UNIVERSIDAD TÉCNICA PARTICULAR DE LOJA
SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

SISTEMAS BASADOS EN CONOCIMIENTOS

Autor:

David Alejandro Burneo Valencia

David Josue Jiménez Jiménez

Karla Lizbeth Ochoa Ludeña

Docente:

Ing. Janneth Chicaiza

Tema:

PROYB1-2 PROCESO DE EXTRACCIÓN DE DATOS

Abr/2021 – Ago/2021

Documentación

Crossref API

Con la API de Crossref que se encuentra disponible públicamente expone los metadatos que los miembros depositan con Crossref cuando registran su contenido. Y tampoco son solo los metadatos bibliográficos: los datos de financiación, la información de licencia, los enlaces de texto completo, los iD de ORCID, los resúmenes y las actualizaciones de Crossmark también están en los metadatos de los miembros. Se puede buscar, facetar, filtrar o muestrear metadatos de miles de miembros, y los resultados se devuelven en JSON (Kemp, 2020).

Semantic Scholar API

Semantic Scholar proporciona una API RESTful para enlazar convenientemente a las páginas de Semantic Scholar y extraer información sobre registros individuales bajo demanda de una licencia de conjunto de datos. La API está disponible gratuitamente, pero aplica un límite de velocidad y responderá con el estado HTTP 429 "Demasiadas solicitudes" si se excede el límite (100 solicitudes por ventana de 5 minutos por dirección IP) (Scholar, s.f.).

Proceso de Extracción de datos

En el apartado de extracción de datos se procedió a desarrollar un Script en el lenguaje de programación Python, en el cual se plantea como meta lograr la extracción de datos empleando las APIS de Crossref y Semantic Scholar, con el fin de integrar la información en una base de datos con los datos disponibles.

Consulta

Como primer punto en el proceso de extracción de datos se tiene la consulta. Para lograr esto se procedió primeramente a emplear la API de Crossref para la consulta de artículos científicos relacionados con los casos de trombosis presentados con la vacuna para el Covid-19. Sin embargo, al realizar la búsqueda general se obtuvo mas de dos millones de resultados, por lo que se procedió a filtrar esta información en función del año (2021) y tipo de archivo (Jorunal Article).

Después de aplicar estos filtros se implementa un método “sort(´relevance´)”, en el cual se ordenan los resultados según la relevancia, y finalmente se extraen únicamente los DOI.

```
works = Works()
w1 = works.query(bibliographic='covid-19 vaccine thrombosis')\
    .filter(type='journal-article', from_pub_date='2021')\
    .sort('relevance')\
    .select('DOI')
```

Ilustración 1. Método de extracción de dois de Crossref

Seguidamente de obtener los doi de todas las publicaciones científicas que nos ofrece la base de datos de Crossref se procede a enviar esta información mediante la API de Semantic Scholar. Cabe recalcar que esta fuente de información solo retorna los metadatos a través de una solicitud especificando el doi del cual se desea obtener la información.

En vista de ello, se procede a utilizar el comando plasmado en la Ilustración 2, en la cual se esta enviando el doi y solicitando todos los metadatos que retorna esta API. Al mismo tiempo se añade un valor timeout, con el fin de que la conexión no se caiga de forma muy rápida cuando se pierde la conectividad con el servidor por un determinado número de segundos.

```
paper = sch.paper(article['DOI'], timeout=10)
```

Ilustración 2. Consulta mediante API Semantic Scholar

Extracción

Después de haber realizado los métodos de consulta y obtener la información retornada por las APIS, esta retorna en algo denominado en Python como diccionarios. Estos contienen toda la información de los metadatos disponibles en Semantic Scholar. Cabe recalcar que estos datos retornan de una forma muy limpia, por lo cual es prácticamente innecesario implementar procesos de limpieza de datos.

Seguidamente se procede a llamar al método “registerArticle” (Ilustración 3) el cual se encarga de leer el diccionario de información, extraer los datos requeridos y almacenarlos en otro diccionario ya segmentado según la información que se va a recuperar. A partir de aquí se llama a los métodos create_article (Ilustración 4) y create_authors (Ilustración 5) según sea el caso para generar los objetos necesarios y llenar el objeto tipo Article.

En el método denominado create_authors se procede a realizar una consulta aparte con los metadatos disponibles de los autores según el artículo científico que se encuentre analizando. Esto con el objetivo de conseguir toda la información posible en cuanto a autores se refiere.

Después que se han realizado correctamente los procesos de extracción de la información necesaria, se procede a crear el objeto Article con toda la información obtenida.

Ya con el objeto Article creado se procede a la carga de información mediante el método create_article.

```
def registerArticle(paper):
    container_paper = {
        'abstract': '',
        'arxivId': '',
        'citationVelocity': '',
        'corpusId': '',
        'doi': '',
        'fieldsOfStudy': '',
        'isOpenAccess': '',
        'isPublisherLicensed': '',
        'numCitedBy': '',
        'numCiting': '',
        'paperId': '',
        'url': '',
        'title': '',
        'year': 0,
        'authors':[]
    }
    author_list = []
    for key in container_paper.keys():
        if key in list(paper.keys()):
            if key == 'authors':
                for i in paper['authors']:
                    author_list.append(i)
                paper[key] = create_authors(author_list)
            container_paper[key] = paper[key]
    return create_article(container_paper)
```

Ilustración 3. Método registerArticle

```python
def create_article(paper):
    objPaper = Journal_Article(
        paper['abstract'], paper['arxivId'], paper['citationVelocity'], paper['corpusId'],
        paper['doi'], paper['fieldsOfStudy'], paper['isOpenAccess'], paper['isPublisherLicensed'],
        paper['numCitedBy'], paper['numCiting'], paper['paperId'], paper['title'], paper['url'],
        paper['year'], paper['authors']
    )

    print(objPaper)
    inst = DataBase()
    exist = inst.find_article(objPaper)
    if exist == True:
        inst.insert_journal(objPaper)
        idJournal = inst.get_id_journal(objPaper)
        inst.insert_authors(objPaper, idJournal)
        if(objPaper.get_fieldsOfStudy() != None):
            inst.insert_fieldsOfStudy(objPaper, idJournal)
```

Ilustración 4.Método create_article

```python
def create_authors(authors):
    authors_list = []
    for i in authors:
        a = sch.author(i['authorId'], timeout=10)
        if 'authorId' in list(a.keys()):
            objAutor = Author(a['authorId'], a['name'], a['url'],
                              a['influentialCitationCount'], a['aliases'])
            authors_list.append(objAutor)
    return authors_list
```

Ilustración 5. Método create_authors

Carga de datos

Para la carga de datos se implementa un modelo de entidad relación en la base de datos MySQL.

Para este procedimiento primero se realiza una consulta a la base de datos con el DOI únicamente para verificar si existe un campo igual. Con esto se realiza el control de evitar duplicados en los artículos científicos registrados (Ilustraciones 6, 7). Si pasa este control se procede a la carga de información tanto de los artículos como de los autores y las áreas de estudio mediante INSERTS. Con esto ya se tiene toda la información consultada en la base de datos. (Ilustración 8)

```python
    inst = DataBase()
    exist = inst.find_article(objPaper)
    if exist == True:
        inst.insert_journal(objPaper)
        idJournal = inst.get_id_journal(objPaper)
        inst.insert_authors(objPaper, idJournal)
        if(objPaper.get_fieldsOfStudy() != None):
            inst.insert_fieldsOfStudy(objPaper, idJournal)
```

Ilustración 6. Llamada a control de duplicados

```python
def find_article(self, article):
    try:
        sql = ("SELECT journalArticleId FROM Journal_Article WHERE paperId = '{}';".format(article.get_paperId()))
        print(sql)
        self.cursor.execute("SELECT journalArticleId FROM Journal_Article WHERE paperId = '{}';".format(article.get_paperId()))
        exist = self.cursor.fetchone()
        if(exist is None):
            return True
        else:
            return False
    except Error as ex:
        print("Error en la consulta find", ex)
```

Ilustración 7. Consulta SQL para verificar si el documento ya esta registrado

journalArticleId	abstract	arxivId	citationVelocity	corpusId	doi	isOpenAcces	isPublisherLicensed	numCitedBy	numCiting	paperId	title
1	NULL	NULL	0	233247658	10.1007/s40278-021-94091-4	0	1	0	0	4bfaf2c61d74e9f1be177efbaec21d7851135851	AstraZe
2	Acquires haemophilia A (AHA) is rare bleeding c...	NULL	0	232419057	10.1111/jth.15291	0	1	0	0	02c04323d1a1d014b78ce992beeb531726cf4430	A Case F
3	NULL	NULL	0	234364954	10.1016/j.vaccine.2021.05.017	0	1	0	17	e0ac083ac7dc1bbaaed4f7bf81397bf9956e4af2	Covid-1
4	2021 Elsevier Ltd. All rights reserved. The rapid...	NULL	0	232210823	10.1016/j.vaccine.2021.03.004	0	1	0	41	74aef9d3cb3608f7c452636a4f79104320028863	Develop
5	NULL	NULL	0	233617372	10.47260/JAMS/1011	0	0	0	0	98391179cfc881080aa5630d7975a0c89007d128	Oxford–
6	NULL	NULL	0	231304025	10.1053/j.gastro.2020.12.066	1	1	2	0	390407a244c33ba773f82cc106aea5db8a1533fd	Letter to
7	NULL	NULL	0	233786155	10.36959/669/746	0	0	0	0	3298f58d00fde4d5cab7755af0647c22554f5fe6	COVID-1
8	In absence of a COVID-19 vaccine, testing, co...	NULL	0	229689942	10.1016/j.vaccine.2020.12.032	0	1	5	48	8c3e2c8ac683713aa3054ed48694c5f8060322d5	Preferer
9	Bacille Calmette-Guérin (BCG) vaccine is prove...	NULL	0	229298389	10.1016/j.vaccine.2020.12.026	0	1	1	24	bdb13d99189e0b5b6eb0dba3e56c317b75f71ea9	BCG vac
10	Public health officials warn that the greatest b...	NULL	0	233374518	10.1016/j.vaccine.2021.04.044	0	1	0	65	632dc2df6a7d6d4ae236623291e56927e77ff70d	Factors
11	NULL	NULL	0	232327168	10.1016/j.thromres.2021.03.011	0	1	0	94	dead671ca254df9b8a8a2dd3cd4ee6592886384d	COVID-1
12	Background The novel coronavirus SARS-CoV-...	NULL	0	231628603	10.1016/j.thromres.2021.01.005	0	1	3	100	599c0922b788312417b4c314955a37eb925d6288	Mechani
13	Background: As of 8 April 2021, a total of 2.9 m...	NULL	0	233396149	10.3390/jcm10081599	0	1	0	35	3c001dc8a15b9b4481f67f07a467e3f10123fb61	Thrombo
14	NULL	NULL	0	[illegible]	[illegible]	0	1	0	17	[illegible]	COVID-

Ilustración 8. Metadatos disponibles en la base de datos

Solución de problemas

Caídas de Servidor

Una dificultad que se tuvo que atravesar durante la ejecución del Script fueron las contantes caídas del servidor, con lo cual se presentaba un error en la ejecución del código con muy pocos datos recolectados (Ilustración 9).

```
  File "C:\Users\Davicho\Documents\EntornosVirtuales\SBC\lib\site-packages\requests\adapters.py", line 529, in send
    raise ReadTimeout(e, request=request)
requests.exceptions.ReadTimeout: HTTPSConnectionPool(host='api.semanticscholar.org', port=443): Read timed out. (read timeout=10)
```

Ilustración 9. Error referenciado a caida de servidor

Para dar solución a este problema se implemento una función similar al Do While de otros lenguajes de programación, en el cual se verificaba si no han existido errores en la ejecución o si han retornado muchos diccionarios de datos en blanco. Si esto ocurría se implementaba una función sleep la cual se encargaba de pausar la ejecución del código determinado número de segundos hasta que se vuelva a obtener información del servidor (Ilustración 10).

```python
bool = False
while bool == False:
    paper = sch.paper(article['DOI'], timeout=10)
    if 'title' in paper:
        registerArticle(paper)
        a = a + 1
        print(a)
        bool = True
        break
    else:
        print('waiting %s' % contador)
        time.sleep(sleeps)
        bool = False
        contador = contador + 1
        if contador > 2:
            bool = True
            break
```

Ilustración 10. Control de caida de servidor

Esquema de especificación

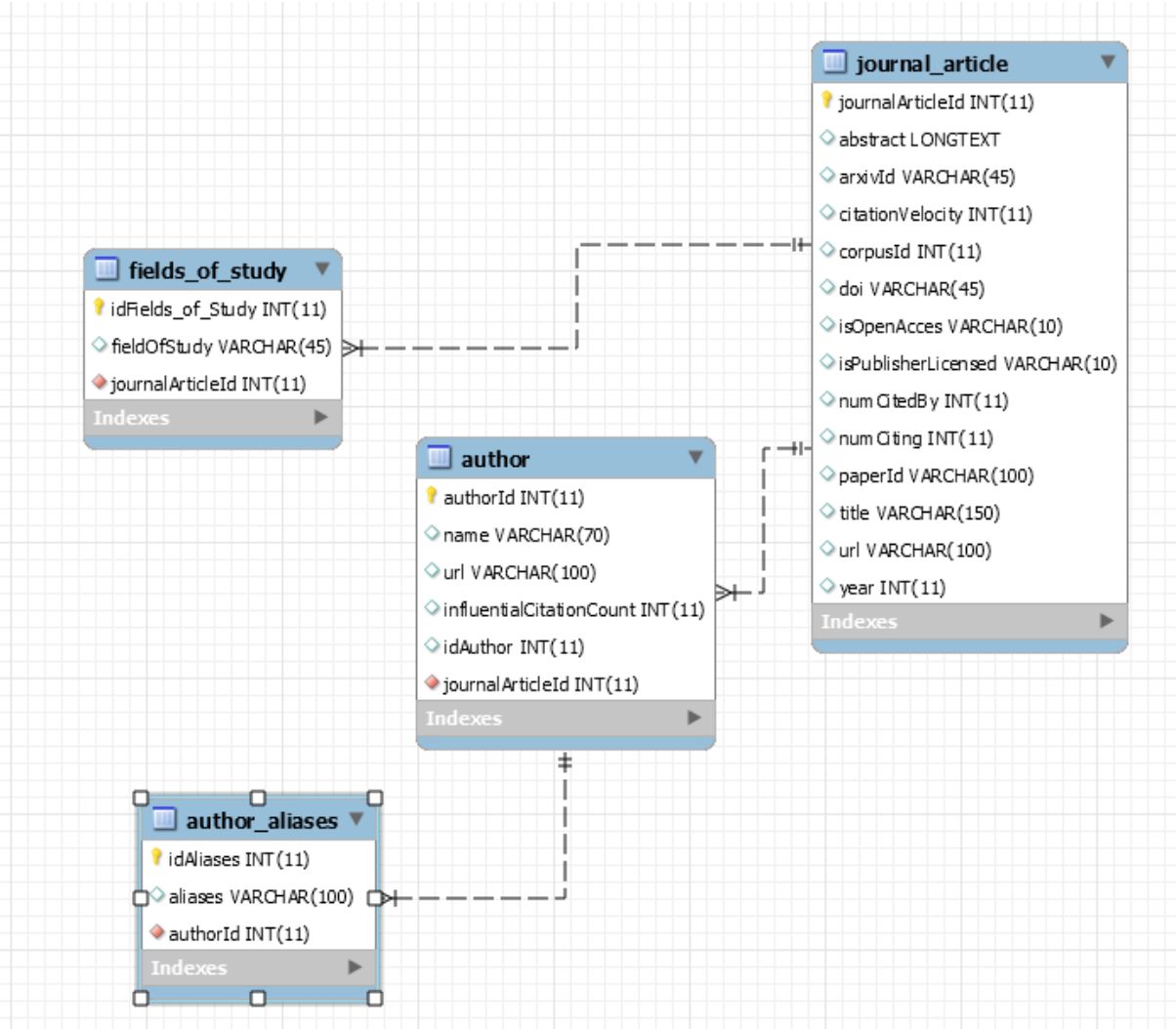
Crossref

1. ESPECIFICACIÓN DE FUENTES DE DATOS			
1.1 Identificación de Fuentes			
Nombre de la Fuente de Datos:	Crossref		
Proveedor:	Repositorio de acceso abierto	**Sitio Web:**	https://www.crossref.org
Institución:	Publishers International Linking Association, Inc.		
Descripción de la fuente de datos:	Es la agencia más grande de registros de DOI; es una asociación de editoras científicas que promueve el uso de las nuevas tecnologías para mejorar la comunicación y la investigación científica; trabaja en la mejora de la identificación de las		
Tamaño del archivo de datos (MB)	106 MB		
Licencia	Creative Commons Attribution 4.0 International License		
Formato del archivo:	sql		
Incluye información de los metadatos	Si		
1.2 Volumetría de datos			
Entidad[atributos]	**Cantidad de Registros**	**Descripción**	
doi	83067	Identificador único de un artículo científico.	
url	83067	URL del artículo científico.	
created	83067	Fecha de creación del artículo científico.	
author	80469	Nombre de autor o autores del artículo científico.	
title	83064	Título del artículo científico.	
abstract	30284	Resumen del artículo científico.	
publisher	83049	Fecha de publicación del artículo científico.	
pages	60597	Número de páginas que contiene el artículo científico.	
volume	62915	Número de tomos o partes que constituyen el artículo científico.	
container_title	83067	Título del contenedor que abarca los artículos científicos.	
license	83067	Campo que verifica si el articulo científico tiene licencia o no	
score	83067	Número de versión del artículo científico.	
1.3 Obtención de datos			
Método de extracción de datos	Construcción de script mediante API de Crossref		
En caso de requerir construir un script o app de extración de datos, indicar detalles como lenguaje de prog., descripción de lo que realizó el programa	Lenguaje de Programación: Python	Se realizó un Script en Python bajo la metodología de programación orientada a objetos. Se consumió la API de Croosref mediante la librería "crossref", posterior a esto se procedió a realizar la busqueda de los articulos científicos, se filtró la información obtenida mediante parámetros como año y tipo de documento. Seguidamente se creó el objeto tipo Articulo y se procede a su almacenamiento en la base de datos.	
Base de datos en la que guardó los resultados	Mysql		

Semantic Scholar

1. ESPECIFICACIÓN DE FUENTES DE DATOS			
1.1 Identificación de Fuentes			
Nombre de la Fuente de Datos:	Semantic Scholar		
Proveedor:	Repositorio de acceso abierto	**Sitio Web:**	https://www.semanticscholar.org
Institución:	Allen Institute for AI.		
Descripción de la fuente de datos:	Semantic Scholar es un motor de búsqueda respaldado por un sistema de inteligencia artificial dedicado a la investigación de publicaciones académicas. Utiliza los últimos avances en el procesamiento del lenguaje natural para proporcionar resúmenes de		
Tamaño del archivo de datos (MB)	19,8 MB		
Licencia	Acuerdo de licencia de conjunto de datos y Api de Semantic Scholar http://s2-public-api-prod.us-west-2.elasticbeanstalk.com/corpus/legal/		
Formato del archivo:	sql		
Incluye información de los metadatos	Si		
1.2 Volumetría de datos			
Entidad[atributos]	**Cantidad de Registros**	**Descripción**	
id	1481	Identificador único tipo llave primaria para la base de datos, resaltando el numero total de publicaciones	
abstract	866	Resumen del artículo científico.	
citationVelocity	1481	Velocidad de citación.	
corpusId	1481	Identificador de un artículo de la base de datos de Semantic Scholar.	
doi	1481	Identificador único de un artículo científico.	
isOpenAcces	1481	Campo que verifica si el artículo científico es de acceso abierto.	
isPublisherLicensed	1481	Campo que verifica si el articulo científico tiene licencia o no	
numCitedBy	1481	Número de citas.	
numCiting	1481	Número de citas del artículo científico.	
paperId	1481	Identificador de un artículo en la base de datos de Semantic Scholar.	
title	1481	Título del artículo científico.	
url	1481	URL del artículo científico.	
year	1480	Año de publicación del artículo científico.	
fieldOfStudy	1251	Campo de estudio que pertenece el artículo científico.	
Author_name	4693	Nombre de autor o autores del artículo científico.	
Author_url	4693	URL del autor.	
Author_influentialCitationCount	4693	Recuento de citas influyentes del autor	
Author_idAuthor	4693	Identificador del autor.	
Author_aliases	14508	Alias del autor.	
1.3 Obtención de datos			
Método de extracción de datos	Construcción de script mediante API de Semantic Scholar		
En caso de requerir construir un script o app de extración de datos, indicar detalles como lenguaje de prog., descripción de lo que realizó el programa	Lenguaje de Programación: Python	Se realizó un Script en Python bajo la metodología de programación orientada a objetos. Se consumió la API de Semantic Scholar mediante la librería "semanticscholar", posterior a esto se procedió a realizar la busqueda de los articulos científicos basandonos en los doi extraidos previamente de Crossref. Luego se procede a extraer los metadatos del articulo cientifico, se implementa un control para evitar duplicados en la base de datos y otro control para evitar caidas del servidor. Se obtiene información tanto del articulo científico como del autor	
Base de datos en la que guardó los resultados	Mysql		

Modelo Integrado



Bibliografía

Kemp, J. (08 de 04 de 2020). *Crossref*. Obtenido de API REST: https://www.crossref.org/education/retrieve-metadata/rest-api/

Scholar, S. (s.f.). *SemanticScholar*. Obtenido de API Académica Semántica: https://api.semanticscholar.org