UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Detección de registros académicos duplicados obtenidos desde repositorios digitales

AUTORES: Soloaga Ignacio

DIRECTOR: Dra. De Giusti Marisa Raquel

CODIRECTOR:

ASESOR PROFESIONAL: Lic. Lira Ariel Jorge

CARRERA: Licenciatura en Sistemas

Resumen

Esta tesina de grado detalla el análisis y la implementación de una herramienta para la detección de registros académicos duplicados basada en un sistema de reglas. La deduplicación de registros es una tarea clave en el proceso de ingesta masiva de documentos a un repositorio puesto que permite el filtrado de contenido duplicado. Además, permite enriquecer los metadatos de los registros existentes en las distintas fuentes. Adicionalmente se presenta el desarrollo de un módulo de mapeo de metadatos que da soporte al proceso de deduplicación de registros y permite establecer interoperabilidad entre los esquemas utilizados en las distintas fuentes.

Palabras Clave

Repositorios digitales, Deduplicación de registros, Mapeo de metadatos, Importación masiva, Interoperabilidad, Registros académicos

Conclusiones

Se analizaron técnicas de deduplicación presentes en la literatura, y se optó por un modelo basado en reglas y funciones de distancia. Los desarrollos realizados y mejorados en las distintas iteraciones permitieron disminuir considerablemente el tiempo en que los recursos digitales son preservados y difundidos en el repositorio digital durante un proceso de ingesta masiva. Estos desarrollos pueden aplicarse sobre múltiples escenarios y fuentes de datos distintas.

Trabajos Realizados

Se desarrolló una herramienta basada en un sistema de reglas que permite encontrar registros académicos duplicados en repositorios digitales. Esta misma ofrece un módulo para el mapeo de metadatos a partir de archivos de configuración reutilizables. El desarrollo de la herramienta se dividió en dos componentes, una librería que concentra los módulos principales y una aplicación web que provee una interfaz de usuario. La herramienta fue puesta a prueba en procesos de ingesta masiva de registros al repositorio SEDICI y los resultados fueron muy favorables.

Trabajos Futuros

Una de las líneas de trabajo a futuro consiste en la optimización de los tiempos de ejecución de la herramienta. También será necesario definir un proceso para el enriquecimiento de metadatos de los registros detectados como duplicados. Por otro lado, se explorará el uso de aprendizaje automático en este contexto para comparar con los resultados obtenidos. Finalmente, se buscará la automatización de la deduplicación en repositorios digitales mediante la incorporación de esta herramienta como un módulo de software de repositorios (por ej. DSpace).

Fecha de la presentación: Noviembre 2020

Detección de registros académicos duplicados obtenidos desde repositorios digitales

Capítulo 1 - Introducción

Motivación

La Universidad Nacional de La Plata (UNLP) cuenta con un repositorio central, SEDICI [https://sedici.unlp.edu.ar], en el que se depositan obras producidas por todas las áreas de la institución. Estas obras son producidas por docentes, investigadores, becarios y personal administrativo, que en muchos casos también han depositado documentos en repositorios temáticos como ser Arxiv, PubMed Central o REPEC, en repositorios de instituciones externas donde ellos o miembros de sus grupos de trabajo realizan sus actividades académicas y científicas, como ser CONICET DIGITAL o CIC-Digital, o repositorios internos de la propia UNLP, como por ejemplo Memoria Académica (Facultad de Humanidades y Ciencias de la Educación) o Naturalis (Facultad de Ciencias Naturales y Museo). Actualmente existen más de 4700 repositorios digitales alrededor del mundo [ROAR, 2020] por lo que la probabilidad de que las obras producidas por una institución estén disponibles en internet es muy alta.

Resulta entonces de gran interés para una institución recopilar en su repositorio institucional las obras de sus miembros que se encuentran dispersas en la amplia red de repositorios mencionada en el párrafo anterior, sin requerir a los autores realizar nuevamente el depósito de estas obras. Existen mecanismos de interoperabilidad que permiten el intercambio de obras entre repositorios, algunos son automáticos y otros semi-automáticos, algunos están basados en estándares como por ejemplo el protocolo OAI-PMH y otros no, pero ninguno de estos mecanismos es completo. La recuperación e ingesta de producción científica a un repositorio es compleja, y estos mecanismos son sólo eslabones o herramientas que forman parte del proceso completo, que involucra tareas de normalización, mapeo y deduplicación de registros de metadatos. Este proceso se divide en distintas etapas [De Giusti, Lira y Oviedo; 2011] que varían en complejidad y cantidad según el tipo de repositorio, la tipología de los documentos y el origen de los datos. Es importante entonces definir un flujo de trabajo débilmente acoplado donde las etapas del mismo se adapten al caso de uso concreto. Este escenario se hace presente en repositorios institucionales de todo el mundo, por lo que esta experiencia podría ser de utilidad para muchas otras instituciones.

Una de las etapas mencionadas en el párrafo anterior, y en la que se centrará mayormente el desarrollo de esta tesina, es en la detección de registros duplicados (deduplicación) en distintos repositorios. Cuando esta tarea se realiza sobre grandes cantidades de registros se torna muy costoso, puesto que buscar manualmente uno por uno en el repositorio no es viable y la adopción de identificadores universales en los repositorios es muy escasa. Asimismo, asegurar que no se incluirán en la importación documentos ya existentes en el repositorio favorece enormemente a la calidad del contenido, y por consecuencia, al aumento de la visibilidad de sus obras [Knoth, 2012].

Objetivos

Objetivo general

Desarrollar una herramienta de análisis de metadatos para detectar registros duplicados entre repositorios digitales.

Objetivos secundarios

- Definir un flujo de trabajo para la recuperación, limpieza, deduplicación e ingesta masiva de documentos[1] a un repositorio digital.
- Desarrollar una herramienta de mapeo de metadatos para dar soporte a las distintas etapas del proceso.
- Permitir encontrar duplicaciones de registros en un repositorio.
- Mejorar la cantidad y calidad de metadatos de registros existentes en un repositorio.
- Aumentar la cobertura de obras de autores de la UNLP en el repositorio institucional SEDICI mediante la incorporación de sus obras.
- Recuperar y repatriar contenido generado en el ámbito de la UNLP que se encuentra disperso en la web.

Escenario de trabajo

SEDICI es el repositorio institucional de la Universidad Nacional de La Plata y alberga producción científica y académica de la institución y de todos sus integrantes, incluyendo investigadores, docentes, becarios de grado y postgrado, y alumnos. El repositorio está implementado sobre DSpace (https://duraspace.org/dspace/), un software que provee herramientas para la administración de colecciones digitales, y comúnmente es usada como solución de repositorio bibliográfico institucional. Desde noviembre de 2020 contiene más de 100.000 recursos.

La carga de publicaciones al repositorio puede hacerse a partir de múltiples vías:

1. Autoarchivo: un autor de la UNLP deposita cada obra realizando la carga manualmente y completando en el proceso un conjunto de campos básicos necesarios para la catalogación, por ejemplo título, nombre completo del autor, fecha de exposición o publicación, congreso o evento en el que fue presentado, revista en la que fue publicado, grado alcanzado cuando se trata de tesis, y licencia Creative Commons (http://www.creativecommons.org.ar/contact/) bajo la cual desea publicar su obra, entre otros metadatos.
2. Carga desde administración: la misma consta de la carga manual de publicaciones, por el personal encargado de SEDICI, que fueron obtenidas como resultado de procesos

[1] En el contexto de esta tesina, se utilizará la palabra documento para referirse al recurso completo que se importa a un repositorio: objeto digital y registro de metadatos asociado.

internos al repositorio, entre los cuales se destacan: solicitud de obras a los autores, obtención de publicaciones a partir de los portales web de revistas o congresos, digitalización y/o depósito de obras prestadas por alguna institución interna a la UNLP, o revisión y aprobación de obras cargadas por autores mediante autoarchivo.

3. Depósito semiautomático desde servicios de la UNLP: se realiza el depósito de obras pertenecientes a la UNLP de forma semiautomática a través del protocolo SWORD. SWORD es [SWORD, 2019] un estándar de interoperabilidad utilizado por los repositorios digitales para recibir depósitos de contenido desde múltiples fuentes. Esto permite enviar documentos para ser preservados y difundidos por los repositorios digitales desde herramientas externas al repositorio, como por ejemplo un sistema de gestión de publicaciones periódicas. Con el uso de este protocolo, DSpace permite la configuración de clientes que periódicamente suben contenido nuevo al repositorio.
4. Importación masiva a partir de cosechas: se realizan cosechas e importaciones periódicas de registros de metadatos expuestos por fuentes de la UNLP a través del protocolo OAI-PMH. Este protocolo permite la interoperabilidad entre fuentes de datos a partir de la exposición y cosecha de registros de metadatos.
5. Importación masiva por parte de un administrador: consiste en la carga al repositorio de un lote de documentos (metadatos y objeto digital) para que las mismas sean incorporadas de forma automática o semi-automática según el caso. Los procesos de importación permiten la carga de grandes cantidades de documentos obtenidos a partir de distintas fuentes y post procesados para cumplir con los requisitos técnicos de importación en el software de repositorio utilizado.

Ingesta masiva de registros

Muchas veces los autores de publicaciones científicas realizan la carga de las mismas en el repositorio de su centro de investigación, en repositorios temáticos reconocidos y seguramente en los repositorios que corresponda por la colaboración con autores de otras instituciones, sin realizar la carga en el propio repositorio institucional. Para imaginar mejor esta situación, se puede tomar como ejemplo a un investigador de física de altas energías, que es a su vez docente-investigador de la UNLP. Al ser investigador superior del CONICET y trabajar junto al CERN (Consejo Europeo para la Investigación Nuclear), cada una de sus publicaciones debe ser cargada en el repositorio de su centro de investigación CONICET Digital, y seguramente también se encuentre en el repositorio de datos del CERN Zenodo (https://zenodo.org/), en la revista donde haya sido publicado el artículo, en el repositorio temático especializado arXiv.org que suelen utilizar los investigadores de estas áreas de la ciencia, y en todos aquellos repositorios de las instituciones a las que pertenecen los autores con las que este autor haya publicado sus obras.

Cuando un repositorio institucional desea agregar al repositorio la producción científica de sus investigadores, que todavía no ha sido cargada al mismo, pero que ya se encuentra presente en múltiples repositorios y/o revistas, se encuentra con situaciones como la mencionada anteriormente. Es por esto que resulta necesario un proceso de recuperación y deduplicación de registros, que permita la incorporación de producción que debe estar cargada

en el repositorio por las políticas del mismo, evitando la incorporación de documentos duplicados, y evitando a los autores el proceso repetitivo de carga en múltiples sistemas.

En determinadas situaciones, puede resultar necesaria la carga masiva de documentos, con el objetivo de poblar el repositorio y fortalecer el contenido que el mismo alberga.

La ingesta masiva de registros en un repositorio digital consiste en la recuperación, el procesamiento y la importación de una gran cantidad de documentos (registros de metadatos con objeto digital asociado) de forma automática. Cada una de estas etapas a su vez se subdivide en distintas tareas cuya complejidad varía dependiendo el origen de los datos, la manera en la que los mismos son obtenidos, la estructuración y completitud de los datos, y las cantidades de registros sobre las cuales se va a trabajar.

A modo de ejemplo, una importación masiva típica al repositorio SEDICI podría dividirse en los siguientes pasos:

1. Obtener los registros vía interfaz OAI-PMH
2. Mapear los metadatos del esquema actual a un formato genérico
3. Detectar registros ya presentes en SEDICI y filtrarlos
4. Filtrar registros que tengan licencias de acceso abierto
5. Mapear los metadatos al perfil de metadatos de SEDICI
6. Corregir estructura de los metadatos
7. Obtener los archivos PDF asociados a cada registro
8. Transformar a PDF/A[2] en caso de ser necesario
9. Generar paquete de importación
10. Importar el paquete a SEDICI (DSpace)

Una de las principales distinciones que tiene la importación de otros métodos de carga donde se sube de a un documento a la vez, mediante una interfaz de usuario, es que los mismos pueden ser cargados sin ningún tipo de revisión humana antes de ser asignados a una colección dentro del repositorio. Para evitar esto, puede optarse por llevar todos los documentos cargados en una importación a *workflow*, lo que obliga a un usuario administrador a aceptar cada uno individualmente. De esta manera, se puede realizar una etapa previa de verificación de correctitud, pero aumenta considerablemente el tiempo en el que los documentos son subidos y disponibilizados.

Para agilizar o incluso evitar esta etapa de revisión de registros es necesario realizar un conjunto de tareas previas a la carga, que consisten en la limpieza, la normalización y la detección de registros duplicados, asumiendo siempre que se cuenta con el listado de registros de metadatos y objetos digitales asociados a importar. De no ser así, se suma una etapa inicial que es la de recuperación de los registros. Cada etapa está compuesta por un conjunto de tareas, y cada una de estas presenta distintas dificultades y persigue distintos objetivos, que serán vistos más adelante.

Las ventajas que ofrecen las ingestas masivas de documentos a un repositorio digital son muchas y en el contexto del SEDICI, pueden mencionarse principalmente las siguientes:

[2] PDF/A es un formato de archivo para la preservación de documentos electrónicos.

- Crecimiento rápido del contenido dentro del repositorio: se disponibiliza una gran cantidad de publicaciones de forma inmediata y, en muchos casos, automática.
- Aumento del impacto y visibilidad de las obras UNLP: se recopila una gran cantidad de obras de autores de la UNLP que se encuentran dispersas por todo internet y se las agrega al repositorio.
- Incorporación de obras UNLP sin que el autor tenga que realizar autoarchivo: cuando la carga de obras propias debe realizarse en múltiples repositorios, el autoarchivo en cada uno de ellos puede resultar en un proceso tedioso para el autor, porque se deben completar los mismos campos una y otra vez en cada repositorio. Este escenario se ataca a partir de la ingesta masiva de documentos.

Estructura de la tesis

Con esta sección finaliza el capítulo 1 de esta tesis de grado, y a continuación se detalla la organización del resto del documento. En el capítulo 2 se introducen conceptos relacionados a las fuentes de producción científico-académica, los repositorios digitales en el contexto nacional e internacional, y se presentan conceptos relacionados a metadatos e interoperabilidad. En el capítulo 3 se introduce el problema de detección de registros duplicados en general, y a continuación se pone el foco en el caso específico de registros académicos duplicados. En este capítulo se mencionan también técnicas existentes para la detección de registros duplicados y se presentan las soluciones existentes, donde se hace evidente la necesidad de desarrollar una solución propia. El capítulo 4 retoma las técnicas y problemas descritos en el capítulo anterior, e introduce un análisis de los aspectos y herramientas a tener en cuenta a partir de dichas experiencias previas. Asimismo, este capítulo presenta los principales elementos y características de los desarrollos propuestos en el marco de este trabajo, lo que involucra el diseño e implementación de una herramienta de deduplicación de registros, y el diseño e implementación de un módulo de mapeo de metadatos para dar soporte a las distintas etapas involucradas. En el capítulo 5 se detalla la implementación de la herramienta como aplicación web y se exponen las distintas interfaces de usuario diseñadas. Estos desarrollos son puestos a prueba en el capítulo 6, donde se detallan los resultados obtenidos y se exponen casos de aplicación práctica de las herramientas. En este capítulo se propone también un proceso basado en el patrón arquitectural *Extract, Transform and Load (ETL)*. Finalmente, en el capítulo 7 se presentan las conclusiones obtenidas a partir del desarrollo y del uso de las herramientas propuestas en este trabajo de tesis y se introducen trabajos futuros que pueden surgir a partir del mismo.

Capítulo 2 - Marco teórico

Introducción

La producción científico-académica es depositada[3] y publicada en múltiples lugares: repositorios y bases de datos, revistas electrónicas, editoriales y archivos. Este depósito puede ser realizado por los autores de las obras, por personas responsables del proceso editorial (editores de revistas, organizadores de congresos, etc.) o por instituciones (por ejemplo una dirección o una secretaría), y en más de una ocasión el depósito es realizado por varios autores de manera independiente y en distintos lugares. Por ejemplo, si una obra tiene 3 autores de distintas instituciones, cada autor podría depositar la misma obra en el repositorio digital de su institución. A su vez, estas obras son recolectadas por agregadores de contenido, que las exponen y sirven como punto de acceso uniforme a los portales de las distintas fuentes. Uno de los tipos de fuentes de producción académica más comunes son los repositorios digitales, que se dividen en repositorios institucionales y repositorios temáticos (por ejemplo relacionados a la física, la economía, las ciencias de la computación, etcétera). El material cargado en este tipo de sistemas se compone comúnmente de dos elementos: uno o más objetos digitales por un lado, como por ejemplo una tesis o un artículo en formato PDF, una entrevista en formato de audio o una exposición de un trabajo en formato de video (entre otros tipos de recursos), y un registro de metadatos por el otro lado, que describe información acerca del objeto digital.

Para incrementar la visibilidad de los documentos que gestiona, cada fuente de datos establece mecanismos de interoperabilidad con otros sistemas (repositorios, bases de datos, agregadores) que le permiten incorporar nuevos materiales y disponibilizar el material propio en otros espacios. Este cruce continuo de información entre distintos sistemas genera una gran cantidad de datos duplicados distribuidos a lo largo de múltiples fuentes y además en muchos casos se pierden las referencias de un mismo documento publicado en distintos lugares, lo que dificulta la identificación de estos duplicados. El objetivo principal de esta tesis es trabajar sobre este problema y proponer una solución al mismo a partir del análisis de los metadatos expuestos para cada objeto digital depositado en los múltiples repositorios disponibles en internet. A lo largo de este capítulo se introducen algunos conceptos y definiciones necesarias para contextualizar el trabajo de esta tesis y reconocer, en este sentido, la importancia de la detección de registros duplicados, ahondar en las múltiples vías de interoperabilidad que existen actualmente en el contexto de los repositorios, describir los problemas existentes y mostrar la necesidad de buscar una solución a los mismos.

[3] En el contexto de este trabajo, el término depósito hace referencia al envío de una obra por parte de un responsable de la misma (autor, editor, compilador, etc.) hacia una organización responsable de su curaduría, como por ejemplo un repositorio institucional.

Repositorios digitales

Un repositorio es un espacio centralizado donde se almacena de forma organizada información digital, en su mayoría producción científica y/o académica de investigadores e instituciones (Repositorio digital, s.f.). Se define también a un repositorio digital como una infraestructura web capaz de brindar un conjunto de servicios a una comunidad, destinados a recopilar, gestionar, difundir y preservar contenidos a través de una colección organizada y accesible en abierto, y que permita la interoperabilidad con otros repositorios similares (De Giusti, Lira, Villarreal y Texel, 2012).

Cada objeto digital depositado en un repositorio está descrito por un registro de metadatos que guarda información acerca del mismo, y que permite la clasificación de los recursos en base a su contenido, tipología, relación con eventos, libros o instituciones, entre otras formas.

Típicamente, estos objetos digitales ingresan al repositorio a través de las distintas vías:

- Autoarchivo: el autor realiza la carga de sus obras mediante la interfaz web del repositorio.
- Carga desde administración: se realiza la carga de obras obtenidas a partir de la digitalización y/o cooperación con otras áreas de la institución.
- Depósito vía SWORD: SWORD es un estándar de interoperabilidad que permite a repositorios recibir de forma automática depósitos de contenido a partir de múltiples fuentes externas al mismo.
- Importación: se realiza ingesta masiva de registros de metadatos recuperados a partir de OAI-PMH u otras fuentes, junto con los objetos digitales asociados, a través del módulo de importación del repositorio.

Repositorios institucionales

Un repositorio institucional (Repositorio institucional, s.f.) es un tipo de repositorio digital en donde se depositan materiales derivados de la investigación y producción científica o académica de una institución (universidades, centros de investigación, facultades, etc.). Estos suelen incluir tesis, artículos científicos, ponencias y objetos de conferencia, revistas electrónicas institucionales y material docente producido por profesores y/o investigadores pertenecientes a la institución. Sus principales objetivos son (De Giusti et al., 2015) facilitar el acceso a estas producciones, aumentar la visibilidad de la producción científica de la institución, asimismo el de preservar los documentos almacenados para asegurar su accesibilidad y uso a largo plazo. Estos repositorios ofrecen un acceso uniforme a la información de la institución y de sus autores, y para ello utilizan estándares de catalogación, preservación e interoperabilidad de recursos que favorecen su recuperación, distribución y accesibilidad en el tiempo. El material alojado en estos repositorios se distribuye junto a sus licencias, que especifican a los lectores los usos permitidos.

Repositorios institucionales en Argentina

En 2011 se crea el Sistema Nacional de Repositorios Digitales (SNRD) (Resolución 469/11 MINCyT, 2011) que tiene como propósito (Azrilevich y De Giusti, 2019) conformar una red interoperable de repositorios digitales en ciencia y tecnología, a partir del establecimiento de políticas, estándares y protocolos comunes a todos los integrantes del Sistema. Uno de los objetivos del SNRD es brindar servicios sobre los registros recolectados de los repositorios nacionales adheridos, y que logra a través de la cosecha mediante OAI-PMH a cada uno de los repositorios.

Para que un repositorio pueda ser cosechado, el SNRD exige al mismo el cumplimiento de un conjunto mínimo de directrices que involucran a los metadatos expuestos. Estas directrices (Directrices SNRD, 2015) requieren la adopción de estándares y prácticas acordadas, facilitando la inclusión de los repositorios digitales adheridos a redes internacionales de repositorios, como por ejemplo la Red de Repositorios de Acceso Abierto a la Ciencia de Latino América LAReferencia [http://www.lareferencia.info/es/].

Una de las Directrices SNRD que deben cumplir los metadatos de un repositorio digital es la exposición de los mismos bajo el esquema Dublin Core y respetando la categorización de cada metadato en el campo y formato correspondiente, según se define en la sección de Revisión de las Directrices SNRD. La exposición de los metadatos en este formato es independiente de los esquemas o las estructuras utilizadas para catalogar a los mismos dentro del repositorio.

En el año 2013 surge la Ley Nº 26.899 (Ley Nº 26.899, 2013) de Repositorios digitales institucionales de acceso abierto, que establece que todos los organismos e instituciones públicas que componen al Sistema Nacional de Ciencia, Tecnología e Innovación (SNCTI) y que reciben financiamiento del Estado Nacional, deben desarrollar repositorios digitales institucionales de acceso abierto en los que se depositará la producción científico-tecnológica resultante del trabajo de sus investigadores. Además se definen obligaciones para que las instituciones aprueben políticas de acceso abierto a la producción científica en cuanto a la obligatoriedad de los investigadores de depositar en el repositorio institucional.

Actualmente, el SNRD se compone de 44 repositorios digitales de 44 instituciones diferentes, y cuenta con más de 310.000 publicaciones agregadas. Dentro de los repositorios institucionales que tienen mayor cantidad de producción científica agregada se encuentran CONICET Digital y SEDICI.

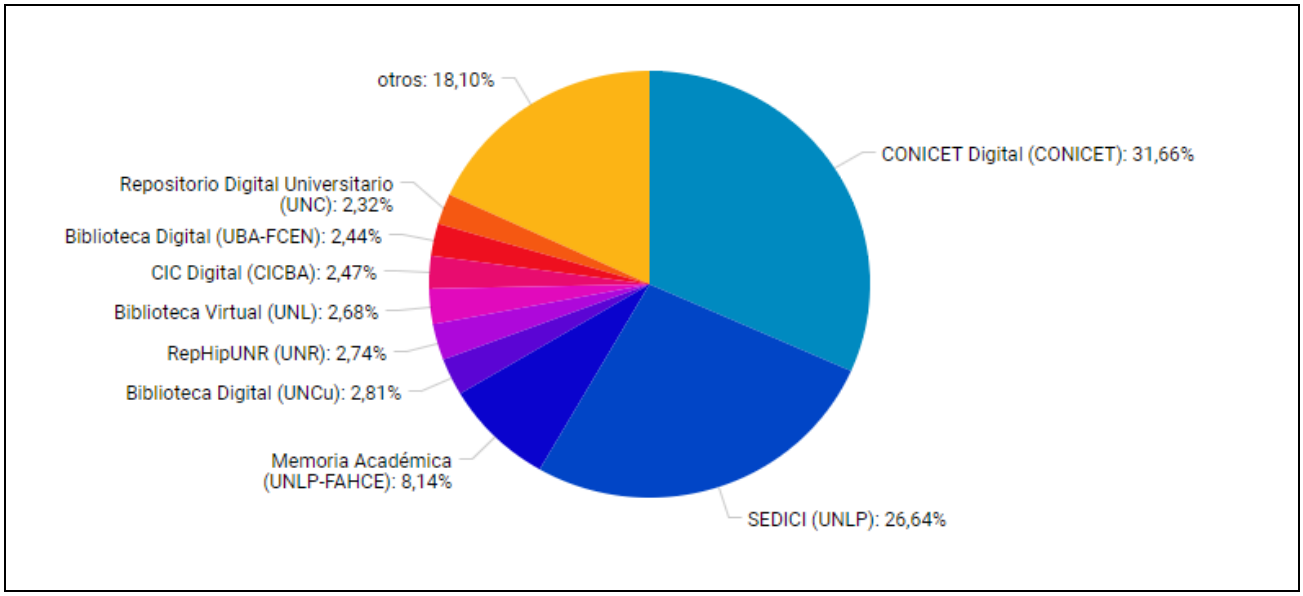

Figura 1. Cantidad de publicaciones según repositorio
(Fuente: https://repositoriosdigitales.mincyt.gob.ar/vufind/Content/stats)

Metadatos

Los metadatos pueden definirse como datos que describen a otros datos, y en particular, en el contexto de los repositorios digitales, los metadatos describen e identifican a un conjunto de objetos digitales. Cada documento cargado en un repositorio digital está asociado a un registro de metadatos que describe información sobre el mismo, tales como autor del documento, fechas de publicación, palabras claves, instituciones de origen, entre otras cosas. Los metadatos necesarios para guardar un objeto digital en un repositorio dependen de cada repositorio en particular, así como también el esquema utilizado para organizar a los mismos.

Existe una gran variedad de metadatos dependiendo del contexto y el objetivo en que los mismos son estudiados. En el caso particular de los objetos digitales, estos pueden clasificarse (Dappert & Enders, 2010) en distintas categorías que reflejan los distintos aspectos de su función, y son las siguientes:

- Descriptivos: metadatos que describen al objeto digital, tales como título, autores o fechas, y que sirven de apoyo para el descubrimiento y entrega del contenido.
- De estructura: metadatos que capturan relaciones estructurales físicas de un objeto digital, tales como información acerca de imágenes embebidas dentro de un sitio web, y relaciones estructurales lógicas, como por ejemplo rangos de páginas en un libro.
- Técnicos: incluyen información técnica que aplica a cualquier tipo de archivo, como información acerca del software o hardware necesario para la ejecución o lectura del archivo, sumas de comprobación o firmas digitales que aseguren autenticidad, y también información técnica específica de cada tipo de archivo, como por ejemplo ancho y largo de una imagen.

- Administrativos: metadatos que indican la procedencia de un objeto digital, o las modificaciones realizadas sobre el mismo, así como también información sobre los permisos y licencias de uso.

Registro de metadatos

A lo largo del capítulo 1 se hizo referencia al término 'registro de metadatos', y este mismo será utilizado también con frecuencia en capítulos posteriores, es por eso que resulta necesaria una definición con mayor detalle. Un registro de metadatos es un conjunto de metadatos asociados que describen a un objeto específico, y en el dominio de esta tesina más precisamente a un objeto digital. La calidad de los metadatos es muy importante para que los registros puedan ser recuperados por los motores de búsqueda actuales y comprendidos fácilmente por los lectores (CIESIN, s.f.).

Un documento guardado en un repositorio digital se compone de un registro de metadatos que lo describe, un archivo binario que usualmente representa al objeto digital específico (documento de texto, video, audio o imagen) e información propia del sistema de repositorio utilizado.

Para ejemplificar el concepto de registro, a continuación se muestra parte de un registro de metadatos que describe a una ponencia presentada en un congreso, cargada en el repositorio SEDICI.

dc.subject	Turbulence	es
dc.subject	Large eddy simulation	es
dc.subject	Structural vibrations	es
dc.title	Stochastic wind-load model for building vibration estimation using large-eddy cfd simulation and random turbulenc flow generation algorithms	en
dc.type	Objeto de conferencia	es
sedici.identifier.uri	https://cimec.org.ar/ojs/index.php/mc/article/view/5289	es
sedici.identifier.issn	2591-3522	es
sedici.creator.person	Inaudi, José A.	es
sedici.creator.person	Sacco, Carlos G.	es
sedici.description.note	Publicado en: *Mecánica Computacional* vol. XXXV, no. 12	es
sedici.subject.materias	Ingeniería	es

Figura 2. Parte de un registro perteneciente a una ponencia cargada en SEDICI

Esquema de metadatos

Un esquema de metadatos define la manera en que un conjunto de elementos describe información específica de un objeto digital en un dominio particular. Los esquemas de metadatos surgen como respuesta a la necesidad de una comunidad de representar de la mejor forma posible un tipo de recurso específico (Digital Curation Center, s.f.). El desarrollo de

un esquema de metadatos tiende a ser controlado por el consenso de la comunidad, y a través de procesos formales de presentación, aprobación y publicación de nuevos elementos.

Existe una gran cantidad de esquemas y estándares de metadatos que pueden categorizarse según el dominio, el nivel de detalle o el tipo de información que almacenen. Uno de los estándares de propósito general más utilizados es Dublin Core, y a partir del mismo por ejemplo, se desarrolló un esquema extendido para el dominio de las ciencias naturales llamado Darwin Core.

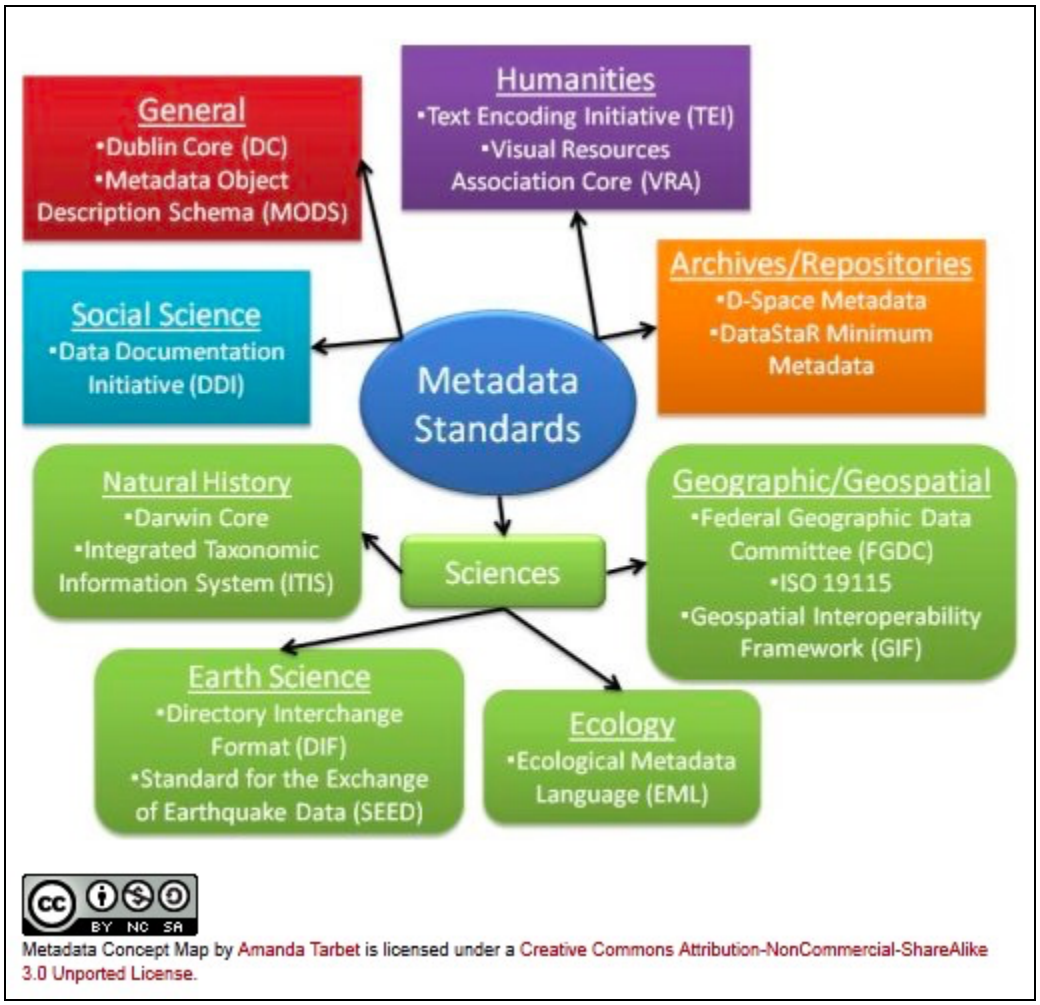

Figura 3. Estándares de metadatos categorizados según el dominio
(Fuente: https://libguides.bc.edu/c.php?g=44295&p=280677)

Existe una gran cantidad de estándares más allá de los que se pueden observar en la figura 3 y entre los referentes a nivel global en el contexto de los repositorios digitales se encuentran:

- Dublin Core [https://www.dublincore.org/]
- MODS [http://www.loc.gov/standards/mods/]
- METS [http://www.loc.gov/standards/mets/]
- MARC [https://www.loc.gov/marc/]

En muchos casos es necesario realizar un mapeo entre un esquema y otro, es decir definir una representación equivalente entre dos esquemas, para poder guardar los registros obtenidos a partir de distintas fuentes que utilizan esquemas de metadatos diferentes. Además del esquema utilizado, la estructura con la que se guardan los metadatos varía. Por ejemplo, en algunos casos los títulos y subtítulos son guardados en distintos metadatos, y en otros el título completo se guarda en un único metadato. Otro caso es el de los nombres de autores, en algunos casos se guardan de la forma 'apellido, nombre' y en otros como 'nombre apellido,' con y sin coma, utilizando nombres completos o sólo iniciales, entre otras variantes.

Dublin Core

Dublin Core es un modelo de metadatos creado por la organización Dublin Core Metadata Initiative [https://www.dublincore.org], dedicada a promover el uso de estándares interoperables y a promover el desarrollo de vocabularios especializados de metadatos. Es el esquema de metadatos más utilizado y una de las razones principales radica en la adopción del mismo por el estándar de interoperabilidad OAI-PMH como esquema de metadato obligatorio para la transmisión de información entre repositorios y otros sistemas.

Este modelo se compone de 15 elementos que describen diferentes aspectos de un objeto digital, y que pueden clasificarse según el tipo de información que representen:

- Relacionados al contenido del recurso
    - Título
    - Palabras claves
    - Descripción
    - Fuente
    - Tipo del recurso
    - Relación
    - Cobertura
- Relacionados al recurso cuando es visto como una propiedad intelectual
    - Autor o creador
    - Editor
    - Colaboradores
    - Derechos
- Relacionados con la instanciación del recurso
    - Fecha
    - Formato
    - Identificador del recurso
    - Idioma

Existe una especificación más completa y actualizada del vocabulario Dublin Core llamada DCMI Terms (DCMI: DCMI Metadata Terms, s.f.) que además de contener los 15 elementos básicos de Dublin Core define docenas de propiedades, clases, tipos de datos y esquemas de codificación de vocabulario. Las propiedades agregadas en DCMI Terms

permiten describir con mayor detalle a los metadatos, por ejemplo *created, modified, accessioned,* etcétera*,* para describir al metadato fecha*.*

Si bien Dublin Core define un estándar de interoperabilidad de metadatos, en muchos casos la cantidad de elementos que provee para describir un objeto digital no es suficiente, y se debe recurrir a otros esquemas.

Perfiles de aplicación

Generalmente, el uso de un único esquema de metadatos en un repositorio digital no es suficiente para describir eficientemente los objetos digitales contenidos en el mismo por las limitaciones propias de cada esquema (Koutsomitropoulos, Alexopoulos, Solomou & Papatheodorou, 2010). Para solucionar este problema, surge el uso de Perfiles de Aplicación que consisten en un conjunto de elementos obtenidos a partir de uno o más esquemas existentes, combinados y optimizados para una aplicación específica (Heery & Patel, 2000). Los perfiles de aplicación no pueden introducir elementos nuevos que no pertenezcan a ningún esquema de metadatos previamente definido. En caso que se quiera introducir elementos nuevos, se debe crear el esquema de metadatos correspondiente y se debe asumir la responsabilidad del mantenimiento del mismo.

En el SEDICI por ejemplo, el perfil de aplicación utilizado se compone a partir de esquemas como Dublin Core extendido, MODS, ETD (específico para tesis y disertaciones) y un esquema de metadatos propio. En la tabla que se muestra a continuación se puede observar el registro de metadatos que describe a un libro guardado bajo este perfil de aplicación.

Metadato	**Valor cargado en SEDICI**
dc.date.accesioned	2019-11-21T12:37:05Z
dc.date.available	2019-11-21T12:37:05Z
dc.date.issued	2019
dc.identifier.uri	http://sedici.unlp.edu.ar/handle/10915/85847
dc.identifier.uri	https://doi.org/10.35537/10915/85847
dc.description.abstract	Una introducción a la metodología de la investigación social cualitativa, apropiada para estudiantes de grado que deben presentar un trabajo de tesis para alcanzar su título. Además de repasar los métodos más comunes al alcance de un investigador solitario, como un tesista de grado, se destaca la presentación de las problemáticas subjetivas y aquellas relacionadas con el trabajo de campo.
dc.language	es

dc.publisher	Editorial de la Universidad Nacional de La Plata (EDULP)
dc.relation.ispartof	Libros de Cátedra
dc.subject	Investigación
dc.subject	Metodología
dc.subject	Turismo
dc.title	Investigar en turismo
dc.type	Libro
sedici.identifier.isbn	978-950-34-1835-2
sedici.title.subtitle	Una introducción
sedici.creator.person	Garay, Carlos Alberto
sedici.subject.materias	Ciencias Sociales
sedici.description.fulltext	true
mods.originInfo.place	Facultad de Ciencias Económicas
sedici.subtype	Libro
sedici.rights.license	Creative Commons Attribution-NonCommercial-ShareAlike 4.0 International (CC BY-NC-SA 4.0)
sedici.rights.uri	http://creativecommons.org/licenses/by-nc-sa/4.0/

Tabla 1. Registro de metadatos de un libro guardado en SEDICI

Identificadores persistentes

Un identificador persistente es (THOR Project, s.f.) una referencia duradera a un recurso (un documento, un archivo, una página web o cualquier otro objeto digital). Generalmente, estos identificadores son accionables, es decir, un navegador web podría dirigirse al objeto referenciado a partir de los mismos. Uno de los objetivos más importantes (Hakala, 2010) de los identificadores persistentes es proveer enlaces persistentes a los recursos. De este modo, con un identificador persistente un usuario puede confiar en que obtendrá el recurso deseado incluso si la localización física del mismo cambió en algún momento.

En el contexto de las publicaciones científicas y académicas, donde cada publicación puede referenciarse en múltiples sitios y/o repositorios, los identificadores persistentes resuelven la ambigüedad que se genera al no saber si dos publicaciones similares son

efectivamente la misma. Como solución a este problema surgen varios identificadores persistentes creados por instituciones, y entre los más reconocidos se destacan:

Handle System

El Handle System es un registro patentado por Corporation for National Research Initiative (CNRI) que asigna identificadores persistentes a recursos digitales distribuidos en internet, y que se encarga de resolver estos identificadores para localizar y acceder al recurso en sí mismo. Para favorecer a la persistencia, el string que representa a un identificador handle no debe estar ligado a ningún atributo modificable de la entidad y debe ser único en el sistema.

La estructura de los handles consiste en un prefijo que nombra a la autoridad responsable del recurso, y un sufijo que identifica al recurso específico. Por ejemplo, el handle '10915/5529' identifica a un documento en SEDICI. El prefijo 10915 hace referencia a SEDICI como autoridad responsable del recurso con sufijo 5529, que resuelve en la URL al artículo http://sedici.unlp.edu.ar/handle/10915/5529.

DOI (Digital Object Identifier)

El sistema de *Digital Object Identifier* (identificador de objeto digital) fue introducido en el año 2000 y es una implementación del Handle System encargada de dar a las publicaciones científicas un identificador para poder localizar el artículo en internet. Utiliza la infraestructura subyacente del Handle System y su funcionamiento se debe a la colaboración de tres grandes organizaciones:

- International DOI Foundation: se encarga de la gestión y promoción de los estándares de la marca.
- Corporation for National Research Initiatives: es la asociación encargada de mantener el Handle System.
- Agencias de registro: permiten a las editoriales conseguir un DOI para sus publicaciones bajo los criterios.

ORCID

ORCID es un sistema que surge en 2012 y que provee un identificador digital persistente para investigadores científicos y académicos.

Los objetivos de ORCID son:

- ayudar en la transición de la Ciencia a la 'e-Ciencia', para poder enlazar las publicaciones académicas en grandes volúmenes de información en constante crecimiento.
- suministrar a cada investigador un curriculum vitae digital constantemente actualizado.
- permitir a otras organizaciones utilizar la base de datos abierta de ORCID para recuperar información y publicaciones de un investigador.

Los identificadores son alfanuméricos y se otorgan a cada investigador de forma única. Por ejemplo, el ORCID del científico Stephen Hawking es https://**orcid**.org/0000-0002-9079-593X.

Interoperabilidad vía OAI-PMH

La interoperabilidad entre sistemas puede definirse como la capacidad que tienen los mismos para comunicarse unos con otros intercambiando información de forma que ambos puedan hacer uso de la misma. En el contexto de los repositorios digitales, la interoperabilidad hace referencia al intercambio de registros de metadatos de formas estandarizadas, y permite la (Rodrigues & Clobridge, 2011) agregación, minería de datos, creación de nuevas herramientas y servicios, y la generación de nuevo conocimiento a partir del contenido de los mismos.

La *Open Archive Initiative* se creó (Coll & Cruz, 2003) con la misión de desarrollar y promover estándares de interoperabilidad para facilitar la difusión eficiente de contenidos en Internet. OAI-PMH es un protocolo para la cosecha de metadatos definido por OAI y se centra en la transmisión de metadatos de cualquier material en soporte digital, siendo un requisito esencial que estos se codifiquen en Dublin Core.

OAI-PMH define dos roles, uno de *data provider*, que es el repositorio encargado de reunir y exponer metadatos, y otro de *service provider*, que son los sistemas externos encargados de recolectar los datos expuestos por los *data providers.*

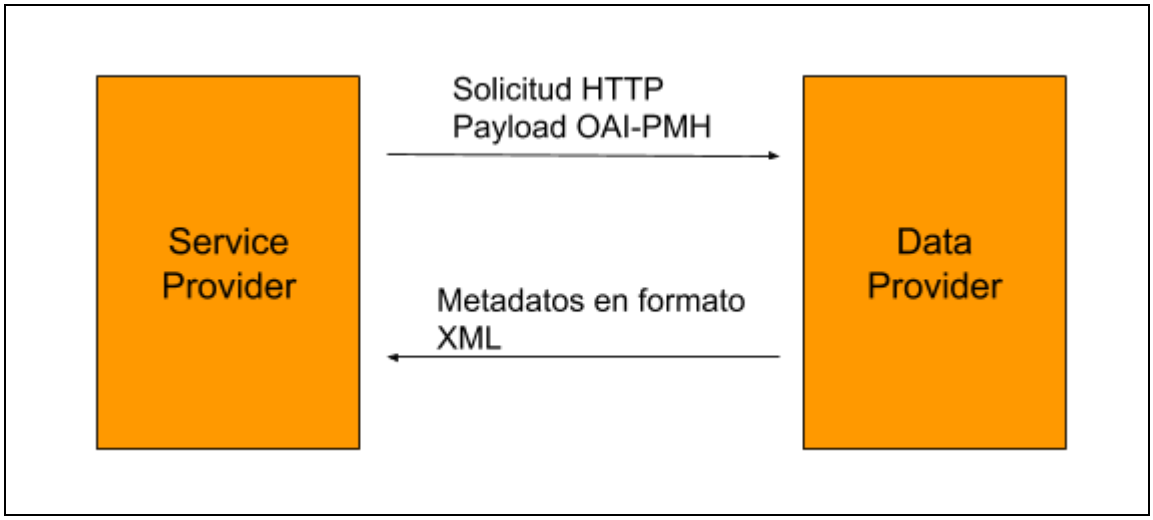

Figura 4. Protocolo OAI-PMH

Utilizando este protocolo un repositorio puede obtener todos los registros de metadatos expuestos por otros repositorios para procesarlos y opcionalmente incorporarlos, sin tener que establecer una comunicación previa, y utilizando módulos internos o externos al software de repositorio.

Cabe destacar que la cantidad de metadatos expuestos por cada fuente suele diferir de la cantidad de los metadatos utilizados por el repositorio para representar internamente a un

objeto digital. La 'vista' que obtiene un usuario de cada documento es una versión resumida del registro completo, y varía según el esquema de metadatos utilizado y la fuente de datos en particular.

Interoperabilidad entre esquemas de metadatos

Cuando se deben integrar metadatos de distintas fuentes heterogéneas es común encontrar problemas para establecer la interoperabilidad entre los distintos modelos de metadatos utilizados (esquemas y perfiles de aplicación). Dos modelos de metadatos se consideran interoperables (Orgel, Höffernig, Bailer & Russegger, 2015) si la información puede ser expresada de igual manera en ambos modelos, o bien si cambiar la representación de un modelo al otro es posible sin pérdida de información.

En el contexto de los repositorios digitales, para lograr la interoperabilidad entre sistemas, continuamente se deben realizar transformaciones entre diferentes esquemas y perfiles de aplicación, correcciones sobre la estructura y la sintaxis de los metadatos e incluso revisiones semánticas sobre el significado de cada elemento guardado en un repositorio. Cuando se obtiene información almacenada en múltiples fuentes, para que la misma pueda ser incorporada a un repositorio, se deben establecer mapeos entre los esquemas y la estructura bajo la cual los metadatos se encuentran cargados, para que el sistema sobre el cual corre el repositorio pueda procesar y almacenar los registros de forma correcta.

Mapeo de metadatos

Como se comentó en párrafos anteriores, para cumplir con la premisa de interoperabilidad resulta necesario mapear los metadatos guardados en un repositorio entre distintos esquemas, tanto para cumplir con los requisitos de agregadores nacionales, como es el Sistema Nacional de Repositorios Digitales (SNRD), y permitir que el repositorio pueda ser cosechado, como para importar producción académica traída desde otro repositorio que guarda los registros bajo un esquema de metadatos diferente.

Se define al mapeo como la actividad intelectual de comparar y analizar dos esquemas de metadatos para así encontrar equivalencias entre los mismos (Baca, 2016). Estas equivalencias se representan a través de *crosswalks* (cruce de peatones en inglés) que definen el mapeo de los elementos, la semántica y la sintaxis de un esquema de metadatos a otro, a partir de una tabla o gráfico. Actualmente, los *crosswalks* son el método más utilizado para alcanzar interoperabilidad entre distintos esquemas de metadatos (Chan & Zeng, 2006).

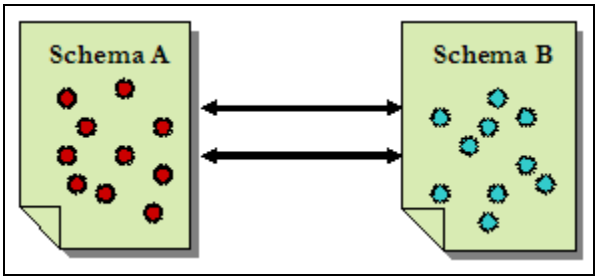

Figura 5. *Crosswalk* entre esquemas de metadatos (Chan & Zeng, 2006)

La necesidad de mapear entre distintos esquemas se hace presente en múltiples etapas a lo largo del proceso de recuperación e importación de registros académicos a un repositorio digital. Una vez obtenidos los registros de metadatos a procesar, el primer paso es preparar los mismos para que puedan ser deduplicados contra el conjunto de registros del repositorio destino. Es muy probable que ambos conjuntos se encuentren bajo distintos esquemas o perfiles, por lo que resulta necesario llevar todos los registros a un esquema común, de manera que el procesamiento sea uniforme y se pueda acceder a los metadatos bajo un conjunto limitado de valores. Además, una vez finalizada la deduplicación de registros, es necesario mapear nuevamente aquellos registros que deban ser importados al esquema de metadatos o perfil de aplicación del repositorio destino.

Capítulo 3 - Detección de registros académicos duplicados

Introducción

La ingesta masiva de documentos en un repositorio minimiza el tiempo en que la producción de muchos autores se comparte puesto que se disminuyen considerablemente los tiempos de carga de cada documento. Para que esto suceda correctamente y no genere un impacto negativo en la calidad del repositorio, el contenido a importar debe cumplir con ciertas premisas, entre las cuales se encuentra impedir la duplicación de contenido dentro del repositorio y esto se logra al asegurar que no se incluye en una importación producción ya existente dentro del mismo. Preservar la calidad del contenido cargado en un repositorio digital es un punto clave para poder cumplir con las funciones de interoperabilidad y difusión del contenido.

Aunque el objetivo más común y obvio que conlleva la detección de registros duplicados es el de eliminar información repetida, existen escenarios en los que se desea encontrar registros duplicados obtenidos a partir de distintas fuentes para poder enriquecer el valor de los mismos en ambas bases de datos/ repositorios, agregando datos que pueden estar presentes para una misma entidad en otra fuente.

Existen múltiples técnicas para la detección de registros duplicados, que cubren desde la deduplicación de tuplas en bases de datos hasta la deduplicación de producción científico-académica en repositorios o agregadores digitales, y es en este último tema sobre el cual se centrará este capítulo.

Deduplicación de registros

En informática, la deduplicación de registros (Record Linkage, s.f.) es la tarea de encontrar registros en un conjunto de datos que hacen referencia a la misma entidad en distintas fuentes de datos. En la literatura, se suele hacer referencia a esta tarea como *record linkage*, *identity resolution*, *data deduplication* o *record matching*, entre otras variantes que pueden surgir en base al autor o la línea de investigación. La aplicación más común de este proceso suele darse en bases de datos para eliminar tuplas o elementos repetidos con el objetivo de optimizar almacenamiento. Un ejemplo específico de deduplicación en bases de datos se muestra a continuación, donde el objetivo es detectar registros que hagan referencia a la misma dirección de un cliente (sin presencia de identificadores únicos):

Figura 6. Ejemplo de vinculación de registros en una base de datos
(Fuente: https://pbpython.com/record-linking.html)

Los mecanismos utilizados para la deduplicación de datos varían según el dominio, el tipo y la estructura de los datos. Por ejemplo, cuando el objetivo se basa en detectar objetos digitales duplicados a nivel de bits (sean dos archivos PDF) es posible calcular algún tipo de suma de comprobación sobre los mismos, utilizando algoritmos de encriptación como MD5 o SHA-256, y verificar que las cadenas resultantes sean las mismas. En cambio, cuando el objetivo se basa en detectar registros duplicados en bases de datos sin disponer de identificadores únicos, los mecanismos utilizados para su detección se tornan más complejos, porque se debe determinar la similitud existente entre cada par de tuplas, para así identificar aquellas que puedan referirse a la misma entidad y luego tomar una decisión en base al resultado obtenido.

La estructura de los datos cumple un rol muy importante en el procesamiento de los mismos, porque determina el grado de facilidad y consistencia con la que los mismos pueden ser procesados (Elmagarmid, Ipeirotis & Verykios, 2007), y es por esto que calcular la similitud entre registros de metadatos se torna más fácil cuando se cuenta con identificadores únicos o una estructura definida, puesto que esto facilita en gran medida la comparación y manipulación de los distintos campos.

Heterogeneidad de los datos

A todos los conflictos relacionados con la estructura de los datos se los suele encapsular bajo el término general (Elmagarmid et al., 2007) de *heterogeneidad de los datos.* Elmagarmid et al. definen dos tipos de heterogeneidad: estructural y léxica. La heterogeneidad estructural hace referencia a las diferencias existentes en la estructura bajo la cual son guardados los datos en distintas fuentes, por ejemplo una dirección guardada en el campo *dirección* en una fuente, y guardada en tres campos *calle, ciudad y código postal* en otra. La heterogeneidad léxica se da cuando los registros tienen la misma estructura en las distintas fuentes, pero la información está representada de distintas formas y refieren a la misma entidad del mundo real, por ejemplo '*Facultad de Informática, Calle 50 y 120'* y '*50 y 120, Fac. de Informática*'.

Una vez resueltos los problemas de heterogeneidad estructural, es decir, una vez que se logra convertir la estructura de los datos a una estructura uniforme, queda encontrar una forma de determinar similitudes en la estructura léxica de los mismos. Este es el punto central de estudio de esta tesina, para el cual se estudiarán diferentes técnicas y se presentará una solución acorde al contexto de repositorios digitales y producción académica en el capítulo siguiente.

El proceso de deduplicación conlleva entonces tareas de limpieza y normalización de los datos previas a la detección efectiva de registros duplicados. En el capítulo '*Data Consolidation and Integration*' del libro '*Master Data Management (2009)'* se divide al proceso general en tres subprocesos:

- Parseo[4] y estandarización de los datos: se lleva la información de cada registro u objeto a una representación estandarizada y uniforme.
- Transformación de los datos: se normalizan los formatos y se realizan correcciones sobre cada registro.
- Coincidencia de registros: se evalúa la similitud entre los registros para determinar si existen conjuntos que hagan referencia a la misma entidad.

La tarea que consiste en lograr una representación estandarizada y uniforme de los datos puede resultar análoga con el mapeo de los metadatos mencionada en el capítulo 2, centrando el problema en el contexto de los repositorios académicos.

Deduplicación de registros académicos

En los repositorios digitales la información guardada se basa completamente en producción científico-académica. Como se mencionó en secciones anteriores, los registros académicos se guardan bajo distintos esquemas o formatos, que dan estructura y organización a los mismos. La diversidad de fuentes a partir de las cuales pueden obtenerse estos registros resulta en formatos y esquemas variados con estructuras diferentes; y como ya se mencionó es en este punto donde radica la mayor dificultad cuando se trata de detectar registros duplicados, ya que se agregan varias dificultades al proceso.

La deduplicación de registros académicos debe necesariamente basarse en el análisis de los metadatos que componen a cada uno, debido a que estos contienen la información que representa a cada objeto digital. Las comparaciones de igualdad entre los valores de los distintos campos de dos registros no son suficientes, porque las mismas no contemplan variaciones de tipeo, de estructuración y de diferencias entre caracteres (mayúsculas y minúsculas, signos de interrogación, exclamación, etcétera); y es por esto que se debe recurrir a funciones que calculan la similitud existente entre diferentes cadenas de texto.

Además, cabe destacar que la deduplicación de registros académicos puede darse de dos formas: interna o cruzada (Atzori, Manghi & Bardi, 2018). En el primer caso, los duplicados son generados por información cargada dentro de la misma fuente de datos, y en el segundo caso, los duplicados son generados al cruzar información a partir de múltiples fuentes, por ejemplo, en casos de importación masiva a un repositorio.

Heterogeneidad en los metadatos de distintas fuentes

La dificultad principal presente en la detección de recursos iguales en el dominio de los repositorios académicos y científicos es la desambiguación de valores. La estructura bajo la cual se cargan registros en distintos repositorios depende de los tesauros adoptados, el esquema de metadatos, y en gran parte, del factor humano. De esta manera y a modo de ejemplo, el artículo titulado ‘Políticas territoriales y construcción del paisaje cultural’ y subtitulado ‘Caso Región Gran La Plata’ puede estar cargado de esta manera en un repositorio;

[4] Parseo es el proceso de analizar una secuencia de caracteres a fin de determinar su estructura sintáctica.

y puede figurar como un artículo titulado ‘Políticas territoriales y construcción del paisaje cultural - Caso región gran La Plata’ en otro. En este último, el registro no distingue entre título y subtítulo, hay palabras que no llevan tildes correspondientes y existen diferencias en la presencia de mayúsculas y minúsculas.

Asimismo, detectar que dos autores son el mismo es muy dificultoso, principalmente por la forma en que los mismos son cargados en un registro. Por ejemplo, la autora ‘García, María Ana’ puede aparecer como ‘García, María A.’, ‘García, M. Ana’ ó ‘García, M. A.’, en distintos repositorios.

Para contextualizar, a continuación se presentan dos registros que hacen referencia al mismo artículo depositado en dos repositorios digitales distintos: SEDICI y CONICET Digital.

ProBiota | Serie Técnica y Didáctica | Lista de peces de la provincia de Mendoza

Autores: Fernández, Luis | Marin, Bruno | Nadalin, Diego O. | Martínez, Facundo | López, Hugo Luis **2015**

Tipo de documento: Edicion de revista

Like Share Twittear

Resumen

Esta serie tiene como finalidad dar a conocer las especies presentes en los diferentes estados provinciales. Tomando como base los trabajos de López et al. (2003), Reis et al. (2003), y Liotta (2006), mencionamos para cada territorio los cambios y novedades posteriores a estas publicaciones. Consideramos que este modesto aporte contribuirá a precisar el conocimiento ictiofaunístico regional, ya que, además de las listas de especies, adjuntamos bibliografía de referencia y el marco biogeográfico e hídrico correspondientes, que podrán ser de utilidad para quienes hagan uso de este trabajo. Por otra parte entendemos que la participación de autores involucrados en la región considerada, le da un verdadero sentido federal a esta contribución, además de reforzar vínculos en los protagonistas de nuestra especialidad. En este nuevo número presentamos la provincia de Mendoza que se encuentra limitada al norte por San Juan, al este por San Luis, al sur La Pampa y Neuquén y al oeste por Chile.

Información general

Fecha de publicación: 2015

Idioma del documento: Español

Revista: ProBiota: Serie Técnica y Didáctica; no. 29

Institución de origen: Facultad de Ciencias Naturales y Museo

ISSN: 1515-9329

Palabras claves: Mendoza (Argentina) ; ictiología ; Peces ; listas bibliográficas

Materias: Ciencias Naturales

Figura 7. Artículo depositado en SEDICI

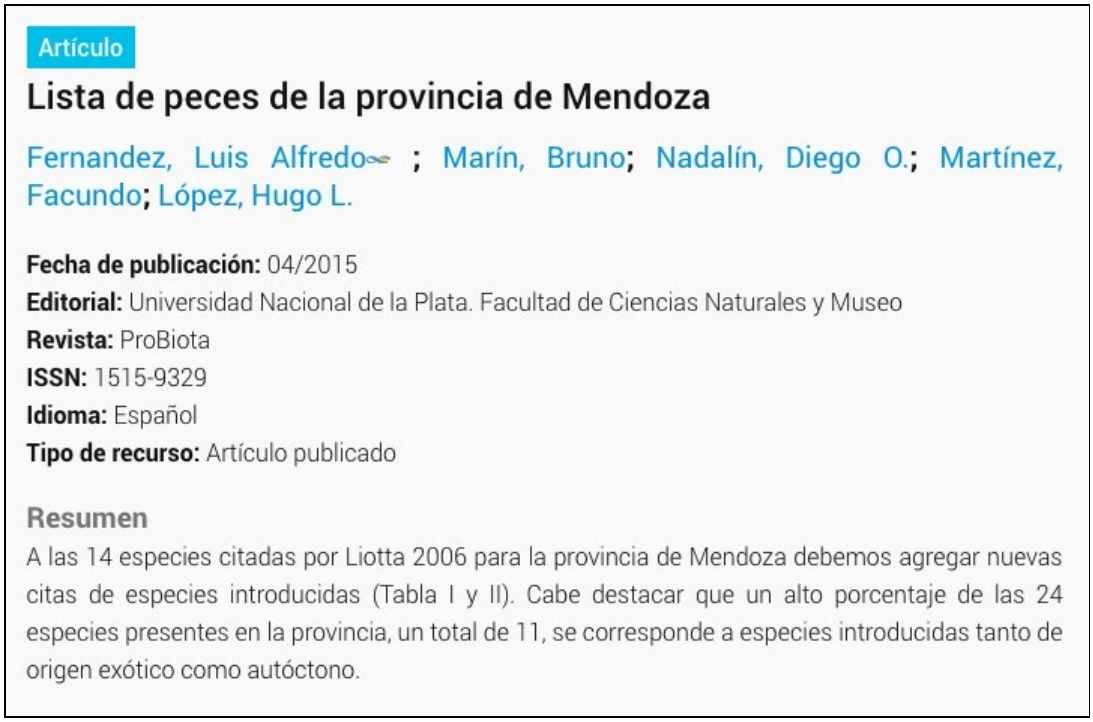

Figura 8. Artículo depositado en CONICET Digital

Como se puede observar, los títulos de los registros coinciden solo parcialmente e incluso a simple vista son considerablemente diferentes. La cantidad de autores es la misma, pero la forma en la que algunos de ellos están cargados difiere:

Fernández, Luis - Fernandez, Luis Alfredo
López, Hugo Luis - López, Hugo L.

Aunque el resumen guardado en ambos repositorios también es distinto, tanto los títulos como los autores de ambos registros presentan un gran porcentaje de similitud, y hay un conjunto de otros metadatos que coinciden completamente: fecha de publicación, título e issn de la revista. Estos dos registros hacen referencia efectivamente al mismo objeto digital, y puede comprobarse al descargar el archivo PDF asociado en cada repositorio.

Uso de identificadores persistentes

Cuando se intenta distinguir entre dos recursos iguales en el dominio de los repositorios digitales, el uso de identificadores persistentes externos a los repositorios resuelve en gran medida el problema, puesto que dos registros de metadatos que describen al mismo objeto digital debieran tener el mismo identificador persistente, aún cuando los mismos se encuentran depositados en distintos repositorios.

De esta manera, si cada registro presente en un repositorio tuviese asignado un identificador persistente, la deduplicación de los mismos no resultaría tan difícil. En la práctica, la adopción de identificadores persistentes universales, como DOI por ejemplo, es muy baja

(Atzori, Manghi & Bardi, 2018). Si bien los repositorios académicos establecieron buenas prácticas relacionadas a estos identificadores, los mismos todavía no fueron adoptados o no se encuentran disponibles junto con los metadatos de cada registro.

Técnicas para la detección de registros duplicados

En la literatura se presentan múltiples técnicas que abordan el problema de registros duplicados y las mismas (Elmagarmid et al., 2007) se dividen, a grandes rasgos, en:

- aquellas que utilizan enfoques que se basan en algoritmos de aprendizaje automático, para los cuales se requieren datos de entrenamiento, y
- aquellas que utilizan enfoques que se basan en el conocimiento sobre el dominio específico y en funciones de distancia para establecer coincidencias.

El proceso de detección de registros duplicados fue planteado inicialmente por Dunn en 1946 (Amón & Jiménez, s.f.). Luego Newcombe et al. desarrollaron fundamentos probabilísticos en 1959 (Newcombe et al., 1959), que fueron retomados por Fellegi y Sunter en 1969 y definieron reglas de decisión probabilística (Fellegi & Sunter, 1969). Entre los años 1990 y 2000 Winkler propone mejoras a este modelo y plantea que el modelo mejorado no requiere de tantas suposiciones previas acerca de los datos y que el mismo se adecua mejor en casos difusos, aunque se aumenta considerablemente el tiempo de computación (Winkler, 2000).

El proceso de detección de registros duplicados se resume en comparar cada registro perteneciente a un conjunto N con cada registro perteneciente a otro conjunto M y determinar si la similitud calculada para cada par de registros es mayor o menor a un umbral que categoriza los pares como duplicados o no duplicados.

Para llevar a cabo este proceso Elmagarmid et al. (2007) presentan seis técnicas que se describen a continuación:

A. Modelos probabilísticos de emparejamiento

Estos modelos se basan en conceptos de la teoría de la probabilidad y definen funciones de densidad para determinar la similitud existente entre un par de registros. Algunos autores hacen referencia a la detección de duplicados como un problema de *inferencia Bayesiana*, en el cual las observaciones realizadas sobre un par de registros llevan a inferir la similitud existente entre los mismos.

B. Aprendizaje supervisado y semi supervisado

Estos enfoques se basan en técnicas de aprendizaje automático cuyo objetivo es que las computadoras, a través de funciones y datos de entrenamiento, aprendan a encontrar similitudes entre los registros y determinen si son o no duplicados. La rama del aprendizaje automático que utiliza datos de entrenamiento etiquetados, es decir pares de registros marcados como duplicados o no duplicados, se llama aprendizaje supervisado. Aquella que mezcla datos de entrenamiento etiquetados y no etiquetados se define como aprendizaje semi supervisado.

C. Técnicas basadas en aprendizaje activo

Si bien es fácil generar pares de registros que puedan considerarse fácilmente duplicados o no duplicados, es muy difícil crear casos ambiguos donde las diferencias son muy sutiles. En base a esto, algunos sistemas de detección de duplicados utilizan técnicas de aprendizaje activo, en los cuales en vez de entrenar a un modelo con un conjunto estático de pares de registros, el modelo mismo elige activamente subconjuntos de datos no etiquetados para mejorar su función, y una vez que se le provee un subconjunto de datos etiquetados, el modelo aprende de estos y aumenta la precisión con la que se establecen similitudes.

D. Técnicas basadas en distancia

Estas técnicas utilizan métricas de similitud o funciones de distancia entre cadenas de texto para determinar las diferencias existentes entre una cadena de texto y otra. La ventaja principal de este tipo de algoritmos es que no requieren de datos de entrenamiento, basta con definir una métrica de similitud y umbrales adecuados para que el algoritmo funcione correctamente. Existen múltiples formas para calcular la similitud entre los valores de un registro, algunos algoritmos optan por unir todos los campos del registro en una larga cadena de texto y calcular la similitud entre estas, y otras optan por calcular la similitud campo a campo y determinar luego la similitud general de los registros.

E. Enfoques basados en reglas

El enfoque basado en reglas es un caso especial de las técnicas basadas en distancia porque también hace uso de métricas de similitud entre strings. En este caso, son utilizadas las métricas de cada campo de un registro, y se definen reglas con la lógica necesaria para catalogar a un par de registros como duplicados o no duplicados. Se pueden definir múltiples reglas que combinen o analicen de distintas formas los distintos campos de un registro, así como también se pueden definir umbrales propios a cada regla. Es importante notar que estos enfoques dependen fuertemente del conocimiento que se tiene sobre el dominio particular, ya que en base al mismo se define la lógica de comparación de los distintos campos, y la combinación de las métricas con los umbrales.

F. Aprendizaje sin supervisión

Los algoritmos de aprendizaje sin supervisión son un tipo de algoritmo de aprendizaje automático donde los datos de entrada utilizados no están etiquetados. En este caso se trabaja con pares de registros que no están catalogados como duplicados o no duplicados, y el modelo mismo debe inferir similitudes entre los registros a fin de formar *clusters* con registros que considere parecidos. Este enfoque se basa en conceptos de probabilidad como los utilizados en los modelos probabilísticos de emparejamiento.

Metodologías para optimizar la cantidad de comparaciones

En general, los algoritmos de detección de registros duplicados conllevan un orden de ejecución cuadrático, puesto que se deben comparar todos los registros de un conjunto dado contra todos los registros del otro conjunto. Esto resulta muy costoso en términos de procesamiento, y a raíz de este problema surgen distintas metodologías que intentan minimizar la cantidad de comparaciones realizadas, las cuales consisten en dividir el conjunto de registros a comparar en subconjuntos más pequeños. Para realizar estas particiones, los enfoques más populares (Draisbach & Naumann, 2009) son:

- métodos de bloques, que dividen estrictamente los registros en subconjuntos disjuntos, utilizando una clave como *código postal* para la partición, y por otro lado, los
- métodos de tipo ventana que ordenan los registros en base a una clave y luego definen una ventana deslizante de tamaño fijo para comparar únicamente los pares de registros que se encuentren dentro de la misma ventana.

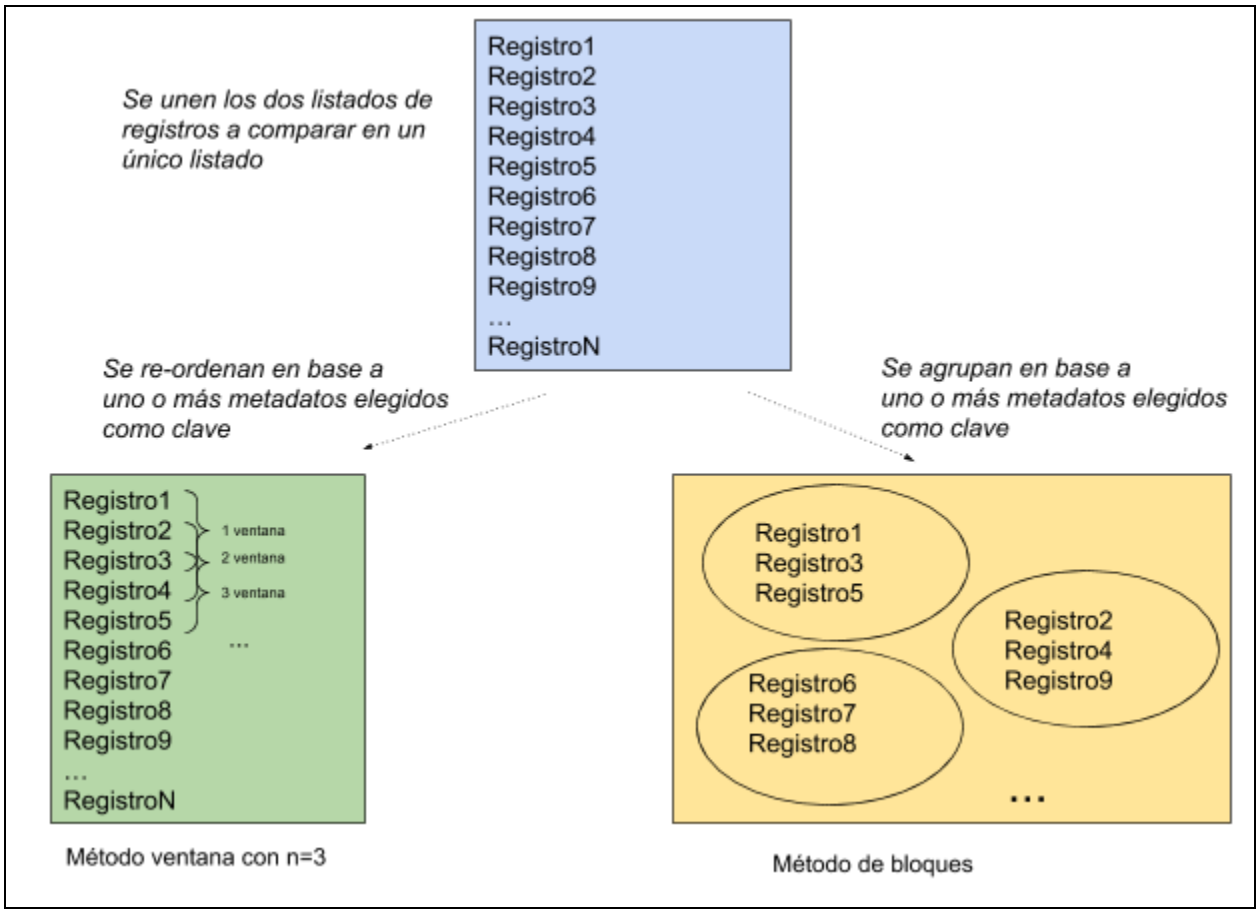

Figura 9. Metodologías de optimización para la cantidad de comparaciones

Si bien estas metodologías son muy útiles para minimizar considerablemente la cantidad de comparaciones, la elección de una clave a partir de la cual formar bloques u

ordenar los registros es muy difícil de llevar a cabo, sobre todo cuando se trabaja con datos con estructuras no uniformes y valores que son propensos a errores de tipeo, puesto que es muy posible que queden registros por fuera del bloque o la ventana a la cual debieran pertenecer y no sean contemplados en las comparaciones.

Soluciones existentes

La deduplicación de registros académicos está presente en la mayoría de los agregadores de contenido como Google Scholar, OpenAIRE, CORE, entre otros. Sin embargo, cada uno de estos sistemas utiliza algoritmos y herramientas propias para la detección de registros duplicados, y ninguno de estos es de código abierto.

OpenAIRE (https://www.openaire.eu/) es un proyecto europeo que apoya a la Ciencia Abierta y que funciona como agregador de publicaciones científicas académicas de múltiples repositorios conectados. Para detectar registros duplicados y evitar incorporar nuevos documentos ya agregados previamente, OpenAIRE utiliza algoritmos de deduplicación basados únicamente en títulos, fechas y autores de cada registro. Cuando los títulos no son suficientes para determinar la igualdad entre dos registros, entonces se utilizan las fechas y los autores como pistas para aumentar o disminuir los porcentajes de similitud (Príncipe, 2015).

Los algoritmos utilizados se ejecutan en un cluster Hadoop MapReduce [https://hadoop.apache.org/docs/r1.2.1/mapred_tutorial.html] sobre hardware específico para maximizar el procesamiento paralelo y minimizar los tiempos de ejecución en deduplicaciones que involucran millones de registros. La deduplicación forma parte de un sistema más grande encargado de obtener, limpiar, deduplicar y enriquecer registros de metadatos que luego serán publicados.

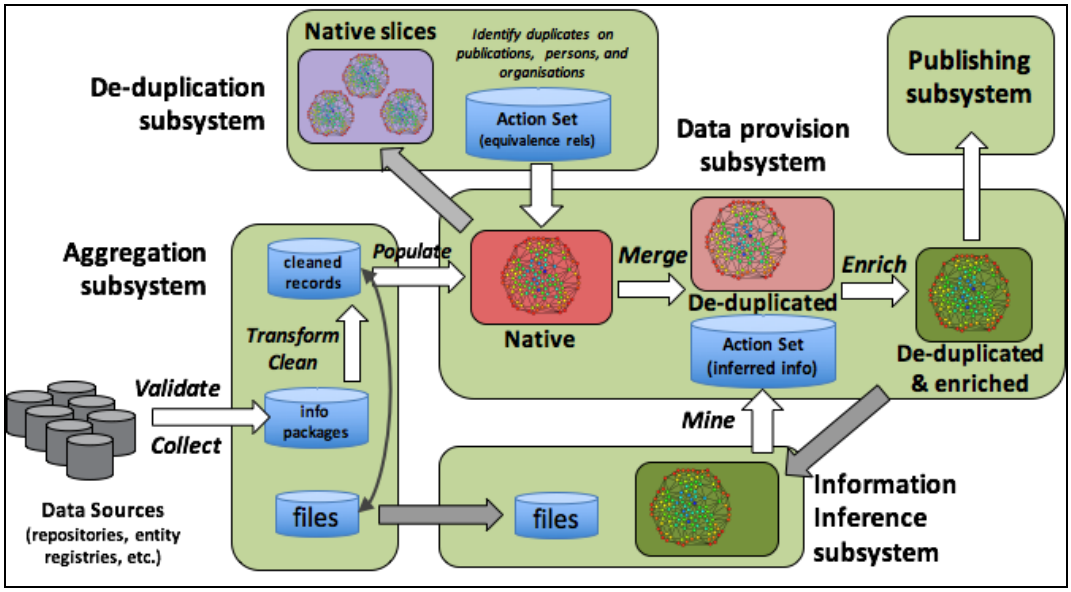

Figura 10. Ecosistema de agregación de contenido de OpenAIRE
(Aggregation and content provision workflows, s.f.)

La deduplicación de registros académicos también está presente en buscadores y gestores de referencias bibliográficas (Kwon, Lemieux, McTavish & Wathen, 2015):

- Ovid [https://www.ovid.com/]: es una plataforma de búsqueda de investigación médica. Los buscadores utilizados deduplican información generada en productos de la misma plataforma como Ovid MEDLINE y Ovid Embase.
- Refworks [https://www.refworks.com/es/]: es un gestor de citas y sus buscadores permiten deduplicar registros de distintas fuentes.
- Mendeley [https://www.mendeley.com/]: es un gestor de citas que identifica registros duplicados entre las referencias importadas y permite eliminarlas en el momento.
- Endnote [https://endnote.com/]: es un gestor de citas que genera grupos separados con referencias duplicadas y le permite al usuario navegar los mismos para eliminar aquellas que identifique como duplicadas.

Existe una variedad de sistemas comerciales (Peter Christen, 2009) para linkear registros y deduplicar los mismos, pero desde la perspectiva del usuario estos sistemas son una caja negra, los algoritmos y la lógica de comparación utilizada para resolver el problema no son expuestos al usuario, ni tampoco son customizables o modificables. Además muchos de estos sistemas están construidos para un dominio específico, como pueden ser el linkeo de clientes en una base de datos o de listas de correos electrónicos.

Para la deduplicación de registros en bases de datos y plantillas de Excel, una de las soluciones de software más conocidas es Dedupe (https://dedupe.io/), una herramienta que utiliza algoritmos de aprendizaje automático supervisado y no supervisado para encontrar tuplas duplicadas o altamente similares. Aunque esta herramienta es privativa y necesita de una membresía paga para poder ser utilizada, hace uso de una librería que se encuentra en acceso abierto publicada en *Github*: https://github.com/dedupeio/dedupe. Si bien es un enfoque válido y muy útil para la deduplicación de datos, no se adecua correctamente al caso de uso de registros académicos en repositorios digitales, porque no presenta ningún tipo de regla o análisis de características propias del dominio, y está pensada para casos de negocio más genéricos, como por ejemplo la detección de tuplas duplicadas en bases de datos de clientes. Se realizó una serie de pruebas de esta herramienta con registros de metadatos obtenidos de SCOPUS (https://www.scopus.com) y SEDICI, y los resultados no fueron satisfactorios, puesto que en muchos casos se detectaron como duplicados registros de artículos con registros de reseñas sobre estos artículos, o registros de ponencias con registros de artículos. En estos casos por ejemplo, si bien existen grandes similitudes en metadatos como título, autores y fechas, el modelo de *Dedupe* no tiene en cuenta la tipología de los documentos y no permite parametrizar nuevos valores para los distintos metadatos.

También existen diversas soluciones comerciales que proveen servicios de deduplicación sobre bases de datos, generalmente enfocados en bases de datos con registros de clientes y direcciones, y que permiten definir reglas de negocio específicas dentro de este dominio. Estas soluciones ofrecen planes de deduplicación en base a la cantidad de tuplas a procesar, el tamaño de la base de datos o incluso se ofrecen precios fijos por registro o tupla deduplicada.

En el contexto de los registros académicos la lógica de deduplicación debe ser muy específica y debe basarse en el análisis de los metadatos que cada registro exponga. Puesto que la diferencia sintáctica que existe en los metadatos cargados en distintos repositorios es muy alta, la cantidad de posibilidades que deben contemplarse es muy extensa.

Capítulo 4 - Análisis y desarrollo

Introducción

A raíz del estudio de las técnicas y metodologías para la detección de registros duplicados, y en base a las soluciones existentes analizadas, se propone el desarrollo de una herramienta para la detección de registros académicos duplicados obtenidos desde repositorios digitales.

Inicialmente se comenzó el desarrollo con un prototipo de prueba que se vio limitado por la diversidad de registros a comparar y la falta de flexibilidad para contemplar casos de uso nuevos. En base a esta experiencia, se propone el desarrollo de una herramienta basada en un sistema de reglas que utiliza funciones de distancia para calcular porcentajes de similitud entre strings. Cada regla combina distintos metadatos y en base a las comparaciones se genera un resultado para cada par de registros.

Dado que los esquemas de metadatos varían en cada fuente de datos a partir de las cuales se obtienen los registros, se propone un esquema de metadatos genérico que permita a la herramienta definir una representación uniforme para el procesamiento de los registros. Además se propone el desarrollo de un módulo de mapeo de metadatos que da soporte a la funcionalidad de deduplicación de la herramienta y a las distintas etapas de un proceso de recuperación e ingesta.

Desarrollo de un primer prototipo

Inicialmente se comenzó el desarrollo con un prototipo pequeño que permitiese comprobar la eficacia de las funciones para calcular similitud entre strings, y la comparación entre metadatos de distintos registros. Este prototipo permitía:

- Procesar dos listados de documentos (subidos en formato csv) y detectar, aplicando distintos criterios, similitud entre recursos.
- Procesar un listado de documentos y un listado de autores, y detectar coincidencias entre autores de cada documento con autores del listado.

El algoritmo de detección que utilizaba este primer prototipo se basaba en la comparación de títulos y autores únicamente, y utilizaba un algoritmo de tipo ventana[5] para disminuir la cantidad de comparaciones entre documentos.

Para lograr el ordenamiento y selección de una ventana adecuada, se utilizó como clave el título de cada registro. De esta manera, se realizaba una unión entre ambos listados de registros a deduplicar y se ordenaba la lista resultante alfabéticamente en base al título de cada uno. Una vez ordenada, se comparaba cada registro de la lista contra los 5 registros siguientes, y se verificaba si la similitud entre los títulos y autores de cada registro era lo suficientemente alta como para considerar a ambos títulos el mismo. Antes de comenzar la comparación de un

[5] Los métodos de comparación de tipo ventana [Hadzic - Sarajlic, 2020] ordenan los datos en base a una clave, y usando una ventana deslizante, iteran sobre los datos ordenados y comparan únicamente aquellos que estén dentro de la misma ventana.

registro contra los 5 consecuentes, se generaba un nuevo cluster y se agregaba el registro inicial de la ventana al mismo. Luego, todos los registros considerados duplicados dentro de esa ventana también se agregaban al cluster.

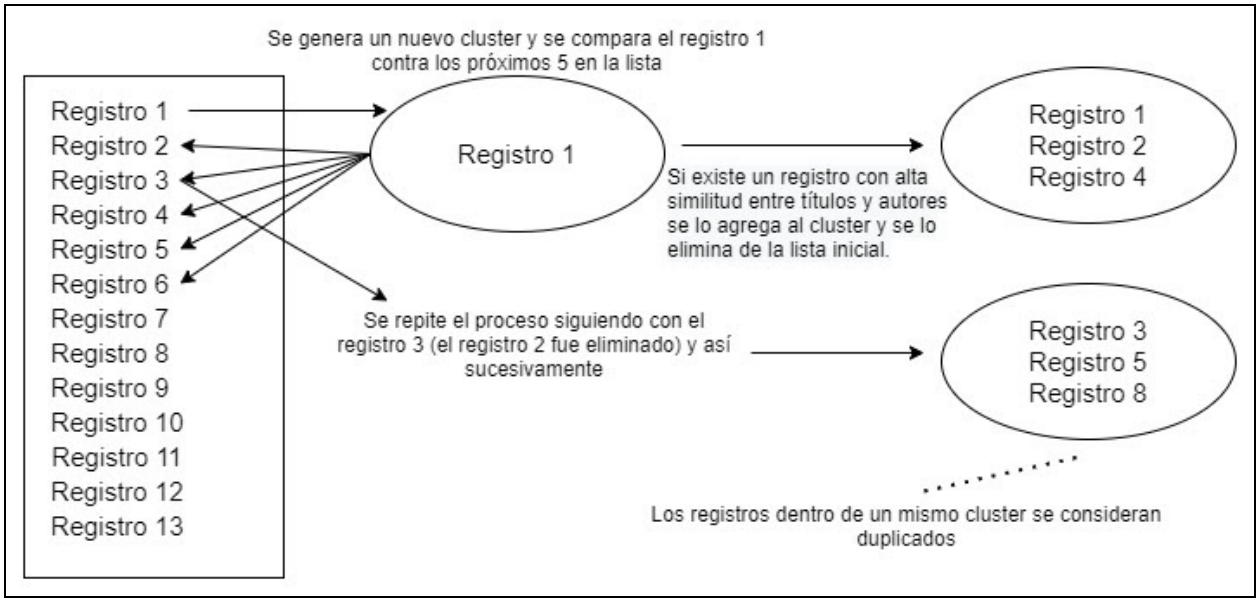

Figura 11. Diagrama de funcionamiento del primer prototipo

Con la prueba y el uso de la herramienta rápidamente surgieron problemas e insuficiencias en este prototipo que dejaron ver la necesidad de modelar una herramienta más robusta, con módulos definidos e independientes, que sea flexible a la hora de contemplar casos de uso nuevos, como ser la comparación de registros académicos obtenidos desde múltiples repositorios, permitir la comparación de nuevas tipologías de documentos, o trabajar con datos de entrada y salida en múltiples formatos. Además, una funcionalidad que este prototipo no brindaba era la capacidad de definir ‘niveles’ de similitud entre cada registro para categorizar cada par de documentos como ‘casi duplicados’, o establecer porcentajes que permitan un análisis más útil que un conjunto de clusters, una vez generados los resultados.

A raíz de la experiencia obtenida con el primer prototipo desarrollado se definieron nuevos objetivos y requisitos funcionales para el desarrollo de la nueva herramienta.

Solución propuesta

Se propone el desarrollo de una herramienta de detección de registros duplicados basada en reglas y funciones de distancia que permita el análisis y comparación de los distintos metadatos y determine el nivel de similitud existente entre cada par. Para esto, la herramienta debe ser capaz de:

- Comparar dos listados de registros de metadatos en busca de duplicados.
- Realizar comparaciones sintácticas de similitud entre strings.
- Definir un esquema de representación uniforme para los registros a procesar.
- Establecer porcentajes de similitud entre cada par de registros comparados.

- Permitir la definición de umbrales a partir de los cuales dos registros se consideran duplicados, casi duplicados, no duplicados o indefinido.
- Definir reglas que evalúen ciertos metadatos de los registros y determinen similitud entre los mismos, independientes unas de otras.
- Brindar flexibilidad para incorporar nuevos formatos de archivo para la entrada y salida de datos, comparar nuevos tipos de registros y definir nuevas reglas.

Para la construcción de la herramienta se dividió el desarrollo en dos componentes, una librería que concentra la lógica propia del sistema de reglas y una aplicación web que permite el uso de dicha librería a partir de la carga de archivos, mapeos, tareas de deduplicación y exportación de resultados.

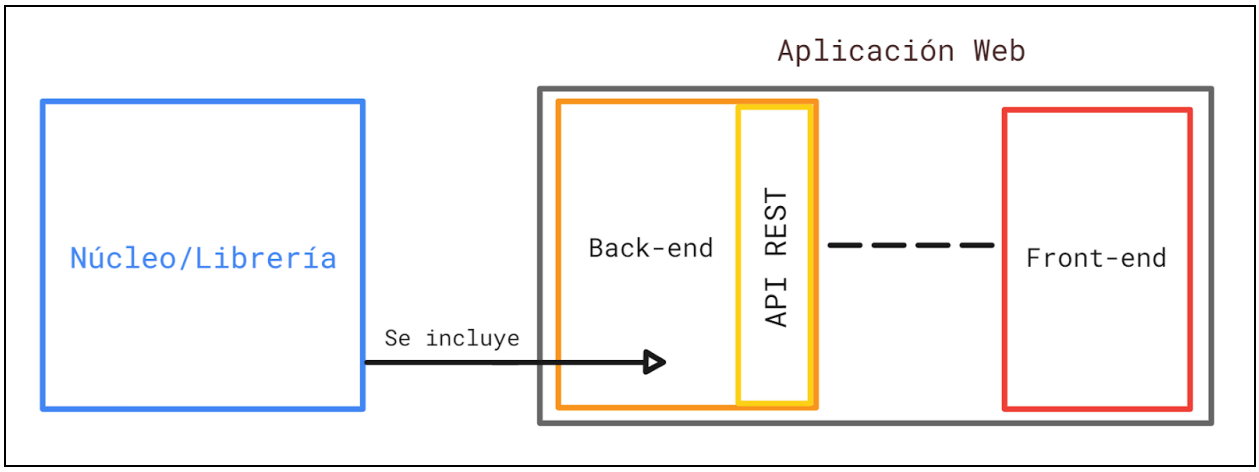

Figura 12. Arquitectura de la herramienta

1- Núcleo / Librería

En primer lugar, se desarrollaron los módulos que contienen la lógica de comparación de la herramienta y todas las funciones y clases necesarias para que la misma funcione correctamente. También se desarrolló una interfaz *CLI* para poder instanciar deduplicaciones desde consola. El factor principal por el cual se planteó la división del desarrollo en dos etapas fue la necesidad de lograr independencia entre la lógica de la herramienta, y la forma en que la misma puede ser utilizada. El desarrollo de módulos independientes que pueden ser instanciados desde consola, o incluidos en una aplicación web para ser accedidos a través de una API REST[6], provee una mayor flexibilidad y escalabilidad a la herramienta. El segundo factor por el cual se decidió desarrollar con este enfoque fue concentrarse en la lógica y funcionalidad de la herramienta dejando de lado las cuestiones propias de configuración de distintos frameworks web en la etapa inicial.

[6] REST es un estilo de arquitectura de software para aplicaciones web que permite el intercambio de información entre distintas aplicaciones a través del protocolo HTTP.

2- Aplicación web (Deduplicador)

Se propone el desarrollo de una API REST que exponga la funcionalidad del núcleo de la herramienta como servicio web. Esto le permite a la herramienta ser interoperable con otros sistemas y acoplarse con una interfaz de usuario amigable que permita el uso completo de la herramienta sin tener que escribir comandos en consola, lo cual puede resultar tedioso para usuarios no informáticos.

En base a las ventajas enunciadas en el párrafo anterior, se optó por desarrollar la herramienta como una aplicación web dividida en una aplicación back-end (desarrollada en Django) y una front-end (desarrollada en Angular).

Núcleo de la herramienta

El enfoque utilizado para la herramienta de deduplicación de registros utiliza una técnica de deduplicación basada en reglas y funciones de distancia. El núcleo de la herramienta se compone de todos los módulos necesarios para que a partir de dos listados de registros en formato CSV se genere un reporte de las coincidencias encontradas y los niveles de similitud definidos para cada registro del listado a deduplicar. Estos módulos se dividen en base a la funcionalidad de cada uno y se componen de una o más clases.

Uno de los objetivos principales del desarrollo fue definir una librería que contenga a las clases encargadas de gestionar el proceso de deduplicación de principio a fin, incluyendo tareas de mapeo de metadatos, y que la misma pueda utilizarse desde consola o incluirse en distintos sistemas.

Para el desarrollo del núcleo de la herramienta se utilizó el lenguaje de programación Python en su versión 3.7 (https://docs.python.org/3.7/). Para el intercambio y almacenamiento de los datos se utilizaron los formatos de archivo CSV y JSON (https://www.json.org/json-es.html).

El núcleo de la herramienta se compone de los siguientes módulos:

Engine	en este módulo se encuentra el algoritmo principal de comparación. Se realiza la iteración sobre los dos listados de registros y se llama al método de evaluación de cada regla. Se comunica con la clase ResultsCollector para gestionar los resultados de cada regla para cada par de documentos, y con la clase OutputFormatter para generar la salida de la deduplicación.
Rules	contiene la jerarquía de clases correspondiente a las reglas.
Utils	contiene funciones *helpers* para dar soporte a todos los módulos de la herramienta.
Comparator	contiene a las clases responsables de la comparación de autores, títulos y fechas.

Results Collector	contiene a la clase encargada de gestionar y almacenar los resultados de las evaluaciones que realizan las reglas sobre cada par de documentos.
Output Formatter	contiene a la clase encargada de generar la tupla de salida para cada documento, formateando los ids, el nombre de las reglas junto con su resultado, y similitud determinada.
Output Handler	módulo encargado de gestionar la escritura de la salida de la herramienta. Contiene una clase OutputHandlerFactory que devuelve el objeto OutputHandler correspondiente dependiendo el tipo de salida requerida (CSV, consola, base de datos, etcétera).
Crosswalk	módulo encargado de mapear registros de metadatos entre distintos esquemas, a partir de configuraciones personalizables y reutilizables. Las clases que componen este módulo y su funcionamiento se detallan al final del capítulo.

Tabla 2. Módulos del núcleo de la herramienta

Esquema de metadatos genérico

Dado que existe una amplia variedad de esquemas bajo los cuales los registros de metadatos son guardados y expuestos por los repositorios, resulta necesario definir un esquema de metadatos interno a la aplicación que sirva para almacenar y acceder a la información de cada registro de manera uniforme. Los registros que se ingresen como entrada a la herramienta deben respetar este esquema, por lo que resultará necesario realizar un mapeo de los registros a este esquema antes de iniciar una deduplicación.

Se propone un esquema genérico compuesto de los siguientes metadatos, siendo algunos de estos obligatorios y otros opcionales:

Metadato	Significado	Condición
id	Identificador del registro en el listado original	Obligatorio
title	Título	Obligatorio
subtitle	Subtítulo	Opcional
type	Tipología del registro	Obligatorio
author	Autor o autores	Obligatorio

date	Fecha de publicación	Obligatorio
doi	Identificador DOI	Opcional
isbn	ISBN	Opcional
issn	ISSN	Opcional
description	Resumen o abstract	Opcional

Tabla 3. Esquema de metadatos genérico

En la herramienta, cada registro es representado como un diccionario donde cada clave es alguno de los metadatos enunciados. Los registros se ingresan a la herramienta en formato CSV y cada fila es transformada en un diccionario que se guarda en el arreglo correspondiente: se crea uno para cada archivo CSV.

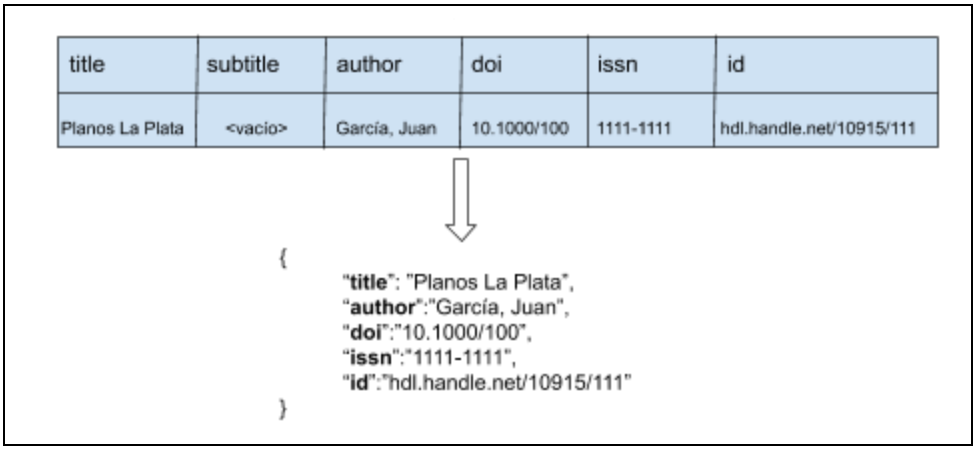

Figura 13. Representación de un registro dentro de la herramienta

Normalización de la tipología de cada registro

Los valores utilizados por los distintos repositorios digitales para definir el tipo de un documento son muy diversos puesto que dependen del estándar de metadatos utilizado, de un tesauro[7] específico o de reglas propias de cada repositorio. Por ejemplo, los artículos depositados en SEDICI se guardan como registros de tipo *'Artículo'*, en SCOPUS como *'Article'* y en CONICET Digital como *'info:eu-repo/semantics/article'* y '*info:ar-repo/semantics/artículo*'. Es por esto que la función encargada de normalizar la tipología de un registro se basa en una lista de posibles valores bajo los cuales se define la tipología de un documento en un repositorio, y que deben asociarse con cada valor definido en el enumerativo TypeEnum.

En base a la prueba de la herramienta con registros de distintos repositorios se agregaron variaciones nuevas para cada tipología de registro. A continuación se muestra una

[7] Un tesauro es una lista de palabras o términos controlados, empleados para representar conceptos.

tabla con las variaciones para cada tipología normalizada, en la primera versión de la herramienta (los valores ya se encuentran convertidos a minúscula y sin tildes).

ARTICLE	**BOOK**	**BOOKCHAPTER**
article	book	bookpart
articulo	libro	capitulo de libro
info:eu-repo/semantics/article	info:eu-repo/semantics/book	info:eu-repo/semantics/bookPart
info:ar-repo/semantics/articulo	info:ar-repo/semantics/libro	book chapter
review		info:ar-repo/semantics/capitulo de libro
paper		
THESIS		**CONFERENCE_OBJECT**
tesis	tesina	objeto de conferencia
tesis de grado	info:eu-repo/semantics/thesis	ponencia
tesis doctoral	info:eu-repo/semantics/tesis de grado	info:eu-repo/semantics/conferenceobject
tesis de maestria	info:eu-repo/semantics/doctoralthesis	info:ar-repo/semantics/objeto de conferencia
info:ar-repo/semantics/tesis doctoral	proyecto de tesis	
info:eu-repo/semantics/masterthesis	info:ar-repo/semantics/tesis de grado	

Tabla 4. Valores normalizados de las tipologías de un documento

A aquellos documentos que no se puede determinar su tipología se les asigna el tipo *UNKNOWN.*

Engine y algoritmo de comparación

Este módulo contiene a la clase *Engine*, encargada de gestionar el proceso de deduplicación de principio a fin. Este proceso involucra la iteración sobre los listados de registros y la evaluación del conjunto de reglas correspondiente sobre cada par de registros. La responsabilidad de gestión de resultados asociados a cada regla y escritura del reporte de salida es delegada a las clases *ResultsCollector, OutputFormatter y OutputHandler* respectivamente. Cada fila que será escrita en el CSV resultado se representa como un

diccionario creado por el *OutputFormatter* a partir de los resultados seleccionados por el *ResultsCollector.* Estos diccionarios se guardan en la variable de instancia *report_to_write* del objeto *OutputHandler y serán escritos a un archivo CSV una vez finalizada la comparacion de todos los registros.*

La transformación de los archivos CSV de entrada a las listas de documentos (diccionarios) es realizada por funciones que sirven como punto de entrada en la interfaz de consola definida.

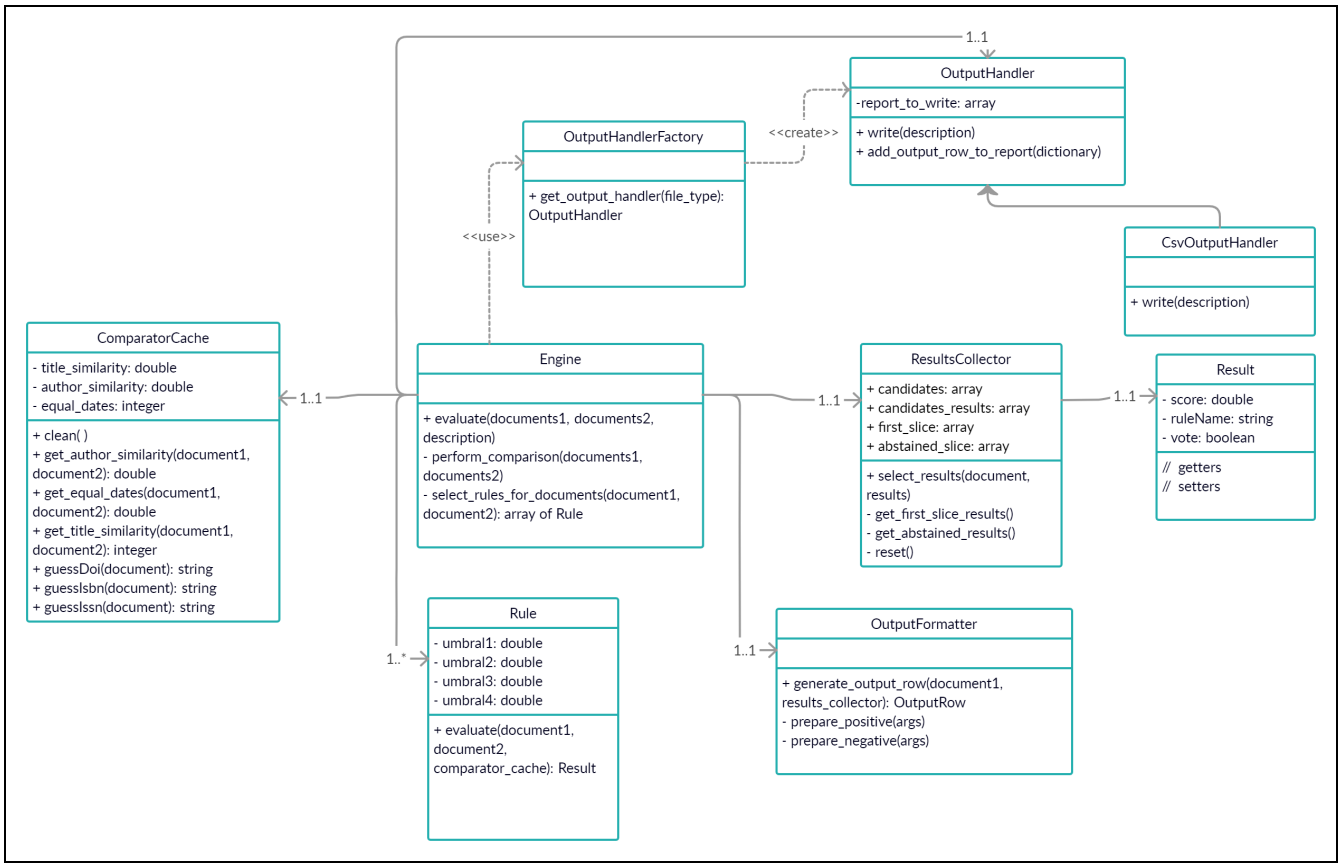

Figura 14. Diagrama UML de clase Engine y sus relaciones

Dado que para la deduplicación de registros es necesario comparar todos los registros de un listado, contra todos los registros de otro listado, el algoritmo base de la herramienta se encarga de iterar sobre ambos listados y ejecutar las reglas correspondientes para cada par de registros a comparar.

Durante una deduplicación se realizan NxM comparaciones, donde N hace referencia a la cantidad de documentos que se quieren importar al repositorio. Por ejemplo, en una deduplicación de registros para una ingesta a SEDICI, N es el listado de registros del repositorio desde el cual se desea obtener documentos para ingestar y M es el listado de registros presentes en SEDICI. Cada resultado generado por una regla tiene un puntaje asociado (0, 0.5, 0.75 o 1) que determina la similitud encontrada para cada par de registros y en base al mismo se determinan los resultados a guardar.

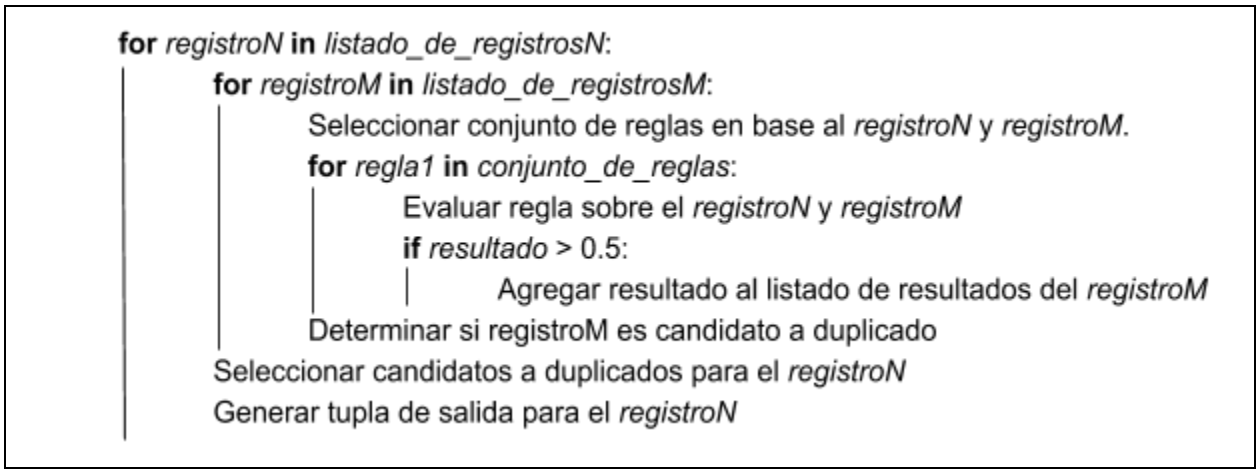

Figura 15. Pseudocódigo del algoritmo de comparación

Una vez finalizada la comparación, se guarda la lista que almacena los resultados finales de los N registros a un archivo CSV. Un ejemplo del resultado final para el registro con handle **10915/91176** cargado en SEDICI, luego de una deduplicación, es:

```
id_documentN: '10915/91176',
ids_documentsM: '11746/10511',
rules: {
            regla1: 1,
            regla2: 0.75
},
similarity: 'DUPLICATE'
```

A partir de este ejemplo se interpreta que para el registro con ***id* 10915/91176**, se encontró un registro duplicado con ***id* 11746/10511** donde las reglas **1 y 2** puntuaron con un valor de **1** y **0.75** respectivamente la similitud de los registros.

Reglas

Para calcular porcentajes de similitud entre pares de registros es necesaria la comparación y el análisis de los distintos metadatos que componen a los mismos. Dado que en las distintas fuentes de datos la mayoría de los objetos digitales usualmente no tienen un identificador persistente asociado, la deduplicación de registros no puede realizarse en un solo paso (que sería la comparación de estos identificadores), sino que involucra el análisis de múltiples metadatos. Dado que la tipología de los registros varía y que en base a esto varían los metadatos disponibles para realizar comparaciones, resulta natural pensar en combinar la comparación de distintos metadatos que tengan un criterio en común, por ejemplo: título de revista + título en artículos, o título del libro + número de capítulo + título en capítulos de libros.

Para cubrir la mayor cantidad de combinaciones posibles entre distintos metadatos, se plantea un sistema de reglas donde la responsabilidad de cada regla consiste en evaluar la similitud existente entre un subconjunto limitado de los metadatos de cada registro. Luego, en base a la comparación de los valores obtenidos a partir de la comparación de metadatos como autores, títulos o fechas (entre otros) contra niveles de aceptación predefinidos por cada regla, se determina un porcentaje de similitud general para el par de registros en particular.

En la tabla que se muestra a continuación se pueden ver los posibles valores generados por una regla y su significado.

Valor	**Significado**
1	Registros duplicados
0.75	Registros casi duplicados
0.5	Indefinido
0.25	(No se utiliza)
0	Registros no duplicados

Tabla 5. Puntajes de una regla

Cada uno de estos valores afirma con menor o mayor certeza la posibilidad de que el par de registros comparado haga o no referencia a la misma entidad.

Resultado asociado a la evaluación de una regla

La evaluación de una regla sobre un par de registros genera como resultado un objeto *Result* que contiene información sobre el nombre de la regla que lo produjo, el puntaje calculado (0, 0.5, 0.75 o 1) y un valor booleano que indica si la regla pudo o no votar. En algunos casos una regla no cuenta con información suficiente para determinar un puntaje de similitud, por ejemplo cuando no se encuentra un metadato requerido para las comparaciones de la regla en cuestión: si un registro no contiene DOI, la regla que evalúe este metadato no podrá votar, por lo tanto se abstiene.

Tipos de reglas

En un repositorio digital, la tipología asignada a un documento define el conjunto de metadatos utilizados para describir al mismo. Por ejemplo, un registro de metadatos de una tesis se compone, entre otros datos, de metadatos que describen al director, codirector y/o asesores profesionales de la tesis, mientras que un registro que describe a un capítulo de un libro se compone, entre otros datos, del título y el ISBN del libro al que pertenece el mismo.

La diferencia existente entre los metadatos utilizados para describir a los distintos tipos de documentos hace que las comparaciones necesarias para calcular la similitud entre un par de registros cambie según el tipo de los registros a analizar, ya que los metadatos disponibles

varían de un tipo de documento a otro. A raíz de esto, se definieron reglas específicas que evalúan pares de documentos en base a la tipología de los mismos, y que permiten que cada una pueda centrarse en el análisis de metadatos específicos según el tipo de documento a comparar.

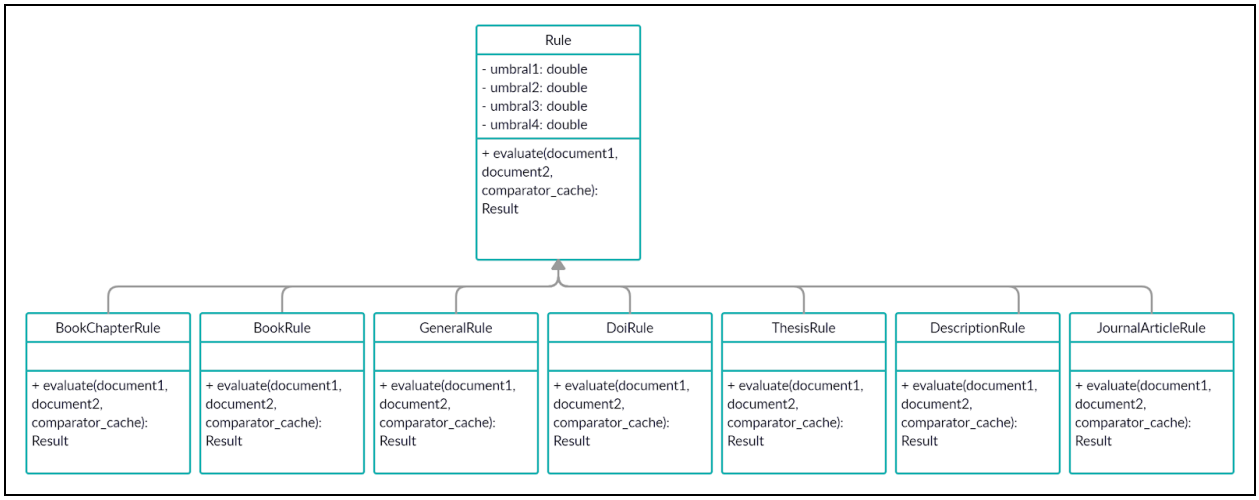

Figura 16. Diagrama UML del módulo de reglas

Las reglas definidas en la primera versión de la herramienta son:

Nombre de la clase	**Aplicabilidad**	**Metadatos utilizados**
GeneralRule	Todos	Título, autores y fecha
JournalArticleRule	Artículos	ISSN, título, autores y fecha
BookRule	Libros	ISBN y título
ThesisRule	Tesis	Título y autor/es
BookChapterRule	Capítulos de libro	ISBN, título del libro, título del capítulo
DoiRule	Todos	DOI y título
AbstractRule	Todos	Resumen o abstract

Tabla 6. Tipos de reglas definidas

- Regla general (clase GeneralRule): esta regla es evaluada siempre, independientemente del tipo de los registros a comparar. Para el cálculo del resultado utiliza el porcentaje de similitud entre títulos, autores y fechas.

- Regla para artículos (clase JournalArticleRule): esta regla es evaluada únicamente cuando los dos registros a comparar son artículos. Se basa en la comparación de los ISSN de cada artículo en conjunto con la similitud entre títulos, autores y fechas. A diferencia de la regla general, los umbrales contra los que se comparan estas similitudes son menos estrictos, ya que se le da mayor importancia a la comparación de los ISSN.

- Regla para libros (clase BookRule): esta regla es evaluada únicamente cuando los dos registros a comparar son libros. Se basa en la comparación de los ISBN de cada libro, en caso que los mismos coincidan, sólo tiene en cuenta la similitud entre los títulos. Caso contrario, se basa en la similitud entre títulos, autores y fechas.

- Regla para tesis (clase ThesisRule): esta regla es evaluada únicamente cuando los dos registros a comparar son tesis. Se basa en la comparación de títulos y autores únicamente.

- Regla para capítulos de libros (clase BookChapterRule): esta regla es evaluada únicamente cuando los dos registros a comparar son capítulos de libros. Se basa en la comparación de los ISBN de cada uno, y luego en la similitud de sus títulos.

- Regla DOI (clase DoiRule): esta regla es evaluada siempre, independientemente del tipo de los registros a comparar, y sólo vota sí ambos registros cuentan con el metadato DOI. En caso que los DOI sean iguales, se basa en la similitud de los títulos para mayor confiabilidad, puesto que un documento podría tener mal cargado el metadato DOI.

- Regla para resúmenes (clase AbstractRule): esta regla es evaluada siempre, independientemente del tipo de los registros a comparar, y se basa en la comparación de los resúmenes de cada registro, teniendo en cuenta la longitud de cada uno, y la presencia de substrings en común elegidos aleatoriamente.

Elección del conjunto de reglas a evaluar

La clase Engine encargada de iterar sobre los listados de registros y efectuar la evaluación de cada regla, posee entre sus atributos un diccionario que contiene diferentes conjuntos de reglas. Cada uno de estos conjuntos define las reglas que podrán ser evaluadas en base a la tipología de los registros a comparar. Para la elección del conjunto de reglas a evaluar se obtiene la tipología de cada registro y en base a la misma se define el conjunto de reglas a utilizar. Este proceso se repite por cada par de registros a comparar durante una tarea de deduplicación.

```
class Engine():

   rule_map = {
       1:[JournalArticleRule()],
       2:[ThesisRule()],
       3:[BookIsbnRule()],
       4:[BookPartRule()],
       5:[GeneralRule(), DoiRule(), AbstractRule()]
   }
```

Las claves del diccionario se corresponden con el valor definido en el enumerativo *TypeEnum*, utilizado para normalizar y definir la tipología de cada registro.

```
class TypeEnum(Enum):
   ARTICLE = 1
   THESIS = 2
   BOOK = 3
   BOOKPART = 4
   UNKNOWN = 5
   CONFERENCEOBJECT = 6
```

Cabe destacar que para la elección del conjunto de reglas a evaluar se tiene en cuenta la tipología de ambos registros, es decir, si se compara el registro A con el registro B, para poder evaluar el conjunto de reglas 1 (JournalArticleRule) entonces tanto el registro A como el B deben ser de tipo *ARTICLE.* Además, el conjunto 5 contiene las reglas que pueden ser evaluadas independientemente de la tipología de los registros, por lo tanto, para cada par de registros siempre se evalúa el conjunto de reglas número 5 + el conjunto de reglas correspondiente en base a la tipología de los mismos.

Lógica de comparación de las reglas

Cada regla define un puntaje que refleja la similitud existente entre el par de registros evaluados basándose en los valores de similitud obtenidos para cada metadato en particular y los niveles de aceptación predefinidos en cada regla. Los porcentajes de similitud obtenidos para cada metadato a analizar por la regla se comparan contra los umbrales definidos (niveles de aceptación) formando un árbol de decisión en base al cual se determina el puntaje final: 0, 0.5, 0.75 o 1. Cada una de las ramas de este árbol de decisión (representado en el código por cadenas de sentencias *if/else*) contempla diferentes niveles de aceptación, y son las que finalmente catalogan a un par de registros como duplicados, posibles duplicados, no duplicados o con resultado indefinido.

A modo de ejemplo, a continuación se muestra un diagrama de decisión que representa la lógica correspondiente al método *evaluate* de la clase *GeneralRule*:

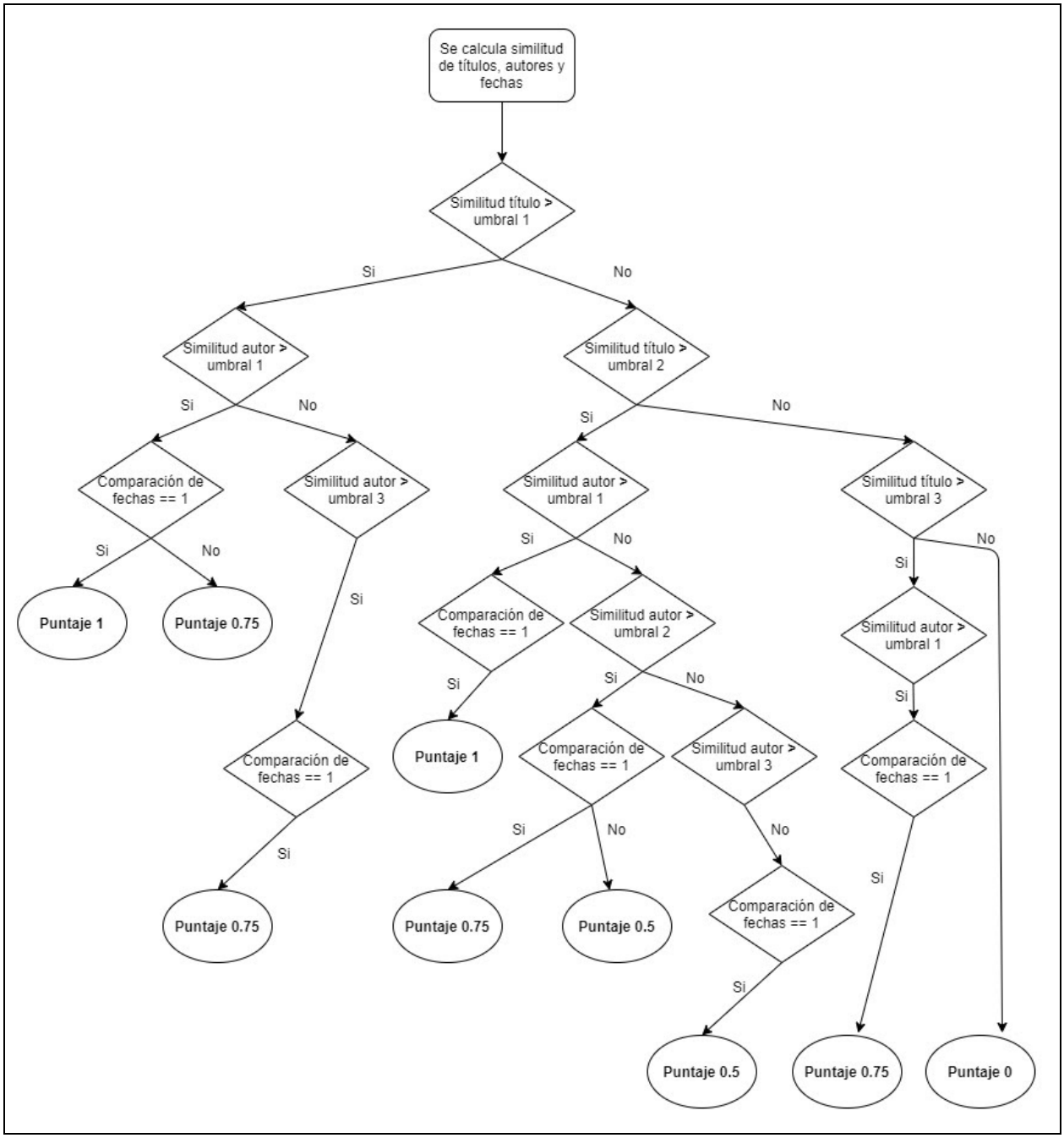

Figura 16. Diagrama representativo de la lógica definida para la clase GeneralRule

La similitud entre títulos, autores y fechas es calculada por las funciones definidas en el módulo de comparación correspondiente a cada metadato en particular.

Comparación de metadatos

En este módulo se agrupan las clases encargadas de la comparación y el cálculo de la similitud de tres metadatos básicos, que son utilizados por la mayoría de las reglas: títulos, autores y fechas. Para cada uno de estos metadatos existe una clase responsable de analizar y determinar la similitud existente para cada par de registros dado. La clase *ComparatorCache* brinda la funcionalidad de caché sobre las similitudes calculadas para un mismo par de registros a las distintas reglas.

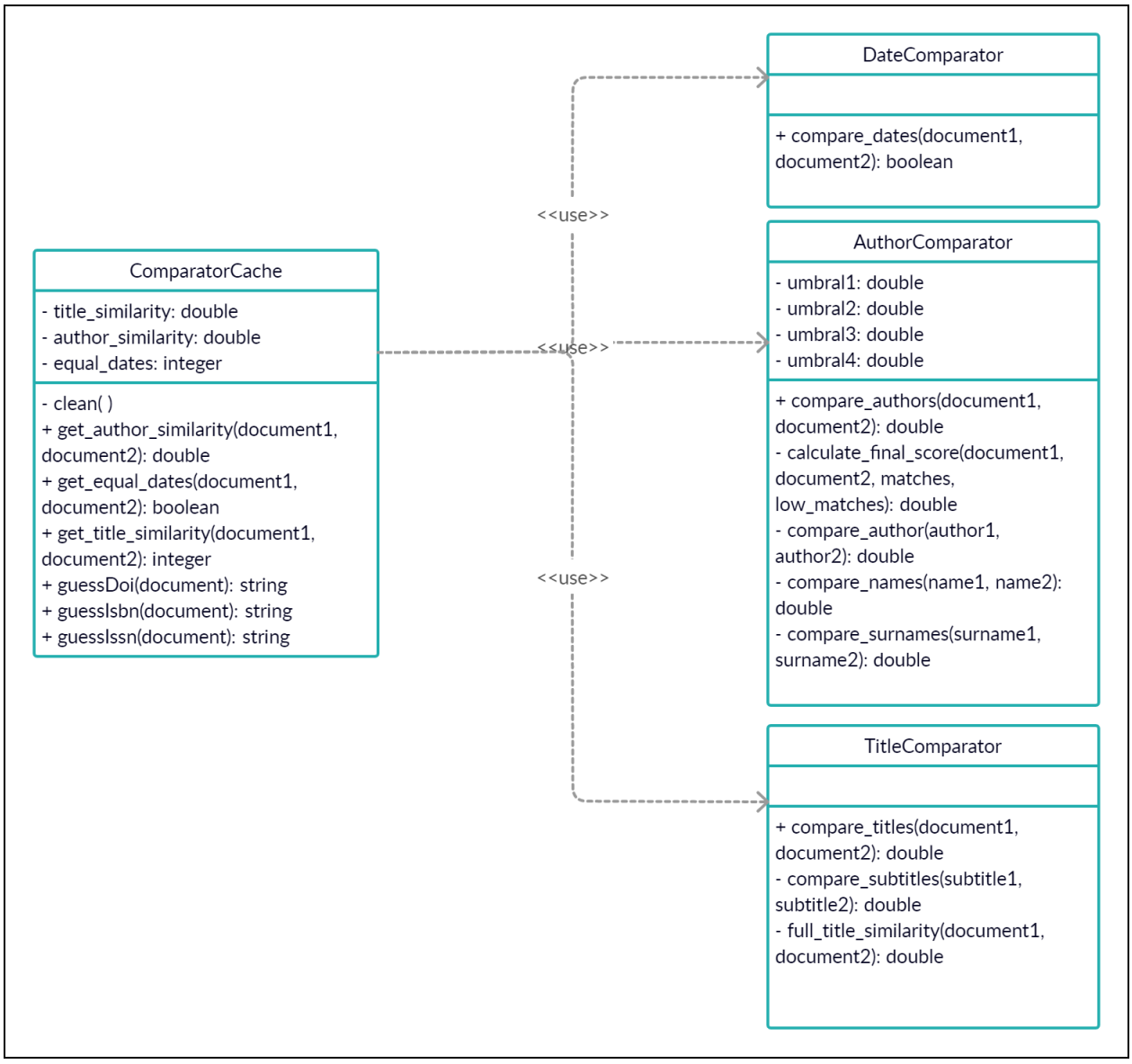

Figura 17. Diagrama UML del módulo de comparación

Comparación de autores

La clase *AuthorComparator* define un método público llamado *compare_authors* que calcula el porcentaje de similitud existente entre el listado de autores de dos registros. Este método hace uso de otros métodos definidos dentro de la misma clase que son: *compare_author,* que compara un par de autores en particular, *compare_names* que compara un par de nombres, *compare_surnames* que compara un par de apellidos y *calculate_final_score* que se encarga de generar el puntaje final en base a las coincidencias encontradas entre los distintos autores comparados. Se distingue entre coincidencias altas y coincidencias bajas, teniendo estas últimas menos peso a la hora de calcular el puntaje final. Dada la gran cantidad de casos que deben contemplarse y puesto que no todas las coincidencias tienen el mismo grado de precisión se optó por distinguir entre estos dos niveles.

A continuación, se muestran algunos ejemplos de comparación de autores y el nivel de coincidencia establecido.

Nombre autor 1	**Nombre autor 2**	**Nivel de coincidencia**
García, Juan	García, J.	Alto
Fernández García, Juan	Fernández G., Juan	Alto
Perez, J.	Perez García, J.	Alto
de la Paz Diulio, María	Diulio, María de la Paz	Bajo
Perez, J.	Peres, J.	No hay coincidencia
García, Gabriel	García, María	No hay coincidencia
Fernández, Alfredo Horacio	Fernández, Horacio	Bajo

Tabla 6. Ejemplos de coincidencia entre autores

Los niveles de coincidencia se determinan en base al nivel de similitud calculado para cada par de autores, y para esto se contemplan apellidos iguales, apellidos compuestos con partes escritas en el nombre, nombres compuestos, nombres completos e iniciales, entre otras variaciones. La dificultad principal que se encontró cuando se trabajó sobre la comparación de autores fue la definición de funciones que contemplen diversos casos de escritura en los nombres sin dejar pasar como coincidentes otros casos en los que no existía una coincidencia real (falsos positivos).

El método principal *compare_authors* itera sobre las listas de autores recibidas y compara uno a uno los autores llamando al método *compare_author,* el cual a su vez se encarga de dividir el nombre completo en nombre y apellido e invocar a los métodos *compare_names* y *compare_surnames* respectivamente. Estos métodos hacen uso de la función de similitud para strings descrita en párrafos anteriores para comparar los nombres y

los apellidos y tienen en cuenta además la presencia de nombres abreviados, iniciales y nombres compuestos. El método *calculate_final_score* se basa en la cantidad de coincidencias bajas y altas encontradas y la cantidad de autores presentes en cada listado.

Debido a restricciones sobre los metadatos expuestos por cada repositorio, la comparación de autores es meramente sintáctica, ya que no se cuenta con información adicional de cada autor (coautoría, temas de interés, grupos de trabajo, identificadores universales) para poder definir relaciones semánticas entre los mismos/as. Si bien esto es una limitación, ya que pueden darse situaciones donde se comparen dos autores distintos que tienen el mismo nombre y se obtenga un porcentaje de similitud muy elevado, cabe destacar que la similitud final calculada para un par de registros se determina en conjunto con el análisis de otros metadatos, y no únicamente teniendo en cuenta los autores de los mismos.

Comparación de títulos

Si bien el hecho de comparar dos strings para obtener una medida de similitud se soluciona con la función mencionada en párrafos anteriores, las variaciones posibles que pueden darse en los títulos de ciertos registros guardados en distintos repositorios son muy extensas y hacen que la tarea de comparación de títulos se vuelva más compleja.

Las dificultades encontradas durante las pruebas realizadas junto al desarrollo de la herramienta, en las cuales se analizaron registros de metadatos de documentos obtenidos a partir de múltiples repositorios son:

- Título y subtítulo guardados en el mismo metadato separado con los caracteres '**.**' y '**:**' indistintamente. Esto dificulta el parseo del metadato porque no se tiene un delimitador común para definir qué parte del string es el título y cual el subtítulo. Ej:
    - <dc.title>Breviarios BBA: La Reforma Universitaria</dc.title>
    - <dc.title>Breviarios BBA. La Reforma Universitaria</dc.title>
    - <dc.title>Breviarios BBA</dc.title>
    - <dc.title.subtitle>La Reforma Universitaria</dc.title.subtitle>

- Títulos guardados en distintos idiomas. Ej:
    - <dc.title>*Relação entre gene e resistência Rps 1k com resistência a Pythium ultimum e P. irregulare em soja*</dc.title>
    - <dc.title>Relación entre el gen Rps 1k y la resistencia a Pythium ultimum y P. irregulare en soja</dc.title>

- Errores de tipeo en los títulos

- Diferencias en la estructura de los metadatos. Ej:
    - <dc.title>Título. Subtítulo </dc.title>
    - <dc.title>Subtítulo. Título </dc.title>

- Información adicional en el título de un registro. En algunos casos se encuentra información sobre la tipología del registro, los autores o la edición de la revista a la cual pertenece un artículo en el título. Ej:
    - <dc.title>ProBiota | Serie Técnica y Didáctica | Lista de peces de la provincia de Entre Ríos</dc.title>
    - <dc.title>Lista de peces de la provincia de Entre Ríos</dc.title>

En base a las observaciones listadas y como resultado de un proceso de iteración en el que se probaron distintas técnicas y umbrales de aceptación, se definió una clase encargada de comparar y calcular la similitud entre títulos de dos registros, que contempla una gran cantidad de casos especiales. Esta clase se llama *AuthorComparator* y permite la comparación de títulos con las siguientes precondiciones:

- el título y el subtítulo se encuentran ambos en el metadato *title* separados por un punto o dos puntos.
- el título y el subtítulo se encuentran separados en los metadatos *title y subtitle* respectivamente.

Además se contemplan situaciones en las que un registro posee el título en varios idiomas (todos en la clave *title* y separados por el caracter separador de la herramienta '|'). En este caso, luego de iterar sobre el listado de títulos y comparar los mismos uno a uno, el puntaje final es el mayor porcentaje de similitud calculado entre cada par de títulos analizados.

Comparación de fechas

La clase que contiene la lógica definida para determinar si dos registros poseen fechas de publicación iguales contempla únicamente los años de publicación de cada uno de los registros, sin tener en cuenta el día y el mes en que fueron publicados. Esta elección se fundamenta en que los repositorios no siempre exponen la fecha completa de publicación de un registro, y si lo hacen, los formatos utilizados para guardar las fechas muchas veces varían dependiendo el repositorio.

Si se analiza la fecha '05-11-2018' no es posible determinar si se hace referencia al día 5 del mes de noviembre o al día 11 del mes de mayo, excepto que se cuente con información sobre el formato utilizado (DD-MM-AAAA, MM-DD-AAAA). En cualquier caso donde el año de publicación se guarda sin abreviaturas, extraerlo de la fecha completa es muy fácil, ya que se debe iterar sobre el string separado por guiones medios y buscar un número de 4 dígitos.

También existen casos en los que un registro de metadatos posee más de una fecha asociada al mismo porque por ejemplo se guardan fechas de publicación de las distintas versiones por las que pasa un artículo (preprints, postprints y versiones publicadas).

En base a las observaciones mencionadas, se desarrolló el comparador de fechas de tal manera que:

- Genera un puntaje para determinar si dos registros tienen fechas de publicación iguales (1) o no las tienen (0).
- Compara únicamente los años de cada fecha.

- Realiza una comparación de todas contra todas (en caso que se cuente con más de una fecha en alguno de los dos registros a comparar).

Ejemplo para el cual se obtiene un 1 como puntaje:
- Registro 1: <dc.date>10-20-2019|01-01-2020</dc.date>
- Registro 2: <dc.date>2019</dc.date>

Ejemplo para el cual se obtiene un 0 como puntaje:
- Registro 1: <dc.date>10-20-2019|01-01-2020</dc.date>
- Registro 2: <dc.date>10-20-2018</dc.date>

Cabe destacar que en aquellos casos donde al menos uno de los dos registros a comparar no tiene una fecha asociada, el puntaje devuelto es 0.

Auxiliar *utils*

Este módulo contiene todas las funciones que dan soporte a los distintos módulos de la herramienta y que por cuestiones de reusabilidad no deben quedar atadas a un módulo en particular con lógica y funcionalidad específica. Aquí se definen las funciones que sirven para:

- Calcular la similitud entre dos strings.
- Obtener el ISSN de un registro.
- Obtener el ISBN de un registro.
- Obtener el DOI de un registro.
- Normalizar un string (convertir a minúscula, eliminar espacios en blanco repetidos, eliminar tildes y tags html)
- Normalizar el tipo de un registro.

Funciones de similitud entre strings

Como se mencionó a lo largo de los capítulos 2 y 3, los metadatos a analizar y comparar en busca de similitudes presentan diferencias sintácticas producto de errores de tipeo o diferencias en los esquemas y formatos bajo los cuales fueron cargados. Para calcular porcentajes de similitud existen numerosos algoritmos que utilizan distintos enfoques para determinar en que medida dos strings son similares. Las reglas para calcular esta similitud varían dependiendo el algoritmo a utilizar, algunos calculan la distancia que existe entre un string A y un string B, siendo está la diferencia de caracteres que existe entre uno y otro; y otros por ejemplo calculan la presencia de substrings de A en B.

Con el fin de elegir un algoritmo que se adecue al caso de uso específico de este trabajo de tesis se analizaron y realizaron pruebas con los siguientes tres algoritmos:

Distancia Levenshtein

La distancia Levenshtein es (Gilleland, 2006) una medida de la similitud entre dos strings que calcula el número de inserciones, eliminaciones o sustituciones de caracteres necesarias para transformar un string A en un string B. Mientras mayor sea este número, más

diferentes son los strings A y B. Por ejemplo**,** **l**a distancia Levenshtein entre ‘heladera’ y ‘helareda’ es de 2, porque se necesitan 2 sustituciones de caracteres para transformar una palabra en otra:

heladera -> hela**r**era-> helare**d**a

Para calcular esta distancia se utilizó el módulo *python-Levenshtein* [https://pypi.org/project/python-Levenshtein/] de Python que provee la función *distance(stringA, stringB)*.

```
>>> Levenshtein.distance('heladera', 'helareda')
2
```

Distancia Jaro-Winkler

La distancia de Jaro define un porcentaje de similitud entre dos strings y está definida por la fórmula:

$$sim_j = \begin{cases} 0 & \text{if } m = 0 \\ \frac{1}{3}\left(\frac{m}{|s_1|} + \frac{m}{|s_2|} + \frac{m-t}{m}\right) & \text{otherwise} \end{cases}$$

Figura 18. Distancia Jaro-Winkler (Jaro-Winkler distance, s.f.)

donde (Jaro-Winkler distance, s.f.) *m* es la cantidad de caracteres que coinciden en el string *s1* y *s2* sin importar el orden de los mismos, |*s1*| es la longitud del string *s1* (lo mismo se cumple para *s2)* y t es la cantidad de transposiciones (permutaciones) necesarias para convertir el string *s1* en el string *s2.*

La distancia de Jaro-Winkler es una adaptación de la distancia Jaro que utiliza un prefijo *p* para favorecer a strings que son exactamente iguales desde el inicio hasta un prefijo de longitud *l,* y se define como:

$$sim_w = sim_j + \ell p(1 - sim_j),$$

donde *simj* es la similitud Jaro entre los strings *s1 y s2, l* es la longitud del prefijo definido y *p* es un factor de ajuste para definir el peso que tiene la coincidencia de caracteres en el prefijo definido.

Para calcular esta distancia se utilizó el módulo *pyjarowinkler* [https://pypi.org/project/pyjarowinkler/] que provee la función *get_jaro_distance()* que permite calcular la similitud Jaro, o Jaro-Winkler enviando el parámetro *winkler* en *True*.

Metaphone

Metaphone es un algoritmo fonético (Metaphone, s.f.) que sirve para indexar palabras en base a su pronunciación en inglés. Este algoritmo no calcula por sí mismo similitudes entre strings, sino que genera una clave en base a la pronunciación de la palabra indicada, y por ende palabras similares deberían compartir la misma clave.

Para probar este algoritmo se utilizó el módulo *metaphone* de Python [https://pypi.org/project/Metaphone/#id1] que provee dos funciones: metaphone*(string)* y *doublemetaphone(string)*. Esta última es una variante del algoritmo original metaphone, que genera una clave adicional para mayor exactitud.

Considerando los strings 'Las obras literarias' y 'Obras literarias', las claves generadas para cada string son ('*LSPRSLTRRS')* y ('*APRSLTRRS')* respectivamente. Si bien comparten ciertos caracteres, son diferentes a simple vista, y no representan ninguna métrica de similitud relevante que sirva a las necesidades de la herramienta, ya que ambos strings podrían ser títulos de una misma obra, con un error de escritura cuando el mismo fue cargado.

Función utilizada

A partir de las pruebas realizadas con cada uno de los algoritmos mencionados, se decidió tomar la distancia Levenshtein como método para calcular similitud entre strings. Los casos a tener en cuenta cuando se comparan metadatos de registros depositados en repositorios digitales se componen en gran medida de errores y diferencias de tipeo en los datos, y la variedad de estructuras bajo las cuales los mismos son guardados, y de esta manera la distancia Levenshtein resulta muy útil para determinar diferencias o similitudes entre los mismos, ya que tiene en cuenta la cantidad de caracteres diferentes que posee un string en comparación a otro. Además se realizó una prueba para comparar el tiempo que tardan las librerías utilizadas en calcular la distancia Levenshtein versus la distancia Jaro y los resultados fueron favorables para Levenshtein, como se muestra en la tabla a continuación:

Longitud de los *strings*	**Distancia**	**Tiempo de computación**
8 caracteres	Levenshtein	0.449 ms
	Jaro	0.47 ms
45 caracteres	Levenshtein	0.014 ms
	Jaro	0.287 ms

Tabla 7. Tiempo de computación de la distancia Levenshtein versus Jaro

La función que se programó para la herramienta hace uso del cálculo de la distancia Levenshtein entre dos strings, y determina en base a la misma un porcentaje de similitud. A continuación se muestra el código de la función implementada en el módulo **utils.py:**

```
def calculateDistance(string1, string2):
      try:
          text_distance = Levenshtein.distance(string1, string2)
          max_len = max(len(string1),len(string2))
          normalized_distance = 1 - text_distance / max_len
      except:
          normalized_distance = 0
      return normalized_distance
```

Ejemplo

Supongamos el *string* A que contiene el valor '*Este es un string de prueba*' y el *string* B '*Este es otro string de puerba*':

1. Se calcula la distancia Levenshtein entre ambos strings, que da como resultado el valor 6, porque se necesita sumar cuatro caracteres para agregar la palabra 'otro' al string A, y dos permutaciones en la palabra 'prueba' para transformarla en 'puerba', como aparece en el string B.
2. Luego se calcula la longitud de ambos strings y se obtiene la longitud mayor, que en este caso es la longitud del string B (29) y se la utiliza para dividir la distancia Levenshtein calculada.
3. Una vez calculado el porcentaje de diferencia entre el string A y el string B (0.2) se determina el porcentaje de similitud, que puede definirse como 1 - *{porcentaje de diferencia} = {porcentaje de similitud}*.

En este caso y según la función definida, la similitud existente entre el string A y el string B es de un 80%.

Obtención de identificadores

Las funciones que se encargan de obtener identificadores evalúan diferentes expresiones regulares[8] definidas para cada caso particular sobre los metadatos correspondientes (ISSN, ISBN, DOI). Esto es así porque el formato en que se encuentran los identificadores dentro de cada registro muchas veces varía, especialmente para el metadato DOI. Por ejemplo, un mismo identificador DOI puede encontrarse con las siguientes variaciones:
doi:10.15517/rbt.v58i1.5194, *https://doi.org/10.15517/rbt.v58i1.5194* ó *10.15517/rbt.v58i1.5194.*

Resultado de una deduplicación

Los resultados generados por cada regla son gestionados por la clase *ResultsCollector* que se encarga de seleccionar y mantener en memoria los resultados para cada registro del

[8] En informática, las expresiones regulares se utilizan para la búsqueda de patrones u operaciones de sustituciones en cadenas de caracteres.

listado a deduplicar. Está selección de resultados se realiza en base al puntaje de los mismos y se descartan todos aquellos resultados de reglas que hayan puntuado con 0 la similitud entre el par de registros. Cuando se finaliza la comparación de un registro contra todos los registros del otro listado, la clase *OutputFormatter* se encarga de generar, en base a los resultados obtenidos, la fila correspondiente al registro que será escrita en el reporte de deduplicación una vez finalizado el procesamiento de todos los registros. Está escritura es realizada en un archivo CSV por la clase *OutputHandler*.

A continuación se muestra una tabla que representa una porción de un reporte generado a partir de una deduplicación entre CONICET Digital y SEDICI.

id_document1	id_document2	rules	similarity
https://ri.conicet.gov.ar/handle/11336/14093	http://sedici.unlp.edu.ar/handle/10915/36493	JournalArticleRule=1 | GeneralRule=0.75 | DoiRule=A	DUPLICATE
https://ri.conicet.gov.ar/handle/11336/69795	http://sedici.unlp.edu.ar/handle/10915/53734	JournalArticleRule=1 | GeneralRule=0.75 | DoiRule=A	DUPLICATE
https://ri.conicet.gov.ar/handle/11336/9668	http://sedici.unlp.edu.ar/handle/10915/72967	GeneralRule=0.75 | DoiRule=A|	NEAR_DUPLICATE
https://ri.conicet.gov.ar/handle/11336/69595	None	None	NO_DUPLICATE

Tabla 8. Resultado ejemplo de una deduplicación

Mapeo de metadatos

Dado que la herramienta de deduplicación utiliza un esquema de metadatos genérico para representar a cada registro y es necesario realizar esta transformación desde el esquema original, y visto que el mapeo es una tarea recurrente en el proceso de importación de registros a repositorios, y que no existen herramientas que cumplan esta función en este dominio y con la flexibilidad necesaria, se desarrolló un módulo dentro de la herramienta de deduplicación cuyo propósito específico es el de mapear un conjunto de registros a un esquema de metadatos diferente, y que permite facilitar y automatizar el conjunto de tareas relacionadas al mapeo: combinación, eliminación y procesado de los metadatos.

Las clases que componen a este módulo se pueden observar en el siguiente diagrama UML:

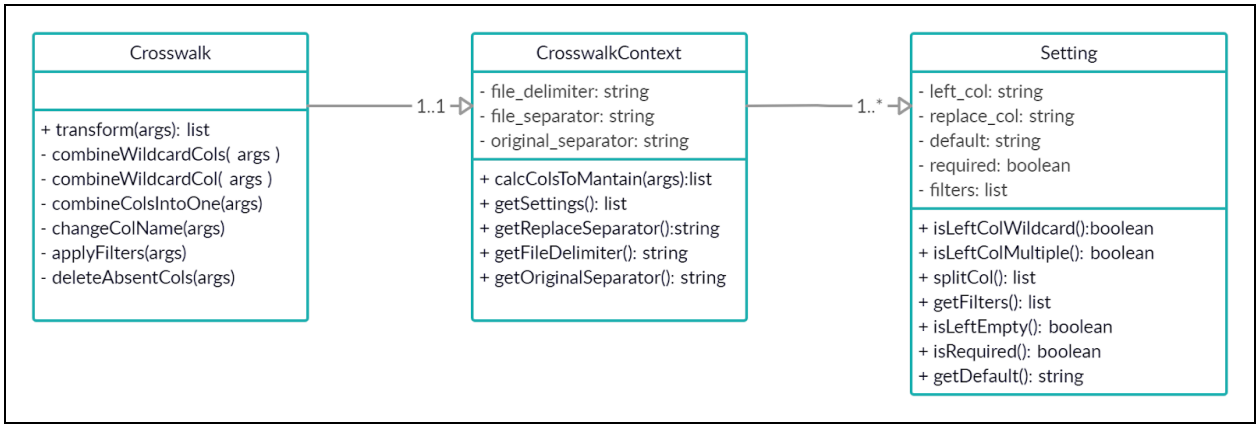

Figura 19. Diagrama UML del módulo de mapeo

Funcionamiento

Al pensar en el desarrollo del módulo de mapeo se tuvo en cuenta principalmente la flexibilidad para poder definir distintas opciones de mapeo fácilmente, y la reusabilidad de una 'configuración' de mapeo en distintos escenarios, es por eso que surge la idea de poder definir configuraciones externas a la lógica de la herramienta, por ejemplo en un archivo con notación JSON[9], que defina las reglas básicas de mapeo y que sirva al módulo para saber qué metadatos deben reemplazarse y cómo. En base a esta premisa, el módulo desarrollado permite realizar mapeos entre esquemas de metadatos, a partir de archivos de configuración personalizables, combinar o eliminar metadatos y definir para cada metadato una serie de filtros a ser aplicados sobre los mismos, como por ejemplo la eliminación de caracteres especiales, espacios en blanco, transformación a mayúsculas y/o minúsculas.

El prototipo desarrollado se basó en una herramienta cuya interfaz es accedida mediante consola indicando los parámetros correspondientes:

- ruta al archivo CSV a mapear,
- ruta al archivo de configuración JSON, y
- una cadena de texto indicando el nombre del nuevo archivo.

A continuación se muestra un ejemplo de invocación al comando utilizado para instanciar la herramienta a partir de la CLI:

```
      python3 crosswalk_cli.py archivo-original.csv
configs/configuracion.json archivo-mapeado.json
```

Cada configuración se guarda en un archivo en notación JSON, indicando para cada columna el mapeo y los filtros correspondientes. A continuación se muestra una configuración de ejemplo donde se mantiene la columna *dc.title* con el mismo valor, se realiza el mapeo de *dc.author* a *sedici.creator.person*, de *Año* a *dc.date.issued,* se combinan las columnas *Número* y *Congreso* para mapearse a *sedici.relation.event* y se mapea de *issn* a *sedici.identifier.issn.*

[9] JSON es un formato de texto sencillo para el intercambio de datos con estructura clave-valor.

```
[
    [
         {
            "left":"dc.title",
            "replace":"dc.title",
            "default":"",
            "required":false
        },
        {
            "left":"dc.author",
            "replace":"sedici.creator.person",
            "default":"",
            "required":false
        },
        {
            "left":"Año",
            "replace":"dc.date.issued",
            "default":"",
            "required":false
        },
        {
            "left":"Numero+Congreso",
            "replace":"sedici.relation.event",
            "default":"",
            "required":false
        },
        {
            "left":"issn",
            "replace":"sedici.identifier.issn",
            "default":"",
            "required":false
        }
    ],
    {
        "original_separator":"||",
        "replace_separator":"&",
        "file_delimiter":","
    }
]
```

En la siguiente tabla se detalla la función de cada clave utilizada en la configuración:

Clave	Significado
left	Define la o las columnas del CSV de entrada que van a ser mapeadas a la columna definida en la clave replace. En caso que una columna no quiera mantenerse en el archivo CSV destino, no debe indicarse la misma en el archivo de configuración para que la herramienta la ignore.
replace	Nombre con el que se reemplazarán las columnas indicadas en la clave left.
default	Define el valor que toma por defecto la columna (en caso de no tener un valor presente).
required	Determina si es obligatorio que el campo tenga un valor original para mantener la fila, en caso que el campo esté vacío y el valor de required sea true, se elimina la fila y no se la incluye en el archivo CSV destino.
filter	Define los filtros a aplicar sobre el dato. Los filtros existentes actualmente son trim y lowercase, y pueden combinarse múltiples filtros utilizando el carácter '|' como separador. Es opcional.
original_separator	Aplica sobre todos los metadatos y define el delimitador de campos multivaluados usado en el archivo CSV de entrada.
replace_separator	Aplica sobre todos los metadatos y define el delimitador de campos multivaluados que va a ser utilizado en el archivo CSV de salida.
file_delimiter	Define el delimitador del archivo de entrada para casos en los que se ingrese un archivo en formato TSV (Tab Separated Values) en vez de CSV, o se utilice algún delimitador personalizado.

Tabla 9. Significado de las claves del archivo de configuración

Combinación de columnas

Para el mapeo de dos o más columnas a una única columna destino se utiliza el carácter '+' como operador de unión, que tiene como efecto la concatenación de los valores de las columnas indicadas. También es posible definir expresiones regulares en la clave *left* para combinar columnas que cumplen con un determinado patrón. Por ejemplo, cuando un mismo metadato puede ser guardado en distintos idiomas, se indica el mismo entre corchetes '[ ]'. Si se quiere mapear todos los metadatos dc.title[es], dc.title[en], dc.title[fr] a un único dc.title que contenga todos los valores posibles para el título, podría definirse la configuración:

```
{
    "left": "dc.title*",
    "replace": "dc.title",
    "default":"",
    "required":"true"
}
```

la cual es equivalente a:

```
{
    "left": "dc.title[es]+dc.title[en]+dc.title[fr]",
    "replace": "dc.title",
    "default":"",
    "required":"true"
}
```

Capítulo 5 - Desarrollo de la aplicación web

Introducción

En el capítulo 4 se detalló el análisis e implementación de una herramienta de deduplicación de registros de metadatos y se presentaron los módulos principales de la misma. Al inicio del capítulo se planteó la metodología de trabajo a seguir y se dividió el desarrollo en dos grandes partes: librería (funcionamiento y módulos principales) y aplicación web.

El objetivo de este capítulo es detallar la implementación de la herramienta en el contexto de una aplicación web y presentar las pantallas principales de la misma. La arquitectura de la aplicación web a desarrollar se basa en un modelo cliente-servidor, donde el cliente es una aplicación front-end[10] y el servidor es una aplicación back-end. En esta última es donde se implementa la funcionalidad de la herramienta de deduplicación desarrollada.

Tecnologías utilizadas

- Para el desarrollo de la aplicación back-end se utilizó el framework para desarrollo web Django (https://www.djangoproject.com/) y el framework para desarrollo de APIs REST Django REST (https://www.django-rest-framework.org/).
- Para el desarrollo de la aplicación front-end se utilizó el framework Angular en su versión 9 con el lenguaje de programación TypeScript (https://www.typescriptlang.org/). TypeScript es una extensión del lenguaje JavaScript (https://developer.mozilla.org/es/docs/Web/JavaScript) que agrega tipado estático.

Aplicación back-end

El objetivo de esta aplicación es brindar la funcionalidad de la herramienta desarrollada como servicio web, para que la aplicación front-end pueda invocar las distintas funciones que ofrece la misma, consumir los datos generados y presentarlos al usuario a través de interfaces gráficas.

Dado que los módulos principales de la herramienta ya fueron implementados la implementación de la aplicación back-end se resume a los siguientes pasos:

1. Instalación de los paquetes Django y Django REST Framework: la instalación de ambos paquetes se realiza a través de consola con el gestor de paquetes de Python llamado *pip (https://pypi.org/project/pip/).*
2. Creación de un proyecto Django: la creación de un proyecto Django se realiza a través de consola invocando al comando *django-admin startproject nombreDelProyecto.*
3. Configuración del proyecto: se realiza la configuración del proyecto en el archivo *settings.py* generado automáticamente en la creación del proyecto. Esto incluye configuración de datos para conexión con la base de datos y declaración de paquetes y librerías externas.

[10] Front-end y back-end son términos utilizados para dividir la capas de presentación y la capa de acceso a datos en una aplicación web.

4. Incorporación de la herramienta desarrollada: en este paso se copian dentro de la estructura de carpetas del proyecto los distintos módulos implementados en el capítulo 4.
5. Implementación del modelo de datos a persistir: se crean las clases que serán persistidas en la base de datos.
6. Creación de una API REST: se crean los controladores para exponer la funcionalidad de la aplicación como un servicio web a través de una API REST. Para esto es necesario definir las rutas a los distintos puntos de entrada de la aplicación.

Extensiones al modelo

Tarea de deduplicación

Al modelo de datos de la aplicación se agregó una clase llamada *DeduplicationJob* que guarda datos acerca de una deduplicación de dos listados de registros en particular. Es decir, cuando se invoca al método principal de la herramienta sobre dos listados de registros particulares, se crea un objeto de tipo *DeduplicationJob* con la información correspondiente y se persiste.

Está clase guarda la siguiente información:

- Descripción
- Estado (en progreso, finalizada o fallida)
- Progreso (porcentaje completado hasta el momento)
- Referencia al archivo CSV con listado de registros 1 y 2
- Referencia al archivo CSV con el resultado final (una vez que la tarea se completó)

Si se recupera de la base de datos una tarea de deduplicación en progreso, un ejemplo de los valores que la misma puede contener se muestra a continuación:

Descripción	SEDICI vs Ponencias Memoria Académica
Estado	En progreso
Progreso	75,5 %
Archivo CSV 1	sedici-formato-generico.csv
Archivo CSV 2	ponencias-mem-academica-generico.csv
Resultado	(Vacío)

Tabla 10. Tarea de deduplicación

Tarea de mapeo

Dado que la herramienta desarrollada también permite iniciar tareas de mapeo sobre un listado de registros y un archivo de configuración dado, se agregó al modelo una clase llamada *CrosswalkJob* que guarda información sobre una tarea de mapeo particular.

Esta clase guarda la siguiente información:

- Descripción
- Estado (en progreso, finalizada o fallida)
- Progreso (porcentaje completado hasta el momento)
- Referencia al archivo CSV con listado de registros.
- Referencia al archivo de configuración utilizado.
- Referencia al archivo CSV con el resultado final (una vez que la tarea se completó)

Si se recupera de la base de datos una tarea de mapeo finalizada, un ejemplo de los valores que la misma puede contener se muestra a continuación:

Descripción	SEDICI a esquema genérico
Estado	Finalizada
Progreso	100 %
Archivo CSV	registros-sedici-original.csv
Resultado	sedici-formato-generico.csv

Tabla 10. Tarea de mapeo

Endpoints principales de la API REST

Método HTTP	**Endpoint**	**Parámetros**	**Acción**
POST	/tool/deduplicator	-csv1: archivo csv -csv2: archivo csv -multithread: boolean (indica si el procesamiento se divide en múltiples threads) -description: string	Inicia una tarea de deduplicación entre los registros del csv1 y csv2.
GET	/tool/deduplicator/jobs	-last_five: boolean (indica si sólo se devuelven las últimas 5 tareas)	Devuelve todas las tareas de deduplicación

POST	/tool/crosswalk	-csv: archivo csv con registros a mapear -config: archivo json con la configuración de mapeo -description: string	Inicia una tarea de mapeo sobre los registros contenidos en el archivo csv.
GET	/tool/crosswalk/jobs	-last_five: boolean (indica si sólo se devuelven las últimas 5 tareas)	Devuelve todas las tareas de mapeo
PUT	/tool/crosswalk/jobs/{id}	-id: id de la tarea de mapeo	Cancela una tarea de mapeo
PUT	/tool/deduplicator/jobs/{id}	-id: id de la tarea de deduplicación	Cancela una tarea de deduplicación
GET	/tool/crosswalk/jobs/{id}	-id: id de la tarea de mapeo	Devuelve el detalle de una tarea de mapeo
GET	/tool/deduplicator/jobs/{id}	-id: id de la tarea de deduplicación	Devuelve el detalle de una tarea de deduplicación
GET	/tool/deduplicator/jobs/{id}/results	-id: id de la tarea de deduplicación	Descarga los resultados de la tarea de deduplicación, siempre y cuando haya finalizado.
GET	/tool/crosswalk/jobs/{id}/results	-id: id de la tarea de mapeo	Descarga el resultado de la tarea de mapeo, siempre y cuando haya finalizado.

Tabla 11. Endpoints principales de la API REST

Aplicación front-end

El objetivo de esta aplicación es presentar al usuario una interfaz amigable e intuitiva para el uso de la herramienta desarrollada, y que ahora se expone como servicio web. Por esto último, es posible desarrollar una aplicación que corra en el navegador del usuario y que se comunique directamente con la aplicación back-end para intercambiar datos e invocar las distintas funciones de la herramienta.

En esta sección se muestran las distintas interfaces desarrolladas para el uso general de la herramienta de deduplicación como para el uso del módulo específico de mapeo de metadatos.

Interfaz de usuario de la herramienta de deduplicación

La interfaz de usuario desarrollada permite:

- Iniciar sesión en el sistema
- Iniciar una tarea de deduplicación
    - Single thread
    - Multithread
- Listar tareas
    - Filtrar y ordenar por estado (finalizadas, fallidas o en curso)
    - Filtrar y ordenar por descripción
    - Filtrar y ordenar por fecha
- Ver el detalle de una tarea
    - Descargar archivos CSV utilizados
    - Descargar resultados
    - Cancelar tarea
- Acceder a la documentación de la herramienta
    - Guía de uso
    - Descargar archivos ejemplo

A continuación se adjuntan imágenes de las pantallas principales de la herramienta:

Pantalla de inicio

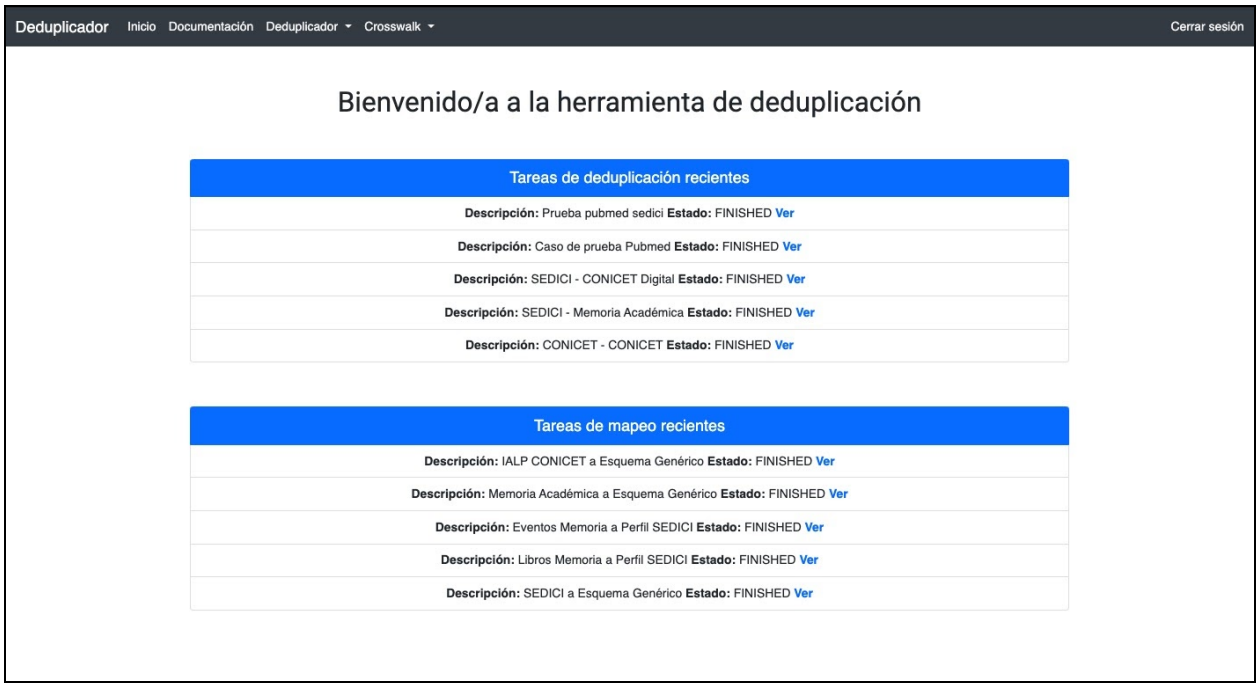

Figura 20. Pantalla de inicio de la aplicación

En esta pantalla se pueden observar dos tablas que contienen las últimas cinco tareas de deduplicación y mapeo realizadas. Para cada una de estas se muestra la descripción y el estado, junto con un link al detalle de la tarea en cuestión. Además, se puede observar la barra de navegación de la aplicación que se mantiene igual en todas las pantallas de la misma.

Formulario para iniciar una tarea

Figura 21. Formulario para iniciar una tarea de deduplicación

El usuario debe subir los archivos CSV con el listado de registros correspondientes en cada uno, indicar una descripción para identificar la tarea y opcionalmente indicar que el procesamiento se realice en múltiples threads. Una vez se presiona el botón *Iniciar* y en caso que no haya errores en los archivos seleccionados, se redirige al usuario al detalle de la tarea creada.

Detalle de una tarea

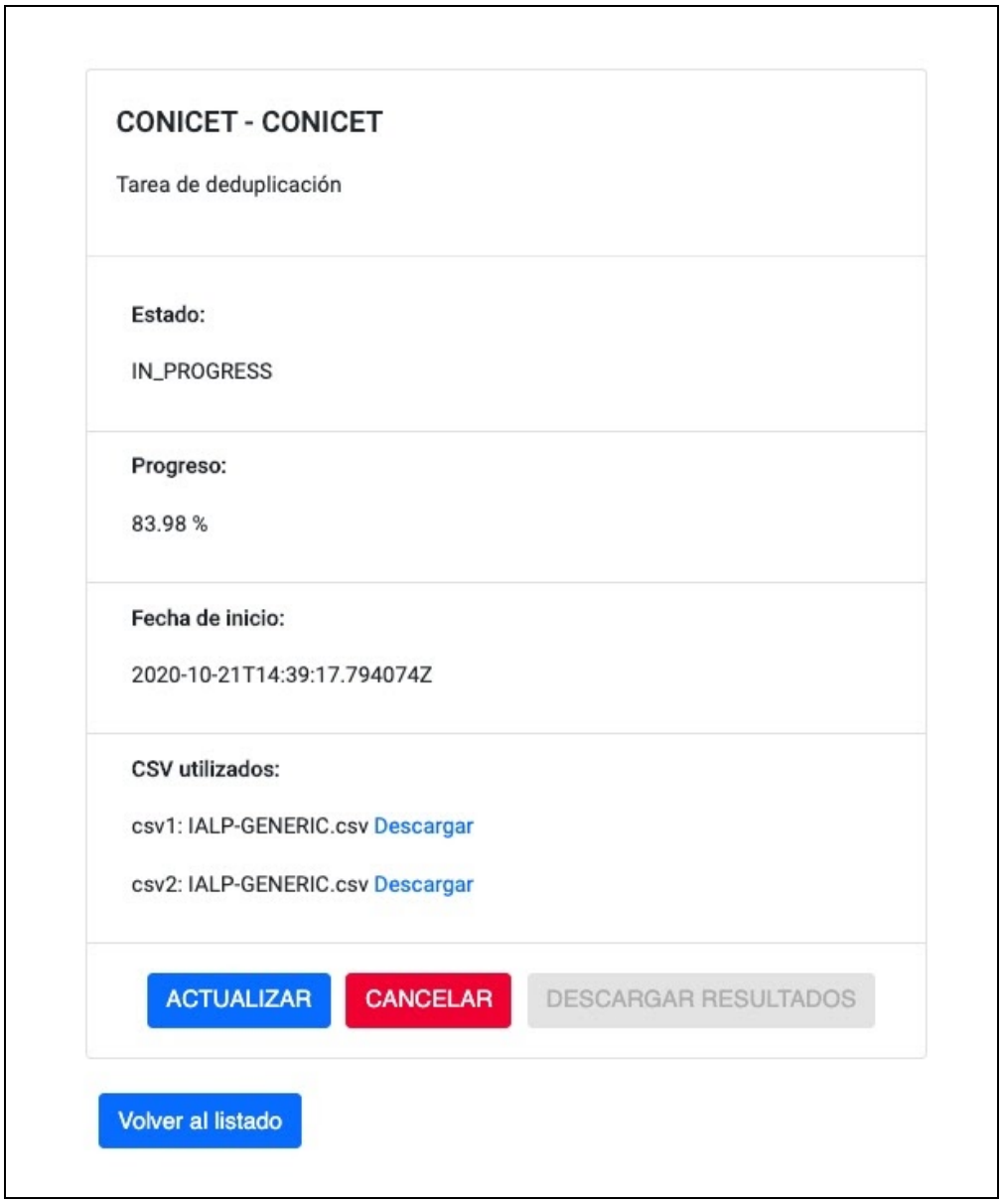

Figura 22. Detalle de una tarea de deduplicación

En el detalle se puede observar el estado de la tarea, cuyos posibles valores son *IN_PROGRESS* (en progreso), *FAILED* (fallida) y *FINISHED* (finalizada); el progreso expresado como porcentaje, la fecha y hora de inicio y los archivos CSV utilizados.

Los botones para actualizar el progreso y cancelar la tarea se deshabilitan una vez que la misma finalizó o fallo; y el botón para descargar los resultados únicamente se habilita cuando la tarea finalizó correctamente (estado *FINISHED*).

Listado de tareas

Figura 23. Pantalla de listado de tareas de deduplicación

En el listado de tareas se muestra un historial de todas las tareas finalizadas, fallidas y en curso. Se pueden ordenar las filas utilizando cualquier atributo al hacer doble click sobre el encabezado de cada columna. El icono azul en la columna de acciones permite ir al detalle de una tarea en particular, y el icono rojo permite eliminar una tarea con previa confirmación (al eliminar una tarea también se eliminan los archivos CSV y los resultados asociados, en caso de existir). El *input* de búsqueda en la esquina superior izquierda de la pantalla permite filtrar las tareas en base a cualquier columna, excepto en base a la columna *Acciones* ya que las mismas no varían para distintas tareas.

Interfaz de usuario del módulo de mapeo

Las interfaces desarrolladas permiten acceder a toda la funcionalidad del módulo de mapeo a través de la interacción con formularios y tablas, y la misma puede ser extendida en el futuro para permitir la definición de los archivos de configuración dentro de la misma aplicación.

Actualmente, la interfaz de usuario permite realizar las siguientes operaciones:

- Iniciar una tarea de mapeo
- Ver el detalle de una tarea
    - Descargar el resultado
- Listar tareas
    - Filtrar y ordenar por descripción
    - Filtrar y ordenar por estado
    - Filtrar y ordenar por fecha
    - Eliminar una tarea

A continuación se muestran las pantallas principales de este módulo:

Formulario para iniciar una tarea

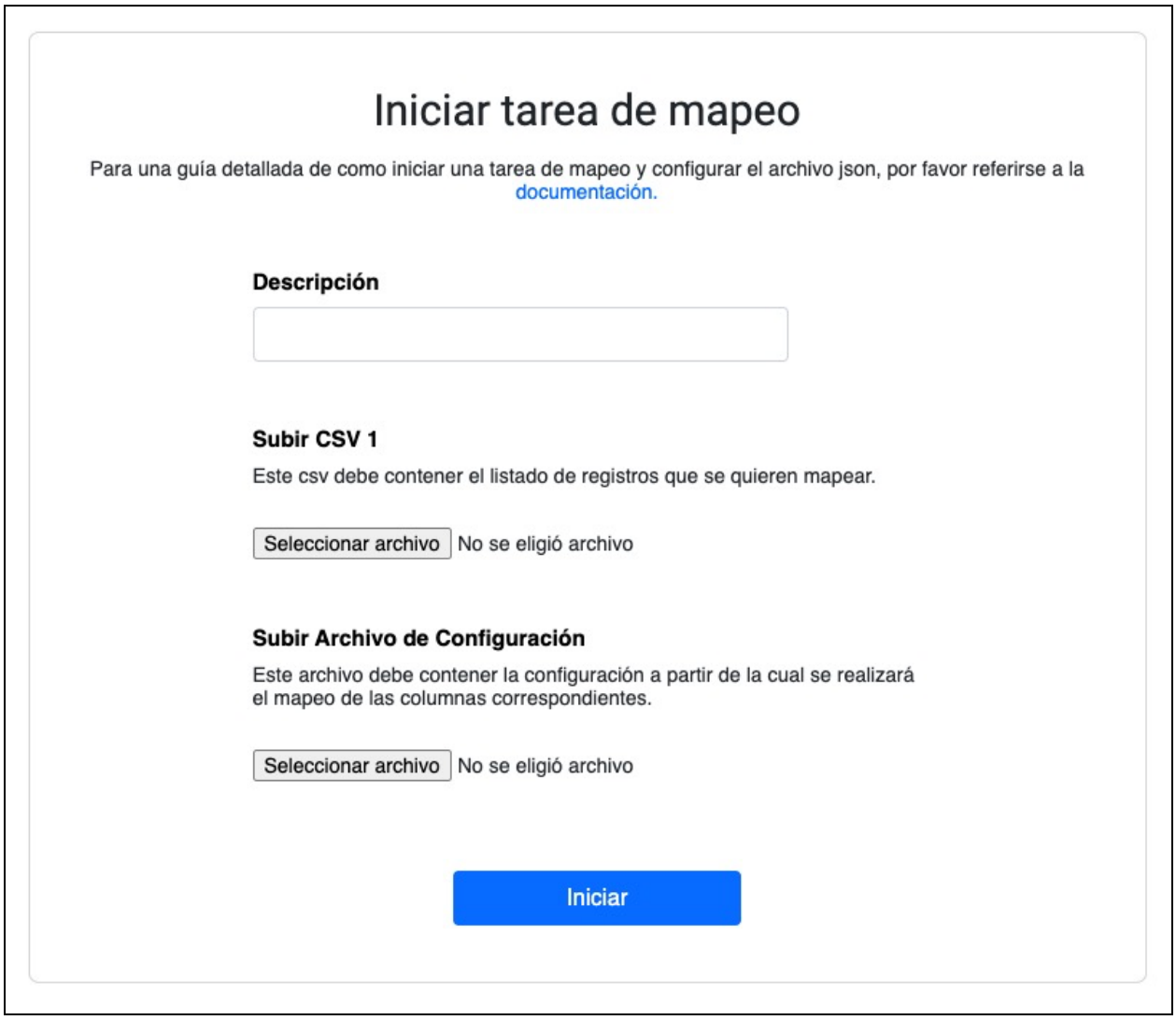

Figura 24. Formulario para iniciar una tarea de mapeo

Este formulario permite iniciar una tarea de mapeo indicando *descripción*, *archivo CSV* que debe mapearse, y *archivo de configuración* definido para la tarea. Una vez iniciada una tarea, la interfaz redirecciona al usuario al detalle de la misma, donde se puede observar información acerca de la misma.

Detalle de una tarea

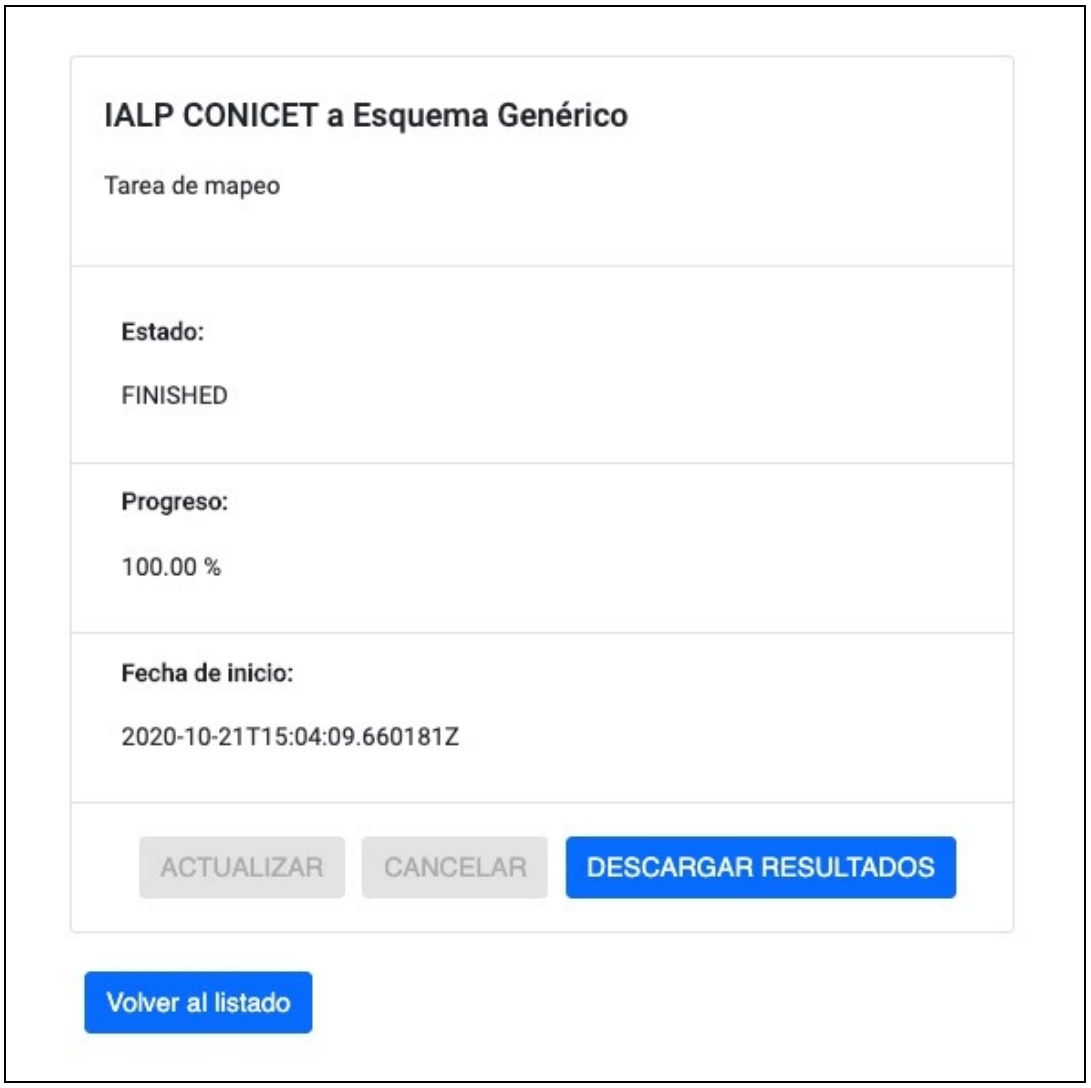

Figura 25. Detalle de una tarea de mapeo

En esta pantalla se puede observar el estado de una tarea (*IN_PROGRESS, FINISHED, FAILED, CANCELLED),* el progreso de la misma y la fecha en la que fue iniciada. Con el botón *ACTUALIZAR* se refresca el estado y el progreso, con el botón *CANCELAR* se cancela la tarea y se la pone en estado *CANCELLED,* y con el botón *DESCARGAR RESULTADOS* (activo únicamente al finalizar la tarea) se descarga el archivo CSV mapeado.

Listado de tareas

Figura 26. Pantalla de listado de tareas de mapeo

En esta pantalla se muestra una tabla con el historial completo de las tareas iniciadas, permitiendo filtrar y ordenar los datos en base a cualquier columna (excepto la de *Acciones*). En la columna *Acciones* se tiene opciones para *Ir al detalle de una tarea* o para *Eliminar una tarea* (este borrado es físico, elimina la tarea de la base de datos y el archivo CSV resultado asociado a la misma).

Capítulo 6 - Proceso de importación y resultados obtenidos

Introducción

En las últimas etapas del desarrollo de la herramienta de deduplicación se realizaron numerosas pruebas del funcionamiento de la misma con registros obtenidos a partir de múltiples repositorios y con los registros presentes en SEDICI. Cada una de estas pruebas permitió la corrección de las reglas y de los módulos encargados de comparar metadatos, y el ajuste de los valores definidos como umbrales, en base a la detección de falsos positivos o de falsos negativos. La prueba de la herramienta se basó en iteraciones y en cada una de las mismas se obtuvieron diferentes resultados que permitieron evaluar la corrección y la performance de la herramienta.

Las tareas de deduplicación llevadas a cabo se compusieron de registros obtenidos a partir de la base de datos de SCOPUS y de los repositorios Memoria Académica y CONICET Digital, y se cubrió un amplio conjunto de tipos de documentos a evaluar, entre los que se incluyen libros, capítulos de libros, artículos, objetos de conferencia y reseñas. Los registros de metadatos fueron obtenidos a partir de las interfaces provistas por cada repositorio, en el caso de CONICET Digital y Memoria Académica se utilizó el protocolo OAI-PMH, y en el caso de SCOPUS se realizó la exportación de registros a través de su interfaz web en conjunto con metadatos expuestos por una API que SCOPUS expone. En el caso de SEDICI, se realizó una exportación de todos los registros almacenados a través del módulo de administración de DSpace.

A la fecha de la finalización de este trabajo la herramienta de deduplicación procesó más de 150.000 documentos y en base a sus resultados se lograron incorporar más de 11.000 documentos nuevos al repositorio SEDICI. A lo largo de este capítulo se presenta un análisis de los datos obtenidos a partir del uso práctico de la herramienta de deduplicación en este escenario y se detalla el proceso de recuperación, transformación e ingesta de documentos hacia repositorios digitales definido durante las pruebas de la herramienta, en el que también se hace presente la utilización del módulo de mapeo de metadatos.

Proceso para importaciones masivas

Como resultado de las importaciones de documentos realizadas a SEDICI se logró la definición de un conjunto de etapas y tareas estandarizadas, que dieron lugar a la definición de un proceso de recuperación, transformación e ingesta de registros académicos a repositorios digitales que puede ser aplicado en múltiples escenarios y adaptado a cualquier repositorio digital.

Este proceso se adecua al patrón arquitectural conocido como ETL (Extract, Transform and Load) [De Giusti, Lira, Oviedo - 2011], que involucra tareas de extracción de datos de múltiples fuentes, posteriores transformaciones y evaluaciones sobre los mismos, y finalmente la carga al repositorio o base de datos destino. El proceso utilizado para recuperar, procesar (mapear y deduplicar) e incorporar los documentos mencionados en secciones anteriores se

basa en un modelo de ETL, aplicado al caso de uso particular de repositorios digitales. Cada una de las tareas de extracción, transformación y carga de datos se ve representada en el proceso definido por un conjunto de etapas que tienen como objetivo final alguna de estas tareas.

A continuación se detallan las distintas etapas identificadas para realizar una ingesta masiva de documentos al repositorio con el uso de las herramientas desarrolladas en el marco de este trabajo, desde la deduplicación de registros, hasta el mapeo y normalización de los metadatos para ser ingestados.

1. Obtención de registros desde un repositorio

Esta etapa resume las tareas de búsqueda, filtrado y obtención de los registros académicos a partir de una fuente de datos particular. El filtrado de los mismos consiste en seleccionar únicamente aquellos registros que son de interés para el repositorio destino. En el contexto de SEDICI los registros a buscar son aquellos que hacen referencia a producción de investigadores de la UNLP y que además se encuentren publicados bajo licencias de acceso abierto.

La forma en la que los registros académicos son obtenidos varía dependiendo la fuente de datos, siendo las principales:

- vía interfaz OAI-PMH (e.g. CONICET Digital)
- vía API (e.g. SCOPUS)
- vía exportación por interfaz web (e.g. SEDICI)
- vía administración interna del repositorio (e.g diferentes áreas de la institución)

2. Mapeo de metadatos a formato genérico

Una vez que se cuenta con el listado de registros se deben mapear los metadatos al esquema genérico utilizado por la herramienta de deduplicación. Con el módulo de mapeo desarrollado, es posible definir la configuración de mapeo correspondiente que podrá reutilizarse en futuras importaciones de registros del mismo repositorio.

3. Deduplicación con registros del repositorio destino

Se realiza la deduplicación con el listado de registros actual del repositorio destino a fin de eliminar la producción ya cargada en el repositorio y evitar la importación de registros duplicados. Una vez generado el reporte de la herramienta de deduplicación, se filtran los registros detectados como casi duplicados y duplicados, y se revisan aquellos detectados como indefinidos. El listado final de registros a importar se compone de los registros catalogados como no duplicados y el conjunto de registros catalogados como indefinidos, que luego de la revisión, se confirmaron como no duplicados en el repositorio destino.

4. Reconciliación de metadatos

Una vez se cuenta con el listado de registros deduplicado, se debe realizar la reconciliación con el registro de metadatos original. Para esto es necesario hacer un cruce entre el listado de registros ya deduplicado y el listado de registros de metadatos original obtenido en el paso 1. Este cruce entre listados se realiza a partir del identificador de cada registro.

id_document1	id_document2	rules	similarity
https://ri.conicet.gov.ar/handle/11336/14093	http://sedici.unlp.edu.ar/handle/10915/36493	JournalArticleRule=1 | GeneralRule=0.75 | DoiRule=A	DUPLICATE

Tabla 12. Fila resultado de una tarea de deduplicación

En base al identificador indicado en la clave 'id_document1' se busca el registro correspondiente en el archivo CSV original, para obtener el registro de metadatos completo:

```
lista_de_registros = [ ]
for resultado in resultados:
    for registro_completo in registros_completos:
        if resultado['similarity] == 'DUPLICATE' and resultado['id_documentN'] ==
         registro_completo['id']:
            Se agrega item a la lista

Se escribe el contenido de lista_de_registros en un archivo CSV
```

Figura 27. Pseudocódigo del algoritmo de reconciliación de metadatos

En caso que también se desee agregar a la lista los registros 'casi duplicados' basta con agregar al *if* la condición que chequea por *similarity == 'NEAR_DUPLICATE'*.

Alternativamente, este proceso puede realizarse sin preguntar por la similitud, y habiendo filtrado previamente los registros deseados con el uso de una herramienta de edición de archivos en formato CSV.

5. Mapeo a formato esperado por el repositorio destino

Como se mencionó en el capítulo 2, cada repositorio guarda los registros de metadatos bajo un perfil de aplicación determinado, que puede componerse de uno o más esquemas de metadatos. A fin de preparar el contenido a importar al repositorio se realiza el mapeo de los metadatos al perfil de aplicación utilizado por el repositorio destino, que puede incluir la eliminación de metadatos innecesarios.

En esta etapa se vuelve a hacer uso del módulo de mapeo definiendo la configuración correspondiente. Cabe destacar que por cada fuente de datos a importar generalmente se

tendrán dos configuraciones de mapeo: una para el pasaje al formato genérico (mencionada en el paso 2) y una para el pasaje al perfil de aplicación utilizado por el repositorio destino.

6. Correcciones sobre los metadatos

Se realizan las correcciones y normalizaciones necesarias sobre cada metadato en particular. Por ejemplo se cambia el formato en que se guardan las fechas, el número y volumen de revista o el idioma.

Para realizar correcciones, modificaciones y/o manipular grandes cantidades de registros se utilizó la herramienta de código abierto OpenRefine (https://github.com/OpenRefine/OpenRefine) que brinda facilidades para la manipulación de archivos muy grandes en formato CSV, XML, JSON, entre otros, y procesar, filtrar, ordenar y agrupar los registros en base a distintos criterios. Permite escribir código Jython (una variante de Python implementada en Java) o GREL (Google Refine Express Language) para ejecutar scripts sobre los registros.

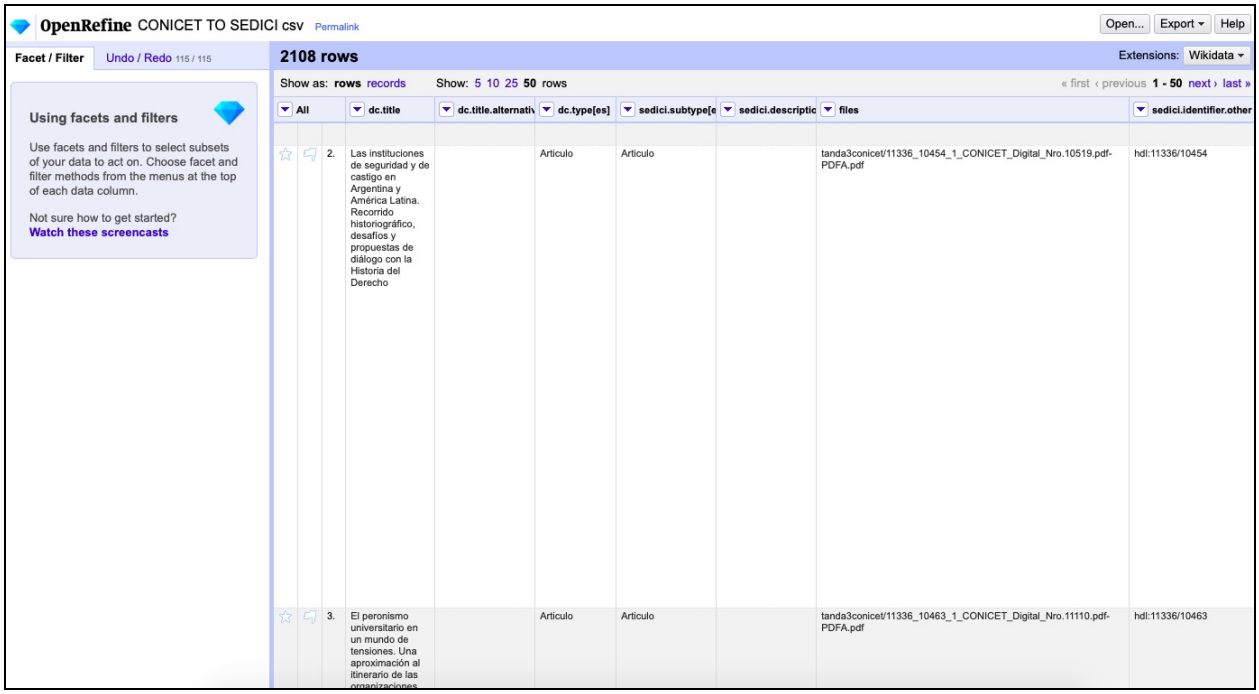

Figura 28. Interfaz de la herramienta OpenRefine

7. Obtención de los objetos digitales asociados a cada registro

En la mayoría de los casos, la carga de registros se ve acompañada de la carga de los objetos digitales asociados a cada uno de ellos, es decir, el recurso al que el registro está describiendo, y que dependiendo del mismo se puede encontrar en formatos de archivos de documentos (PDF), audio (MP3, WAV, etc.) o video (MP4, MOV, FLV, etc.). La forma en que los objetos digitales son obtenidos a partir de su fuente varía según la fuente de datos sobre la cual se esté trabajando, aunque las principales vías de obtención son a partir de los links expuestos por OAI-PMH en cada registro y a partir de los links del portal web del repositorio.

Para la descarga de los archivos PDF se utilizó la herramienta OpenRefine que permite iterar sobre las columnas que contienen la url al archivo PDF asociado a cada registro de metadatos y ejecutar un script que realice la descarga del mismo. Para esto último se utilizó el módulo *urllib* de Python (https://docs.python.org/3/library/urllib.html).

En SEDICI, para cumplir con las políticas de preservación del contenido depositado dentro del repositorio, todos los archivos PDF obtenidos son convertidos al formato PDF/A antes de ser importados.

8. Generar archivo de importación y carga del mismo

Finalmente, para que los registros obtenidos puedan ser incorporados al repositorio deben ser subidos al sistema bajo un formato de archivo particular que funciona como 'paquete', y el mismo varía dependiendo el software de repositorio utilizado en cada caso. En DSpace, este formato de archivo se llama SAF (Simple Archive Format) y está compuesto por un conjunto de directorios donde cada uno representa a un documento. Cada directorio incluye uno o múltiples archivos XML con los metadatos del registro, bajo los esquemas deseados; un archivo contents donde se indica cada archivo asociado al registro; y los objetos digitales en cuestión, por ejemplo un archivo PDF.

En la figura 29 se muestra la estructura de un archivo SAF.

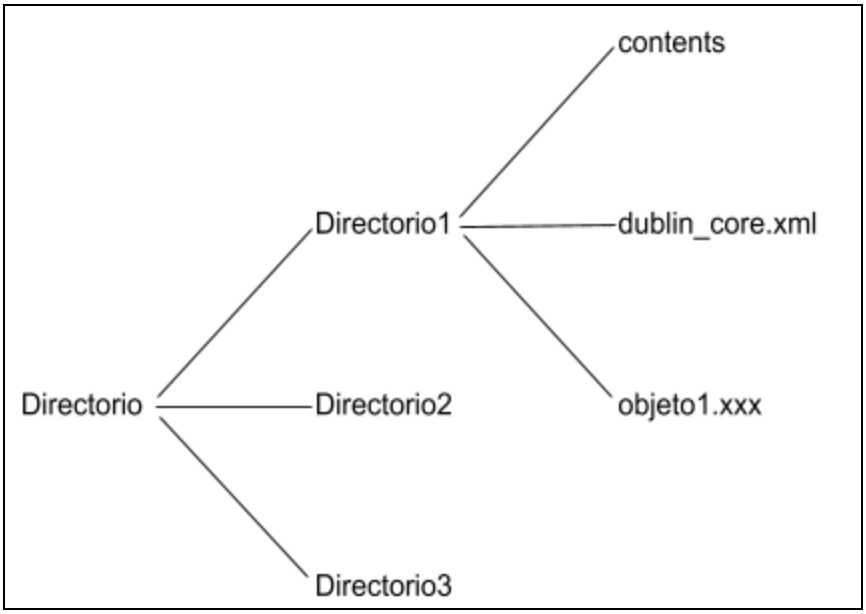

Figura 29. Estructura de paquete SAF
(Fuente: https://www.arvo.es/dspace/tag/import/)

En el contexto de las importaciones realizadas a SEDICI, para generar el archivo en formato SAF se utilizó la herramienta de código abierto *dspace-csv-archive,* a la cual se le provee un archivo CSV con el listado de registros a importar incluyendo una columna que referencia a la ruta donde se encuentra el objeto digital asociado, y genera como salida un

archivo zip que cumple con el formato SAF esperado por DSpace. La misma puede encontrarse en https://github.com/weilandp/dspace-csv-archive.

Casos de aplicación

SCOPUS

SCOPUS (https://www.scopus.com/) es la base de datos de citas y resúmenes de literatura revisada por pares más grande del mundo, y brinda un conjunto de herramientas que facilitan la visualización, búsqueda y reporte de estadísticas de la producción científica que alberga. Desde noviembre de 2020, cuenta con más de 70 millones de registros indexados, aproximadamente 70 mil perfiles de instituciones y 16 millones de perfiles de autores.

La primera aplicación práctica de la herramienta de deduplicación de registros dentro de un proceso de ingesta masiva real a SEDICI optó por realizarse con registros de metadatos exportados de SCOPUS por la calidad y cantidad de metadatos que se exponen de cada registro. A través de la interfaz web del agregador, se filtraron aquellos registros que pertenecen a autores de la UNLP y que se encuentran en acceso abierto y se exportaron en formato CSV. Luego se realizó el mapeo de los metadatos empleados por SCOPUS al esquema genérico utilizado por la herramienta de deduplicación.

La cantidad de registros exportados fue de 5289 y la importación de los mismos se dividió en múltiples tandas para permitir la verificación y corrección de los metadatos importados y el funcionamiento de la herramienta. Para la división de los registros en conjuntos más pequeños los mismos se agruparon por año de publicación.

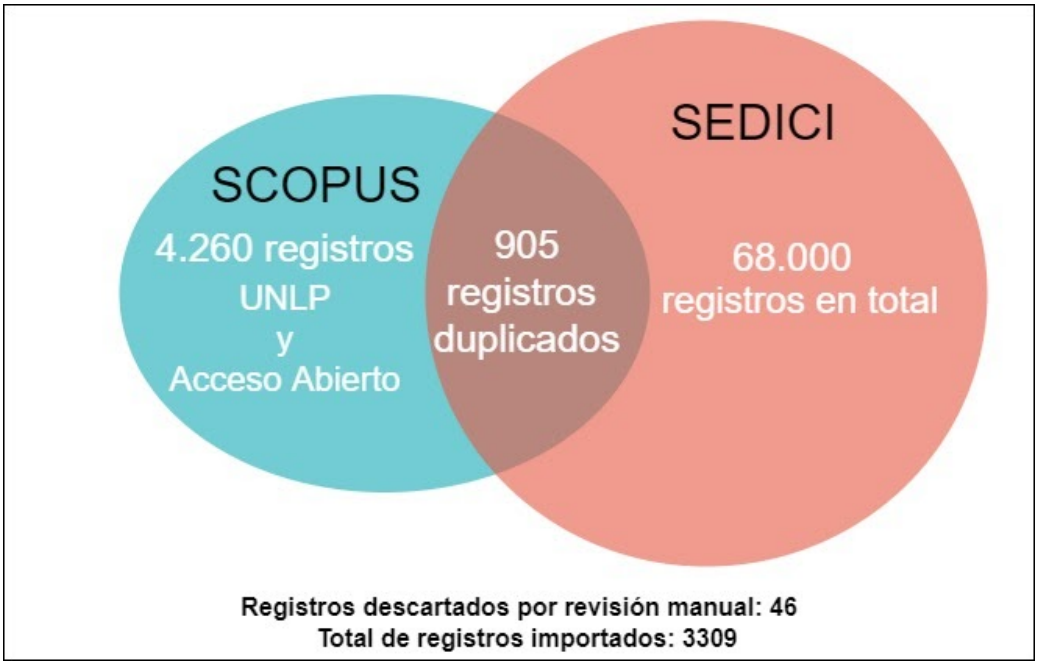

Figura 30. Documentos procesados e importados de SCOPUS

A modo de ejemplo, se detalla la cantidad de registros involucrada en la primera tanda importada a SEDICI (con año de publicación 2010):

Condición	Cantidad de documentos
Obtenidos	218
No duplicados	180
Duplicados	28
Casi duplicados	10
Indefinidos	0
Importados	170

Tabla 12. Tanda de importación número 1 de SCOPUS

Las siguientes tandas involucraron registros de los años 2000 a 2019 con una cantidad total de 4260 registros procesados (incluídos aquellos de la tanda 1) y 3309 importados a SEDICI, como se puede observar en la figura 30.

Memoria Académica

Memoria Académica [http://www.memoria.fahce.unlp.edu.ar/] es el repositorio institucional de la Facultad de Humanidades y Ciencias de la Educación y del Instituto de Humanidades y Ciencias Sociales, ambos de la Universidad Nacional de La Plata, y alberga la producción científico-académica realizada por autores en el ámbito de estas instituciones. Desde noviembre de 2020 cuenta con más de 44.000 recursos de distintas tipologías (planes de estudio, proyectos de extensión, tesis, artículos, entre otras) de los cuales 33.500 tienen el texto completo asociado, y el restante son sólo referencias.

Es de interés para el SEDICI mantenerse sincronizado con el contenido de Memoria Académica ya que todo el contenido albergado en este último es a su vez producción de la UNLP. Esto brinda una ventaja a la hora de analizar la información obtenida puesto que no es necesario realizar un filtrado de producción UNLP en base a los autores o instituciones relacionadas con cada obra.

A raíz de la colaboración con el equipo de Memoria Académica se obtuvieron 9.644 registros de artículos, 529 registros de libros y 13.615 registros de eventos para procesar e importar a SEDICI. A continuación se muestran gráficos que resumen las cantidades de documentos deduplicados e importados, divididos en base a la tipología de los mismos:

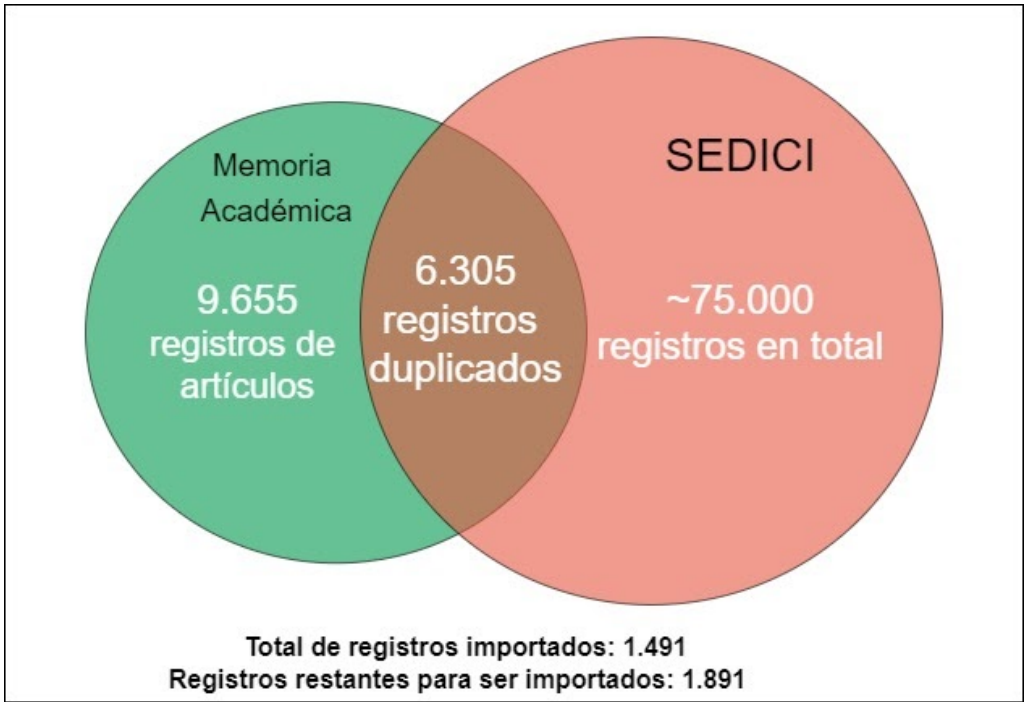

Figura 31. Artículos procesados e importados de Memoria Académica

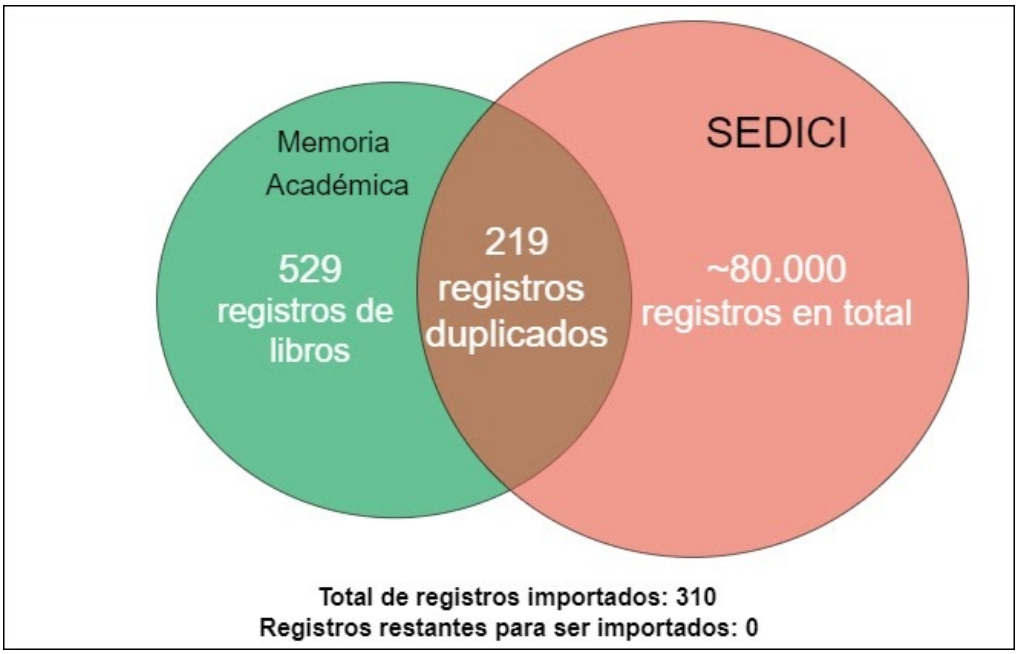

Figura 32. Libros procesados e importados de Memoria Académica

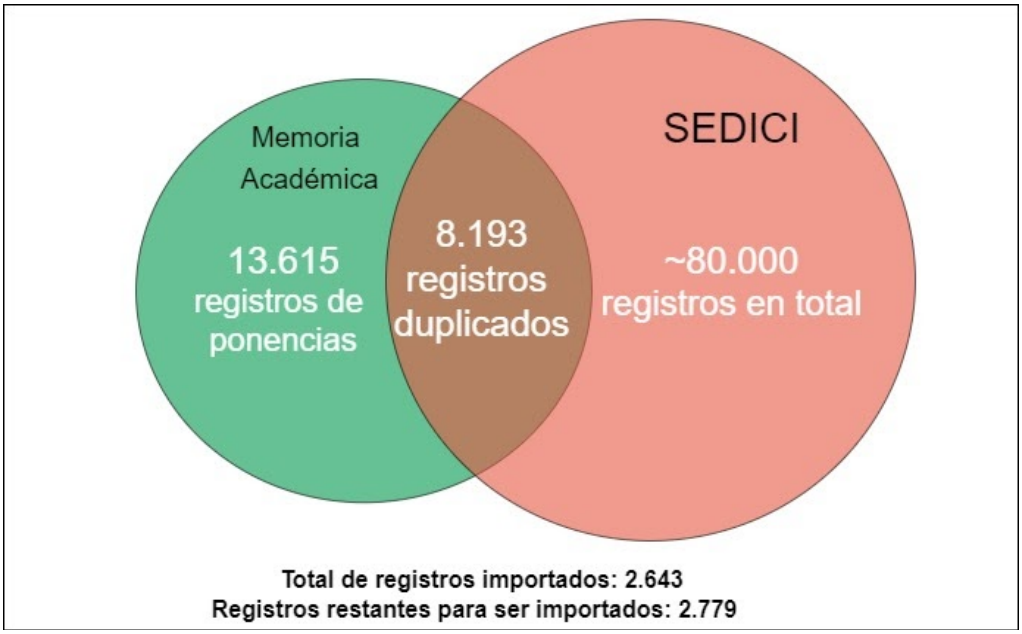

Figura 33. Eventos procesados e importados de Memoria Académica

CONICET Digital

CONICET Digital (https://ri.conicet.gov.ar) es el repositorio institucional del CONICET y reúne la producción resultante de las actividades científicas de sus investigadores. Desde noviembre de 2020 contiene más de 110.000 recursos, aunque no todos se encuentran en acceso abierto. La cantidad de recursos en base al tipo de acceso de cada uno se resume en:

- Acceso restringido: 55910
- Acceso abierto: 55276
- Embargado: 552

Para el SEDICI resulta de interés la incorporación de material en acceso abierto únicamente, por lo que el conjunto de documentos sobre el cual se trabajó fue este último, con 55276 registros de metadatos para analizar. Sin embargo, en CONICET Digital se guarda producción científico-académica de autores de todo el país que realizan investigaciones en el marco de CONICET, pero obviamente no todos/as pertenecen a la UNLP, por lo que este repositorio también requiere una etapa previa de filtrado de producción relacionada a la UNLP, a partir de las colecciones armadas dentro del repositorio y de los centros de investigación que se describen en cada registro.

Las importaciones de CONICET Digital involucraron registros de distintas colecciones:

Fuente de obtención	Cantidad de registros
Colección OAI ‘CCT La Plata’	12476
Registros con filiación ‘UNLP’	1595
Registros que pertenecen a colecciones de centros UNLP	1393

Tabla 13. Registros procesados de CONICET Digital discriminados por fuente

De esta manera se logró reunir la producción de la UNLP que está cargada en CONICET Digital, con un total de 16153 registros. Luego de realizada la deduplicación, se filtraron aquellos registros que tienen licencias de acceso cerrado o embargado para realizar la importación únicamente de los documentos con licencias de acceso abierto.

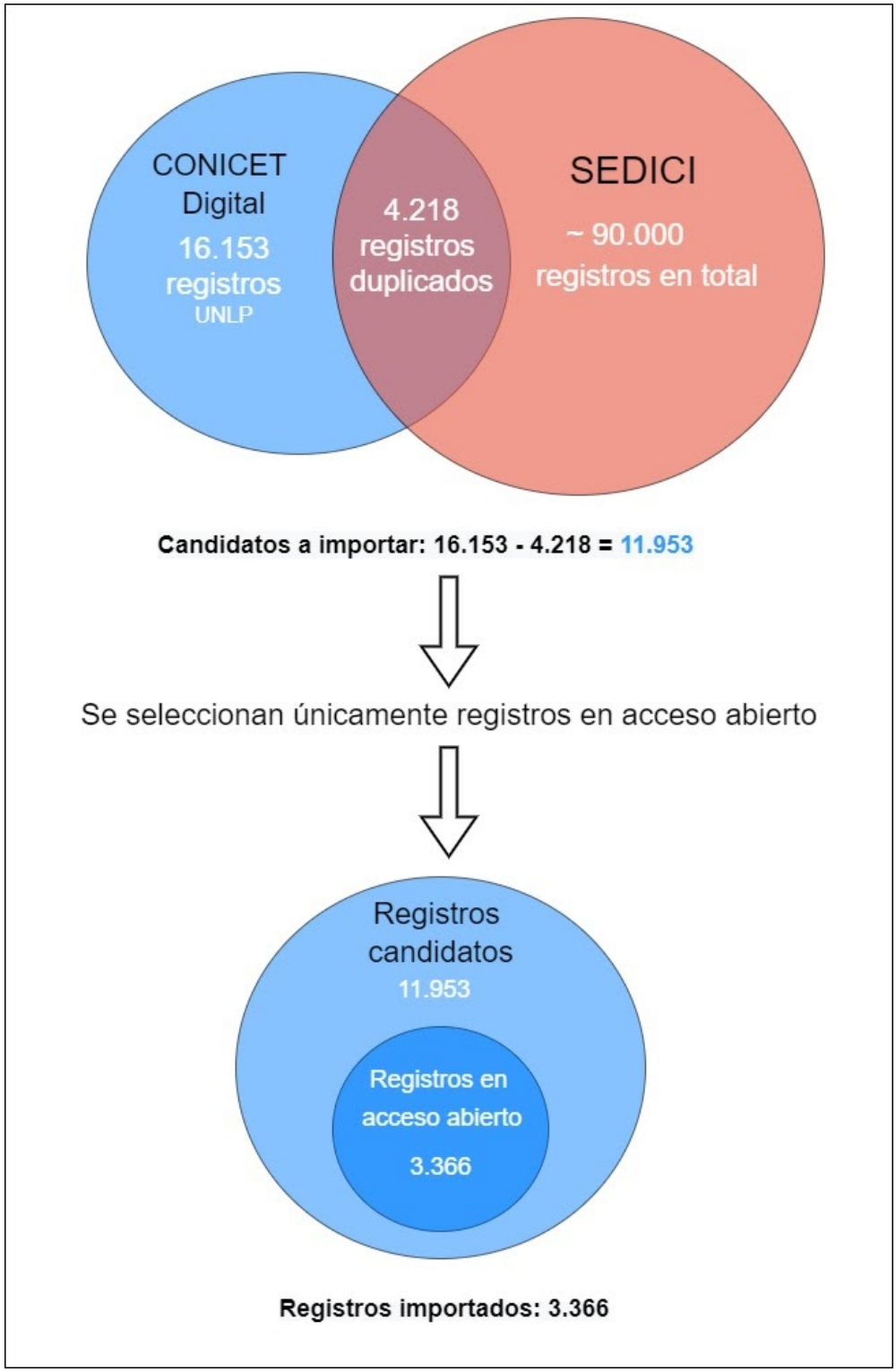

Figura 34. Registros procesados e importados de CONICET Digital

Capítulo 7 - Conclusiones y trabajos futuros

Conclusión

A partir del estudio de los procesos de recuperación e ingesta masiva de documentos en el contexto de los repositorios digitales se detectaron etapas con necesidades específicas, que aún no habían sido resueltas, y que eran críticas para garantizar la calidad del material incorporado. Estas necesidades se enfocan principalmente en la detección de material duplicado y en la transformación y mapeo de registros de metadatos para garantizar interoperabilidad entre los distintos esquemas utilizados en cada fuente de datos. En el marco de esta tesina de grado se realizaron desarrollos que atacan estas necesidades y que brindan una solución a los problemas presentados, ayudando a cumplir con cada uno de los objetivos planteados al inicio de la misma.

Para el desarrollo de la herramienta fue necesario incursionar en las distintas técnicas y metodologías existentes para resolver el problema de deduplicación de registros. La técnica utilizada combina aspectos de técnicas basadas en reglas y en funciones de distancia, principalmente por la flexibilidad que la misma otorga para definir reglas y restricciones específicas del dominio.

A partir de las fuentes de datos mencionadas en el capítulo 6 de este trabajo, se logró obtener un total de 54.000 documentos los cuales fueron utilizados para realizar tareas de deduplicación contra los registros existentes en SEDICI. Estas tareas consistieron en el procesamiento de un total de 150.000 registros de metadatos aproximadamente (sumando aquellos recuperados de las distintas fuentes con aquellos presentes en SEDICI) a partir de los cuales se determinaron documentos duplicados, candidatos a duplicados, no duplicados e indefinidos. Aquellos que se tuvieron en cuenta para incorporar a SEDICI fueron los detectados como no duplicados únicamente, y se dejaron aquellos detectados como candidatos a duplicados e indefinidos para ser analizados en busca de falsos positivos/negativos.

Las acciones realizadas con la herramienta fueron muy favorables y permitieron la incorporación de aproximadamente 11.000 registros nuevos, o lo que es equivalente, permitió al repositorio de la UNLP incrementar su producción en casi un 10%. Entre estos documentos se encuentran artículos, objetos de conferencia, tesis, libros y capítulos de libro, publicados entre los años 2000 a 2020, y que pertenecen en su totalidad a producción de la UNLP publicada bajo licencias de acceso abierto.

Si bien el trabajo realizado en el marco de esta tesina de grado da soporte a ingestas masivas al repositorio SEDICI, la utilización de las herramientas desarrolladas y del proceso definido puede extenderse a múltiples repositorios y escenarios distintos, para dar soporte a la ingesta de documentos y deduplicación de contenido propio en distintos repositorios.

Las importaciones realizadas permitieron disminuir considerablemente el tiempo en que la producción es disponibilizada en el repositorio. Cada ingesta de documentos se ingresa en una tarea de *workflow*, donde el personal responsable de la carga tiene la posibilidad de aceptar o rechazar cada registro. Este proceso fue muy importante durante el desarrollo de la

herramienta puesto que la retroalimentación generada permitió corregir y ajustar el funcionamiento de la herramienta y las reglas de mapeo.

Sin embargo, es importante destacar que en pos de que las importaciones faciliten y aceleren el proceso de carga a los responsables de esta actividad en el repositorio, es necesario que la precarga de metadatos sea correcta y brinde a los administradores únicamente la tarea de revisión y normalización de valores en aquellos casos que sea necesario. La carga incorrecta y la ausencia de metadatos, así como la presencia de problemas en los objetos digitales asociados (archivos PDF que no fueron transformados a PDF-A por ejemplo) entorpecen la carga de cada documento, aumentando los tiempos de carga nuevamente. A raíz de esto se destaca la importancia de definir buenas reglas de mapeo entre los distintos esquemas, sin perder información en el proceso.

Trabajos futuros

Si bien durante el desarrollo de esta tesina se alcanzaron y cumplieron en su totalidad los objetivos planteados al inicio de la misma, hay ciertos puntos de la herramienta y de su utilización que pueden mejorarse e incluso ampliarse en el futuro. A continuación se describen estos aspectos:

Mejorar performance de la herramienta de deduplicación

Cuando la cantidad de registros a comparar es muy grande el tiempo que tarda la herramienta de deduplicación en generar un resultado es considerablemente lento, aproximadamente 30 horas para comparar 12.000 registros contra otros 95.000 registros. Esto es esperable debido a la naturaleza del problema a resolver: se debe comparar cada par de registros posible, lo cual conlleva un algoritmo de fuerza bruta, donde se hace una comparación de NxM elementos que en casos donde M >= N el tiempo de ejecución es del orden de N^2.

Incluso con código concurrente, el factor principal por el cual el tiempo de ejecución no puede mejorar drásticamente es el *Global Interpreter Lock (GIL)* de *Python*, mecanismo que permite ejecutar código de a un hilo por vez, afectando gravemente a la concurrencia.

Alguna soluciones potenciales, sujetas a investigación son:

- Analizar el rendimiento utilizando una implementación de *Python* como *Jython* (basada en *Java*).
- Dividir los conjuntos de datos antes del inicio de una tarea y ejecutar tantos procesos *Python* como núcleos tenga el procesador disponible.
- Migrar porciones de código específicas a un lenguaje que permita maximizar la concurrencia.

Expandir módulo de comparación de autores

En la versión actual de la herramienta la comparación de autores se basa únicamente en el análisis sintáctico de los mismos, y si bien se logró definir una lógica que contempla variaciones en las estructuras y abreviaciones de los nombres, y existencia de apellidos compuestos, aún existe la posibilidad de agregar otras variables para ser analizadas en

conjunto. Puede utilizarse información de co-autoría entre los distintos autores, identificadores universales como ORCID o áreas temáticas de investigación de cada autor para reforzar la similitud calculada, sobre todo en aquellos casos donde dos autores distintos tienen el mismo nombre.

Enriquecimiento de registros detectados como duplicados

La detección de registros duplicados puede utilizarse como vía para enriquecer los metadatos de registros presentes en un repositorio. Resulta de interés explorar un proceso que utilice la información generada por la herramienta de deduplicación para expandir los metadatos de los registros involucrados en cada tarea. Un posible proceso de enriquecimiento de metadatos a partir de la deduplicación se resume en la siguiente figura:

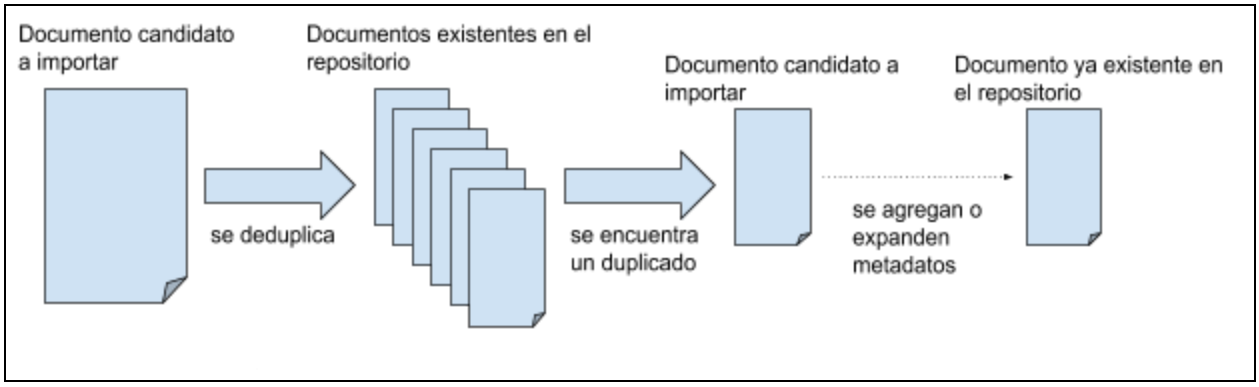

Figura 35. Proceso de enriquecimiento de metadatos en base a registros duplicados

Explorar enfoque de Aprendizaje Automático

Como se mencionó en el capítulo 3 de este trabajo existen múltiples técnicas para la detección de registros duplicados en el contexto de los repositorios digitales. La herramienta de deduplicación desarrollada utiliza una técnica de deduplicación basada en reglas y en funciones de distancia, pero existen otros enfoques que hacen uso de algoritmos de aprendizaje automático. Algunos de estos por ejemplo utilizan técnicas de aprendizaje activo que permiten disminuir considerablemente la cantidad de pares de entrenamiento requeridos para alcanzar funciones que determinen con un grado de precisión alto la similitud entre registros (Sarawagi & Bhamidipaty, 2002). Por otro lado, se proponen enfoques de programación genética que extraen distintas porciones de evidencia de los datos para encontrar una función que pueda determinar si dos registros son duplicados o no, y las mismas son computacionalmente menos exigentes que las utilizadas en otros enfoques similares encontrados en la literatura (Carvalho, Laender, Gonçalves & Silva, 2012).

Resulta de interés experimentar con un prototipo que identifique registros de metadatos duplicados en repositorios digitales a partir de modelos de aprendizaje automático y analizar los resultados obtenidos en comparación con la versión actual de la herramienta desarrollada, a fin de determinar ventajas y desventajas de cada enfoque.

Incorporar funcionalidad de deduplicación dentro del sistema de repositorio

Actualmente la herramienta desarrollada se utiliza como un servicio completamente independiente de los sistemas de repositorios donde se encuentran guardados los datos, y se utilizan los reportes generados para decidir qué conjunto de documentos puede o no ser importado a un repositorio, o como se mencionó en párrafos anteriores, para expandir los metadatos de aquellos documentos que fueron detectados como duplicados. Cualquiera sea el objetivo, la ejecución de la herramienta es independiente al sistema de repositorio, y no funciona como un módulo interno del mismo.

Resulta de interés utilizar la funcionalidad de la herramienta desarrollada como servicio de deduplicación al realizar la carga de una obra mediante el portal web del repositorio. Para esto podría utilizarse la librería desarrollada, separada de los módulos de servicio web, y acoplar la misma con el sistema de repositorio, o bien configurar a este último para que consuma del servicio web expuesto por la aplicación.

Una vez integrada la aplicación con el sistema de repositorio, podrían ejecutarse distintos tipos de tareas:

- Sugerir títulos similares existentes en el repositorio mientras se completa este metadato.
- Realizar un análisis completo con todos los metadatos una vez finalizada la carga de un registro, y en caso de encontrar un candidato a duplicado lanzar una alerta.
- Programar tareas de procesamiento por lotes que se ejecuten como procesos en segundo plano y generen periódicamente reportes de deduplicación dentro del mismo repositorio.

Bibliografía

Registry of Open Access Repositories (ROAR). Accedido 22 de octubre de 2020. http://roar.eprints.org/

De Giusti, M. R., Lira, A. J., & Oviedo, N. F. (2011). Extract, transform and load architecture for metadata collection. VI Simposio Internacional de Bibliotecas Digitales (Brasil, 2011). Recuperado a partir de: http://sedici.unlp.edu.ar/handle/10915/5529

Knoth, P., & Zdrahal, Z. (2012). CORE: Three Access Levels to Underpin Open Access. D-Lib Magazine, 18(11/12). https://doi.org/10.1045/november2012-knoth

SWORD (Protocolo). (2019). En Wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=SWORD_(Protocolo)&oldid=117490516

Repositorio (contenido digital). (2020). En Wikipedia, la enciclopedia libre. https://es.wikipedia.org/w/index.php?title=Repositorio_(contenido_digital)&oldid=129335063

De Giusti, M. R., Lira, A. J., Villarreal, G. L., & Texier, J. D. (2012). Las actividades y el planeamiento de la preservación en un repositorio institucional. BIREDIAL - Conferencia Internacional Acceso Abierto, Comunicación Científica y Preservación Digital. http://sedici.unlp.edu.ar/handle/10915/26045

¿Qué es un repositorio institucional? | Biblioteca Universitaria. (s. f.). Recuperado 22 de octubre de 2020, de https://biblioteca.unileon.es/ayuda-formacion/repositorio-institucional

De Giusti, M. R., Villarreal, G. L., Salamone Lacunza, P., Adorno, F. G., Pinto, A. V., Folegotto, L. E., & Peloche, S. B. (2015). Gestión, preservación, interoperabilidad, visibilidad e impacto de las obras a través de los repositorios institucionales. Ciclo de Conferencias en la Semana Internacional del Acceso Abierto (UBA, 2015). http://sedici.unlp.edu.ar/handle/10915/49456

Resolución 469/11. MINCyT. 2011. Recuperado el 27 de octubre de 2020, de https://www.biblioteca.mincyt.gob.ar/docs/res_be_469-11.pdf

Azrilevich, P. A., & De Giusti, M. R. (2019, noviembre 21). El contexto de los repositorios de acceso abierto en la Argentina: Logros y asuntos pendientes. Asamblea General ISTEC y I Congreso Internacional de Tecnología Aplicada, Innovación y Educación Continua (Córdoba, 2019). http://sedici.unlp.edu.ar/handle/10915/86423

Directrices SNRD. (s. f.). 47. Recuperado a partir de: https://repositoriosdigitales.mincyt.gob.ar/files/Directrices_SNRD_2015.pdf

Ley Nº 26.899. InfoLEG - Ministerio de Economía y Finanzas Públicas—Argentina. (2013). Recuperado 6 de noviembre de 2020, de http://servicios.infoleg.gob.ar/infolegInternet/anexos/220000-224999/223459/norma.htm

Estadísticas. (s. f.). Recuperado 22 de octubre de 2020, de https://repositoriosdigitales.mincyt.gob.ar/vufind/Content/stats

Dappert, A., & Enders, M. (2010). Digital Preservation Metadata Standards. Information Standards Quarterly, 22(2), 11. Recuperado a partir de: https://www.loc.gov/standards/premis/FE_Dappert_Enders_MetadataStds_isqv22no2.pdf

What is a Metadata Record? (s. f.). Center for International Earth Science Information Network (CIESIN). Recuperado 22 de octubre de 2020, de http://www.ciesin.columbia.edu/metadata/what_record.html

What are Metadata Standards | DCC. (s. f.). Recuperado 23 de octubre de 2020, de https://www.dcc.ac.uk/guidance/briefing-papers/standards-watch-papers/what-are-metadata-standards

DCMI: DCMI Metadata Terms. (s. f.). Recuperado 22 de octubre de 2020, de https://www.dublincore.org/specifications/dublin-core/dcmi-terms/

Koutsomitropoulos, D. A., Alexopoulos, A. D., Solomou, G. D., & Papatheodorou, T. S. (2010). The Use of Metadata for Educational Resources in Digital Repositories: Practices and Perspectives. D-Lib Magazine, 16(1/2). https://doi.org/10.1045/january2010-koutsomitropoulos

Heery, R., & Patel, M. (2000). Application Profiles: Mixing and Matching Metadata Schemas. Ariadne, 25. http://www.ariadne.ac.uk/issue/25/app-profiles/

What are persistent identifiers? Overview · THOR Project. (s. f.). THOR Project. Recuperado 22 de octubre de 2020, de https://project-thor.readme.io/docs/introduction-to-persistent-identifiers

Hakala, J. (s. f.). Persistent identifiers—An overview. 17. Recuperado a partir de: http://www.persid.org/downloads/PI-intro-2010-09-22.pdf

Rodrigues, E., & Clobridge, A. (2011). El caso de Interoperabilidad para Repositorios de Acceso Abierto. Zenodo. https://doi.org/10.5281/zenodo.12563

Coll, I. S., & Cruz, J. M. B. (2003). Open archives initiative. Protocol for metadata harvesting (OAI-PMH): Descripción, funciones y aplicaciones de un protocolo. El profesional de la información, 12(2), 99-106. Recuperado 22 de octubre de 2020 de https://www.researchgate.net/publication/28801238_Open_Archives_Initiative_Protocol_for_Metadata_Harvesting_OAI-PMH_descripcion_funciones_y_aplicacion_de_un_protocolo

Orgel, T., Höffernig, M., Bailer, W., & Russegger, S. (2015). A metadata model and mapping approach for facilitating access to heterogeneous cultural heritage assets. International Journal on Digital Libraries, 15(2), 189-207. https://doi.org/10.1007/s00799-015-0138-2

Baca, M. (2016, julio 20). Introduction to Metadata [InteractiveResource]. Getty Research Institute, Los Angeles. http://www.getty.edu/publications/intrometadata

Chan, L. M., & Zeng, M. L. (2006). Metadata Interoperability and Standardization - A Study of Methodology Part I: Achieving Interoperability at the Schema Level. D-Lib Magazine, 12(6). https://doi.org/10.1045/june2006-chan

Record linkage—Wikipedia. (s. f.). Recuperado 23 de octubre de 2020, de https://en.wikipedia.org/wiki/Record_linkage

Elmagarmid, A., Ipeirotis, P., & Verykios, V. (2007). Duplicate Record Detection: A Survey. Knowledge and Data Engineering, IEEE Transactions on, 19, 1-16. https://doi.org/10.1109/TKDE.2007.250581

Loshin, D. (2009). Chapter 10—Data Consolidation and Integration. En D. Loshin (Ed.), Master Data Management (pp. 177-199). Morgan Kaufmann. https://doi.org/10.1016/B978-0-12-374225-4.00010-2

Atzori, C., Manghi, P., & Bardi, A. (2018). GDup: De-Duplication of Scholarly Communication Big Graphs. 2018 IEEE/ACM 5th International Conference on Big Data Computing Applications and Technologies (BDCAT), 142-151. https://doi.org/10.1109/BDCAT.2018.00025

Amón, I., & Jiménez, C. (s. f.). Funciones de Similitud sobre Cadenas de Texto: Una Comparación Basada en la Naturaleza de los Datos. 13. Recuperado el 20 de octubre de https://core.ac.uk/download/pdf/11052347.pdf

Newcombe, H. B., Kennedy, J. M., Axford, S. J., & James, A. P. (1959). Automatic Linkage of Vital Records: Computers can be used to extract «follow-up» statistics of families from files of routine records. Science, 130(3381), 954-959. https://doi.org/10.1126/science.130.3381.954

Fellegi, I. P., & Sunter, A. B. (1969). A Theory for Record Linkage. Journal of the American Statistical Association, 64(328), 1183-1210. https://doi.org/10.1080/01621459.1969.10501049

Winkler, W. E. (s. f.). Frequency-Based Matching In Fellegi-Sunter Model Of Record Linkage.14. Recuperado 22 de octubre de 2020, de https://www.census.gov/srd/papers/pdf/rr2000-06.pdf

Draisbach, U., & Naumann, F. (2009). A Comparison and Generalization of Blocking and Windowing Algorithms for Duplicate Detection. Recuperado 22 de octubre de 2020 de https://www.researchgate.net/publication/242075463_A_Comparison_and_Generalization_of_Blocking_and_Windowing_Algorithms_for_Duplicate_Detection

On Deduplication in the OpenAIRE infrastructure. (s. f.). OpenAIRE. Recuperado 22 de octubre de 2020, de https://www.openaire.eu/blogs/on-deduplication-in-the-openaire-infrastructure-1

Christen, P (2009). Development and user experiences of an open source data cleaning, deduplication and record linkage system | ACM SIGKDD Explorations Newsletter. Recuperado 3 de marzo de 2020, de https://dl.acm.org/doi/abs/10.1145/1656274.1656282

Fava, I (s. f.). Aggregation and content provision workflows. OpenAIRE. Recuperado 22 de octubre de 2020, de https://www.openaire.eu/aggregation-and-content-provision-workflows

Kwon, Y., Lemieux, M., McTavish, J., & Wathen, N. (2015). Identifying and removing duplicate records from systematic review searches. Journal of the Medical Library Association : JMLA, 103(4), 184-188. https://doi.org/10.3163/1536-5050.103.4.004

Gilleland, M. Levenshtein Distance. (s. f.). Recuperado 22 de octubre de 2020, de https://people.cs.pitt.edu/~kirk/cs1501/Pruhs/Spring2006/assignments/editdistance/Levenshtein%20Distance.htm

Jaro–Winkler distance. (2020). En Wikipedia. Recuperado 31 de octubre de 2020, de https://en.wikipedia.org/w/index.php?title=Jaro%E2%80%93Winkler_distance&oldid=974899461

Metaphone. (2020). En Wikipedia, la enciclopedia libre. Recuperado 31 de octubre de 2020, de https://es.wikipedia.org/w/index.php?title=Metaphone&oldid=125249088

Sarawagi, S., & Bhamidipaty, A. (2002). Interactive deduplication using active learning. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 269–278. https://doi.org/10.1145/775047.775087

Carvalho, M., Laender, A., Gonçalves, M., & Silva, A. (2012). A Genetic Programming Approach to Record Deduplication. Knowledge and Data Engineering, IEEE Transactions on, 24, 399-412. https://doi.org/10.1109/TKDE.2010.234