02
  [E7QYE8CX] 2016 · Toward the Discovery of Citation Cartels in Citation Networks
  [3P5V6NS2] s.f. · Citation cartels-02
  [NBF3SRJG] 2016 · Toward the Discovery of Citation Cartels in Citation Networks
  [IVVY6Y52] s.f. · Citation cartels-02
  [A74D2CPH] 2016 · Toward the Discovery of Citation Cartels in Citation Networks

Cualquier campo o texto completo con 'paper mills' (10)
  [AHCUIYLG] s.f. · Texto completo
  [4IFA6YYB] s.f. · Texto completo
  [MXFNDH3U] s.f. · Full Text PDF
  [3N7L6B2Z] s.f. · Texto completo
  [FE8SPYN3] s.f. · Full Text PDF
  [S2DECHFS] s.f. · PDF
  [KD55JQGH] s.f. · PDF
  [7LALVXIA] s.f. · PDF
  [AMJ3DVYF] s.f. · PDF
  [Y6SJ5B9C] s.f. · PDF

Últimos 5 ítems añadidos a la colección (sin adjuntos) (5)
```

```
[D4NWKTT9] s.f. · Sin título
[2A5RHJHF] s.f. · Sin título
[GBIE2Z5I] s.f. · Sin título
[Q4TCQZ7I] s.f. · Sin título
[9SF8K98R] s.f. · Sin título
```

6

Metadatos: cobertura, etiquetas, duplicados y resúmenes

En este apartado se identifican los metadatos más relevantes para el trabajo académico, se cuenta cuántos documentos de la colección los tienen, se revisan las etiquetas y los posibles duplicados, se exporta el corpus y se completan los resúmenes ausentes a partir del texto del PDF. Se apoya en `documentos` (sección 3.5), `pdf_por_doc` y `texto_completo()` (parte 5). Aquí se define el interruptor `ESCRIBIR_EN_ZOTERO`, que controla todas las escrituras en la biblioteca.

6.1 Metadatos relevantes y su cobertura

Cada tipo de documento de Zotero admite un conjunto propio de campos (la sección 4.4 muestra cómo consultarlos). Para la gestión de una revisión de literatura, los más relevantes son los siguientes:

Campo (API)	En Zotero	Para qué sirve
`title`	Título	Identificar el trabajo; base para detectar duplicados
`creators`	Autores, editores	Citar, agrupar por autoría, analizar colaboración
`date`	Fecha	Ordenar y filtrar por periodo
`abstractNote`	Resumen	Cribado temático y contexto para un LLM (parte 9)
`publicationTitle`	Publicación	Revista o serie; análisis de fuentes
`DOI`	DOI	Identificador persistente; enlace con OpenAlex y otras bases
`volume`, `issue`, `pages`	Volumen, número, páginas	Referencia completa
`url`	URL	Acceso al original
`language`	Idioma	Filtrar por idioma; criterios de inclusión
`ISSN`, `ISBN`	ISSN, ISBN	Identificar la revista o el libro
`tags`	Etiquetas	Codificación temática propia del investigador

La API devuelve, para cada documento, todos los campos que admite su tipo, aunque estén vacíos. Por eso la celda distingue entre los documentos a los que **se aplica** un campo (un libro no tiene DOI en Zotero, sino ISBN) y los que **tienen valor** en él.

La tabla de cobertura muestra qué análisis permite la colección. Si faltan muchos resúmenes, un cribado temático o un análisis con un LLM trabajará con información parcial; si faltan DOI, la correspondencia con OpenAlex será menos fiable. La cobertura mide presencia de un valor, no su exactitud.

```
CAMPOS_RELEVANTES = ["title", "creators", "date", "abstractNote", "publicationTitle", "DOI",
                     "volume", "issue", "pages", "url", "language", "ISSN", "ISBN", "tags"]


def tiene_valor(item, nombre):
    """True si el campo existe y no está vacío."""
    valor = item["data"].get(nombre)
    return bool(valor.strip()) if isinstance(valor, str) else bool(valor)


cobertura = {}
print(f"Cobertura de metadatos: {len(documentos)} documentos de la colección {COLECCION_ID}\n")
print(f"{'Campo':<17} {'Se aplica':>9} {'Con valor':>10} {'%':>6} {'Sin valor':>10}")
for nombre in CAMPOS_RELEVANTES:
    aplica = [it for it in documentos if nombre in it["data"]]
    vacios = [it["key"] for it in aplica if not tiene_valor(it, nombre)]
    cobertura[nombre] = {"aplica": len(aplica), "sin_valor": vacios}
    con_valor = len(aplica) - len(vacios)
    porcentaje = f"{con_valor / len(aplica):.0%}" if aplica else "-"
    print(f"{nombre:<17} {len(aplica):>9} {con_valor:>10} {porcentaje:>6} {len(vacios):>10}")

# Claves de los documentos sin un campo concreto, por ejemplo sin DOI:
# cobertura["DOI"]["sin_valor"]
```

Salida registrada

```
Cobertura de metadatos: 63 documentos de la colección 6APD6L29

Campo             Se aplica  Con valor      %  Sin valor
title                    63         63   100%          0
creators                 63         61    97%          2
date                     63         44    70%         19
abstractNote             63         52    83%         11
publicationTitle         55         30    55%         25
DOI                      63         37    59%         26
volume                   56         27    48%         29
issue                    55         19    35%         36
pages                    56         26    46%         30
url                      63         37    59%         26
language                 63         63   100%          0
ISSN                     57         27    47%         30
ISBN                      2          1    50%          1
tags                     63         12    19%         51
```

*Se considera **cobertura** de un campo la proporción de documentos con valor en él, entre aquellos cuyo tipo lo admite. Todos los documentos tienen título e idioma; el resumen falta en 11 de 63 (83 % de cobertura), el DOI en 26 (59 %) y 12 documentos tienen etiquetas (19 %).*

6.2 Etiquetas de la colección

Las etiquetas son la codificación propia del investigador: temas, métodos o estados de revisión. Su uso inconsistente (mayúsculas, idiomas, sinónimos) dificulta recuperar documentos, por lo que conviene revisarlas. La celda cuenta en cuántos documentos aparece cada etiqueta y selecciona los que tienen una etiqueta concreta, indicada en `etiqueta_buscada`. La coincidencia es exacta: `Integridad` e `integridad` son etiquetas distintas, y armonizarlas es una decisión del investigador.

```
frecuencia_etiquetas = Counter()
for it in documentos:
    frecuencia_etiquetas.update(set(etiquetas(it)))

print(f"{len(frecuencia_etiquetas)} etiquetas distintas en {len(documentos)} documentos\n")
for etiqueta, n in frecuencia_etiquetas.most_common(15):
    print(f"{n:>4} documento(s) | {etiqueta}")

# Ejemplo: la etiqueta más frecuente de la lista; puede sustituirse por cualquier otra, p. ej. "DORA"
etiqueta_buscada = frecuencia_etiquetas.most_common(1)[0][0] if frecuencia_etiquetas else ""
seleccion = [it for it in documentos if etiqueta_buscada in etiquetas(it)]
if etiqueta_buscada:
    print(f"\nDocumentos con la etiqueta '{etiqueta_buscada}': {len(seleccion)}")
else:
    print("\nLa colección no tiene etiquetas.")
for it in seleccion[:10]:
    print(f"  {it['key']}  {campo(it, 'title')[:70]}")
```

Salida registrada

```
17 etiquetas distintas en 63 documentos

   5 documento(s) | revision:sin-doi
   3 documento(s) | Computer Science - Digital Libraries
   2 documento(s) | resumen-desde-pdf
   2 documento(s) | Computer Science - Artificial Intelligence
   2 documento(s) | Computer Science - Computation and Language
   2 documento(s) | Computer Science - Information Retrieval
   1 documento(s) | Computer Science - Computers and Society
   1 documento(s) | Medicine and Health Sciences
   1 documento(s) | Narrativas
   1 documento(s) | Métricas Responsables
   1 documento(s) | Bibliometría Evaluativa
   1 documento(s) | Bases de Datos
   1 documento(s) | Indicadores Bibliométricos
   1 documento(s) | Bibliometría Narrativa
   1 documento(s) | DORA

Documentos con la etiqueta 'revision:sin-doi': 5
  AVTE4IA3  Companies selling authorship slots thrive in a culture that equates su
  DISXDZNA  Bibliometric evaluation of Forensic Science International as a scholar
  A2WC95B8  Reference accuracy of primary studies about reference accuracy: A desc
  BGR79MAI  Sneaked references: Fabricated reference metadata distort citation cou
  DABLKWGS  Detecting anomalous referencing patterns in PubMed papers suggestive o
```

La colección tiene 17 etiquetas distintas. La más frecuente, `revision:sin-doi` (5 documentos), es una etiqueta de trabajo creada en la sección 6.8; las etiquetas temáticas aparecen casi siempre en un único documento.

6.3 Posibles duplicados por DOI y título

Un registro repetido altera los recuentos y aumenta el peso aparente de un trabajo en el corpus. La celda agrupa los documentos por DOI normalizado (sin el prefijo `https://doi.org/` y en minúsculas) y por título normalizado con `normalizar_titulo()` (sección 2.7), y muestra los grupos con más de un registro. Por defecto revisa la colección de trabajo; asignando `corpus_revision = biblioteca` (sección 3.2) revisa toda la biblioteca.

Un DOI coincidente es un indicio fuerte de duplicado; un título coincidente puede corresponder a versiones distintas de un trabajo (preprint y artículo) o a trabajos diferentes con el mismo título. La celda solo señala candidatos: no fusiona ni borra registros, operación que se hace en Zotero con la herramienta *Duplicate Items*.

```
from collections import defaultdict

corpus_revision = documentos            # o: biblioteca (sección 3.2), para toda la biblioteca


def doi_normalizado(valor):
    texto = (valor or "").strip().lower()
    return re.sub(r"^(?:https?://(?:dx\.)?doi\.org/|doi:\s*)", "", texto).strip()


por_doi, por_titulo = defaultdict(list), defaultdict(list)
for it in corpus_revision:
    if campo(it, "itemType") in ("note", "attachment", "annotation"):
        continue
    if doi_normalizado(campo(it, "DOI")):
        por_doi[doi_normalizado(campo(it, "DOI"))].append(it)
    if normalizar_titulo(campo(it, "title")):
        por_titulo[normalizar_titulo(campo(it, "title"))].append(it)

for criterio, grupos in (("DOI", por_doi), ("Título", por_titulo)):
    candidatos = {k: v for k, v in grupos.items() if len(v) > 1}
    print(f"\nCriterio {criterio}: {len(candidatos)} grupo(s) candidato(s)")
    for valor, grupo in list(candidatos.items())[:10]:
        print(f"  {valor[:75]}")
        for it in grupo:
            print(f"     {it['key']} | {campo(it, 'itemType'):<15} | {anio(it)}")
```

Salida registrada

```
Criterio DOI: 2 grupo(s) candidato(s)
  10.1162/qss_a_00339
     K3RXLGYE | journalArticle  | 2025
     47LNQY9E | journalArticle  | 2025
  10.1038/s41598-024-71230-8
     YE263I5R | journalArticle  | 2024
     RTIHH9QH | journalArticle  | 2024

Criterio Título: 4 grupo(s) candidato(s)
  detecting anomalous citation groups in journal networks
     A8FBURZN | journalArticle  | s.f.
     JDUWMQQL | journalArticle  | 2021
  using bibliometrics to detect questionable authorship and affiliation pract
     K3RXLGYE | journalArticle  | 2025
     47LNQY9E | journalArticle  | 2025
  identifying fabricated networks within authorship for sale enterprises
     YE263I5R | journalArticle  | 2024
     RTIHH9QH | journalArticle  | 2024
```

```
  forensic scientometrics fosci paris declaration
     NKSTU3HY | journalArticle  | 2024
     CJUMCU6W | journalArticle  | 2024
```

Dos pares de documentos comparten DOI (`K3RXLGYE` y `47LNQY9E`; `YE263I5R` y `RTIHH9QH`) y también título: son duplicados casi seguros. Los otros dos grupos coinciden solo en el título; en el primero los años difieren (s.f. y 2021), lo que obliga a revisar si se trata del mismo trabajo.

6.4 Exportar el corpus a JSON

Un archivo JSON con los campos seleccionados permite analizar la colección con otras herramientas (pandas, R, un LLM) y documentar qué información se utilizó en un estudio. La celda construye un registro por documento con su clave, versión, tipo, título, DOI, fecha, autores, resumen, etiquetas, colecciones y disponibilidad de PDF (sección 5.2), y lo guarda con `guardar_json()` en `corpus_<clave>.json`. El archivo no es una copia de seguridad de la biblioteca: no incluye los PDF, las notas ni el resto de los campos.

```
def nombres_creadores(item):
    return [c.get("name") or ", ".join(x for x in (c.get("lastName", ""), c.get("firstName", "")) if
x)
            for c in campo(item, "creators", [])]


corpus_json = [{
    "zotero_key": it["key"],
    "zotero_version": it.get("version"),
    "tipo": campo(it, "itemType"),
    "titulo": campo(it, "title"),
    "doi": campo(it, "DOI"),
    "fecha": campo(it, "date"),
    "autores": nombres_creadores(it),
    "resumen": campo(it, "abstractNote"),
    "etiquetas": etiquetas(it),
    "colecciones": campo(it, "collections", []),
    "tiene_pdf": it["key"] in pdf_por_doc,
} for it in documentos]

archivo_json = BASE_DIR / f"corpus_{COLECCION_ID}.json"
guardar_json(archivo_json, corpus_json)
print(f"{len(corpus_json)} registros guardados en {archivo_json.name}\n")
print(json.dumps(corpus_json[:1], ensure_ascii=False, indent=2)[:1500])
```

Salida registrada

```
63 registros guardados en corpus_6APD6L29.json

[
  {
    "zotero_key": "AVTE4IA3",
    "zotero_version": 10961,
    "tipo": "journalArticle",
    "titulo": "Companies selling authorship slots thrive in a culture that equates success with a strong
publication record. By Christine Ro & Jack Leeming",
    "doi": "",
    "fecha": "",
    "autores": [
      "Thibault, Sébastien"
    ],
```

```
    "resumen": "",
    "etiquetas": [
      "revision:sin-doi"
    ],
    "colecciones": [
      "6APD6L29"
    ],
    "tiene_pdf": true
  }
]
```


Ejecuta la celda para ver los resultados de tu biblioteca.

6.5 Documentos sin resumen

El resumen (`abstractNote`) es el campo que más condiciona el cribado temático y el uso de un LLM. La celda lista los documentos sin resumen e indica cuáles tienen PDF, porque solo en esos puede recuperarse el resumen del propio texto (sección 6.6). La lista `recuperables` reúne esos casos.

```
sin_resumen = [it for it in documentos if not tiene_valor(it, "abstractNote")]
recuperables = [it for it in sin_resumen if it["key"] in pdf_por_doc]

print(f"{len(sin_resumen)} de {len(documentos)} documentos no tienen resumen.\n")
for it in sin_resumen:
    estado = "con PDF" if it["key"] in pdf_por_doc else "sin PDF"
    print(f"  {it['key']}  {estado:<8} {campo(it, 'title', '(sin título)')[:70]}")

print(f"\nCandidatos a completar desde el PDF: {len(recuperables)}")
```

Salida registrada

```
11 de 63 documentos no tienen resumen.

  AVTE4IA3  con PDF  Companies selling authorship slots thrive in a culture that equates su
  5E84NC7W  con PDF  Some top academics cite themselves heavily, and researchers are debati
  A8FBURZN  con PDF  Detecting anomalous citation groups in journal networks
  93LWGPD3  con PDF  Systematic Manipulation of the Publication Process: Flowcharts and Inf
  PISP9FAX  con PDF  RESSH2025 Book of Abstracts
  3TVP6AXN  con PDF  Paper mills research
  9W2FMFCA  con PDF  Informe bibliométrico sobre la producción científica del profesor Juan
  NIIXVJRS  con PDF  Beyond misconduct: forging an ethical future in academia
  PGFWYPMY  con PDF  Elsevier probes dodgy citations
  WK46QC2Z  con PDF  Citation manipulation through citation mills and pre-print servers
  I87AJSGL  con PDF  Predator Effect: Understanding the Past, Present and Future of Decepti

Candidatos a completar desde el PDF: 11
```

Los 11 documentos sin resumen tienen PDF, por lo que todos son candidatos a completarse desde el texto (sección 6.6).

6.6 Extraer el resumen del texto del PDF

En la mayoría de los artículos el resumen aparece al comienzo, tras un encabezado como *Abstract*, *Resumen* o *Summary*, y termina donde empiezan las palabras clave o la introducción. `resumen_desde_pdf()` busca ese encabezado, recorta el texto hasta el siguiente marcador, une las palabras cortadas por guiones de final de línea y normaliza los espacios. Si no

encuentra el encabezado, o el fragmento es demasiado corto para ser un resumen, devuelve una cadena vacía en lugar de un texto dudoso.

La primera celda prueba la función con un texto de ejemplo, construido para imitar la primera página de un artículo extraída de un PDF.

```python
def resumen_desde_pdf(texto, max_caracteres=3000, min_caracteres=150):
    """Devuelve la sección de resumen de un texto extraído de un PDF, o "" si no la encuentra."""
    inicio = re.search(r"(?im)^\s*(abstract|resumen|summary|resumo)\b[\s.:-]*", texto)
    if not inicio:
        return ""
    resto = texto[inicio.end():]
    fin = re.search(r"(?im)^\s*(key\s*words?|palabras\s+clave|1\.?\s*introduc\w*|introduc\w*"
                    r"|abstract|resumen)\b", resto)
    seccion = resto[:fin.start()] if fin else resto[:max_caracteres]
    seccion = re.sub(r"(\w)-\n(\w)", r"\1\2", seccion)          # une palabras cortadas
    seccion = re.sub(r"\s+", " ", seccion).strip()
    return seccion[:max_caracteres] if len(seccion) >= min_caracteres else ""


texto_ejemplo = """Journal of Research Integrity 12 (2025) 1-15
Network indicators for the detection of citation cartels
Ana Pérez, Luis Gómez
Abstract
Citation cartels are groups of journals or authors that cite each other to inflate
their indicators. This study compares three network measures on a sample of 1,200
journals and evaluates how well each one separates known cases from the rest. The
results suggest that reciprocity combined with temporal concen-
tration identifies most documented cases with few false alarms.
Keywords: citation cartels; research integrity; network analysis
1. Introduction
The evaluation of journals by citation indicators creates incentives..."""

propuesta = resumen_desde_pdf(texto_ejemplo)
print(f"Resumen detectado ({len(propuesta)} caracteres):\n")
print(propuesta)
print("\nSin encabezado de resumen:", repr(resumen_desde_pdf("1. Introduction\nTexto sin resumen.")))
```

Salida generada al ejecutar

```
Resumen detectado (372 caracteres):

Citation cartels are groups of journals or authors that cite each other to inflate their indicators. This
study compares three network measures on a sample of 1,200 journals and evaluates how well each one
separates known cases from the rest. The results suggest that reciprocity combined with temporal
concentration identifies most documented cases with few false alarms.

Sin encabezado de resumen: ''
```

La función conserva el resumen y descarta el título, los autores, las palabras clave y la introducción; la palabra concentration, cortada en dos líneas, se recompone. Cuando el texto no contiene un encabezado de resumen, devuelve una cadena vacía.

Con un documento de tu colección. La celda siguiente aplica la función al primer documento de `recuperables` (sección 6.5). Revisa el resultado antes de guardarlo: en algunos PDF el texto extraído altera el orden de las columnas o mezcla el resumen con notas al pie.

```
if not recuperables:
    print("No hay documentos sin resumen que tengan PDF en esta colección.")
else:
    doc = recuperables[0]
    texto = texto_completo(doc["key"], max_caracteres=30_000)
    propuesta = resumen_desde_pdf(texto)
    print(f"\n{doc['key']} | {campo(doc, 'title')[:80]}\n")
    print(propuesta or "No se encontró una sección de resumen en el texto del PDF.")
```

Salida registrada

```
KD55JQGH: texto obtenido del índice de Zotero.

AVTE4IA3 | Companies selling authorship slots thrive in a culture that equates success with

No se encontró una sección de resumen en el texto del PDF.
```

El primer candidato, `AVTE4IA3`, es una noticia de Nature sin sección de resumen, por lo que la función no encuentra el encabezado y no propone texto. Es el comportamiento esperado: devolver una cadena vacía es preferible a guardar un fragmento que no es un resumen. La sección 10.4 trata estos casos con un LLM.

6.7 Guardar el resumen en Zotero

`completar_resumen()` escribe el resumen en el campo `abstractNote` de un documento y le añade la etiqueta `resumen-desde-pdf`, para que el origen del dato quede registrado y pueda revisarse desde Zotero. No modifica documentos que ya tengan resumen.

La celda define también el interruptor `ESCRIBIR_EN_ZOTERO`, que usan todas las funciones de escritura del manual (partes 6, 7 y 13). Con `False`, las funciones muestran lo que harían sin modificar la biblioteca; con `True`, escriben de verdad, para lo cual la clave necesita *Allow write access* (sección 2.2).

El resumen procede del propio documento, pero la extracción automática puede introducir errores. Conviene revisar cada propuesta, o una muestra cuando se procesan muchos documentos, antes de activar la escritura. La etiqueta permite localizar después los resúmenes añadidos y deshacer el cambio si fuera necesario.

```
ESCRIBIR_EN_ZOTERO = True     # True: las funciones de escritura modifican tu biblioteca


def completar_resumen(item_key, resumen, etiqueta="resumen-desde-pdf"):
    """Guarda un resumen en abstractNote si el documento no lo tiene. Devuelve un mensaje."""
    item = zot.item(item_key)
    if tiene_valor(item, "abstractNote"):
        return f"{item_key}: ya tiene resumen; no se modifica."
    if not resumen:
        return f"{item_key}: no hay texto de resumen que guardar."
    if not ESCRIBIR_EN_ZOTERO:
        return f"[simulación] {item_key}: se guardaría un resumen de {len(resumen)} caracteres."
    datos = item["data"]
    datos["abstractNote"] = resumen
```

```
    datos["tags"] = datos.get("tags", []) + [{"tag": etiqueta}]
    zot.update_item(datos)
    return f"{item_key}: resumen guardado y etiqueta '{etiqueta}' añadida."


if recuperables:
    print(completar_resumen(recuperables[0]["key"], propuesta))
```

Salida registrada

```
AVTE4IA3: no hay texto de resumen que guardar.
```

`AVTE4IA3` *no tiene sección de resumen, de modo que no hay texto que guardar.*

En lote. La celda siguiente repite el proceso para los documentos de `recuperables` y muestra cada propuesta antes del mensaje de escritura, de modo que puedan revisarse. El parámetro `limite` evita procesar toda la colección en la primera prueba.

```
limite = 5
for doc in recuperables[:limite]:
    propuesta = resumen_desde_pdf(texto_completo(doc["key"], max_caracteres=30_000))
    print(f"\n{doc['key']} | {campo(doc, 'title')[:70]}")
    print(f"   Propuesta: {propuesta[:200]}{'...' if len(propuesta) > 200 else ''}")
    print("   " + completar_resumen(doc["key"], propuesta))
```

Salida registrada

```
KD55JQGH: texto obtenido del índice de Zotero.

AVTE4IA3 | Companies selling authorship slots thrive in a culture that equates su
   Propuesta:
   AVTE4IA3: no hay texto de resumen que guardar.
CLK46GYK: texto obtenido del índice de Zotero.

5E84NC7W | Some top academics cite themselves heavily, and researchers are debati
   Propuesta:
   5E84NC7W: no hay texto de resumen que guardar.
EZ5CWSSC: texto obtenido del índice de Zotero.

A8FBURZN | Detecting anomalous citation groups in journal networks
   Propuesta:
   A8FBURZN: no hay texto de resumen que guardar.
UJ9AC7E7: texto obtenido del índice de Zotero.

93LWGPD3 | Systematic Manipulation of the Publication Process: Flowcharts and Inf
   Propuesta:
   93LWGPD3: no hay texto de resumen que guardar.
4KHT582K: texto obtenido del índice de Zotero.

PISP9FAX | RESSH2025 Book of Abstracts
   Propuesta:
   PISP9FAX: no hay texto de resumen que guardar.
```

Ninguno de los cinco primeros candidatos tiene una sección de resumen reconocible: son noticias, informes o documentos sin encabezado de resumen. En la ejecución del 30/10/2026, el resumen de `Y2JEG29H` *sí se extrajo y se guardó con la etiqueta* `resumen-desde-pdf`*, y por eso ya no figura entre los documentos sin resumen.*

6.8 Añadir una etiqueta conservando las existentes

Las etiquetas también sirven para registrar el estado de revisión de cada documento y recuperarlo después en Zotero. `anadir_etiqueta()` lee el documento, comprueba que la etiqueta no exista y la añade a la lista de etiquetas, sin reemplazar las anteriores. Como el resto de las escrituras, respeta `ESCRIBIR_EN_ZOTERO` (sección 6.7).

El ejemplo marca con `revision:sin-doi` los primeros documentos sin DOI según la tabla de cobertura (sección 6.1). Si Zotero informa de un conflicto de versión, el documento se modificó después de leerlo: basta con volver a ejecutar la celda.

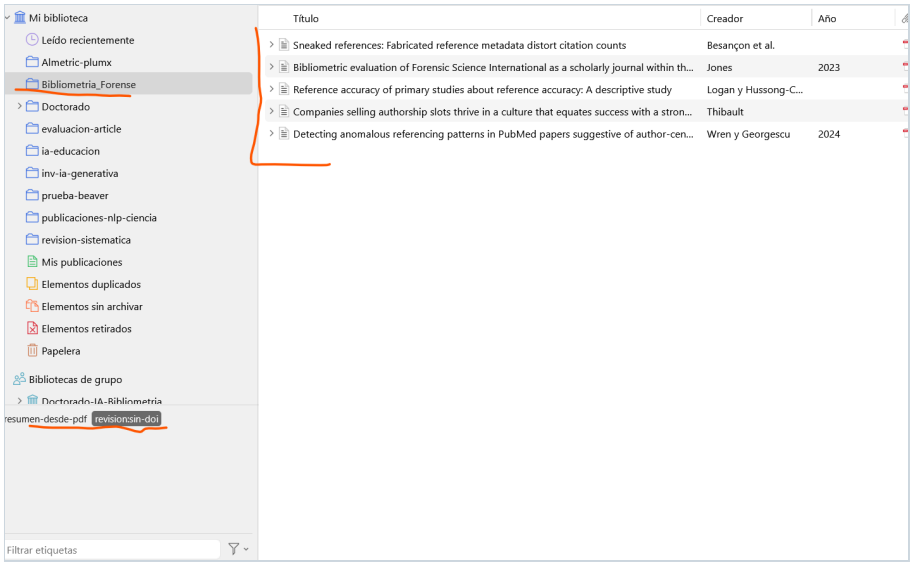
```
def anadir_etiqueta(item_key, etiqueta):
    """Añade una etiqueta a un documento sin borrar las que ya tiene. Devuelve un mensaje."""
    item = zot.item(item_key)
    datos = item["data"]
    if datos.get("itemType") in ("note", "attachment", "annotation"):
        return f"{item_key}: es un ítem de tipo {datos['itemType']}; la función está pensada para
documentos."
    if etiqueta in etiquetas(item):
        return f"{item_key}: ya tiene la etiqueta '{etiqueta}'; no se modifica."
    if not ESCRIBIR_EN_ZOTERO:
        return f"[simulación] {item_key}: se añadiría '{etiqueta}' a sus {len(datos.get('tags', []))}
etiqueta(s)."
    datos["tags"] = datos.get("tags", []) + [{"tag": etiqueta}]
    zot.update_item(datos)
    return f"{item_key}: etiqueta '{etiqueta}' añadida; se conservan las {len(datos['tags']) - 1}
anteriores."


for clave in cobertura["DOI"]["sin_valor"][:5]:
    print(anadir_etiqueta(clave, "revision:sin-doi"))
```

Salida registrada

```
AVTE4IA3: ya tiene la etiqueta 'revision:sin-doi'; no se modifica.
DISXDZNA: ya tiene la etiqueta 'revision:sin-doi'; no se modifica.
A2WC95B8: ya tiene la etiqueta 'revision:sin-doi'; no se modifica.
BGR79MAI: ya tiene la etiqueta 'revision:sin-doi'; no se modifica.
DABLKWGS: ya tiene la etiqueta 'revision:sin-doi'; no se modifica.
```

La etiqueta `revision:sin-doi` se añadió a los cinco primeros documentos sin DOI, conservando las que ya tenían.



Colección Bibliometria_Forense en Zotero. El selector de etiquetas, abajo a la izquierda, muestra `resumen-desde-pdf` *(sección 6.7) y* `revision:sin-doi` *; al seleccionar esta última, la lista se reduce a los cinco documentos marcados.*

7

Notas y organización de la biblioteca

En este apartado se leen las notas de la colección y de un documento, se crean notas nuevas de dos tipos, vinculadas a un documento e independientes dentro de una colección, y se incorpora un documento a otra colección sin duplicarlo. Las notas son el lugar natural para conservar fichas de lectura, decisiones de inclusión y diarios de revisión junto a las fuentes. Para leerlas, la clave necesita *Allow notes access*; para crearlas, *Allow write access* y `ESCRIBIR_EN_ZOTERO = True` (sección 6.7).

Zotero distingue dos clases de notas. Una **nota hija** pertenece a un documento: guarda su clave en `parentItem` y aparece debajo de él. Una **nota independiente** es un ítem de primer nivel: no tiene `parentItem` y puede asignarse a una o varias colecciones mediante el campo `collections`. En ambos casos el contenido, en el campo `note`, es HTML.

7.1 Leer las notas de una colección y de un documento

`texto_de_nota()` convierte el HTML de una nota en texto plano. La celda clasifica las notas de la colección en hijas e independientes, muestra el comienzo de las primeras y, después, lee las notas de un documento concreto con `zot.children(clave, itemType="note")`.

```
coleccion_notas = COLECCION_ID          # cambia aquí la clave para leer otra colección (sección 3.3)
clave_documento = documentos[0]["key"]    # documento cuyas notas se leen (sección 3.5)


def texto_de_nota(nota_html):
    """Convierte el HTML de una nota en texto plano."""
    texto = re.sub(r"<(br|/p|/h\d|/li|/div)[^>]*>", "\n", nota_html or "")
    return html.unescape(re.sub(r"<[^>]+>", "", texto)).strip()


contenido = zot.everything(zot.collection_items(coleccion_notas))   # consulta actualizada
notas = [it for it in contenido if campo(it, "itemType") == "note"]
hijas = [nt for nt in notas if campo(nt, "parentItem")]
independientes = [nt for nt in notas if not campo(nt, "parentItem")]
print(f"Notas en la colección {coleccion_notas}: {len(notas)} "
      f"({len(hijas)} vinculadas a un documento, {len(independientes)} independientes)\n")
for nt in notas[:8]:
    origen = f"documento {campo(nt, 'parentItem')}" if campo(nt, "parentItem") else "independiente"
    print(f"[{nt['key']}] {origen}")
    print("   " + texto_de_nota(campo(nt, "note"))[:150].replace("\n", " | "))

doc = zot.item(clave_documento)
notas_doc = zot.children(clave_documento, itemType="note")
print(f"\nDocumento {clave_documento}: {campo(doc, 'title')[:70]}")
```

```
print(f"Tiene {len(notas_doc)} nota(s).")
for nt in notas_doc:
    print("   " + texto_de_nota(campo(nt, "note"))[:200].replace("\n", " | "))

if not notas:
    print("\nLa API no devolvió ninguna nota. Si la colección tiene notas en Zotero, la clave no tiene "
          "el permiso 'Allow notes access' (sección 2.2): sin él, la API omite las notas al leer.")
```

Salida registrada

```
Notas en la colección 6APD6L29: 20 (5 vinculadas a un documento, 15 independientes)

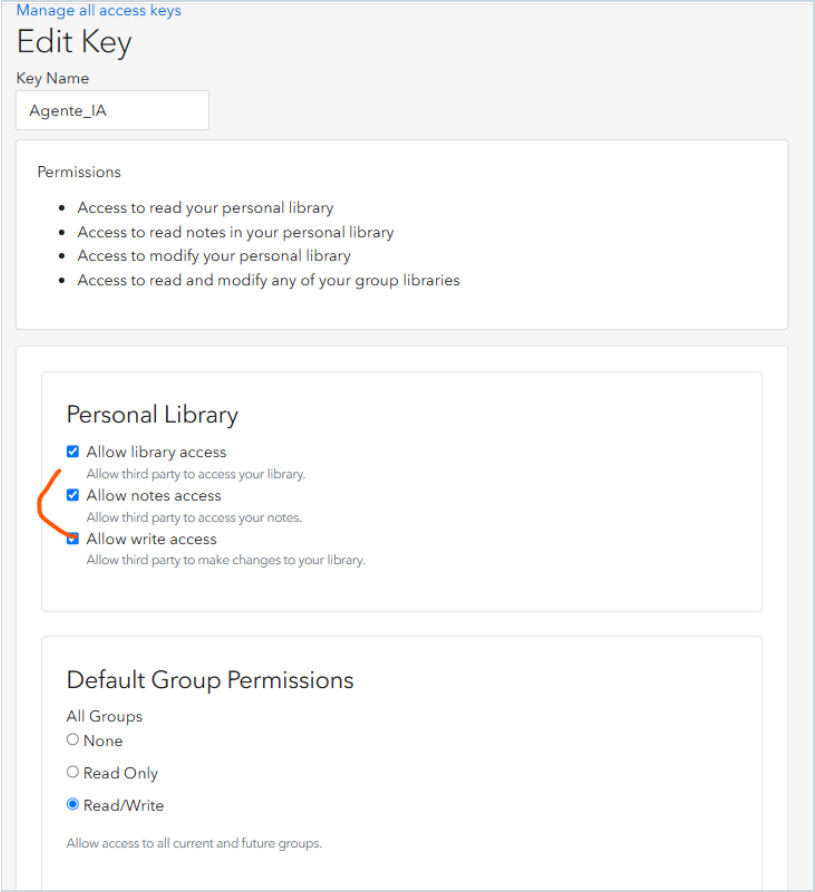
[D4NWKTT9] documento AVTE4IA3
   Resumen generado por LLM | Este texto académico, publicado en Nature, tiene como objetivo principal exponer el funcionamiento y la proliferación de las
[2A5RHJHF] independiente
   Nuevas fuentes (2026-10-30) | The Origins and Veracity of References ‘Cited’ by Generative Artificial Intelligence Applications: Implications for the Qu
[GBIE2Z5I] independiente
   Palabras clave integradoras - bibliometría forense e integridad científica | Términos principales | Forensic Scientometrics / Bibliometría forense | Researc
[Q4TCQZ7I] independiente
   Nuevas fuentes (2026-10-30) | As dimensões do fenômeno produtivismo acadêmico e seu efeito danoso na atividade científica | Citas: 3 | Año: 2024 | Búsqued
[9SF8K98R] independiente
   Palabras clave integradoras - bibliometría forense e integridad científica | Términos principales | Citation manipulation / Manipulación de citas | Publicat
[SQQXPZD2] independiente
   Palabras clave integradoras | Términos principales | Forensic Scientometrics / Bibliometría Forense | Scientific Integrity / Integridad científica | Research
[J8N7PEST] documento AVTE4IA3
   Resumen generado por LLM | Este texto académico, publicado en Nature, tiene como objetivo principal exponer el funcionamiento y el impacto de las "fábri
[DRN2CQ4H] independiente
   Estado de la colección, 30/10/2026 | Documentos: 63 | Con PDF: 63; sin PDF: 0 | Sin resumen: 12 (recuperables desde el PDF: 12) | Sin DOI: 26 de 63 documentos

Documento AVTE4IA3: Companies selling authorship slots thrive in a culture that equates su
Tiene 2 nota(s).
   Resumen generado por LLM | Este texto académico, publicado en Nature, tiene como objetivo principal exponer el funcionamiento y la proliferación de las "fábricas de papers" (paper mills), empresas fraud
   Resumen generado por LLM | Este texto académico, publicado en Nature, tiene como objetivo principal exponer el funcionamiento y el impacto de las "fábricas de artículos" (paper mills), empresas fraudule
```

Con el permiso de notas activado, la API devuelve 20 notas en la colección: 5 vinculadas a un documento y 15 independientes.

La celda vuelve a consultar la colección en lugar de reutilizar `todo_col` (sección 5.2), de modo que incluye las notas creadas después y permite cambiar de colección en `coleccion_notas` sin ejecutar de nuevo las partes anteriores. La variable `clave_documento` indica el documento cuyas notas se leen.

Si la celda devuelve 0 notas en una colección que sí las tiene, la causa más probable es el permiso de la clave. Sin *Allow notes access*, la API omite las notas en las consultas y responde `404 Not Found` al pedir una nota por su clave, aunque la clave permita crearlas con *Allow write access*. El permiso se activa en zotero.org, en **Settings** > **Security** > **Edit Key** (sección 2.2), sin necesidad de crear una clave nueva.



Pantalla Edit Key de una clave existente. En Personal Library, Allow notes access *permite leer las notas y* Allow write access *permite crear y modificar ítems; con ambos marcados funcionan todas las secciones de la parte 7. Tras guardar los cambios, vuelve a ejecutar la sección 7.1.*

7.2 Crear una nota vinculada a un documento

`crear_nota()` sirve para los dos tipos de nota. Parte de la plantilla oficial que devuelve `zot.item_template("note")`, rellena el contenido HTML y las etiquetas y, según los argumentos, la vincula a un documento (`padre`) o la asigna a una colección (`coleccion`). Devuelve la clave de la nota creada, que puede usarse después para modificarla (sección 7.4).

El ejemplo crea una ficha de lectura para el primer documento de la colección, con apartados que el investigador completa después en Zotero. El título del documento se escapa con `html.escape` para que un carácter como `<` o `&` no rompa el HTML de la nota.

```
def crear_nota(contenido_html, padre=None, coleccion=None, tags=()):
    """Crea una nota hija (padre = clave de un documento) o independiente (coleccion = clave).

    Devuelve la clave de la nota creada, o None si la escritura está simulada o falla.
    """
    nota = zot.item_template("note")
    nota["note"] = contenido_html
    nota["tags"] = [{"tag": t} for t in tags]
    if padre:
        nota["parentItem"] = padre
```

```
        destino = f"el documento {padre}"
    elif coleccion:
        nota["collections"] = [coleccion]
        destino = f"la colección {coleccion}"
    else:
        destino = "la biblioteca, sin colección"
    if not ESCRIBIR_EN_ZOTERO:
        print(f"[simulación] Nota de {len(contenido_html)} caracteres para {destino}; no se envía.")
        return None
    respuesta = zot.create_items([nota])
    if respuesta.get("successful"):
        clave = respuesta["successful"]["0"]["key"]
        print(f"Nota {clave} creada en {destino}.")
        return clave
    print(f"No se pudo crear la nota: {respuesta.get('failed')}")
    return None
```

```
clave_documento = documentos[0]["key"]    # documento al que se vincula la ficha (sección 3.5)
doc = zot.item(clave_documento)
print(f"Documento: {doc['key']} | {campo(doc, 'title')}\n")

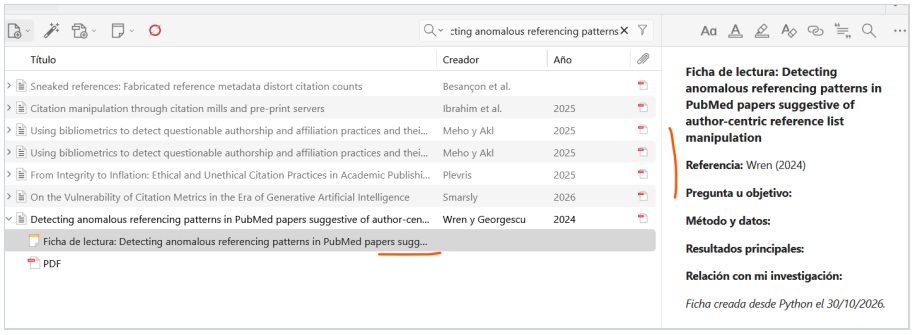
ficha = (
    f"<h2>Ficha de lectura: {html.escape(campo(doc, 'title', '(sin título)'))}</h2>"
    f"<p><b>Referencia:</b> {html.escape(primer_autor(doc))} ({anio(doc)})</p>"
    "<p><b>Pregunta u objetivo:</b> </p>"
    "<p><b>Método y datos:</b> </p>"
    "<p><b>Resultados principales:</b> </p>"
    "<p><b>Relación con mi investigación:</b> </p>"
    f"<p><i>Ficha creada desde Python el {datetime.date.today():%d/%m/%Y}.</i></p>"
)
clave_ficha = crear_nota(ficha, padre=doc["key"], tags=("ficha-lectura",))
```

Salida registrada

```
Documento: AVTE4IA3 | Companies selling authorship slots thrive in a culture that equates success with a strong publication record. By Christine Ro & Jack Leeming

Nota 4N33S2N3 creada en el documento AVTE4IA3.
```

La celda muestra primero la clave y el título del documento, para comprobar en Zotero dónde se creó la nota. En esta ejecución, la nota `4N33S2N3` *se vinculó a* `AVTE4IA3`*, Companies selling authorship slots thrive in a culture that equates success with a strong publication record. La captura siguiente corresponde a la ejecución del 30/10/2026, cuando el primer documento de la lista era* `DABLKWGS`*.*



La ficha aparece debajo del documento, junto a su PDF, con los apartados que el investigador completa después en Zotero.

7.3 Crear una nota independiente en una colección

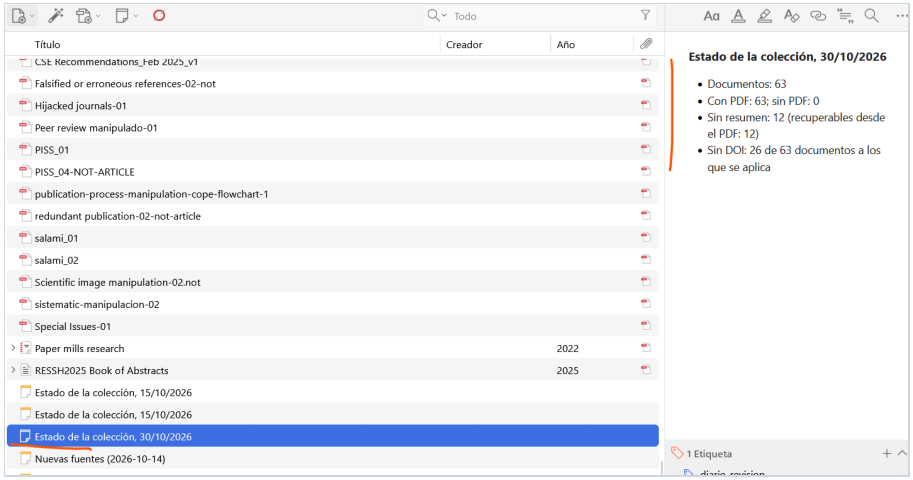
Una nota independiente sirve para registrar información que se refiere a la colección en su conjunto: criterios de inclusión, decisiones de la revisión o un diario de trabajo. El ejemplo crea una entrada de diario con el estado de la colección calculado en las partes 3 a 6, de modo que el diagnóstico queda fechado dentro de Zotero.

```
hoy = datetime.date.today()
diario = (
    f"<h2>Estado de la colección, {hoy:%d/%m/%Y}</h2><ul>"
    f"<li>Documentos: {len(documentos)}</li>"
    f"<li>Con PDF: {len(con_pdf)}; sin PDF: {len(sin_pdf)}</li>"
    f"<li>Sin resumen: {len(sin_resumen)} (recuperables desde el PDF: {len(recuperables)})</li>"
    f"<li>Sin DOI: {len(cobertura['DOI']['sin_valor'])} de {cobertura['DOI']['aplica']} documentos a
los que se aplica</li>"
    "</ul>"
)
clave_diario = crear_nota(diario, coleccion=COLECCION_ID, tags=("diario-revision",))
```

Salida registrada

```
Nota 5DHW5QDU creada en la colección 6APD6L29.
```

La nota independiente `5DHW5QDU` se creó en Bibliometria_Forense con el estado de la colección. La captura siguiente muestra la nota creada en la ejecución del 30/10/2026.



La nota aparece en la lista de la colección como un ítem más, sin documento padre, con la etiqueta `diario-revision`. Las entradas de días anteriores se conservan, de modo que el diario muestra la evolución de la colección: el 15/10/2026 había 13 documentos sin resumen y el 30/10/2026, 12.

7.4 Añadir contenido a una nota existente

Para modificar una nota se lee con `zot.item()`, se cambia su campo `note` y se envía con `zot.update_item()`. Zotero comprueba la versión del ítem: si la nota se modificó en otro lugar después de leerla, la actualización se rechaza en lugar de sobrescribir el cambio. La celda añade un párrafo al diario creado en la sección 7.3. Si no se creó ninguna nota en las secciones 7.2 y 7.3, por ejemplo porque `ESCRIBIR_EN_ZOTERO` vale `False`, usa la primera nota independiente de la colección que listó la sección 7.1. También puede indicarse la clave de cualquier otra nota de esa lista.

```
from pyzotero.zotero_errors import ResourceNotFoundError


def anadir_a_nota(clave_nota, parrafo_html):
    """Añade HTML al final de una nota existente."""
    if not ESCRIBIR_EN_ZOTERO:
        print(f"[simulación] Se añadirían {len(parrafo_html)} caracteres a la nota {clave_nota}.")
        return
    if not clave_nota or clave_nota.startswith("PEGA_AQUI"):
        print("Indica la clave de una nota existente.")
        return
    try:
        nota = zot.item(clave_nota)
    except ResourceNotFoundError:
        print(f"La nota {clave_nota} no se encuentra. Si existe en Zotero, la clave no tiene "
              "'Allow notes access' (sección 7.1).")
        return
    nota["data"]["note"] += parrafo_html
    zot.update_item(nota["data"])
    print(f"Nota {clave_nota} actualizada.")


# la primera nota independiente de la colección listada en la sección 7.1
clave_destino = clave_diario or clave_ficha or (independientes[0]["key"] if independientes else None)
anadir_a_nota(clave_destino, "<p><b>Siguiente paso:</b> revisar los documentos sin resumen y sin DOI.
</p>")
```

Salida registrada

```
Nota 5DHW5QDU actualizada.
```

Con el permiso de notas activado, la nota `5DHW5QDU` de la sección 7.3 se lee y se actualiza sin error. Si la clave no tiene Allow notes access, la función indica la causa en lugar de detenerse con un error `404 Not Found`.

7.5 Incorporar un documento a otra colección

Un documento útil para varias líneas de trabajo no necesita una copia en cada colección: basta con añadir la clave de la colección a su campo `collections`. `anadir_a_coleccion()` conserva las pertenencias anteriores, no modifica el resto del registro y respeta `ESCRIBIR_EN_ZOTERO`. Quitar un documento de una colección y borrarlo de la biblioteca son operaciones distintas, que esta función no realiza.

El ejemplo usa datos que ya aparecieron en salidas anteriores: el documento `A8FBURZN` de la lista de la sección 3.5 y la colección *revision-sistematica* (`UDMXJXJS`) de la lista de la sección 3.3. La celda comprueba con `buscar_coleccion()` (sección 3.4) que la colección existe y que el documento pertenece a la colección de trabajo, y muestra ambos antes de modificar nada. Para incorporar otro documento a otra colección, basta con cambiar las dos claves.

```
def anadir_a_coleccion(item_key, coleccion):
    """Añade un documento a una colección conservando sus colecciones actuales."""
    item = zot.item(item_key)
    datos = item["data"]
    nombre = ruta_coleccion(coleccion) if coleccion in por_clave else coleccion
    if datos.get("itemType") in ("note", "attachment", "annotation"):
        return f"{item_key}: los adjuntos y las notas hijas siguen a su documento; usa la clave del
documento."
    if coleccion in datos.get("collections", []):
        return f"{item_key} ya pertenece a '{nombre}'; no se modifica."
    if not ESCRIBIR_EN_ZOTERO:
        return (f"[simulación] {item_key} se añadiría a '{nombre}' y conservaría sus "
                f"{len(datos.get('collections', []))} colección(es) actuales.")
    datos["collections"] = datos.get("collections", []) + [coleccion]
    zot.update_item(datos)
    return f"{item_key} añadido a '{nombre}'; pertenece ahora a {len(datos['collections'])}
colecciones."


# Datos de ejemplo tomados de salidas anteriores:
#   documento A8FBURZN, "Detecting anomalous citation groups in journal networks" (lista de la sección
3.5)
#   colección revision-sistematica, clave UDMXJXJS (lista de la sección 3.3)
clave_item_destino = "A8FBURZN"
coleccion_destino = buscar_coleccion("revision-sistematica")      # o directamente "UDMXJXJS"

if coleccion_destino is None:
    print("La colección de destino no existe; elige otra de la lista de la sección 3.3.")
elif clave_item_destino not in {it["key"] for it in documentos}:
    print(f"{clave_item_destino} no está en la colección de trabajo; elige otro de la sección 3.5.")
else:
    print(f"Documento: {clave_item_destino} | {campo(zot.item(clave_item_destino), 'title')[:70]}")
    print(f"Destino:   {coleccion_destino} | {ruta_coleccion(coleccion_destino)}")
    print(anadir_a_coleccion(clave_item_destino, coleccion_destino))
```

Salida registrada

```
Documento: A8FBURZN | Detecting anomalous citation groups in journal networks
Destino:   UDMXJXJS | revision-sistematica
A8FBURZN ya pertenece a 'revision-sistematica'; no se modifica.
```

El documento `A8FBURZN` *ya pertenecía a revision-sistematica, porque se incorporó en una ejecución anterior, y la función no lo modifica. Con un documento nuevo, y* `ESCRIBIR_EN_ZOTERO = True` *, el documento aparece en Zotero en las dos colecciones; sigue siendo un único registro, con la misma clave.*

8

Búsquedas en OpenAlex

En este apartado se realizan búsquedas en OpenAlex, primero con una consulta sencilla y después con filtros por año, tipo o citas; también se reconstruyen los resúmenes y se recorren varias páginas de resultados. OpenAlex es la fuente de novedades: allí aparecen los artículos que todavía no están en tu Zotero. Esta parte no depende de Zotero ni del LLM; solo requiere la parte 2 y, preferiblemente, `OPENALEX_API_KEY` (parte 2.4). Al terminar estarán definidas `buscar_openalex()`, `reconstruir_abstract()` y `buscar_openalex_todo()`.

OpenAlex es un índice abierto de publicaciones académicas. El punto de entrada para documentos es `https://api.openalex.org/works`. Se pasa la consulta con `search` y las restricciones con `filter` (varias separadas por comas).

8.1 Búsqueda sencilla, sin filtros

La forma mínima de consultar OpenAlex es una petición `GET` con el parámetro `search`. No hace falta ninguna librería específica: basta `requests`. El parámetro `per_page` limita cuántos resultados devuelve la página.

Las expresiones de búsqueda deben responder al problema de investigación y no solo producir muchos resultados. Comparar formulaciones, examinar candidatos y aplicar criterios documentales ayuda a identificar ambigüedades terminológicas. Ordenar por citas puede destacar trabajos influyentes, pero no demuestra pertinencia y puede relegar publicaciones recientes con pocas citas.

```
respuesta = requests.get(
    "https://api.openalex.org/works",
    params={"search": "paper mills", "per_page": 5},
    timeout=20,
)
datos = respuesta.json()

print("Código HTTP:", respuesta.status_code)
print("Obras que coinciden:", datos["meta"]["count"])
print()
for obra in datos["results"]:
    print(f"- {obra['display_name']} ({obra.get('publication_year')}) | citas:
{obra.get('cited_by_count', 0)}")
```

Salida registrada

```
Código HTTP: 200
Obras que coinciden: 1244700

- Treatment of pulp and paper mill wastewater—a review (2004) | citas: 1118
```

```
- Publication and collaboration anomalies in academic papers originating from a paper mill: Evidence from a
Russia-based paper mill (2023) | citas: 142
- The raw truth about paper mills (2021) | citas: 98
- Paper mills research (2022) | citas: 71
- Management of Pulp and Paper Mill Waste (2014) | citas: 202
```

Sin comillas, `search` busca las palabras por separado. Por eso el primer resultado trata de aguas residuales de la industria papelera (pulp and paper mill), que no tiene relación con las fábricas de artículos. Para buscar la expresión exacta se escribe entre comillas: `"search": '"paper mills"'`. El número de obras cambia a diario.

La respuesta tiene dos partes. `meta` describe la búsqueda (`count`: obras que coinciden; `page` y `per_page`: página actual y tamaño). `results` es la lista de obras. Los campos más útiles de cada obra son:

Campo	Contenido
`id`	Identificador de OpenAlex (`https://openalex.org/W…`)
`display_name`	Título
`publication_year`, `publication_date`	Año y fecha de publicación
`doi`	DOI como URL
`cited_by_count`	Número de citas recibidas
`type`	Tipo de obra (`article`, `preprint`, `review`…)
`authorships`	Autores con sus instituciones
`primary_location`	Revista o repositorio donde se publicó
`abstract_inverted_index`	Resumen en formato de índice invertido (sección 8.4)

8.2 Función de búsqueda con filtros

Los parámetros se envían como diccionario para que `requests` los codifique (espacios, comillas, acentos). Construir la URL a mano con f-strings falla cuando una palabra clave contiene comas.

Errores sin datos sensibles. La URL de cada petición incluye la clave de OpenAlex. Por eso la función no usa `raise_for_status()`, cuyo mensaje de error contiene la URL completa y acabaría en el log o en la salida del notebook. En su lugar lanza un error con un mensaje neutro que solo indica el código HTTP.

```python
def buscar_openalex(texto=None, filtros=None, orden="cited_by_count:desc", por_pagina=5, campos=None):
    params = {"per_page": por_pagina, "sort": orden}
    if texto:
        params["search"] = texto
    if filtros:
        params["filter"] = ",".join(f"{k}:{str(v).replace(',', ' ')}" for k, v in filtros.items())
    if campos:
        params["select"] = ",".join(campos)
    if OPENALEX_API_KEY:
        params["api_key"] = OPENALEX_API_KEY
    if OPENALEX_MAILTO:
        params["mailto"] = OPENALEX_MAILTO
```

```
    try:
        r = requests.get(OPENALEX_URL, params=params, timeout=20)
    except requests.RequestException as e:
        # El mensaje original incluye la URL con la clave: se sustituye por uno neutro
        raise requests.RequestException(f"OpenAlex no responde ({type(e).__name__})") from None
    if r.status_code != 200:
        raise requests.HTTPError(f"OpenAlex respondió con error {r.status_code}", response=r)
    return r.json().get("results", [])
```

Ejemplo de uso. Busca publicaciones del año en curso sobre una consulta libre y muestra título, fecha, citas y DOI de las cinco más citadas.

```
anio = datetime.date.today().year
consulta = "large language models in medical education"

print(f"= Buscando en OpenAlex sobre: '{consulta}'...")
resultados = buscar_openalex(consulta, filtros={"publication_year": anio})

for paper in resultados:
    print(f"\n = {paper.get('display_name', 'Sin título')}")
    print(f"   Fecha: {paper.get('publication_date', 'N/A')} | Citas: {paper.get('cited_by_count',
0)}")
    print(f"   DOI/Link: {paper.get('doi') or paper['id']}")
```

Salida registrada

```
= Buscando en OpenAlex sobre: 'large language models in medical education'...

 = A survey on LLM-as-a-judge
   Fecha: 2026-01-10 | Citas: 684
   DOI/Link: https://doi.org/10.1016/j.xinn.2025.101253

 = An open vision-language model for diverse medical applications
   Fecha: 2026-10-01 | Citas: 333
   DOI/Link: https://doi.org/10.1038/s41591-026-04626-w

 = Prompt Engineering for ChatGPT: A Quick Guide to Techniques, Tips, and Best Practices
   Fecha: 2026-04-24 | Citas: 317
   DOI/Link: https://doi.org/10.18260/jet.43.1.5

 = Parameter-Efficient Fine-Tuning Methods for Pretrained Language Models: A Critical Review and Assessment
   Fecha: 2026-01-27 | Citas: 317
   DOI/Link: https://doi.org/10.1109/tpami.2026.3657354

 = Mean-Field Limit of Non-exchangeable Multi-agent Systems Over Hypergraphs with Unbounded Rank
   Fecha: 2026-10-23 | Citas: 274
   DOI/Link: https://doi.org/10.1007/s00332-026-10338-9
```

El primer resultado no es un artículo sino el registro de una revista, y la guía de sepsis no trata de educación médica. Ordenar solo por citas premia lo muy citado aunque sea poco pertinente. Dos ajustes lo corrigen: `filtros={"type": "article"}` *y dejar el orden por relevancia (* `orden="relevance_score:desc"` *).*

8.3 Filtros útiles

Filtro	Ejemplo	Significado
`publication_year`	`2026`, `>2024`, `2023-2026`	Año o rango
`from_publication_date`	`2026-01-01`	Desde una fecha
`type`	`article`, `preprint`, `review`	Tipo de obra
`title.search`	`paper mills`	La frase aparece en el título
`cited_by_count`	`>10`	Mínimo de citas
`open_access.is_oa`	`true`	Solo acceso abierto
`language`	`es`	Idioma

`search` acepta operadores en mayúsculas: `"paper mills" AND "authorship for sale"`, `OR`, `NOT`.

8.4 Reconstruir el resumen

OpenAlex no entrega el resumen como texto, sino como *índice invertido*: cada palabra con las posiciones en que aparece. Hay que reordenarlo.

Los filtros operacionalizan decisiones de alcance temporal, idioma y tipo de publicación. Deben justificarse porque alteran qué documentos pueden recuperarse. La reconstrucción del resumen transforma un formato de intercambio en texto utilizable; si el índice está ausente, debe registrarse esa limitación en lugar de asumir que el trabajo carece de contenido relevante.

```python
def reconstruir_abstract(indice_invertido):
    if not indice_invertido:
        return ""
    posiciones = {pos: palabra for palabra, lista in indice_invertido.items() for pos in lista}
    return " ".join(posiciones[i] for i in sorted(posiciones))


# Formato tal como llega en paper["abstract_inverted_index"]:
ejemplo = {"Paper": [0], "mills": [1], "sell": [2], "authorship": [3], "and": [4, 7],
           "fabricated": [5], "manuscripts": [6], "distort": [8], "citation": [9], "networks.": [10]}
print(reconstruir_abstract(ejemplo))
```

Salida registrada

```
Paper mills sell authorship and fabricated manuscripts and distort citation networks.
```

8.5 Paginación por número de página

Cada petición devuelve una sola página de resultados. Con los parámetros `page` (número de página, empieza en 1) y `per_page` (de 1 a 200 resultados por página) se avanza por el listado. Es la forma más sencilla de paginar, pero solo permite llegar a los primeros 10 000 resultados.

Revisar solo la primera página puede producir una selección sesgada hacia el orden configurado. La paginación permite ampliar el conjunto recuperado y distinguir el total informado por el servicio de los registros efectivamente examinados. **Nota de compatibilidad:** la documentación de OpenAlex consultada el 14/10/2026 indica un máximo de 100 para `per_page`; el código original conserva valores de hasta 200. Se mantiene intacto por petición del autor, pero este punto debe verificarse al ejecutar esas celdas con la API vigente.

```
consulta = "citation cartels"

params = {"search": consulta, "per_page": 5}
if OPENALEX_API_KEY:
    params["api_key"] = OPENALEX_API_KEY      # sin clave, OpenAlex puede rechazar la petición
if OPENALEX_MAILTO:
    params["mailto"] = OPENALEX_MAILTO

for pagina in (1, 2, 3):
    params["page"] = pagina
    try:
        respuesta = requests.get(OPENALEX_URL, params=params, timeout=20)
    except requests.RequestException as e:
        print(f"Página {pagina}: OpenAlex no responde ({type(e).__name__})")
        break
    if respuesta.status_code != 200:
        # Se muestra el código y el mensaje del servidor, nunca la URL (contiene la clave)
        print(f"Página {pagina}: OpenAlex respondió con error {respuesta.status_code}")
        print("   ", respuesta.text[:200])
        break

    datos = respuesta.json()
    meta = datos["meta"]
    if pagina == 1:
        print(f"'{consulta}': {meta['count']} obras, {meta['per_page']} por página")
    print(f"\nPágina {meta['page']}")
    primero = (pagina - 1) * meta["per_page"] + 1
    for i, obra in enumerate(datos["results"], start=primero):
        print(f"  {i:>2}. {obra['display_name'][:80]}")
```

Salida registrada

```
'citation cartels': 9291 obras, 5 por página

Página 1
   1. Toward the Discovery of Citation Cartels in Citation Networks
   2. Citation Cartels in Medical and Dental Journals
   3. Citation Cartels in Medical and Dental Journals
   4. Detecting Citation Cartels: A Preliminary Review
   5. (Wl-625)-Citation Cartels in Academic Publishing

Página 2
   6. Citation Cartels: Understanding Their Emergence and Impact on the Academic World
   7. The Network of Law Reviews: Citation Cartels, Scientific Communities, and Journa
   8. Citation Cartels: A Hidden Menace in the Scientific Publication
   9. Finding Citation Cartels in Academic Research
  10. A Simple Model of Citation Cartels: When Self-interest Strikes Science

Página 3
  11. Trust-Aware Citation Cartel Ranking in Scholarly Knowledge Graphs
  12. Citation cartels help some mathematicians—and their universities—climb the ranki
  13. Citation Cartels, Dismissive Literature Reviews, and Big Money Dominate US Educa
```

```
14. A model of citation cartels: When self-interest strikes science
15. The relevance of Pierre Bourdieu's cold realities of social inequality in citati
```

8.6 Más resultados con cursor

Para recuperar más de 10 000 registros, o para descargar listados largos de forma eficiente, se usa paginación por cursor: se empieza con `cursor="*"` y cada respuesta trae el siguiente en `meta.next_cursor`.

```python
def buscar_openalex_todo(texto, filtros=None, maximo=200):
    """Recorre las páginas de resultados con cursor hasta reunir `maximo` publicaciones."""
    params = {"search": texto, "per_page": min(200, maximo), "cursor": "*"}
    if filtros:
        params["filter"] = ",".join(f"{k}:{v}" for k, v in filtros.items())
    if OPENALEX_API_KEY:
        params["api_key"] = OPENALEX_API_KEY      # sin clave, OpenAlex puede rechazar la petición
    if OPENALEX_MAILTO:
        params["mailto"] = OPENALEX_MAILTO

    todos, pagina = [], 1
    while params["cursor"] and len(todos) < maximo:
        try:
            respuesta = requests.get(OPENALEX_URL, params=params, timeout=30)
        except requests.RequestException as e:
            print(f"Página {pagina}: OpenAlex no responde ({type(e).__name__})")
            break
        if respuesta.status_code != 200:
            # Código y mensaje del servidor, nunca la URL (contiene la clave)
            print(f"Página {pagina}: OpenAlex respondió con error {respuesta.status_code}")
            print("   ", respuesta.text[:200])
            break

        datos = respuesta.json()
        if pagina == 1:
            print(f"OpenAlex encuentra {datos['meta']['count']} publicaciones para {texto}")
        resultados = datos.get("results", [])
        if not resultados:
            break
        todos.extend(resultados)
        params["cursor"] = datos["meta"].get("next_cursor")
        pagina += 1

    return todos[:maximo]
```

```python
# Recuperar hasta 150 publicaciones relacionadas con bibliometría forense
publicaciones = buscar_openalex_todo(
    texto='"citation mills"',
    filtros={
        "type": "article",
        "publication_year": "2020-2026"
    },
    maximo=150
)

print(f"Publicaciones recuperadas: {len(publicaciones)}")
```

```python
# Mostrar las primeras diez
for numero, publicacion in enumerate(publicaciones[:10], start=1):
    print(f"\n{numero}. {publicacion.get('display_name', '(sin título)')}")
    print(f"   Año: {publicacion.get('publication_year', 's.f.')}")
    print(f"   DOI: {publicacion.get('doi') or 'No disponible'}")
    print(f"   OpenAlex: {publicacion.get('id', '')}")
```

Salida registrada

```
OpenAlex encuentra 22 publicaciones para "citation mills"
Publicaciones recuperadas: 22

1. Citation manipulation through citation mills and pre-print servers
   Año: 2025
   DOI: https://doi.org/10.1038/s41598-025-88709-7
   OpenAlex: https://openalex.org/W4407567707

2. Basal stem area is a better measure of woodiness than canopy cover in the savannas of the Kruger National
Park
   Año: 2020
   DOI: https://doi.org/10.17159/sajs.2020/7409
   OpenAlex: https://openalex.org/W3003614507

3. Revues prédatrices et paper mills mettent en péril la gestion des savoirs
   Año: 2024
   DOI: https://doi.org/10.1016/j.bulcan.2024.12.002
   OpenAlex: https://openalex.org/W4405503911

4. Ex situ cycad conservation and successes with an extinct in the wild cycad
   Año: 2025
   DOI: No disponible
   OpenAlex: https://openalex.org/W7220206464

5. Stop Reducing Us into Simplistic Numbers: An Academic Outcry!
   Año: 2026
   DOI: https://doi.org/10.25259/stn_13_2026
   OpenAlex: https://openalex.org/W7147302957

6. The Challenges of Scientific Publishing: An Editor-In-Chief's Perspective
   Año: 2026
   DOI: https://doi.org/10.1111/fcp.70105
   OpenAlex: https://openalex.org/W7167513146

7. Altmetric Activity and Scientific Integrity: A Retraction Analysis of PubMed-Indexed Social Media-Related
Publications
   Año: 2026
   DOI: https://doi.org/10.3346/jkms.2026.41.e84
   OpenAlex: https://openalex.org/W7130339902

8. Wissenschaftsbetrug und zweifelhafte Publikationspraktiken
   Año: 2025
   DOI: https://doi.org/10.1007/s00063-025-01307-3
   OpenAlex: https://openalex.org/W4413040610

9. Classification of constraints of Hirsch index as a bibliometrical indicator
   Año: 2026
   DOI: https://doi.org/10.33186/1027-3689-2026-5-65-92
   OpenAlex: https://openalex.org/W7164495213

10. The Adolescent Brain Is Literally Awesome
   Año: 2020
   DOI: https://doi.org/10.3389/frym.2020.00075
   OpenAlex: https://openalex.org/W3037956760
```

El mismo recorrido con el cursor a la vista. La celda siguiente repite la búsqueda sin función auxiliar. Muestra cómo cada respuesta entrega `meta.next_cursor`, que se envía en la petición siguiente, y cuántas publicaciones llega en cada página. El bucle termina al alcanzar el máximo o cuando OpenAlex ya no devuelve cursor.

```
params = {
    "search": '"citation manipulation"',
    "filter": "type:article,publication_year:2023-2026",
    "per_page": 100,
    "cursor": "*"
}

if OPENALEX_API_KEY:
    params["api_key"] = OPENALEX_API_KEY

publicaciones = []
maximo = 150
pagina = 1

while params["cursor"] and len(publicaciones) < maximo:
    respuesta = requests.get(OPENALEX_URL, params=params, timeout=30)
    respuesta.raise_for_status()
    datos = respuesta.json()

    resultados = datos.get("results", [])
    print(f"Página {pagina}: {len(resultados)} publicaciones")

    publicaciones.extend(resultados)
    params["cursor"] = datos.get("meta", {}).get("next_cursor")
    pagina += 1

    if not resultados:
        break

publicaciones = publicaciones[:maximo]

print(f"\nTotal recuperado: {len(publicaciones)}")

for numero, publicacion in enumerate(publicaciones[:10], start=1):
    print(f"\n{numero}. {publicacion.get('display_name', '(sin título)')}")
    print(f"   Año: {publicacion.get('publication_year', 's.f.')}")
    print(f"   DOI: {publicacion.get('doi') or 'No disponible'}")
    print(f"   OpenAlex: {publicacion.get('id', '')}")
```

Salida registrada

```
Página 1: 100 publicaciones
Página 2: 100 publicaciones

Total recuperado: 150

1. Citation manipulation through citation mills and pre-print servers
   Año: 2025
   DOI: https://doi.org/10.1038/s41598-025-88709-7
   OpenAlex: https://openalex.org/W4407567707

2. Citation manipulation found to be rife in math
   Año: 2024
   DOI: https://doi.org/10.1126/science.ado3859
   OpenAlex: https://openalex.org/W4391428500
```

```
3. A paper mill detection model based on citation manipulation paradigm
   Año: 2025
   DOI: https://doi.org/10.2478/jdis-2025-0003
   OpenAlex: https://openalex.org/W4406101981

4. Knowledge Manifold Learning: Unveiling the Topological Laws of Scientific Revolutions and Filtering
Citation Manipulation
   Año: 2026
   DOI: https://doi.org/10.13140/rg.2.2.36594.36802
   OpenAlex: https://openalex.org/W7139992917

5. Detecting Pseudo-Manipulated Citations in Scientific Literature through Perturbations of the Citation
Graph
   Año: 2023
   DOI: https://doi.org/10.3390/math11183820
   OpenAlex: https://openalex.org/W4386485463

6. Retracted: Impact of water–tourism–ecosystem nexus on the sustainable development of eco-city
   Año: 2023
   DOI: https://doi.org/10.2166/ws.2023.123
   OpenAlex: https://openalex.org/W4376954720

7. Retracted: Assessment of customer satisfaction with the water supply system in low-income countries: A
case of Adama City, Ethiopia
   Año: 2024
   DOI: https://doi.org/10.2166/ws.2024.006
   OpenAlex: https://openalex.org/W4390976041

8. Retracted: Visual design of water-based landscape environment based on the sustainable utilization of
water resources
   Año: 2023
   DOI: https://doi.org/10.2166/ws.2023.331
   OpenAlex: https://openalex.org/W4389935811

9. Retracted: Evaluation of agricultural wetland utilization and landscape design
   Año: 2023
   DOI: https://doi.org/10.2166/ws.2023.234
   OpenAlex: https://openalex.org/W4387089844

10. Retracted: Developing policy-making for maximizing the water productivity in agricultural lands
   Año: 2023
   DOI: https://doi.org/10.2166/ws.2023.091
   OpenAlex: https://openalex.org/W4362664004
```

9

De metadatos a contexto para el LLM

En este apartado los ítems de Zotero se normalizan como registros bibliográficos estructurados y se concatenan en un único texto de contexto. El modelo de lenguaje solo recibe texto, y unos registros ordenados y sin ruido producen respuestas más precisas con menos tokens. Se apoya en las partes 3 y 4 y produce la lista `mis_datos` y el texto `contexto_investigacion`.

Un LLM solo puede razonar sobre lo que recibe. Esta etapa convierte los ítems de Zotero en fichas limpias: título, autor, etiquetas y resumen. Se descartan adjuntos y notas, y (opcionalmente) los documentos sin resumen; los resúmenes completados en la parte 6 reducen esas exclusiones.

9.1 Normalización de metadatos en registros bibliográficos

La preparación del contexto es una decisión analítica: selecciona qué información de la biblioteca podrá utilizar el modelo. Excluir documentos sin resumen puede mejorar la cantidad de texto disponible por registro y, a la vez, sesgar el corpus hacia fuentes con metadatos más completos. La normalización de esta sección uniforma la representación, pero no certifica la exactitud de los datos.

Las etiquetas se envían al modelo como temas del documento. Por eso se excluyen las que crea el propio manual para registrar el trabajo (`revision:sin-doi`, `resumen-desde-pdf`, `resumen-generado-llm`, `ficha-lectura`, `diario-revision` y `agente-ia`, de las partes 6, 7, 10 y 13): describen el estado de la revisión, no el contenido, y el modelo podría tomarlas como temas de investigación. `etiquetas_tematicas()` aplica ese filtro con la lista `PREFIJOS_TRABAJO`, que puede ampliarse con las etiquetas de trabajo propias del investigador. El agente de la parte 14 usa la misma función, de modo que también aplica la exclusión.

```
TIPOS_DOCUMENTO = ("journalArticle", "preprint", "book", "bookSection", "conferencePaper", "report",
"thesis")


# Etiquetas de trabajo creadas por el manual: describen la revisión, no el tema del documento
PREFIJOS_TRABAJO = ("revision:", "resumen-desde-pdf", "resumen-generado-llm",
                    "ficha-lectura", "diario-revision", "agente-ia")


def etiquetas_tematicas(item):
    """Etiquetas del documento sin las etiquetas de trabajo."""
    return [t for t in etiquetas(item) if not t.lower().startswith(PREFIJOS_TRABAJO)]


def normalizar_registros(items, exigir_resumen=True, verbose=True):
```

```
    fichas = []
    for it in items:
        if campo(it, "itemType") not in TIPOS_DOCUMENTO:
            continue
        ficha = {
            "key": it["key"],
            "titulo": campo(it, "title", "Sin título"),
            "autor": primer_autor(it),
            "anio": campo(it, "date", "")[:4],
            "doi": campo(it, "DOI"),
            "resumen": campo(it, "abstractNote").strip(),
            "tags": etiquetas_tematicas(it),
        }
        if exigir_resumen and not ficha["resumen"]:
            if verbose:
                print(f"- Saltando '{ficha['titulo'][:30]}...' por falta de abstract.")
            continue
        fichas.append(ficha)
    return fichas


mis_datos = normalizar_registros(documentos)      # documentos de la colección de trabajo (sección 3.5)

print(f"\n * Se han preparado {len(mis_datos)} documentos para el análisis.")
for p in mis_datos[:3]:
    print(f"\n ° {p['autor']} - {p['titulo']}")
    print(f"   Abstract (fragmento): {p['resumen'][:100]}...")
```

Salida registrada

```
- Saltando 'Companies selling authorship s...' por falta de abstract.
- Saltando 'Some top academics cite themse...' por falta de abstract.
- Saltando 'Detecting anomalous citation g...' por falta de abstract.
- Saltando 'Systematic Manipulation of the...' por falta de abstract.
- Saltando 'RESSH2025 Book of Abstracts...' por falta de abstract.
- Saltando 'Paper mills research...' por falta de abstract.
- Saltando 'Informe bibliométrico sobre la...' por falta de abstract.
- Saltando 'Beyond misconduct: forging an ...' por falta de abstract.
- Saltando 'Elsevier probes dodgy citation...' por falta de abstract.
- Saltando 'Citation manipulation through ...' por falta de abstract.
- Saltando 'Predator Effect: Understanding...' por falta de abstract.

 * Se han preparado 52 documentos para el análisis.

 ° Jones - Bibliometric evaluation of Forensic Science International as a scholarly journal within the subject category legal medicine
   Abstract (fragmento): This article presents a bibliometric evaluation of Forensic Science International (FSI) as a scholar...

 ° Logan - Reference accuracy of primary studies about reference accuracy: A descriptive study
   Abstract (fragmento): Reference accuracy is defined as the completeness and correctness of citation components such as aut...

 ° Besançon - Sneaked references: Fabricated reference metadata distort citation counts
   Abstract (fragmento): We report evidence of an undocumented method to manipulate citation counts involving “sneaked” refer...
```

De los 63 documentos de la colección, 52 tienen resumen y se convierten en fichas; los 11 restantes se descartan. Las fichas proceden de la colección de trabajo (`documentos`, sección 3.5). Una versión anterior usaba `zot.top(limit=20)`, es decir, los últimos ítems de toda la biblioteca; cuando el agente creó notas nuevas, esos ítems eran solo notas y el contexto quedaba vacío. Revisa los metadatos

antes de confiar en el LLM: un resumen erróneo en Zotero llega al modelo tal cual.

9.2 Construcción del contexto para el modelo

Las fichas se unen en un único texto. `max_resumen` limita la longitud de cada resumen para controlar el número de tokens enviados.

Limitar los resúmenes controla la longitud de entrada, pero puede eliminar información necesaria para distinguir temas próximos. El límite está expresado en caracteres, que no equivalen a tokens. También debe comprobarse que el contexto total cabe en la ventana del modelo; esta función no calcula por sí misma ese presupuesto ni implementa selección semántica de fragmentos.

```
def construir_contexto(fichas, max_resumen=600):
    bloques = ["SISTEMA DE ARCHIVOS DE INVESTIGACIÓN:", "=" * 36]
    for i, p in enumerate(fichas, 1):
        bloques.append(
            f"DOCUMENTO {i}:\n"
            f"- TÍTULO: {p['titulo']}\n"
            f"- AUTOR: {p['autor']}\n"
            f"- TEMAS: {', '.join(p['tags']) or 'sin etiquetas temáticas'}\n"
            f"- RESUMEN: {p['resumen'][:max_resumen]}\n"
            + "-" * 36
        )
    return "\n".join(bloques)


contexto_investigacion = construir_contexto(mis_datos)
print(contexto_investigacion[:1500])
```

Salida registrada

```
SISTEMA DE ARCHIVOS DE INVESTIGACIÓN:
====================================
DOCUMENTO 1:
- TÍTULO: Bibliometric evaluation of Forensic Science International as a scholarly journal within the
subject category legal medicine
- AUTOR: Jones
- TEMAS: sin etiquetas temáticas
- RESUMEN: This article presents a bibliometric evaluation of Forensic Science International (FSI) as a
scholarly journal within the “legal medicine” subject category. Citation data were retrieved from Science
Citation Index (SCI) and Journal Citation Reports (JCR), both of which are part of the Web-of-Science (WOS)
database. The most cited articles in FSI were identified along with the most prolific authors. The current
journal impact factor (JIF) of FSI is 2.2, which was in good agreement with the 5-year JIF of 2.3. FSI was
ranked fourth among 17 journals within the legal medicine subject category. Sin
------------------------------------
DOCUMENTO 2:
- TÍTULO: Reference accuracy of primary studies about reference accuracy: A descriptive study
- AUTOR: Logan
- TEMAS: sin etiquetas temáticas
- RESUMEN: Reference accuracy is defined as the completeness and correctness of citation components such as
author names, article title, journal name, volume designation, and page numbers. A representation of
reference accuracy is often expressed as an error rate which is the percentage of citations with at least
one error within a reference list. The purpose of this study was to describe the reference accuracy of peer-
rev
```

Ejecuta la celda para ver el contexto. En la línea `TEMAS` solo aparecen etiquetas temáticas; un documento cuyas únicas etiquetas sean de trabajo muestra `sin etiquetas temáticas`.

10

Consultas a un LLM con OpenRouter

En este apartado se envía el contexto a un modelo de lenguaje mediante OpenRouter para obtener un perfil temático de la colección, y se pide una respuesta en formato JSON que el código pueda procesar. El modelo resume de qué trata la colección y propone términos de búsqueda. Requiere `OPENROUTER_API_KEY` con créditos (parte 2.3), o un modelo local de Ollama (sección 2.9), y el contexto de la parte 9. Quedan definidas `preguntar_llm()` y `extraer_json()`.

OpenRouter expone muchos modelos con la misma interfaz que la librería `openai`; cambiar de modelo es cambiar una cadena de texto. Los dos que se usaron en las versiones del agente:

Identificador en OpenRouter	Uso en este tutorial
`deepseek/deepseek-chat-v3.1`	Modelo por defecto (`MODELO_LLM`)
`mistralai/mistral-small-3.2-24b-instruct`	Alternativa rápida para clasificaciones cortas

10.1 Una función para todas las llamadas

Los mensajes tienen dos papeles: `system` define el rol y las reglas; `user` lleva los datos. Una `temperature` baja da respuestas más estables, útil cuando luego hay que analizar el texto con código.

OpenRouter permite acceder al modelo, la función define cómo se formula la consulta. Para interpretar una respuesta deben conocerse ambos elementos, junto con el contexto enviado. Los identificadores de la tabla documentan los modelos del ejemplo original y deben comprobarse antes de utilizarlos. Una temperatura baja reduce la variación de muestreo, pero no garantiza respuestas idénticas ni exactitud científica.

La misma función sirve para OpenRouter y para un modelo local (sección 2.9): usa el cliente `llm` y el modelo `MODELO_LLM` que fijó la configuración. Dos ajustes la hacen compatible con los modelos que razonan antes de responder, como `qwen3` o `gpt-oss`. Con un modelo local, eleva el límite de tokens a 1024 como mínimo, porque ese razonamiento también consume tokens. Además, elimina de la respuesta el bloque `<think>…</think>` cuando el modelo lo incluye en el texto.

Con `PROVEEDOR_LLM = "local"` (sección 2.9), todas las celdas de las partes 10 a 14 usan el modelo de Ollama en lugar de OpenRouter, sin cambios en el código. Las salidas registradas de estas partes se obtuvieron con `deepseek/deepseek-chat-v3.1` en OpenRouter; un modelo local producirá respuestas distintas.

```
def preguntar_llm(prompt, sistema=None, modelo=None, temperatura=0.2, max_tokens=1500):
    mensajes = []
    if sistema:
```

```
        mensajes.append({"role": "system", "content": sistema})
    mensajes.append({"role": "user", "content": prompt})
    if PROVEEDOR_LLM == "local":
        # Los modelos que razonan antes de responder (qwen3, gpt-oss) consumen tokens en ese
        # razonamiento; con un límite bajo, la respuesta llegaría vacía.
        max_tokens = max(max_tokens, 1024)
    respuesta = llm.chat.completions.create(
        model=modelo or MODELO_LLM,
        messages=mensajes,
        temperature=temperatura,
        max_tokens=max_tokens,
    )
    texto = respuesta.choices[0].message.content or ""
    return re.sub(r"<think>.*?</think>", "", texto, flags=re.S).strip()    # quita el razonamiento
visible
```

Cambiar entre OpenRouter y el modelo local. `usar_modelo()` cambia el cliente `llm` y el modelo activo sin volver a ejecutar la configuración: `usar_modelo("local")` pasa al modelo de Ollama indicado en `MODELO_LOCAL`, `usar_modelo("local", "qwen2.5:3b-instruct")` a cualquier otro modelo descargado, y `usar_modelo("openrouter")` vuelve al modelo de OpenRouter. Todas las celdas que se ejecuten después (perfil temático, resumen asistido, filtro de pertinencia, taxonomía y agente) usan el modelo elegido.

`comparar_modelos()` envía la misma pregunta a los dos proveedores y muestra cada respuesta con su tiempo, de modo que puede valorarse si el modelo local basta para una tarea antes de usarlo. Al terminar, restaura el modelo que estaba activo. Ollama debe estar en marcha y el modelo local, descargado (sección 2.9).

```
def usar_modelo(proveedor, modelo=None):
    """Cambia el modelo activo. proveedor: "openrouter" (en la nube) o "local" (Ollama)."""
    global llm, MODELO_LLM, PROVEEDOR_LLM
    if proveedor == "local":
        llm = OpenAI(base_url=OLLAMA_BASE_URL, api_key="ollama")
        MODELO_LLM = modelo or MODELO_LOCAL
    elif proveedor == "openrouter":
        llm = OpenAI(base_url=OPENROUTER_BASE_URL, api_key=OPENROUTER_API_KEY)
        MODELO_LLM = modelo or globals().get("MODELO_OPENROUTER", "deepseek/deepseek-chat-v3.1")
    else:
        raise ValueError('El proveedor debe ser "openrouter" o "local".')
    PROVEEDOR_LLM = proveedor
    print(f"Modelo activo: {MODELO_LLM} ({'Ollama, en tu ordenador' if proveedor == 'local' else
'OpenRouter'})")


def comparar_modelos(pregunta, modelo_local=None):
    """Envía la misma pregunta a OpenRouter y al modelo local, y restaura después el modelo activo."""
    activo = (PROVEEDOR_LLM, MODELO_LLM)
    for proveedor, modelo in (("openrouter", None), ("local", modelo_local)):
        usar_modelo(proveedor, modelo)
        inicio = time.time()
        try:
            texto = preguntar_llm(pregunta, temperatura=0.2, max_tokens=300)
        except Exception as e:
            texto = f"(sin respuesta: {type(e).__name__}; comprueba la sección 2.8 o 2.9)"
```

```
        print(f"   {time.time() - inicio:.1f} s | {texto[:600]}\n")
    usar_modelo(*activo)
```

```
comparar_modelos("En una frase, ¿qué es la bibliometría forense?")

# Para usar un modelo en las celdas siguientes, descomenta una línea:
# usar_modelo("local")                          # MODELO_LOCAL (sección 2.5)
# usar_modelo("local", "qwen2.5:3b-instruct")   # otro modelo de la lista de la sección 2.9
# usar_modelo("openrouter")                     # vuelve a OpenRouter
```

Salida registrada

```
Modelo activo: deepseek/deepseek-chat-v3.1 (OpenRouter)
   1.9 s | La bibliometría forense es la aplicación de métodos estadísticos y cuantitativos para detectar e
investigar fraudes académicos, como la manipulación de citas o la fabricación de artículos.

Modelo activo: qwen3.6 (Ollama, en tu ordenador)
   54.5 s | La bibliometría forense es la aplicación de métodos cuantitativos y análisis estadísticos de
datos bibliométricos para investigar fraude académico, plagio, manipulación de citas y disputas de autoría
con fines investigativos o legales.

Modelo activo: qwen3.6 (Ollama, en tu ordenador)
```

Ejecuta la celda con Ollama en marcha. Para cada proveedor se muestran el modelo, el tiempo de respuesta y la respuesta. La primera consulta al modelo local tarda más, porque Ollama lo carga en memoria. Al final se restaura el modelo que estaba activo; para cambiarlo, descomenta una de las líneas de `usar_modelo()`.

Nota: si bien el modelo local elimina el tema del pago a proveedor externo para el uso de modelos LLM, es mucho más lento, en esta pequeña prueba el modelo en la nube realizó la consulta en **1.9s**, mientras que el local en **54.5s**, pero adicional consume recurso de tu memoria que en algunos casos es necesario una buena tarjeta gráfica para lo que vayas a realizar.

10.2 Perfil temático de la biblioteca

El perfil temático sintetiza el contenido de los registros seleccionados y puede orientar la expansión de términos de búsqueda. Su utilidad depende de que represente el corpus sin sustituirlo por asociaciones generales del modelo. La revisión debe comparar la propuesta con los títulos y resúmenes, detectar temas sobrerrepresentados y comprobar si se omiten dimensiones centrales del problema.

```
SISTEMA_ANALISTA = (
    "Eres un analista bibliográfico. Analiza los resúmenes de una biblioteca personal y "
    "define qué está investigando el usuario. Responde con un resumen de 3 frases sobre la "
    "temática y una lista de 5 términos de búsqueda específicos en inglés."
)

print("== El modelo está analizando tu biblioteca...")
perfil_ia = preguntar_llm(contexto_investigacion, sistema=SISTEMA_ANALISTA)

print("\n--- PERFIL GENERADO POR LA IA ---")
print(perfil_ia)
```

Salida registrada

```
== El modelo está analizando tu biblioteca...

--- PERFIL GENERADO POR LA IA ---
```

```
El usuario está investigando la integridad científica y las prácticas cuestionables en publicaciones
académicas, con enfoque en la detección de fraudes a gran escala como paper mills, manipulación de métricas
de citación y distorsiones en los sistemas de evaluación. La investigación abarca metodologías de forensic
scientometrics para identificar patrones anómalos, contaminación de la literatura científica y el impacto de
la inteligencia artificial en la manipulación académica. También explora cómo estas prácticas afectan la
credibilidad de la comunicación científica y las posibles soluciones para preservar la confianza en la
ciencia.

**Términos de búsqueda:**
1. Paper mill detection
2. Citation manipulation
3. Forensic scientometrics
4. Research integrity violations
5. Scholarly publishing fraud
```

Con las 52 fichas de Bibliometria_Forense, el perfil describe la investigación sobre fábricas de artículos, manipulación de citas y cienciometría forense, y propone cinco términos de búsqueda en inglés. El perfil depende por completo de los documentos que recibe el modelo.

10.3 Pedir una respuesta estructurada (JSON)

Para usar la respuesta en código conviene pedir JSON. Los modelos a veces añaden texto o bloques de código alrededor, así que se extrae el primer objeto `{...}` antes de convertirlo.

Una salida JSON facilita conectar la respuesta con el programa, pero su validez sintáctica no demuestra que las categorías sean correctas ni que los campos cumplan el significado esperado. La extracción actual ilustra una solución inicial. Una ampliación metodológica podría validar un esquema, tipos de datos y categorías permitidas, conservando por separado la respuesta original.

```python
def extraer_json(texto):
    """Extrae y convierte el primer objeto JSON de una respuesta del LLM."""
    m = re.search(r"\{.*\}", texto, re.DOTALL)
    if not m:
        return None
    try:
        return json.loads(m.group(0))
    except json.JSONDecodeError:
        return None


# Respuesta típica de un modelo (texto de ejemplo):
respuesta = """Claro, aquí tienes el resultado:
```json
{"tema": "integridad científica y bibliometría forense",
 "terminos": ["citation cartels", "paper mills", "sneaked references"]}
```"""

datos = extraer_json(respuesta)
print(datos["tema"])
print(datos["terminos"])

Con el LLM real:
datos = extraer_json(preguntar_llm(
```

```
#     f"Devuelve SOLO un JSON con las claves 'tema' y 'terminos' (lista de
5).\n\n{contexto_investigacion}"))
```

Salida registrada

```
integridad científica y bibliometría forense
['citation cartels', 'paper mills', 'sneaked references']
```

10.4 Resumen asistido para documentos sin resumen en el PDF

Cuando el PDF no contiene una sección de resumen (un capítulo, un informe o un documento escaneado y procesado con OCR), `resumen_desde_pdf()` devuelve una cadena vacía. En esos casos un LLM puede redactar un resumen a partir del texto completo. A diferencia del resumen extraído en la sección 6.6, este texto es **generado** y no procede literalmente del documento, por lo que no se guarda en `abstractNote`: se registra como nota hija con la etiqueta `resumen-generado-llm` y la indicación del modelo utilizado.

El resumen generado sirve para orientar la lectura o el cribado, no para citarse. El prompt pide al modelo que no añada información ausente del texto, pero esa instrucción no garantiza que la respuesta sea fiel; conviene contrastarla con el documento.

```
if "preguntar_llm" not in globals():
    raise NameError("Ejecuta antes la sección 10.1, que define preguntar_llm().")


def resumen_con_llm(texto, max_caracteres=12_000):
    """Pide al LLM un resumen de un texto académico (objetivo, método y resultados)."""
    prompt = ("Resume en español, en un único párrafo de 150 a 200 palabras, el objetivo, el método "
              "y los resultados principales del siguiente texto académico. Usa solo información "
              "presente en el texto.\n\nTEXTO:\n" + texto[:max_caracteres])
    return preguntar_llm(prompt, temperatura=0.1, max_tokens=500)


# Documentos sin resumen cuyo PDF tampoco tiene una sección de resumen reconocible
textos = {d["key"]: texto_completo(d["key"], max_caracteres=30_000) for d in recuperables}
sin_seccion = [d for d in recuperables if not resumen_desde_pdf(textos[d["key"]])]
print(f"\n{len(sin_seccion)} de {len(recuperables)} documentos no tienen una sección de resumen en el
PDF.")

if sin_seccion:
    doc = sin_seccion[0]
    generado = resumen_con_llm(textos[doc["key"]])
    print(f"\n{doc['key']} | {campo(doc, 'title')[:80]}\n\n{generado}\n")
    crear_nota(f"<h2>Resumen generado por LLM</h2><p>{html.escape(generado)}</p>"
               f"<p><i>Modelo: {MODELO_LLM}. Texto generado: verificar con el documento.</i></p>",
               padre=doc["key"], tags=("resumen-generado-llm",))
```

Salida registrada

```
KD55JQGH: texto obtenido del índice de Zotero.
CLK46GYK: texto obtenido del índice de Zotero.
EZ5CWSSC: texto obtenido del índice de Zotero.
UJ9AC7E7: texto obtenido del índice de Zotero.
4KHT582K: texto obtenido del índice de Zotero.
9C8KJP2G: texto obtenido del índice de Zotero.
GZZDYNZD: texto obtenido del índice de Zotero.
SCK2GJ2Z: texto obtenido del índice de Zotero.
```

```
JISEWSCL: texto obtenido del índice de Zotero.
UD6REMBN: texto obtenido del índice de Zotero.
HZRXRCTW: texto obtenido del índice de Zotero.

11 de 11 documentos no tienen una sección de resumen en el PDF.

AVTE4IA3 | Companies selling authorship slots thrive in a culture that equates success with

El texto académico tiene como objetivo principal exponer el funcionamiento de las "fábricas de artículos"
(paper mills), empresas fraudulentas que venden autorías en publicaciones científicas, y analizar las causas
y consecuencias de este fenómeno. El método empleado para la investigación incluye el análisis de casos
concretos, como el de un profesor saudí que compra autorías para lograr su promoción, y el de un post en
Facebook que ofrecía coautoría en un artículo ya aceptado. También se recurre a entrevistas con
investigadores, como una bióloga de plantas de EE. UU., y al expertise de especialistas en integridad
académica como Ivan Oransky y Anna Abalkina. Los resultados principales revelan que esta práctica es un
síntoma de un sistema académico que presiona a los investigadores para que publiquen grandes cantidades de
artículos, often linking their career advancement and job security to these metrics. Esta presión es
particularmente intensa en países como China, donde un análisis de Nature identifica que siete de las diez
instituciones con mayor proporción de artículos retractados eran hospitales. Las paper mills operan a escala
industrial, generando cientos de millones de dólares anuales mediante técnicas que incluyen la suplantación
de identidad, el plagio encubierto con IA y la creación de manuscritos fraudulentos, lo que representa una
grave amenaza para la integridad de la ciencia, especialmente en campos como la medicina y la biología.

Nota M889E46K creada en el documento AVTE4IA3.
```

Ninguno de los 11 documentos sin resumen tiene una sección de resumen en el PDF. Para el primero, `AVTE4IA3`*, el modelo redactó un resumen que se guardó como nota hija (* `M889E46K`*) con la etiqueta* `resumen-generado-llm`*. El texto es generado: sus cifras deben comprobarse en el documento antes de usarlas.*

11

Filtro de relevancia y memoria local

En este apartado el LLM valora la pertinencia de cada artículo candidato y se guarda en disco qué artículos se han revisado. Así se evitan resultados fuera de tema y repeticiones entre ejecuciones. Se apoya en las partes 8 (OpenAlex) y 9 (LLM) y deja definidas `es_relevante()` y `evaluar_relevancia()`, junto con el archivo de historial.

El agente no debe mostrar dos veces el mismo artículo ni aceptar cualquier resultado de OpenAlex. Se combinan dos piezas: un juicio del LLM sobre cada candidato y un historial en disco.

11.1 Interpretar bien la respuesta del LLM

En las versiones v1 a v3 se comprobaba `"RELEVANTE" in decision`. Esa condición también es cierta cuando el modelo responde `IRRELEVANTE`, así que **todos** los candidatos pasaban el filtro. Lo mismo ocurre al buscar `"SI"` dentro de cualquier frase que contenga esas letras, como `"NO ES SIMILAR"`. La corrección, `es_relevante()`, definida en la celda siguiente, solo acepta la respuesta si su **primera palabra** es afirmativa.

El filtro estima pertinencia y el historial conserva antecedentes de procesamiento. El primero influye en qué literatura se presenta al investigador; el segundo evita repetir operaciones y permite revisar decisiones previas. Una respuesta mal interpretada es un fallo del programa; una respuesta formalmente correcta pero temáticamente equivocada es un fallo de clasificación. La evaluación debe distinguirlos.

Esto ocurrió en la versión original, que filtraba con `if "RELEVANTE" in decision.upper():`. Salida registrada:

```
Analizando novedades...
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Asia Pacific Journal of Human Resources
Link: https://doi.org/10.1111/(issn)1744-7941
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Sycophantic AI decreases prosocial intentions and promotes dependence
Link: https://doi.org/10.1126/science.aec8352
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Reliability of LLMs as medical assistants for the general public: a randomized
preregistered study
Link: https://doi.org/10.1038/s41591-025-04074-y
------------------------------
```

```
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Surviving Sepsis Campaign: international guidelines for management of sepsis and septic shock 2026
Link: https://doi.org/10.1007/s00134-026-08361-1
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: An investigation into the potential of collaborative computer game-based learning in Higher Education
Link: https://doi.org/10.17869/enu.237315
------------------------------
```

```python
def es_relevante(respuesta):
    """True solo si la PRIMERA palabra de la respuesta es afirmativa."""
    palabras = re.findall(r"[A-ZÁÉÍÓÚÑ]+", respuesta.upper())
    return bool(palabras) and palabras[0] in {"RELEVANTE", "SI", "SÍ", "YES"}
```

11.2 Reproducir la salida de la versión original

La celda siguiente reproduce el bucle de filtrado de la v1 con los cinco resultados de OpenAlex de la salida registrada. Como la ejecución original no guardó las respuestas del modelo, se simulan: el modelo rechaza cuatro artículos y acepta uno. Con la comprobación `"RELEVANTE" in decision.upper()`, el bucle anuncia los cinco como relevantes y su salida coincide con la registrada.

Reproducir una salida incorrecta permite separar el comportamiento observado de la solución adoptada. Las respuestas predefinidas de esta demostración explican un error lógico; no son nuevas llamadas al modelo ni evidencia de rendimiento. El ejemplo permite comprobar cómo una misma decisión textual produce selecciones diferentes según la regla de interpretación.

```python
# Los cinco resultados de OpenAlex de la salida registrada (búsqueda de la sección 8.2)
resultados = [
    {"display_name": "Asia Pacific Journal of Human Resources",
     "doi": "https://doi.org/10.1111/(issn)1744-7941"},
    {"display_name": "Sycophantic AI decreases prosocial intentions and promotes dependence",
     "doi": "https://doi.org/10.1126/science.aec8352"},
    {"display_name": "Reliability of LLMs as medical assistants for the general public: a randomized preregistered study",
     "doi": "https://doi.org/10.1038/s41591-025-04074-y"},
    {"display_name": "Surviving Sepsis Campaign: international guidelines for management of sepsis and septic shock 2026",
     "doi": "https://doi.org/10.1007/s00134-026-08361-1"},
    {"display_name": "An investigation into the potential of collaborative computer game-based learning in Higher Education",
     "doi": "https://doi.org/10.17869/enu.237315"},
]
perfil_usuario = "Large language models, medical education, and assessment tools"

# Respuestas del modelo SIMULADAS (la ejecución original no las guardó)
respuestas_llm = {
    resultados[0]["display_name"]: "IRRELEVANTE",
    resultados[1]["display_name"]: "IRRELEVANTE",
```

```
    resultados[2]["display_name"]: "RELEVANTE",
    resultados[3]["display_name"]: "IRRELEVANTE",
    resultados[4]["display_name"]: "IRRELEVANTE",
}


def evaluar_con_ia(perfil, paper):
    """Sustituye la llamada al LLM de la v1 por la respuesta simulada."""
    return respuestas_llm[paper["display_name"]]


# ── Bucle de la versión original (v1) ──
print("Analizando novedades...")
for paper in resultados:
    decision = evaluar_con_ia(perfil_usuario, paper)
    if "RELEVANTE" in decision.upper():          # también es cierto para "IRRELEVANTE"
        print("NUEVO ARTICULO RELEVANTE ENCONTRADO:")
        print(f"Titulo: {paper['display_name']}")
        print(f"Link: {paper['doi']}")
        print("-" * 30)
```

Salida generada al ejecutar

```
Analizando novedades...
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Asia Pacific Journal of Human Resources
Link: https://doi.org/10.1111/(issn)1744-7941
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Sycophantic AI decreases prosocial intentions and promotes dependence
Link: https://doi.org/10.1126/science.aec8352
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Reliability of LLMs as medical assistants for the general public: a randomized preregistered study
Link: https://doi.org/10.1038/s41591-025-04074-y
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Surviving Sepsis Campaign: international guidelines for management of sepsis and septic shock 2026
Link: https://doi.org/10.1007/s00134-026-08361-1
------------------------------
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: An investigation into the potential of collaborative computer game-based learning in Higher
Education
Link: https://doi.org/10.17869/enu.237315
------------------------------
```

Es idéntica a la salida registrada de la v1: los cinco artículos se anuncian como relevantes, aunque el modelo rechazó cuatro.

El mismo bucle con `es_relevante()`. Solo cambia la condición del `if`.

```
# ── Mismo bucle con el filtro corregido ──
print("Analizando novedades (filtro corregido)...")
for paper in resultados:
    decision = evaluar_con_ia(perfil_usuario, paper)
    if es_relevante(decision):
        print("NUEVO ARTICULO RELEVANTE ENCONTRADO:")
        print(f"Titulo: {paper['display_name']}")
        print(f"Link: {paper['doi']}")
        print("-" * 30)
```

```
    else:
        print(f"Descartado ({decision}): {paper['display_name'][:70]}")
```

Salida generada al ejecutar

```
Analizando novedades (filtro corregido)...
Descartado (IRRELEVANTE): Asia Pacific Journal of Human Resources
Descartado (IRRELEVANTE): Sycophantic AI decreases prosocial intentions and promotes dependence
NUEVO ARTICULO RELEVANTE ENCONTRADO:
Titulo: Reliability of LLMs as medical assistants for the general public: a randomized preregistered study
Link: https://doi.org/10.1038/s41591-025-04074-y
------------------------------
Descartado (IRRELEVANTE): Surviving Sepsis Campaign: international guidelines for management of
Descartado (IRRELEVANTE): An investigation into the potential of collaborative computer game-bas
```

Con el filtro corregido solo se anuncia el artículo que el modelo marcó como relevante; los otros cuatro se descartan. En el agente (parte 14), las respuestas no son simuladas: las produce `evaluar_relevancia()` *(sección 11.3) con el LLM real.*

11.3 Evaluar un candidato con el LLM

Se envían título y resumen, y se pide una sola palabra con `temperature=0`. Si OpenRouter falla (red, saldo o límite de uso), la función devuelve `None` en lugar de `False`, para que el agente distinga «no se pudo evaluar» de «no es relevante». Los errores de programación, como una función no definida, no se capturan: se muestran al momento.

La relación de un trabajo con el corpus debe juzgarse respecto de una pregunta o un alcance temático definidos. El título y el resumen proporcionan indicios, pero pueden ser insuficientes para decidir la inclusión final. Diferenciar una evaluación negativa de una evaluación no realizada evita contabilizar fallos del servicio como desacuerdos científicos.

```
def evaluar_relevancia(perfil, paper, modelo=None):
    resumen = reconstruir_abstract(paper.get("abstract_inverted_index"))[:800]
    prompt = (
        f"Perfil de mi investigación: {perfil}\n\n"
        f"Artículo candidato:\nTítulo: {paper.get('display_name')}\nResumen: {resumen or 'No
disponible'}\n\n"
        "¿Es relevante para mi investigación? Responde con UNA palabra: RELEVANTE o IRRELEVANTE."
    )
    try:
        respuesta = preguntar_llm(prompt, modelo=modelo, temperatura=0, max_tokens=5)
    except Exception as e:          # fallo de OpenRouter (red, saldo, límite de uso…)
        logging.error(f"LLM no disponible al evaluar {paper.get('id')}: {type(e).__name__}")
        return None                 # None = no se pudo evaluar (el agente lo cuenta aparte)
    return es_relevante(respuesta)
```

Ejemplo con un resultado de OpenAlex. La celda evalúa la primera publicación recuperada en la sección 8.6 frente al perfil temático `perfil_ia` de la sección 10.2 y distingue tres casos: relevante, irrelevante o no evaluada, cuando OpenRouter no responde.

```
paper = publicaciones[0]
resultado = evaluar_relevancia(perfil_ia, paper)     # perfil generado en la sección 10.2

decision = (
    "NO SE PUDO EVALUAR" if resultado is None
    else "RELEVANTE" if resultado
    else "IRRELEVANTE"
```

```
)

print(paper.get("display_name", "(sin título)"))
print(decision)
```

Salida registrada

```
Citation manipulation through citation mills and pre-print servers
RELEVANTE
```

11.4 Memoria: el historial

El historial es un diccionario JSON cuya clave es el identificador de OpenAlex. Además se compara el título normalizado, porque la misma obra puede aparecer con dos identificadores (preprint y versión publicada).

Conservar los identificadores permite reconocer trabajos ya procesados, mientras que los títulos ayudan a detectar coincidencias entre registros. Ambos son indicios y tienen límites: un título semejante puede corresponder a obras distintas y un preprint puede requerir vincularse con su versión publicada. El historial no es memoria semántica ni entrenamiento del modelo; es un registro local del proceso.

```
historial = cargar_json(ARCHIVO_HISTORIAL, {})
titulos_vistos = {normalizar_titulo(v.get("titulo", "")) for v in historial.values()}
print(f"Memoria cargada: {len(historial)} documentos registrados.")

primero = next(iter(historial.values()))
print("\nEjemplo de registro:")
print(json.dumps(primero, indent=2, ensure_ascii=False))

candidato = "FROM INTEGRITY TO INFLATION: ethical and unethical citation practices in academic
publishing"
print("\n¿Ya visto?", normalizar_titulo(candidato) in titulos_vistos)
```

Salida generada al ejecutar

```
Memoria cargada: 23 documentos registrados.

Ejemplo de registro:
{
  "titulo": "Overview and trends in forensic palynology based on a scientometric study in the PubMed
database",
  "url": "https://openalex.org/W4405993893",
  "citas": 2,
  "fecha": "2026-04-28",
  "doi": "https://doi.org/10.33447/paubrasilia.2025.e0166",
  "keyword": "forensic scientometrics"
}

¿Ya visto? True
```

Salida generada con el `historial_detallado.json` real de la carpeta del proyecto.

12

Taxonomía y palabras clave generadas por el LLM

En este apartado se pide al modelo, en una sola llamada, una taxonomía bilingüe del tema y las frases de búsqueda para OpenAlex, y se separan ambas partes de la respuesta de forma fiable. Las frases de búsqueda alimentan al agente y la taxonomía documenta el campo de estudio. Se apoya en la parte 10 y deja definidas `crear_prompt_taxonomia()` y `parsear_respuesta()`.

El agente pide al modelo dos cosas en una sola llamada: una taxonomía bilingüe del tema y las frases de búsqueda para OpenAlex. Para separarlas con fiabilidad se usan **delimitadores fijos** y una expresión regular.

12.1 El problema de analizar texto libre

En versiones previas del código de prueba se pedían las palabras clave sin indicar el formato de respuesta y luego se separaban por comas. El modelo añadió una frase de cortesía y una numeración, y ese texto acabó mezclado con los términos. La celda muestra la respuesta del modelo reconstruida a partir de su salida registrada.

La estructura propuesta por el LLM puede ayudar a ordenar una colección y explorar vocabulario para búsquedas. No debe presentarse como taxonomía científica validada sin definir sus categorías, alcance y relaciones. Los términos bilingües requieren revisión conceptual, porque una traducción literal puede no coincidir con la terminología utilizada por una comunidad académica.

```
respuesta_v5 = ("Based on the research titles provided, here are 3 technical keywords for
OpenAlex:\n\n"
                "1. Citation, Bibliometrics, Scientometrics\n"
                "2. Research Misconduct, Citation Manipulation, Citation Cartels\n"
                "3. Disinformation, Scholarly Publishing, Paper Mills")

terminos = [t.strip() for t in respuesta_v5.split(",")]   # método de la v5: separar por comas

print(f"Se obtienen {len(terminos)} 'términos':\n")
for i, t in enumerate(terminos, 1):
    print(f"{i}. {t!r}")
```

Salida generada al ejecutar

```
Se obtienen 8 'términos':

1. 'Based on the research titles provided'
2. 'here are 3 technical keywords for OpenAlex:\n\n1. Citation'
3. 'Bibliometrics'
```

```
4. 'Scientometrics\n2. Research Misconduct'
5. 'Citation Manipulation'
6. 'Citation Cartels\n3. Disinformation'
7. 'Scholarly Publishing'
8. 'Paper Mills'
```

Los términos 1, 2, 4 y 6 son incorrectos: contienen la frase introductoria, saltos de línea (`\n`) o la numeración. Utilizados en OpenAlex, producen búsquedas sin sentido.

12.2 Plantilla de prompt con delimitadores

Qué es. Un *prompt* es el texto con las instrucciones que se envía al modelo. Una **plantilla** es ese texto con huecos entre llaves (`{tema}` , `{corpus}` , `{n}` …) que Python rellena con `.format()` en cada ejecución. Las instrucciones permanecen fijas y solo cambian los datos de la colección.

Por qué hace falta. El agente necesita dos productos distintos de una sola llamada al modelo: una taxonomía para leer, que se guarda como nota y en `estructura_tematica.txt` , y una lista de frases que el código enviará a OpenAlex. Con una respuesta en texto libre, el código no puede saber dónde termina una cosa y empieza la otra (sección 12.1). La plantilla obliga al modelo a responder con un formato fijo.

Partes de la plantilla

Bloque	Función
Rol: `Eres especialista en {tema}`	Sitúa al modelo en el dominio de la colección
Datos: `{corpus}`	Títulos y resúmenes de la colección (parte 9)
Tarea 1	Taxonomía bilingüe organizada en categorías
Tarea 2	`{n}` frases de búsqueda en inglés para OpenAlex
Delimitadores `---ESTRUCTURA---` y `---KEYWORDS---`	Marcas fijas que el código busca para separar las dos partes de la respuesta

Categorías. Si se pasa una lista de categorías, como `CATEGORIAS_FORENSE` , el modelo las respeta. Si no se pasa ninguna, el modelo propone entre 6 y 10 a partir de los documentos. Así la misma plantilla sirve para colecciones de cualquier tema.

`parsear_respuesta()` realiza el camino inverso: localiza los delimitadores con expresiones regulares, separa la taxonomía de las frases y elimina numeraciones y viñetas.

Comillas en las frases de búsqueda. En OpenAlex, las comillas indican una expresión exacta y deben ir siempre por pares. Una comilla sin pareja (`sneaked references" AND citation manipulation`) provoca un error 500. Unas comillas mal colocadas (`sneaked references" AND "citation manipulation`) hacen que se busque la expresión `" AND "` y no los conceptos. `normalizar_consulta()` elimina todas las comillas y vuelve a ponerlas alrededor de cada concepto de varias palabras. `consulta_simple()` ofrece una versión sin comillas ni operadores, que el agente usa como respaldo si la búsqueda exacta falla o no da resultados.

```
# Categorías de ejemplo para la colección Bibliometria_Forense (opcional)
CATEGORIAS_FORENSE = ["Términos principales", "Integridad científica", "Prácticas problemáticas",
                      "Detección y análisis", "Problemas emergentes", "Fenómenos asociados",
```

```
                      "Análisis y evaluación", "Sistemas y políticas", "Actores y redes",
                      "Variantes lingüísticas"]

PROMPT_TAXONOMIA = """Eres especialista en {tema}.
A partir de estos documentos de mi colección:
{corpus}

TAREA 1: Genera 'Palabras clave integradoras - {tema}'.
{instruccion_categorias}
Escribe cada categoría como título terminado en dos puntos y, debajo, de 3 a 8 términos,
uno por línea, en formato 'Término en inglés / Término en español'.

TAREA 2: Selecciona las {n} frases de búsqueda en inglés más específicas para OpenAlex.
Escribe cada frase sin comillas y separa los conceptos con AND,
por ejemplo: paper mills AND authorship for sale

RESPONDE EXACTAMENTE CON ESTOS DELIMITADORES, sin texto adicional:
---ESTRUCTURA---
(taxonomía)
---KEYWORDS---
(una frase por línea, sin numerar)
"""


def crear_prompt_taxonomia(corpus, tema, n, categorias=None):
    """Rellena la plantilla con los datos de una colección."""
    if categorias:
        instruccion = "Usa exactamente estas categorías:\n" + "\n".join(f"- {c}" for c in categorias)
    else:
        instruccion = "Propón entre 6 y 10 categorías que organicen el tema de los documentos."
    return PROMPT_TAXONOMIA.format(tema=tema, corpus=corpus, instruccion_categorias=instruccion, n=n)


def normalizar_consulta(frase):
    """Reconstruye una frase de búsqueda con las comillas siempre emparejadas.

    'paper mills" AND authorship sale'  ->  '"paper mills" AND "authorship sale"'
    """
    sin_comillas = re.sub(r"[\"“”]", " ", frase)
    partes = re.split(r"\s+(AND|OR|NOT)\s+", sin_comillas.strip())
    salida = []
    for parte in partes:
        parte = " ".join(parte.split())
        if parte in ("AND", "OR", "NOT"):
            salida.append(parte)
        elif parte:
            salida.append(f'"{parte}"' if " " in parte else parte)
    return " ".join(salida)


def consulta_simple(frase):
    """Versión de respaldo: solo las palabras, sin comillas ni operadores."""
    return " ".join(re.sub(r"[\"“”]|\b(AND|OR|NOT)\b", " ", frase).split())


def parsear_respuesta(texto, n):
    est = re.search(r"---ESTRUCTURA---(.*?)---KEYWORDS---", texto, re.DOTALL)
    kws = re.search(r"---KEYWORDS---(.*)", texto, re.DOTALL)
    estructura = est.group(1).strip() if est else ""
```

```
    keywords = []
    for linea in (kws.group(1) if kws else "").splitlines():
        limpia = re.sub(r"^\s*(\d+[\.\)]|[-*•])\s*", "", linea).strip()
        if len(limpia) > 3 and not limpia.endswith(":"):
            keywords.append(normalizar_consulta(limpia))
    return estructura, keywords[:n]
```

Comprobar la normalización. La celda aplica `normalizar_consulta()` y `consulta_simple()` a cuatro frases mal formadas: dos con una comilla sin pareja, que provocan error 500, y dos con las comillas alrededor de `AND`, que no devuelven resultados.

Corregir comillas y simplificar una expresión mejora su compatibilidad con la búsqueda. No asegura que el término represente el fenómeno estudiado. La comparación entre consulta original y consulta simplificada sirve para examinar cuánto cambia la intención de búsqueda y qué resultados se incorporan o se pierden.

```
# Frases mal formadas
frases_con_error = [
    'sneaked references" AND citation manipulation',      # comilla sin pareja -> error 500
    'paper mills" AND authorship sale',                   # comilla sin pareja -> error 500
    'sneaked references" AND "citation manipulation',     # comillas alrededor de AND -> 0 resultados
    'tortured phrases" AND "plagiarism evasion',          # comillas alrededor de AND -> 0 resultados
]
for frase in frases_con_error:
    print(f"Antes:        {frase}")
    print(f"Normalizada:  {normalizar_consulta(frase)}")
    print(f"Respaldo:     {consulta_simple(frase)}\n")
```

Salida generada al ejecutar

```
Antes:        sneaked references" AND citation manipulation
Normalizada:  "sneaked references" AND "citation manipulation"
Respaldo:     sneaked references citation manipulation

Antes:        paper mills" AND authorship sale
Normalizada:  "paper mills" AND "authorship sale"
Respaldo:     paper mills authorship sale

Antes:        sneaked references" AND "citation manipulation
Normalizada:  "sneaked references" AND "citation manipulation"
Respaldo:     sneaked references citation manipulation

Antes:        tortured phrases" AND "plagiarism evasion
Normalizada:  "tortured phrases" AND "plagiarism evasion"
Respaldo:     tortured phrases plagiarism evasion
```

Todas las frases quedan con comillas emparejadas, que OpenAlex puede interpretar.

Ver el prompt que recibe el modelo. La celda rellena la plantilla con dos documentos de la colección, sin lista de categorías, para mostrar el texto exacto que se enviaría.

Mostrar el prompt permite comprobar qué documentos y qué requisitos de salida se presentan al modelo. Esta revisión hace visible una parte central del diseño que suele quedar oculta en una interfaz conversacional. La interpretación del resultado debe referirse al prompt efectivamente utilizado, junto con los datos de entrada.

```
corpus_ejemplo = "\n".join(f"Título: {r['titulo']} | Resumen: {r['resumen'][:120]}" for r in
mis_datos[:2])
print(crear_prompt_taxonomia(corpus_ejemplo, tema="bibliometría forense", n=6))
```

Salida generada al ejecutar

```
Eres especialista en bibliometría forense.
A partir de estos documentos de mi colección:
Título: Bibliometric evaluation of Forensic Science International as a scholarly journal within the subject
category legal medicine | Resumen: This article presents a bibliometric evaluation of Forensic Science
International (FSI) as a scholarly journal within th
Título: Reference accuracy of primary studies about reference accuracy: A descriptive study | Resumen:
Reference accuracy is defined as the completeness and correctness of citation components such as author
names, article t

TAREA 1: Genera 'Palabras clave integradoras - bibliometría forense'.
Propón entre 6 y 10 categorías que organicen el tema de los documentos.
Escribe cada categoría como título terminado en dos puntos y, debajo, de 3 a 8 términos,
uno por línea, en formato 'Término en inglés / Término en español'.

TAREA 2: Selecciona las 6 frases de búsqueda en inglés más específicas para OpenAlex.
Escribe cada frase sin comillas y separa los conceptos con AND,
por ejemplo: paper mills AND authorship for sale

RESPONDE EXACTAMENTE CON ESTOS DELIMITADORES, sin texto adicional:
---ESTRUCTURA---
(taxonomía)
---KEYWORDS---
(una frase por línea, sin numerar)
```

12.3 Probar el análisis

Se prueba con una respuesta construida a partir de los archivos que generó el agente en una ejecución anterior (`estructura_tematica.txt` y `keywords_relevantes.txt`), con numeración y comillas añadidas para comprobar la limpieza.

La respuesta reconstruida permite verificar el procesamiento del texto bajo condiciones conocidas. Se trata de una prueba de extracción y limpieza, independiente de la calidad temática del modelo. Los archivos de una ejecución anterior son prerrequisitos de esta celda: si no existen en el entorno local, debe reconocerse esa dependencia antes de interpretar el fallo.

```
estructura_previa = ARCHIVO_ESTRUCTURA.read_text(encoding="utf-8")
keywords_previas = ARCHIVO_KEYWORDS.read_text(encoding="utf-8").splitlines()

respuesta_llm = ("Aquí está el resultado.\n---ESTRUCTURA---\n" + estructura_previa +
                 "\n---KEYWORDS---\n" + "\n".join(f'{i}. "{k}"' for i, k in
enumerate(keywords_previas, 1)))

estructura, keywords = parsear_respuesta(respuesta_llm, n=6)

print(f"Categorías detectadas: {sum(1 for l in estructura.splitlines() if l.endswith(':'))}")
print("\n".join(estructura.splitlines()[:5]))
print("...")
```

```python
print("\nFrases de búsqueda:")
for k in keywords:
    print("  ·", k)
```

Salida generada al ejecutar

```
Categorías detectadas: 10
Términos principales:
Forensic Scientometrics / Bibliometría Forense
Scientific Integrity / Integridad científica
Research Misconduct / Mala conducta investigadora
Publication Ethics / Ética de la publicación
...

Frases de búsqueda:
  · "sneaked references" AND "citation manipulation"
  · "paper mills" AND "authorship for sale"
  · "forensic scientometrics" AND "research misconduct"
  · "predatory publishing" AND "special issues"
  · "generative AI" AND "confabulated references"
  · "journal takeovers" AND "questionable publishers"
```

Salida generada con los archivos reales de la carpeta del proyecto.

13

Escribir en Zotero: notas e ítems

En este apartado los resultados se convierten en HTML y se guardan como notas o ítems dentro de la colección. De este modo los hallazgos quedan en Zotero junto a tus documentos. Para escribir hace falta una clave con *Allow write access* y activar `ESCRIBIR_EN_ZOTERO = True` (sección 6.7); con `False`, todas las escrituras se simulan.

El agente devuelve sus resultados a la biblioteca como **notas** dentro de la colección. Todas las funciones de escritura respetan el interruptor `ESCRIBIR_EN_ZOTERO`, que se define en la sección 6.7: con `False` solo simulan, para poder probar sin modificar la biblioteca.

13.1 Taxonomía y resultados en HTML

Las notas de Zotero son HTML. Se escapan los textos con `html.escape` para que un título con `<` o `&` no rompa la nota. La vista previa muestra solo las dos primeras categorías (`primeras_categorias()`); la nota que se crea en la sección 13.2 contiene la taxonomía completa.

Registrar hallazgos en la biblioteca facilita relacionarlos con las fuentes de partida y conservar decisiones del proceso. La nota generada debe distinguir sus sugerencias de las anotaciones verificadas del investigador. Crear una nota o un ítem es una operación técnica de almacenamiento, no una certificación de calidad o pertinencia.

```
def estructura_a_html(estructura):
    partes, abierta = [], False
    for linea in estructura.splitlines():
        linea = linea.strip()
        if not linea:
            continue
        if linea.endswith(":"):
            if abierta:
                partes.append("</ul>")
            partes.append(f"<h3>{html.escape(linea[:-1])}</h3><ul>")
            abierta = True
        else:
            partes.append(f"<li>{html.escape(linea)}</li>")
    if abierta:
        partes.append("</ul>")
    return "".join(partes)


def fuentes_a_html(hallazgos):
    filas = []
    for h in hallazgos:
        enlace = h.get("doi") or h.get("url")
```

```
        filas.append(f"<li><b>{html.escape(h['titulo'])}</b><br/>"
                     f"Citas: {h['citas']} | Año: {h.get('anio', '')} | Búsqueda:
{html.escape(h['keyword'])}<br/>"
                     f"<a href=\"{enlace}\">{enlace}</a></li>")
    return "<ul>" + "".join(filas) + "</ul>"


html_taxonomia = estructura_a_html(estructura)
print(html_taxonomia[:300], "...")      # código HTML que recibirá Zotero

def primeras_categorias(html_nota, n=2):
    """Las n primeras categorías de la taxonomía, para mostrar un extracto."""
    return "".join("<h3>" + bloque for bloque in html_nota.split("<h3>")[1:n + 1])


from IPython.display import HTML, display
display(HTML(primeras_categorias(html_taxonomia, 2)))   # extracto: la nota completa se crea en 13.2
```

Salida registrada

```
<h3>Términos principales</h3><ul><li>Forensic Scientometrics / Bibliometría Forense</li><li>Scientific
Integrity / Integridad científica</li><li>Research Misconduct / Mala conducta investigadora</li>
<li>Publication Ethics / Ética de la publicación</li></ul><h3>Integridad científica</h3><ul><li>Refer ...
```

Vista de la nota en Zotero

Términos principales

- Forensic Scientometrics / Bibliometría Forense
- Scientific Integrity / Integridad científica
- Research Misconduct / Mala conducta investigadora
- Publication Ethics / Ética de la publicación

Integridad científica

- Reference Accuracy / Precisión de referencias
- Citation Integrity / Integridad de citas
- Data Manipulation / Manipulación de datos
- Image Manipulation / Manipulación de imágenes
- Authorship Misrepresentation / Falsificación de autoría
- Institutional Affiliation Misrepresentation / Falsificación de afiliación institucional

13.2 Crear una nota en una colección

`crear_nota_zotero()` es la función que usa el agente para guardar sus resultados. Respeta el interruptor `ESCRIBIR_EN_ZOTERO` de la sección 6.7: para crear notas reales, ese interruptor debe valer `True` y la clave necesita *Allow write access*. A diferencia de `crear_nota()` (sección 7.2), añade un título a la nota y devuelve un mensaje de texto, que el agente muestra en pantalla.

```
def crear_nota_zotero(titulo, contenido_html, coleccion, padre=None, tags=("agente-ia",)):
    """Crea una nota en Zotero. Devuelve un mensaje con el resultado."""
    if not ESCRIBIR_EN_ZOTERO:
        return f"[simulación] Nota '{titulo}' preparada ({len(contenido_html)} caracteres); no se
envía."
    nota = {
```

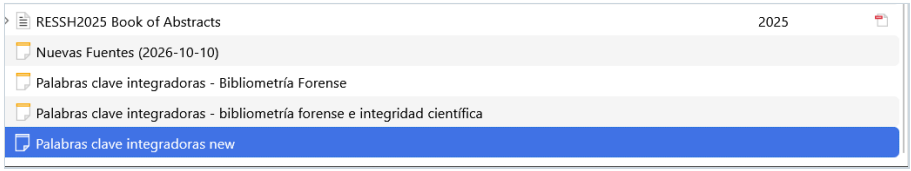
```
        "itemType": "note",
        "note": f"<h2>{html.escape(titulo)}</h2>{contenido_html}",
        "tags": [{"tag": t} for t in tags],
        "collections": [coleccion] if not padre else [],
    }
    if padre:
        nota["parentItem"] = padre
    respuesta = zot.create_items([nota])
    if respuesta.get("successful"):
        return f"Nota '{titulo}' creada en Zotero."
    return f"Error al crear la nota: {respuesta.get('failed')}"


print(crear_nota_zotero("Palabras clave integradoras", estructura_a_html(estructura), COLECCION_ID))
```

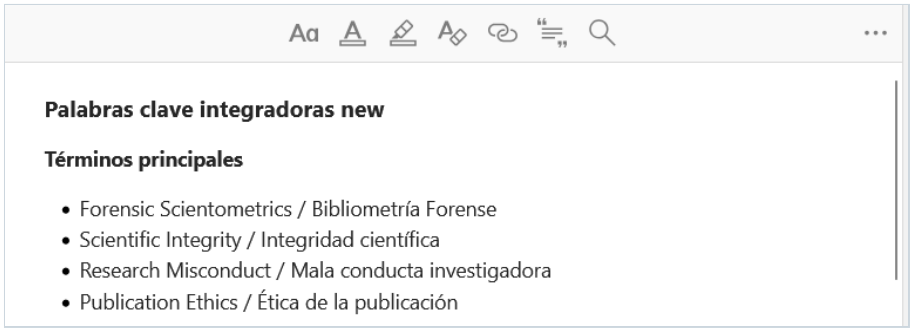
Salida registrada

```
Nota 'Palabras clave integradoras' creada en Zotero.
```

Salida registrada con `ESCRIBIR_EN_ZOTERO = True`; las capturas corresponden a la ejecución del 13/10/2026. La nota se creó en la colección. Las capturas de Zotero lo confirman: la nota aparece en la lista de la colección junto a las que generó el agente, y su contenido conserva la taxonomía con títulos de categoría y listas de términos.



Notas de la colección en Zotero, entre ellas «Palabras clave integradoras new», creada por esta celda.



Inicio de la nota en Zotero: el título y la primera categoría, «Términos principales». El resto de las categorías sigue el mismo formato: un título por categoría y una lista de términos.

13.3 Añadir un artículo de OpenAlex como ítem de Zotero

Paso opcional: convertir un hallazgo en un `journalArticle` real de la colección, con título, autores, fecha, revista y DOI.

Importar un hallazgo exige decidir qué campos externos equivalen a los del gestor bibliográfico. La conversión ilustrada crea un `journalArticle`; debe comprobarse que el tipo real del resultado es compatible y que sus autores, fecha y revista se trasladaron correctamente. El DOI facilita la comparación con registros existentes, pero no todos los trabajos disponen de él.

```
def openalex_a_zotero(paper, coleccion):
    if not ESCRIBIR_EN_ZOTERO:
        return f"[simulación] '{paper.get('display_name', '')[:60]}' no se añade."
    item = zot.item_template("journalArticle")
    item["title"] = paper.get("display_name", "")
    item["date"] = paper.get("publication_date", "")
    item["DOI"] = (paper.get("doi") or "").replace("https://doi.org/", "")
    item["url"] = paper.get("doi") or paper["id"]
    item["abstractNote"] = reconstruir_abstract(paper.get("abstract_inverted_index"))
    fuente = (paper.get("primary_location") or {}).get("source") or {}
    item["publicationTitle"] = fuente.get("display_name", "")
    item["creators"] = [{"creatorType": "author", "name": a["author"]["display_name"]}
                        for a in paper.get("authorships", [])[:20]]
    item["collections"] = [coleccion]
    item["tags"] = [{"tag": "openalex"}, {"tag": "agente-ia"}]
    return zot.create_items([item])
```

Ejemplo con una publicación ya recuperada. La celda toma la primera publicación de la lista de la sección 8.6 y comprueba, con `doi_normalizado()` (sección 6.3), que su DOI no esté ya en la colección de trabajo antes de añadirla. Con `ESCRIBIR_EN_ZOTERO = True` crea el ítem en Zotero; con `False` solo lo simula.

```
paper = publicaciones[0]          # primera publicación recuperada en la sección 8.6
doi = doi_normalizado(paper.get("doi"))
print(f"{paper.get('display_name', '(sin título)')[:80]} | DOI: {doi or 'sin DOI'}")

if doi and any(doi_normalizado(campo(it, "DOI")) == doi for it in documentos):
    print("Ya está en la colección de trabajo; no se añade.")
else:
    print(openalex_a_zotero(paper, COLECCION_ID))
```

Salida registrada

```
Citation manipulation through citation mills and pre-print servers | DOI: 10.1038/s41598-025-88709-7
Ya está en la colección de trabajo; no se añade.
```

La primera publicación de la sección 8.6, Citation manipulation through citation mills and pre-print servers, ya estaba en la colección de trabajo (con el mismo DOI), de modo que no se añade de nuevo.

14

El agente completo

En este apartado se ejecuta con una sola llamada el ciclo completo: lectura de la colección, generación de la estrategia de búsqueda, consulta a OpenAlex, filtrado y registro de resultados. La colección y el tema se indican en la llamada, de modo que el mismo agente sirve para cualquier línea de investigación de tu biblioteca. Requiere haber ejecutado las celdas de definición de las partes 2 a 13.

Une todas las piezas en un ciclo de cuatro fases:

1. **Contexto**: lee los documentos de la colección en Zotero.
2. **Estrategia**: el LLM genera la taxonomía y las frases de búsqueda; se guardan en disco y como nota.
3. **Búsqueda y filtro**: consulta OpenAlex por cada frase, descarta lo ya visto y pide al LLM que valore cada candidato.
4. **Entrega**: actualiza el historial y crea una nota con las fuentes nuevas.

La función reúne decisiones de recuperación, propuestas del modelo y operaciones de registro dentro de un flujo definido por el programa. Su ejecución completa demuestra conexión funcional entre componentes, pero no mide por sí sola el rendimiento bibliográfico. Para usarla como base de investigación hay que evaluar pertinencia, omisiones y estabilidad respecto de criterios explícitos.

```
def ejecutar_agente(coleccion_id, tema=None, categorias=None, n_keywords=6,
                    anios_atras=2, por_kw=5, filtrar_con_llm=True):
    """Ciclo completo del agente sobre una colección.

    coleccion_id : clave de la colección (sección 3.3)
    tema         : descripción del tema; si se omite, se usa el nombre de la colección
    categorias   : lista de categorías para la taxonomía; si se omite, las propone el LLM
    n_keywords   : frases de búsqueda que generará el LLM
    anios_atras  : ventana de publicación en OpenAlex
    por_kw       : resultados de OpenAlex por frase de búsqueda
    """
    # Comprobación previa: el agente usa funciones definidas en las partes 8 a 13
    necesarias = ["buscar_openalex", "normalizar_registros", "preguntar_llm", "es_relevante",
                  "evaluar_relevancia", "crear_prompt_taxonomia", "parsear_respuesta",
                  "normalizar_consulta", "consulta_simple", "crear_nota_zotero", "estructura_a_html"]
    faltan = [f for f in necesarias if f not in globals()]
    if faltan:
        raise NameError("Ejecuta antes las celdas que definen: " + ", ".join(faltan))

    tema = tema or zot.collection(coleccion_id)["data"]["name"]
    carpeta = BASE_DIR / "colecciones" / coleccion_id          # archivos propios de cada colección
    carpeta.mkdir(parents=True, exist_ok=True)
    f_estructura = carpeta / "estructura_tematica.txt"
```

```
    f_keywords = carpeta / "keywords_relevantes.txt"
    f_historial = carpeta / "historial_detallado.json"
    logging.info(f"--- Inicio de ciclo: colección {coleccion_id} ({tema}) ---")

    # Fase 1. Contexto
    print("Fase 1: Extrayendo metadata completa de Zotero...")
    items = zot.everything(zot.collection_items_top(coleccion_id))
    fichas = normalizar_registros(items, exigir_resumen=False, verbose=False)
    if not fichas:
        print(f"La colección {coleccion_id} no contiene documentos. Revisa la clave (sección 3.3).")
        return []
    corpus = "\n".join(f"Título: {f['titulo']} | Resumen: {f['resumen'][:300]}" for f in fichas)

    # Fase 2. Estrategia
    print("Fase 2: El LLM esta construyendo la estructura robusta...")
    respuesta = preguntar_llm(
        crear_prompt_taxonomia(corpus, tema, n_keywords, categorias),
        temperatura=0.1, max_tokens=2500)
    estructura, keywords = parsear_respuesta(respuesta, n_keywords)
    if not keywords:
        print("El LLM no respetó el formato. Revisa el log y vuelve a ejecutar.")
        logging.error(f"Respuesta sin delimitadores: {respuesta[:500]}")
        return []
    guardar_texto(f_estructura, estructura)
    guardar_texto(f_keywords, "\n".join(keywords))
    print(f"Estructura guardada. Top {len(keywords)} keywords listas.")
    logging.info(crear_nota_zotero(f"Palabras clave integradoras - {tema}",
                                   estructura_a_html(estructura), coleccion_id))

    # Fase 3. Búsqueda y filtro
    print("Fase 3: Buscando en OpenAlex con radar ampliado...")
    historial = cargar_json(f_historial, {})
    vistos = {normalizar_titulo(v.get("titulo", "")) for v in historial.values()}
    perfil = f"{tema}: " + "; ".join(keywords)
    anio_desde = datetime.date.today().year - anios_atras
    filtros = {"publication_year": f">{anio_desde - 1}", "type": "article"}
    nuevos, errores, sin_evaluar = [], 0, 0

    for kw in keywords:
        print(f"Buscando: {kw}")
        candidatos = []
        # 1.º la frase exacta; 2.º, si falla o no da resultados, la versión simplificada
        for consulta in dict.fromkeys([kw, consulta_simple(kw)]):
            try:
                candidatos = buscar_openalex(consulta, filtros=filtros, por_pagina=por_kw)
            except requests.RequestException as e:
                respuesta_http = getattr(e, "response", None)
                codigo = respuesta_http.status_code if respuesta_http is not None else "sin respuesta"
                print(f"  ! OpenAlex respondió con error {codigo}; se omite esta consulta:
{consulta}")
                logging.error(f"OpenAlex ({consulta}): HTTP {codigo}")
                errores += 1
                continue
            if candidatos:
                break
        if not candidatos:
            print("  · Sin resultados para esta frase.")
            continue
        vistos_kw = descartados_kw = sin_evaluar_kw = 0
```

```
        for p in candidatos:
            titulo = p.get("display_name") or "Sin título"
            if p["id"] in historial or normalizar_titulo(titulo) in vistos:
                vistos_kw += 1
                continue
            if filtrar_con_llm:
                decision = evaluar_relevancia(perfil, p)
                if decision is None:
                    sin_evaluar_kw += 1
                    continue
                if not decision:
                    descartados_kw += 1
                    continue
            registro = {"titulo": titulo, "url": p["id"], "citas": p.get("cited_by_count", 0),
                        "anio": p.get("publication_year"), "fecha": str(datetime.date.today()),
                        "doi": p.get("doi"), "keyword": kw}
            historial[p["id"]] = registro
            vistos.add(normalizar_titulo(titulo))
            nuevos.append(registro)
            print(f"  + Encontrado: {titulo[:70]}...")
        resumen = f"  {len(candidatos)} candidato(s): {vistos_kw} ya revisado(s), {descartados_kw}
descartado(s) por el LLM"
        if sin_evaluar_kw:
            resumen += f", {sin_evaluar_kw} sin evaluar (fallo de OpenRouter)"
            sin_evaluar += sin_evaluar_kw
        print(resumen)

    # Fase 4. Entrega
    guardar_json(f_historial, historial)
    if nuevos:
        logging.info(crear_nota_zotero(f"Nuevas fuentes ({datetime.date.today()})",
                                       fuentes_a_html(nuevos), coleccion_id))
    logging.info(f"Ciclo terminado: {len(nuevos)} nuevos, {errores} consultas con error, {sin_evaluar}
sin evaluar.")
    if errores:
        print(f"Aviso: {errores} consultas a OpenAlex fallaron. Los detalles están en
{ARCHIVO_LOG.name}.")
    if sin_evaluar:
        print(f"Aviso: {sin_evaluar} candidatos no se evaluaron porque OpenRouter falló (red, saldo o
límite de uso).")
    print(f"Proceso finalizado. Total nuevos: {len(nuevos)}")
    return nuevos
```

Ejecutar el agente. Antes, ejecuta en orden todas las celdas de las partes 2 a 13: el agente usa sus funciones. La colección se indica en la propia llamada, de modo que el agente no depende de ninguna variable fijada al principio. El ejemplo usa la colección *Bibliometria_Forense*, su tema y sus categorías, con 10 frases de búsqueda. Cada línea `Buscando:` corresponde a una frase generada por el LLM y cada `+ Encontrado:` a un artículo nuevo que ha pasado los filtros.

Cada colección guarda sus archivos en `colecciones/<clave>/`, de modo que los historiales de temas distintos no se mezclan. Para conservar el historial ya acumulado de *Bibliometria_Forense*, mueve `historial_detallado.json` a `colecciones/6APD6L29/` antes de la primera ejecución.

Las pruebas entre colecciones deben conservar sus datos y parámetros por separado. El historial puede modificar los resultados visibles porque omite candidatos ya procesados, y la fuente externa puede cambiar con el tiempo. Por ello, una comparación entre modelos requiere controlar el conjunto de candidatos y el estado inicial de la memoria.

```
nuevos = ejecutar_agente(
    coleccion_id="6APD6L29",                              # Bibliometria_Forense (sección 3.3)
    tema="bibliometría forense e integridad científica",
    categorias=CATEGORIAS_FORENSE,
    n_keywords=10,
)
```

Salida registrada

```
Fase 1: Extrayendo metadata completa de Zotero...
Fase 2: El LLM esta construyendo la estructura robusta...
Estructura guardada. Top 10 keywords listas.
Fase 3: Buscando en OpenAlex con radar ampliado...
Buscando: "sneaked references" AND "citation manipulation"
  5 candidato(s): 5 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "authorship for sale" AND "paper mills"
  + Encontrado: Integrity at stake: confronting “publish or perish” in the developing ...
  5 candidato(s): 4 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "reference list manipulation" AND "peer review"
  5 candidato(s): 5 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "citation cartels" AND "anomalous patterns"
  5 candidato(s): 3 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "fabricated reference metadata" AND PubMed
  + Encontrado: High prevalence of articles with image-related problems in animal stud...
  + Encontrado: Fabricated or accurate? Ethical concerns and citation hallucination in...
  5 candidato(s): 1 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "author-centric reference manipulation" AND detection
  5 candidato(s): 0 ya revisado(s), 5 descartado(s) por el LLM
Buscando: "coercive citation" AND "research integrity"
  5 candidato(s): 3 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "predatory publishing" AND "citation inflation"
  + Encontrado: Bibliometric Assessment of a Retracted Paper on a Psychometric Assessm...
  3 candidato(s): 2 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "bibliometric forensics" AND "statistical detection"
  5 candidato(s): 0 ya revisado(s), 5 descartado(s) por el LLM
Buscando: "research misconduct" AND "publication ethics"
  + Encontrado: Explosion of formulaic research articles, including inappropriate stud...
  5 candidato(s): 2 ya revisado(s), 2 descartado(s) por el LLM
Proceso finalizado. Total nuevos: 5
```

El agente generó 10 frases de búsqueda y encontró 5 publicaciones nuevas para Bibliometria_Forense. Para cada frase indica cuántos candidatos devolvió OpenAlex, cuántos ya estaban en el historial y cuántos descartó el LLM. Si falta alguna celda de definición de las partes 8 a 13, se detiene al inicio e indica cuál.

14.1 Cambiar de colección y de tema

La temática no siempre será la misma. Para vigilar otra línea de investigación basta con indicar otra colección en la llamada. Si no se pasan categorías, el LLM las propone a partir de los documentos de esa colección; si no se pasa `tema`, se usa el nombre de la colección.

```
# Otra colección de la lista de la sección 3.3, por su nombre (ia-educacion)
# o directamente por su clave ("2XNUVRA7"). Una clave de documento, como AVTE4IA3, no sirve aquí.
nuevos_ia = ejecutar_agente(
```

```
    coleccion_id=clave_de_coleccion("ia-educacion"),
    tema="inteligencia artificial en educación",
    n_keywords=6,
)

# Varias colecciones seguidas, cada una con sus propios archivos e historial
for nombre in ["ia-educacion", "revision-sistematica"]:
    ejecutar_agente(coleccion_id=clave_de_coleccion(nombre), n_keywords=6)
```

Salida registrada

```
Fase 1: Extrayendo metadata completa de Zotero...
Fase 2: El LLM esta construyendo la estructura robusta...
Estructura guardada. Top 6 keywords listas.
Fase 3: Buscando en OpenAlex con radar ampliado...
Buscando: "artificial intelligence" AND "student trajectory" AND "higher education"
  + Encontrado: “Where is Agency Moving to?”: Exploring the Interplay between AI Techn...
  + Encontrado: Predicting Student Dropout from Day One: XGBoost-Based Early Warning S...
  + Encontrado: Predicting students’ academic progress and related attributes in first...
  5 candidato(s): 0 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "neural networks" AND "personalized learning" AND "engineering education"
  5 candidato(s): 4 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "student perception" AND "problem-based learning" AND "chat AI"
  5 candidato(s): 5 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "generative artificial intelligence" AND "academic integrity" AND ethics
  + Encontrado: Generative AI in higher education: A global perspective of institution...
  5 candidato(s): 4 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "artificial intelligence" AND "teacher training" AND "pedagogical models"
  + Encontrado: Educational Transformation Through Emerging Technologies: Critical Rev...
  5 candidato(s): 3 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "machine translation" AND abuse AND "EFL classroom"
  + Encontrado: Revolutionizing Language Learning: How ChatGPT and AI are changing the...
  + Encontrado: The Impact of AI Writing Tools on Academic Integrity: Unveiling Englis...
  + Encontrado: Automated versus peer assessment: Effects of learners' English public ...
  5 candidato(s): 0 ya revisado(s), 2 descartado(s) por el LLM
Proceso finalizado. Total nuevos: 8
Fase 1: Extrayendo metadata completa de Zotero...
Fase 2: El LLM esta construyendo la estructura robusta...
Estructura guardada. Top 6 keywords listas.
Fase 3: Buscando en OpenAlex con radar ampliado...
Buscando: "adaptive learning" AND "neural networks" AND "engineering education"
  + Encontrado: Advancements in Generative AI: A Comprehensive Review of GANs, GPT, Au...
  5 candidato(s): 3 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "artificial intelligence" AND "student trajectory" AND "higher education"
  + Encontrado: Enhancing academic performance of business students using generative A...
  5 candidato(s): 3 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "generative AI" AND "academic integrity" AND "university teaching"
  + Encontrado: Generative artificial intelligence in higher education: Evidence from ...
  5 candidato(s): 3 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "chat AI" AND "problem-based learning" AND "student perception"
  5 candidato(s): 5 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "intelligent tutors" AND "user acceptance" AND "university students"
  + Encontrado: Exploring AI-mediated informal digital learning of English (AI-IDLE): ...
  + Encontrado: Would ChatGPT-facilitated programming mode impact college students’ pr...
  + Encontrado: Modeling Teachers’ Acceptance of Generative Artificial Intelligence Us...
  + Encontrado: Generative Artificial Intelligence in Higher Education: Exploring Ways...
  + Encontrado: First-year students AI-competence as a predictor for intended and de f...
  5 candidato(s): 0 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "machine translation" AND "EFL classroom" AND "ethical challenges"
  + Encontrado: Artificial Intelligence in Translation: Challenges and Opportunities...
  + Encontrado: Advancing communicative competence in the digital age: A case for AI t...
  5 candidato(s): 2 ya revisado(s), 1 descartado(s) por el LLM
Proceso finalizado. Total nuevos: 10
Fase 1: Extrayendo metadata completa de Zotero...
Fase 2: El LLM esta construyendo la estructura robusta...
```

```
Estructura guardada. Top 6 keywords listas.
Fase 3: Buscando en OpenAlex con radar ampliado...
Buscando: "systematic review" AND "PRISMA protocol" AND "educational field"
  + Encontrado: The Educational Affordances and Challenges of ChatGPT: State of the Fi...
  + Encontrado: Generative artificial intelligence: Educational reflections from an an...
  5 candidato(s): 0 ya revisado(s), 3 descartado(s) por el LLM
Buscando: "artificial intelligence" AND "machine learning" AND "personalized learning"
  + Encontrado: Artificial Intelligence-Enabled Intelligent Assistant for Personalized...
  + Encontrado: AI -driven adaptive learning for sustainable educational transformatio...
  + Encontrado: Generative AI and the future of higher education: a threat to academic...
  5 candidato(s): 1 ya revisado(s), 1 descartado(s) por el LLM
Buscando: "digital age" AND "reading competence" AND "educational technology"
  + Encontrado: Comprensión lectora en la era digital: Una revisión sistemática...
  + Encontrado: Dynamic insights into research trends and trajectories in early readin...
  + Encontrado: Development of Interactive E-LKPD Based on Live-Worksheets for Reading...
  5 candidato(s): 0 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "virtual education" AND e-learning AND "inclusive education"
  + Encontrado: The Future of Education: A Multi-Layered Metaverse Classroom Model for...
  + Encontrado: COVID-19 and Digital Transformation in Higher Education Institutions: ...
  + Encontrado: Universal design for learning and artificial intelligence in the digit...
  + Encontrado: Methodological and Technological Advancements in E-Learning...
  + Encontrado: Accessible and inclusive online learning in higher education: a review...
  5 candidato(s): 0 ya revisado(s), 0 descartado(s) por el LLM
Buscando: "critical thinking" AND "skill development" AND education
  + Encontrado: AI in education: A review of personalized learning and educational tec...
  + Encontrado: The Artificial Intelligence Assessment Scale (AIAS): A Framework for E...
  5 candidato(s): 1 ya revisado(s), 2 descartado(s) por el LLM
Buscando: "Newcastle-Ottawa Scale" AND "selection bias" AND "risk of bias"
  5 candidato(s): 1 ya revisado(s), 4 descartado(s) por el LLM
Proceso finalizado. Total nuevos: 15
```

El agente se ejecutó sobre ia-educacion y después sobre las colecciones de la lista, cada una con sus propios archivos e historial; en la última encontró 15 publicaciones nuevas.

14.2 Revisar lo encontrado

Ordena los artículos nuevos por número de citas para empezar la lectura por los más influyentes.

Ordenar por citas ofrece una vista de los resultados, pero el recuento depende de la fuente y de la fecha de consulta. La prioridad de lectura debe considerar también la relación con la pregunta de investigación, el diseño del estudio y la actualidad de la evidencia. La lista es una entrada para revisión, no una selección final de calidad.

```python
for h in sorted(nuevos, key=lambda x: x["citas"], reverse=True):
    print(f"{h['citas']:>4} citas | {h['titulo'][:70]}  ({h['keyword']})")
```

Salida registrada

```
 112 citas | Explosion of formulaic research articles, including inappropriate stud  ("research misconduct" AND "publication ethics")
  27 citas | Integrity at stake: confronting “publish or perish” in the developing   ("authorship for sale" AND "paper mills")
   7 citas | High prevalence of articles with image-related problems in animal stud  ("fabricated reference metadata" AND PubMed)
   5 citas | Fabricated or accurate? Ethical concerns and citation hallucination in  ("fabricated reference metadata" AND PubMed)
   0 citas | Bibliometric Assessment of a Retracted Paper on a Psychometric Assessm  ("predatory publishing" AND "citation inflation")
```