UNIVERSIDAD DE DEUSTO

RECOMENDACIÓN DE CONJUNTOS DE DATOS ENLAZADOS BASADA EN MINERÍA DE GRAFOS

Tesis doctoral presentada por Mikel Emaldi Manrique
dentro del Programa de Doctorado en Ingeniería Informática y Telecomunicación

Dirigida por Dr. Diego López de Ipiña González de Artaza
y Dr. Oscar Corcho García

UNIVERSIDAD DE DEUSTO

RECOMENDACIÓN DE CONJUNTOS DE DATOS ENLAZADOS BASADA EN MINERÍA DE GRAFOS

Tesis doctoral presentada por Mikel Emaldi Manrique
dentro del Programa de Doctorado en Ingeniería Informática y Telecomunicación

Dirigida por Dr. Diego López de Ipiña González de Artaza
y Dr. Oscar Corcho García

El doctorando | El director | El director

Bilbao, octubre de 2015

Recomendación de conjuntos de datos enlazados basada en minería de grafos

Autor: Mikel Emaldi Manrique

Director: Dr. Diego López de Ipiña González de Artaza

Director: Dr. Oscar Corcho García

Ama eta aitarentzat.

Abstract

Since Tim Berners-Lee presented the Linked Open Data concept in 2006, its popularity and adoption has increased. Based on the paradigm for publishing data on the Web presented by Berners-Lee, Hendler and Lassila known as Semantic Web, Linked Open Data dictates a set of rules for the correct publishing of datasets on the Web (Berners-Lee, 2006): 1) use URIs for naming things from the real world, 2) these URIs has to be HTTP URIs for allowing universal access to them, 3) when someone looks up these URIs, they have to provide relevant information, using the standards from the World Wide Web Consortium (W3C), like one of RDF serializations or SPARQL query language, and 4) including links to other datasets, allowing the acquisition of new knowledge. Later, Tim Berners-Lee presented a scale for stimating the quality of Linked Open Datasets. According to this scale, composed by five levels, a dataset scores a star for each level that it overcomes:

- 1$\star$ Data is available on the Web (whatever format) but with an open license.
- 2$\star$ Data is available as a machine-readable structured data (e.g. excel instead of image scan of a table).
- 3$\star$ Besides of being available as a machine-readable structured data, the format has to be non-proprietary (e.g. CSV instead of excel).
- 4$\star$ Data has to use open standards from the W3C (RDF and SPARQL).
- 5$\star$ Data has to be linked to other data from external datasets.

From the adoption of Linked Open Data by companies and organizations new challenges have arisen. One of them is about the last level of the scale, about establishing links among different datasets. Nowadays, when a new dataset is published, the publisher of this dataset has few clues for selecting other related datasets for linking his/her dataset to. For solving this issue, a system that suggests candidate datasets for data interlinking has been developed in this PhD thesis, obtaining a high precision when recommending candidate datasets to be linked. This system is based on the analysis of the structure of the RDF graphs from the datasets. First, the graphs are synthesized and graph mining techniques are applied for extracting the most common subgraphs from each graph. When extracted, similarities among these subgraphs are searched for determining likely relations among them. Moreover, within this solution, a methodology for the synthesis of datasets and a test set for evaluating Linked Dataset recommender systems have been developed.

Resumen

Desde que Tim Berners-Lee presentó el concepto de *Linked Open Data* ("datos abiertos enlazados" en español) en el año 2006, su popularidad y adopción ha ido en aumento. Basado en el paradigma de publicación de información en la Web presentado por Berners-Lee, Hendler y Lassila, y conocido como "Web Semántica", Linked Open Data dicta una serie de reglas para la correcta publicación de conjuntos de datos en la Web (Berners-Lee, 2006): 1) utilizar URI para describir objetos del mundo real, 2) dichas URI serán HTTP URI para que el acceso a ellas sea posible de manera universal, 3) al acceder a estas URI se mostrará información relevante a través de los estándares definidos por el World Wide Web Consortium (W3C), como las diferentes representaciones del modelo RDF o el lenguaje de consulta SPARQL y 4) se incluirán enlaces hacia otros conjuntos de datos, permitiendo la obtención de nuevo conocimiento. Más adelante, Tim Berners-Lee presentaba una escala por la cual se podría calcular la calidad de los conjuntos de datos publicados como Linked Open Data. Esta escala, compuesta por cinco niveles, otorga una estrella por cada nivel superado por el conjunto de datos:

1⋆ Los datos están disponibles en la Web (en cualquier formato) con una licencia que permita considerarlos *Open Data*, o datos abiertos.

2⋆ Se encuentran disponibles en un formato fácilmente procesable por máquinas (por ejemplo, un documento *MS Excel* en lugar de una imagen de una tabla escaneada).

3⋆ Además de encontrarse en un formato fácilmente procesable por máquinas, este formato es libre (por ejemplo, un fichero CSV (*Comma-Separated Values*) en lugar de un fichero *MS Excel*).

4⋆ Los datos han sido publicados siguiendo los estándares del W3C (RDF y SPARQL).

5⋆ Los datos están enlazados con otros datos, proveyendo contexto y enriqueciéndolos.

A raíz de la adopción del paradigma Linked Open Data por empresas y organizaciones han surgido nuevos retos dentro del proceso de publicación de datos en la Web. Uno de ellos trata sobre el último nivel de la escala: el enlazado entre diferentes conjuntos de datos. En la actualidad cuando se publica un nuevo conjunto de datos, el administrador de este conjunto de datos dispone de muy pocas pistas a la hora de seleccionar otros conjuntos de datos con los cuales enlazar el suyo propio. Es cierto que existen catálogos como The Datahub que almacenan metadatos sobre los conjuntos de datos, pero la mayoría de las veces resultan insuficientes, obligando al administrador del conjunto de datos a dedicar excesivo tiempo en la búsqueda de conjuntos de datos relacionados. Para solucionar esta problemática, en esta tesis doctoral se ha desarrollado un sistema que, analizando la estructura de los conjuntos de datos, recomienda otros conjuntos de datos candidatos a ser enlazados con una gran precisión. Para ello, se sintetiza la estructura de los grafos RDF y se aplican técnicas de minería de grafos para extraer los subgrafos más frecuentes de cada conjunto de datos. Una vez extraídos, se buscan similitudes entre estos subgrafos para establecer posibles relaciones entre los conjuntos de datos. Además, junto con esta solución, se han desarrollado una metodología para la síntesis de conjuntos de datos y un juego de ensayo para evaluar las recomendaciones de enlazado de conjuntos de datos.

Laburpena

Tim Berners-Leek *Linked Open Data* kontzeptua ("datu ireki estekatuak" euskaraz) 2006. urtean aurkeztu zuenetik, berauen ospe eta onarpena gero eta handiago bilakatu dira. Tim Berners-Lee, Hendler eta Lassilak Web-ean datuak argitaratzeko aurkeztutako (eta "Web Semantiko" bezala ezagututako) paradigman oinarritua, Linked Open Data-k Web-ean datuak zuzen argitaratzeko zenbait arau ezartzen ditu (Berners-Lee, 2006): 1) URIak erabiltzea objektu errealak deskribatzeko, 2) URI hauek HTTP URIak izatea beraien atzipena unibertsala izan dadin, 3) URI hauek atzitzerakoan aipagarria den informazioa World Wide Web Consortium-ak (W3C) definitutako estandarren bidez erakustea (hala nola RDF modeloaren edozein irudikapen edota SPARQL kontsulta lengoaia) eta 4) beste datuenganako estekak ezartzea, jakintza berria lortzeko aukera emanez. Geroago, Tim Berners-Leek Linked Open Data bezala argitaratutako datuen kalitatea neurtzeko eskala bat aurkeztu zuen. Eskala honek, bost mailez osatuta, datu multzoek gainditutako maila bakoitzeko izar bana ezartzen du:

1⋆ Datuak *Open Data* kontsideratu ahal izateko lizentziapean daude eskuragarri Web-ean (edozein formatutan).

2⋆ Datuak makinek erraz prozesatzeko formatu batean daude eskuragarri (*MS Excel* dokumentu bat eskaneatutako taula baten irudiaren ordez, adibidez).

3⋆ Makinek erraz prozesatzeko formatu batean egoteaz gain, formatu hau librea da (CSV (*Comma-Separated Values*) fitxategi bat *MS Excel* fitxategi baten ordez, adibidez).

- $4\star$ Datuak W3C-aren estandarrak jarraituz argitaratu dira (RDF eta SPARQL).
- $5\star$ Datuak kanpoko beste datuekin estekatuak izan dira, hauexek aberastuz eta testuinguruaz hornituz.

Enpresa eta erakundeek Linked Open Data paradigma onartuz geroztik erronka berriak sortu dira datuak Web-ean argitaratzeko prozesuaren barnean. Horietako batek eskalako azken mailaz dihardu: datu multzo ezberdinen artean estekak ezartzea. Gaur egun, datu multzo berri bat argitaratzerakoan, datu multzo honen argitaratzaileak oso aztarna gutxi ditu bere datu multzoarekin estekatzeko beste datu multzoak aurkitzeko. Egia da datu multzoei buruzko metadatoak biltzen dituzten The Datahub bezalako katalogoak existitzen direla, baina gehiengoetan eskasak izaten dira, datu multzoaren argitaratzailea erlazionatutako datu multzoen bila denbora gehiegi ematera behartuz. Problematika honi erantzuna emateko, datu multzoen egitura aztertuz estekatzeko hautagai diren datu multzoak gomendatzen dituen sistema bat garatu da doktorego-tesi honen barnean. Horretarako, RDF grafoen egituraren sintesia egiten da eta grafo-meatzaritza teknikak aplikatzen dira sarrien agertzen diren azpigrafoak erauzteko. Azpigrafo hauek erauzi ostean, antzekotasunak bilatzen dira haien artean, datu-multzoen artean posible izan litezkeen estekak ezartzeko. Gainera, soluzio honekin batera, datu-multzoak sintetizatzeko metodologia eta datu-multzoen arteko esteken gomendioak balioztatzeko saiakuntza-bankua garatu dira.

Agradecimientos

Cinco años han pasado desde que Diego me ofreció empezar a trabajar en el MORElab y por consiguiente en esta tesis doctoral. Cinco años en los que he dedicado una gran parte de mi tiempo y de mis esfuerzos al trabajo plasmado en este documento. Cinco años en los que muchas personas de mi alrededor me han ofrecido consejo, sugerencias y ánimos, o lo que es lo mismo: ayuda. Sin embargo, a pesar del largo tiempo pasado, me gustaría comenzar esta sección de agradecimientos por el final, por la gente que, en la etapa final de este trabajo, se prestó voluntaria a colaborar en el desarrollo de la evaluación de esta tesis doctoral. Compañeros que aún estando saturados de trabajo sacaron un par de horas de su valioso tiempo para ayudarme. Quisiera agradecer su colaboración a David Ausín, Aitor Almeida, Oscar Peña, Aitor Gómez Goiri, Juan López de Armentia, Aritz Bilbao, Idafen Santana, Edu Castillejo, Unai Aguilera, Oscar Corcho, Josu Bermúdez, Iván Pretel, Pablo Curiel, Diego López de Ipiña, David Bujan, Pablo Orduña, Koldo Zabaleta, Luis Rodríguez, Gorka Azkune, Diego Casado y Aimar Rodríguez. Gracias por haber dedicado aunque sea un sólo minuto de vuestro tiempo.

Dicho esto, me gustaría pasar a los agradecimientos individuales. En primer lugar me gustaría agradecer a mis directores Diego López de Ipiña y Oscar Corcho. A Diego, por haberme dado la oportunidad de comenzar esta aventura y proporcionarme los recursos necesarios para llegar hasta el final. A Oscar, por haberte involucrado en esta tesis de manera desinteresada y porque gracias a tu dedicación he aprendido muchísimo. Gracias a los "veteranos" del MORElab Unai, Aitor y Pablo por sus consejos y por haber solucionado mis dudas en todo

momento; tanto en lo que respecta a la tesis, como a los proyectos de investigación o al funcionamiento del laboratorio en general. A Aitor Gómez Goiri, por ser mi primer contacto dentro de este laboratorio e iniciarme en esto de la investigación (creo que todavía tenemos pendiente un Journal sobre ISMED :P). A mis compañeros del grupo de Linked Data Jon, Unai y Oscar por todo lo que he aprendido trabajando junto a vosotros. A David Ausín, siempre dispuesto a ayudar en lo que haga falta. No me puedo olvidar tampoco de Edu, Iván, Koldo, Pablo Curiel, Josu, David Buján y Aritz. Y gracias al resto de compañeros y ex-compañeros del MORElab, que no dudo que algo habréis aportado durante el transcurso de este trabajo, por pequeño que sea. ¡Ah!, y no podría dejar de dar las gracias a Erik Mannens y todo su equipo por la gran acogida que me dieron en Gante.

Gracias, Iran, por tu apoyo incondicional. Gracias a toda mi familia sanguínea y política por su interés y ánimos. La cuadrilla, cómo no, siempre ahí. Y como suele ser costumbre, lo mejor se deja para el final: eskerrik asko ama eta aita por haberme proporcionado la oportunidad de estudiar y llegar hasta este punto. Gracias por todas las facilidades que me habéis dado en esta vida y los sacrificios que habéis hecho por mi. Sin vosotros nunca lo habría conseguido.

Eskerrik asko,

Mikel Emaldi Manrique

octubre 2015

Índice general

Índice de figuras

Índice de cuadros

Índice de listados

Acrónimos

API Application Programming Interface

CSV Comma-Separated Values

DAML Darpa Agent Markup Language

FOAF Friend of a Friend

FSM Frequent Subgraph Mining

HTML Hypertext Markup Language

HTTP Hypertext Transfer Protocol

IFP Inverse Functional Property

LOD Linked Open Data

MDL Minimum Description Length

MIME Multipurpose Internet Mail Extensions

MS Microsoft

N3 Notation3

NER Named Entity Recognition

OIL Ontology Inference Layer

OWL Web Ontology Language

ACRÓNIMOS

PLN Procesado del Lenguaje Natural

RDF Resource Description Framework

RDFS RDF Schema

RDQL RDF Data Query Language

RSS Really Simple Syndication

SMOA String Metric for Ontology Alignment

SOAP Simple Object Access Protocol

SPARQL SPARQL Protocol and RDF Query Language

SQL Structured Query Language

URI Uniform Resource Identifier

URL Uniform Resource Locator

UTF Unicode Transformation Format

W3C World Wide Web Consortium

WWW World Wide Web

XHTML Extensible Hypertext Markup Language

XML Extensible Markup Language

XSLT Extensible Stylesheet Language Transformations

In fact, data is about our lives.

Sir Tim Berners-Lee

CAPÍTULO

1

Introducción

En el año 2001 Tim Berners-Lee (creador de la *World Wide Web* y director del *World Wide Web Consortium* (W3C)), James Hendler y Ora Lassila presentaban en su publicación "The Semantic Web" (Berners-Lee et al., 2001) un novedoso concepto: la Web Semántica. Presentada como una extensión de la Web de documentos, la Web Semántica haría posible la compartición del conocimiento existente en la Web de manera que pudiese ser procesado y comprendido por máquinas, con todo el potencial que ello ofrece. Para ello presentaban tecnologías como *Resource Description Framework* (RDF) (Brickley et al., 2004), que permitía describir la información a través de tripletas comprensibles por máquinas; el uso de las ya conocidas URI (Universal Resource Identifiers) para identificar todos los conceptos sin dar lugar a ambigüedades y posibilitando un modelo de datos auto-descriptivo; y su apuesta por las ontologías para representar el conocimiento de manera efectiva. De la misma manera, los autores remarcaban que la Web Semántica se convertiría en una realidad gracias a los agentes software o aplicaciones que de modo automático, sin mediación humana recogieran el conocimiento de diferentes fuentes para interactuar con él.

Más adelante, a medida que los desarrollos y la investigación alrededor de la Web Semántica iban floreciendo, Tim Berners-Lee propuso una nueva evolución

dentro de la Web: *Linked Data*, o tal y como se denomina en español, "datos enlazados". Presentado por Berners-Lee en forma de guía de diseño, el concepto *Linked Data* establecía cuatro reglas básicas para compartir el conocimiento a través de la Web (Berners-Lee, 2006):

1. Utilizar URI para la identificación de los objetos del mundo real.
2. Que dichas URI sean de tipo HTTP URI para que el acceso a ellas sea posible de manera universal.
3. Cuando se acceda a una URI, mostrar información relevante a través de los estándares del W3C (las diferentes representaciones de RDF o el lenguaje de consulta SPARQL (Prud'hommeaux y Seaborne, 2008)).
4. Incluir enlaces hacia otras URI, de manera que se pueda obtener nuevo conocimiento.

Con el cumplimiento de estas cuatro reglas, Berners-Lee aspiraba a convertir la Web en una gran base de conocimiento distribuida.

En el año 2010, Berners-Lee actualizó esta guía de diseño añadiendo la palabra *"open"* al término *Linked Data*, acuñando el término *Linked Open Data*. Este término comenzó a utilizarse para referirse a los datos enlazados que permitían su reutilización por terceros, tanto para su explotación como para el enriquecimiento de los mismos. Al mismo tiempo, estableció una clasificación (que comprende de una a cinco estrellas) para estimar el avance realizado por cada conjunto de datos hacia el estatus de *Linked Open Data*:

- 1⋆ Los datos están disponibles en la Web (en cualquier formato) con una licencia que permita considerarlos *Open Data*, o datos abiertos.
- 2⋆ Se encuentran disponibles en un formato fácilmente procesable por máquinas (por ejemplo, un documento *MS Excel* en lugar de una imagen de una tabla escaneada).
- 3⋆ Además de encontrarse en un formato fácilmente procesable por máquinas, este formato es libre (por ejemplo, un fichero CSV (*Comma-Separated Values*) en lugar de un fichero *MS Excel*).

4⋆ Los datos han sido publicados siguiendo los estándares del W3C (RDF y SPARQL).

5⋆ Los datos están enlazados con otros datos, proveyendo contexto y enriqueciéndolos.

A lo largo de los años se han ido desarrollando un sinfín de aplicaciones basadas en los planteamientos aquí recogidos, por lo que, tal y como auguraron sus principales ideólogos, la Web Semántica es hoy en día una realidad. Desde el primitivo explorador RDF *Tabulator* (Berners-Lee et al., 2006), hasta el reciente *Google Knowledge Graph*[1], innumerables universidades, grupos de investigación o grandes empresas como Yahoo![2] o Facebook[3] han invertido esfuerzo en investigar, desarrollar y mejorar sus servicios beneficiándose de las ventajas ofrecidas por la Web Semántica. Dentro de estas iniciativas cabe destacar la de Schema.org[4]. Impulsada desde su inicio por Bing, Google y Yahoo, esta iniciativa proporciona una serie de esquemas para etiquetar los documentos HTML de manera que estas etiquetas sean reconocidas por la mayoría de los motores de búsqueda.

Además, la publicación de conjuntos de datos sobre *Linked Open Data* ha ido aumentando a lo largo de los años. Desde su creación, la *Linked Open Data Cloud*[5] (catálogo de conjuntos de datos publicados como *Linked Open Data*) ha ido creciendo de manera considerable. Su primera versión, en el año 2007, contaba con doce conjuntos de datos; su última actualización, a mediados de 2014, con 570[6]. Por no hablar de proyectos como DBpedia (Auer et al., 2007), el gran proyecto colaborativo cuyo objetivo consiste en proporcionar como *Linked Open Data* toda la información almacenada en la Wikipedia[7]; o las bases de conocimiento colaborativas WikiData[8] y FreeBase[9], administradas por la Fundación Wikimedia y Google

[1] `http://www.google.com/insidesearch/features/search/knowledge.html`
[2] `http://semsearch.yahoo.com/`
[3] `http://ogp.me/`
[4] `http://schema.org/`
[5] `http://lod-cloud.net/`
[6] `http://data.dws.informatik.uni-mannheim.de/lodcloud/2014/`
[7] `http://www.wikipedia.org/`
[8] `http://www.wikidata.org`
[9] `https://www.freebase.com/`

respectivamente.

1.1 Motivación

A pesar de que, tal y como se ha expuesto a lo largo de este capítulo de introducción, hay una gran cantidad de esfuerzos implicados en el desarrollo de la Web Semántica y los datos enlazados, a día de hoy existe una gran cantidad de terreno inexplorado o, por lo menos, en el que no se han realizado los avances necesarios. La investigación desarrollada en el marco de esta tesis doctoral ha pretendido realizar su aportación dentro de uno de estos terrenos inexplorados de los conjuntos de datos enlazados o *Linked Data*, concretamente, en el terreno de la recomendación de conjuntos de datos enlazados. Para explicar de una manera clara la motivación de esta tesis doctoral, se pasa a exponer el siguiente caso de uso.

Si recordamos las reglas para la publicación de *Linked Data* expuestas en el presente capítulo, a la hora de generar y publicar un nuevo conjunto de datos enlazados se deben realizar los siguientes pasos:

1. **Analizar el dominio de los datos** a partir de los cuales se va a generar el conjunto de datos enlazados. Este análisis incluye tareas como la comprensión del propio dominio, el establecimiento de una ontología para representar el conocimiento descrito por los datos o el diseño de las URI a través de las cuales se van a identificar los recursos representados.

2. **Generar los datos en una de las representaciones de RDF**. Una vez definidas las ontologías y las URI a emplear, el siguiente paso consiste en convertir los datos en crudo o primitivos a RDF. Para ello existen diferentes representaciones de este estándar, como RDF/XML (Beckett y McBride, 2004), Turtle (Beckett et al., 2013), N-Triples (Beckett y Carothers, 2013), Notation3 (Berners-Lee y Connolly, 2011), etc.

3. **Enlazar** los datos generados con otros datos de terceros, para así, poder enriquecerlos y contextualizarlos.

4. **Publicar** estos datos de manera que estén accesibles de manera universal y a través de estándares del W3C como el lenguaje de consulta SPARQL.

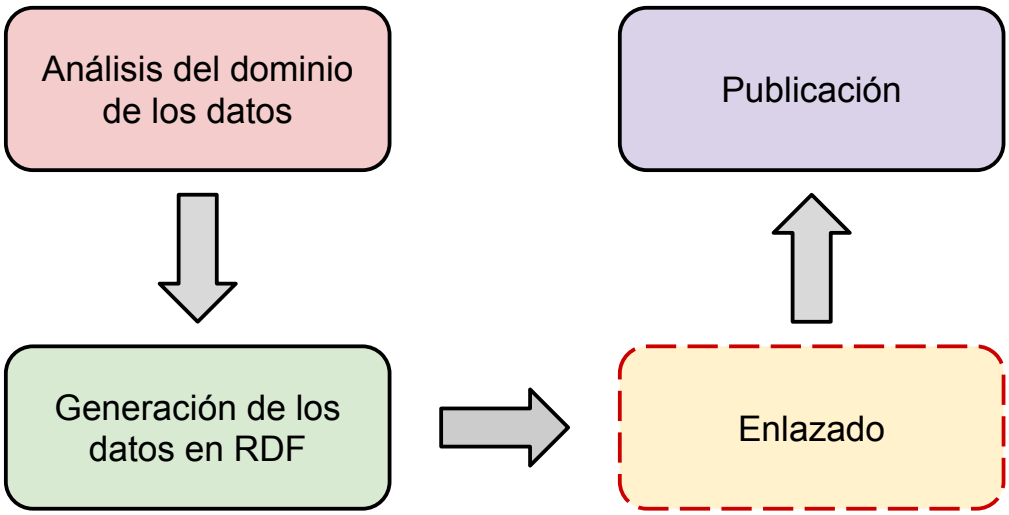


Figura 1.1: Fases para la generación y publicación de conjuntos de datos enlazados.

Para los pasos primero, segundo y cuarto existen numerosas herramientas para facilitar su desarrollo. Entre estas herramientas podemos encontrar buscadores de ontologías (d´Aquin et al., 2007), catálogos de vocabularios como el de la iniciativa *Linked Open Vocabularies*[1], guías de diseño de URI (Sauermann et al., 2007; Berners-Lee, 1998), herramientas para la manipulación de datos (OpenRefine[2] y su extensión para generar RDF (Maali et al., 2011)), numerosos repositorios RDF y puntos de acceso SPARQL (Openlink Virtuoso[3], Fuseki[4] o Stardog[5]), etc. Para el paso tercero, si bien es cierto que también existen numerosas herramientas que facilitan el enlazado entre diferentes conjuntos de datos, tal y como se expone en el capítulo 2, la mayoría de ellas parten de la premisa de que el usuario conoce con qué conjunto de datos quiere enlazar el suyo propio. Tal y como se ha expuesto al comienzo de este capítulo, la *LOD Cloud* ha aumentado de manera considerable desde su creación; si continúa esta tendencia, llegará el momento en el que examinar todos sus conjuntos de datos, con el objetivo de seleccionar cuáles de ellos son los apropiados para ser enlazados con nuestros propios conjuntos de datos, se

[1] `http://lov.okfn.org/dataset/lov/`
[2] `http://openrefine.org/`
[3] `http://virtuoso.openlinksw.com/`
[4] `http://jena.apache.org/documentation/serving_data/`
[5] `http://stardog.com/`

convertirá en una tarea verdaderamente tediosa e incluso inmanejable.

Y este es el terreno que la investigación realizada en el marco de esta tesis doctoral ha querido explorar: a partir de un conjunto de datos enlazados, determinar, dentro de un catálogo de datos enlazados como puede ser la *LOD Cloud*, los conjuntos candidatos a ser enlazados entre si. De esta manera, a través de este trabajo de investigación se pretende facilitar la búsqueda de conjuntos de datos relevantes para completar la tarea de enlazado entre diferentes conjuntos de datos, tan característica de la publicación de datos como *Linked Data*.

1.2 Metodología de investigación

Para el correcto desarrollo de esta tesis doctoral se ha seguido la metodología representada en la figura 1.2, cuyas diferentes fases se describen a continuación:

- **Revisión biblográfica:** durante esta fase se han analizado los trabajos existentes en el campo en el que está situada la investigación realizada. La selección de los trabajos analizados se ha realizado a través de la búsqueda en las conferencias y revistas más relevantes del campo. El resultado de este análisis queda reflejado en el capítulo 2 de esta tesis doctoral.

- **Definición de hipótesis, objetivos y restricciones:** una vez analizados los trabajos más relevantes, se ha planteado la hipótesis que se pretende demostrar a través del trabajo realizado a lo largo de esta tesis doctoral. Junto con la hipótesis, se han planteado los objetivos que debe cumplir el trabajo desarrollado, además de sus restricciones, con el objetivo de acotar y delimitar claramente el alcance de la investigación realizada. Tanto la hipótesis como los objetivos y restricciones se han plasmado en la sección 1.3.

- **Planteamiento de la solución:** en esta fase se ha planteado la solución técnica a través de la cual se pretende validar la hipótesis planteada. Esta solución debe cumplir los objetivos y restricciones previamente planteados.

- **Desarrollo:** durante esta fase se ha desarrollado la solución planteada. Tanto la solución planteada como el desarrollo de la misma se explican en el capítulo 3.

- **Evaluación:** el objetivo de esta fase consiste en obtener una serie de métricas e indicadores que permitan ponderar tanto cualitativa como cuantitativamente los resultados obtenidos a partir de la aplicación de la solución desarrollada sobre un conjunto de datos de prueba. Una vez obtenidos los resultados de la evaluación, estos se comparan con los resultados conseguidos por otras posibles soluciones del estado del arte o con los de soluciones de referencia creadas para tal efecto durante esta tesis doctoral. Si se diese el caso de que estos resultados no fuesen satisfactorios, se replantería la solución con el objetivo de mejorarla. Tanto los resultados de la evaluación como la comparación con otras soluciones se plasman en el capítulo 4.

- **Conclusión:** durante esta fase se plantean las conclusiones extraídas a lo largo de la realización de esta tesis doctoral, tal y como puede observarse en el capítulo 5.

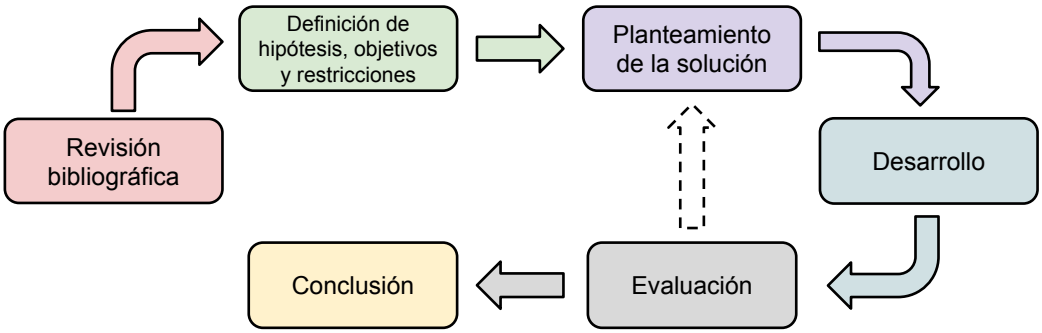


Figura 1.2: Metodología aplicada para el desarrollo de esta tesis doctoral.

1.3 Hipótesis y objetivos

1.3.1 Metas y retos de investigación

La meta de esta tesis consiste en avanzar en el estado del arte de los sistemas de búsqueda y recomendación de Linked Data. Para ello se pretende el **desarrollo de un modelo que, a través del análisis estructural de los grafos formados por los conjuntos de datos RDF, permita determinar la posible relación existente entre diferentes conjuntos de datos**. Esta problemática puede ser definida de manera similar a la de Leme et al. (2013):

> Dado un conjunto de datos S, calcular una puntuación para cada uno de los conjuntos de datos $T_i(i = 1, ..., m)$ del repositorio $\mathbf{T}$, siento esta puntuación favorable a los conjuntos de datos que tengan mayor probabilidad de contener recursos que puedan ser enlazados con recursos de S.

Alcanzar esta meta implica la superación de una serie de retos de investigación:

- Tal y como se expone en el análisis crítico del estado del arte (sección 2.4.4), las soluciones existentes en el campo de la identificación de fuentes para el enlazado de conjuntos de datos estructurados presentan algunas carencias que se pretenden solventar a lo largo de esta tesis doctoral. Algunas de ellas son:
  - Romper la dependencia con las etiquetas textuales.
  - Solucionar el problema del arranque en frío (*cold start* en inglés).
- Dada la gran cantidad de tripletas de algunos de los conjuntos de datos enlazados ubicados en la LOD Cloud, se requiere el planteamiento de una técnica para la simplificación o resumen de la estructura de dichos conjuntos de datos.
- Desde el punto de vista técnico se pueden hallar los siguientes retos:

  - Las herramientas de minería de grafos existentes son incapaces de trabajar con datos en formato RDF, empleando cada una de ellas su propio formato de entrada. Por lo tanto, existe la necesidad de desarrollar una herramienta para la conversión de ficheros RDF al formato de entrada de la herramienta de minería de grafos seleccionada.
  - El gran tamaño de los conjuntos de datos a tratar requiere el desarrollo de herramientas y algoritmos que procesen estos conjuntos de datos de manera eficiente.

1.3.2 Contribuciones al estado del arte

Para superar los retos de investigación planteados, esta tesis contribuirá al estado del arte con una serie de herramientas y modelos descritos a lo largo del capítulo 3. Estos avances consistirán en:

- Desarrollo de un método que permita la síntesis o resumen de la estructura de los grandes conjuntos de datos, de cara a su posterior análisis.
- Desarrollo de una herramienta para la conversión de grafos RDF a un formato procesable por la herramienta de minería de grafos seleccionada.
- Desarrollo de un sistema que dada una serie de conjuntos de datos enlazados, como podría ser la LOD Cloud, recomiende los conjuntos de datos a enlazar.
- Desarrollo de un *"gold standard"* que permita la evaluación tanto del sistema desarrollado en el seno de esta tesis como por los sistemas de recomendación de Linked Data desarrollados en el futuro.

1.3.3 Supuestos de trabajo, hipótesis y restricciones

El trabajo desarrollado en esta tesis se basa en que se cumplen una serie de **supuestos** listados a continuación. Estos supuestos permiten entender las decisiones metodológicas o técnicas tomadas a lo largo de esta tesis:

S1. Los conjuntos de datos empleados están descritos correctamente en formato RDF y siguiendo los principios de Linked Data: no tienen errores de sintaxis, las ontologías empleadas pueden ser accedidas a través de su URI correspondiente y se utilizan correctamente las clases y propiedades de estas ontologías.

S2. Las instancias descritas en estos conjuntos de datos pueden pertenece a más de una clase ontológica.

S3. De la misma manera, las ontologías empleadas por estos conjuntos de datos están descritas correctamente en el lenguaje OWL.

A partir de estos supuestos, se puede plantear la hipótesis que se pretende validar a través del trabajo de investigación realizado durante esta tesis doctoral:

> *A partir de un conjunto de datos estructurado, se pueden hallar, de manera automática, otros conjuntos de datos idóneos para su enlazado solamente a través del análisis de las características estructurales de los propios conjuntos de datos.*

Por último, la solución desarrollada para resolver la problemática planteada cuenta con una serie de **restricciones** que delimitan el radio de acción de la investigación llevada a cabo a lo largo de esta tesis:

R1. Los conjuntos de datos enlazados estarán descritos en RDF. No se admiten datos estructurados en otros formatos como pueden ser bases de datos, hojas de cálculo, etc.

R2. La alternativa planteada en esta tesis no tendrá en cuenta avances en términos de eficiencia, solamente tratará cuestiones relativas a la eficacia de la solución.

R3. La solución planteada en ningún caso establece enlaces entre los diferentes conjuntos de datos, solamente determina posibles candidatos a ser enlazados. Una vez conocidos estos candidatos podrían emplearse alguna de las herramientas descritas en la sección 2.2, tal como Silk o Limes (secciones 2.2.3 y 2.2.5).

1.4 Organización de la tesis

Tras esta introducción, a lo largo del capítulo 2 se expone la revisión del estado del arte realizada. Además de la síntesis de los trabajos seleccionados, esta revisión del estado del arte incluye el análisis crítico de cada uno de estos trabajos. A continuación en el capítulo 3 se presenta la solución propuesta para la problemática planteada. Los evaluación de dicha propuesta se muestra en el capítulo 4. Para finalizar, en el capítulo 5 se exponen las conclusiones y el trabajo futuro.

Izan zirelako gara, garelako izango dira.

Jose Migel Barandiaran

CAPÍTULO

2

Estado del arte

A lo largo de este capítulo se presenta un resumen de las soluciones existentes dentro de los diferentes campos de la Web Semántica relacionados con la investigación desarrollada en esta tesis doctoral. En primer lugar, en la sección 2.1 se analizan los trabajos relacionados con la búsqueda de datos estructurados. En la sección 2.2 se analizan los diferentes *frameworks* para el descubrimiento de enlaces entre instancias de conjuntos de datos estructurados. Ya que, en esta tesis, uno de los pasos previos a la recomendación de conjuntos de datos candidatos a ser enlazados consiste en sintetizar los conjuntos de datos, en la sección 2.3 se analizan las técnicas de sintetizado de conjuntos de datos estructurados desarrolladas hasta la fecha. A continuación, en la sección 2.4 se estudian los trabajos relacionados con la identificación de fuentes de datos candidatas a ser enlazadas.

De la misma manera, en la sección 2.5 se muestran los diferentes algoritmos para la búsqueda de las subestructuras más frecuentes de un grafo, ya que como se detalla en el capítulo 3, es el punto de partida de la solución propuesta.

Al final de cada sección se realiza el análisis crítico correspondiente con el objetivo de mostrar las fortalezas y debilidades de cada trabajo.

2.1 Búsqueda de datos estructurados

En esta sección se analizan las diferentes herramientas y trabajos para la búsqueda de datos publicados como Linked Data existentes hasta la fecha. Además de las herramientas orientadas hacia *Linked Data*, se incluye la descripción de una serie de buscadores de ontologías, con el objetivo de mostrar la evolución de las técnicas de búsqueda dentro de la Web Semántica en general. En el siguiente cronograma se pueden observar los trabajos analizados:

2.1.1 OntoKhoj

OntoKhoj (Patel et al., 2003) proporciona una interfaz para facilitar la reutilización de ontologías, tanto por parte de ingenieros como por agentes Web. El primer paso realizado por OntoKhoj consiste en poblar su propia base de conocimiento con diferentes ontologías existentes en la Web. Esta tarea se desarrolla realizando, en primer lugar, un proceso de *crawling* en el que se recoge el diferente contenido semántico, y en segundo lugar, clasificando las ontologías en función de su dominio, consultando el repositorio DMOZ[1].

Una vez clasificadas, OntoKhoj aplica una variación del algoritmo PageRank (Page et al., 1998) denominada OntoRank para establecer la relevancia de cada una de las ontologías en función de su relación. Solamente se contemplan tres tipos de relaciones, teniendo cada una de ellas un peso diferente: 1) relaciones de instancia (`rdf:type`), 2) de subclase (`rdfs:subclass` y `daml:subClass`) y 3) relaciones de dominio o rango (`rdfs:domain`, `rdfs:range`, `daml:domain` y `daml:range`).

[1] `http://www.dmoz.org`

$$OR(O) = N * \sum_{i=1}^{\alpha}(1/\Omega_i) * \sum_{j=1}^{\beta_i} OR(O_i) * T_j \tag{2.1}$$

En la ecuación 2.1 se muestra la definición formal de OntoRank, siendo O la ontología para la cual se quiere calcular su OntoRank OR, α el número de ontologías que enlazan hacia O, β_i el número de enlaces de cada O_i hacia O, Ω_i el número total de enlaces de O_i hacia cualquier ontología, T el peso del enlace y N el factor de normalización.

Por último, OntoKhoj ofrece dos vías para realizar la búsqueda de ontologías dentro de su sistema:

- **Interfaz de consulta orientada al contexto:** a partir de un término introducido por el usuario, OntoKhoj muestra los diferentes significados que el término pudiera tener. Una vez que el usuario ha seleccionado el significado correcto, Ontokhoj realiza la búsqueda de ontologías relacionadas con dicho término. En caso de no encontrar resultados, extrae los sinónimos de dicho término a través de WordNet (Miller, 1995), buscando ontologías relacionadas con dichos sinónimos. Si aún así, todavía no se ha encontrado una ontología que satisfaga al usuario, se buscarán ontologías relacionadas con el hiperónimo del término original.

- **Interfaz máquina:** esta interfaz permite que los agentes Web puedan consultar el directorio de ontologías de OntoKhoj. Este repositorio está representado en RDF y permite realizar consultas a través de los lenguajes RDQL (Seaborne, 2004) o F-Logic (Kifer y Lausen, 1989).

2.1.2 TAP

TAP (Guha et al., 2003) es una infraestructura formada por TAPache (una extensión del popular servidor HTTP Apache[1]) y una interfaz de consulta denominada *GetData*.

El funcionamiento de TAPache es muy sencillo: basta con crear un directorio homónimo al directorio donde se encuentren los documentos HTML, en el que se

[1] `http://httpd.apache.org/`

almacenarán los ficheros RDF correspondientes a cada documento HTML. Para acceder a los ficheros RDF correspondientes, se empleará la propia URL del fichero RDF o la interfaz *GetData* descrita a continuación. Además, TAPache permite agregar todos los ficheros RDF de un directorio accediendo a la URL correspondiente a dicho directorio.

Por su parte, la interfaz *GetData* se ofrece como un servicio Web SOAP (Box et al., 2000). La idea sobre la cual se desarrolla esta interfaz consiste en que al igual que las grandes páginas web no proporcionan acceso a su base de datos a través del lenguaje SQL, ya que podrían generarse consultas con un alto coste computacional, lo mismo debería ocurrir para la Web Semántica y el acceso a sus datos a través de SPARQL. *GetData* permite realizar consultas sobre datos RDF de manera sencilla, como:

$$GetData(< resource >, < property >) \Rightarrow < value >$$

Esta consulta, dados un recurso y una propiedad de dicho recurso, devolverá los objetos asociados a dicha tupla de sujeto y predicado. Además, TAP proporciona dos interfaces más para explorar los recursos RDF:

- **Búsqueda:** esta interfaz toma como dato de entrada una cadena de caracteres y devuelve todos los recursos en los que dicha cadena aparezca como objeto de la propiedad `TitleProperty`.
- ***Reflection:*** dado un recurso, esta interfaz devuelve todos los recursos que tienen relación con el mismo, ya sea a través de enlaces hacia fuera o hacia dentro.

2.1.3 Swoogle

Swoogle (Ding et al., 2004) extrae la información semántica de cada documento descubierto a través del *crawling*, computando las relaciones entre los diferentes documentos e indexándolos a través de un sistema que utiliza n-gramas y las URI de los recursos como palabras clave. Entre estos documentos, Swoogle permite buscar tanto ontologías (*Semantic Web Ontology*), como instancias de dichas ontologías

(*Semantic Web Database*). Además, permite analizar las características de la Web Semántica a través de las relaciones entre los diferentes recursos.

Dentro de la arquitectura de Swoogle pueden identificarse cuatro componentes principales que se describen a continuación.

2.1.3.1 Búsqueda de SWD

Dentro del marco de la búsqueda de SWD, el equipo de Swoogle ha desarrollado un conjunto de *crawlers*: *Google Crawler*, *Focused Crawler* y *Swoogle Crawler*. El primero de ellos busca documentos de la Web Semántica (`.rdf`, `.owl`, `.daml` y `.n3`) a través del buscador Google. Por su parte, El denominado *Focused Crawler* permite al usuario insertar una URL determinada en busca de los SWD a los que dirige dicha URI. A partir de los SWD extraídos por estos dos *crawlers* el *Swoogle Crawler* extrae nuevos SWD a través de sus relaciones (`owl:imports` y `rdfs:seeAlso`).

2.1.3.2 Metadatos de los SWD

Swoogle identifica tres categorías de metadatos:

- **Metadatos básicos:** Los metadatos básicos generados por Swoogle se clasifican en tres categorías:
  - *Características del lenguaje:* características sintácticas y semánticas de los SWD como la codificación (RDF/XML, N-TRIPLE o N3), el lenguaje (OWL, DAML, RDFS y RDF) o el nivel del lenguaje OWL utilizado (OWL-LITE, OWL-DL u OWL-FULL).
  - *Estadísticas RDF:* estas estadísticas se refieren a la proporción de clases, propiedades e instancias dentro de un SWD. En función del ratio de cada elemento, el SWD se clasificará como SWDB (*Semantic Web Database*) o SWO (*Semantic Web Ontology*).
  - *Anotaciones de la ontología:* propiedades que describen una SWO, como pueden ser `rdfs:label`, `rdfs:comment`, `owl:versionInfo` o `daml:versionInfo`.

- **Relaciones entre los SWD:** Swoogle captura las diferentes relaciones entre los SWD, como pueden ser la extensión, la importación o el versionado de ontologías.

- **Resultados analíticos:** La principal función de los resultados analíticos es clasificar los SWD en función a su importancia. Para ello, Swoogle desarrolla una variación del algoritmo PageRank de Google (Page et al., 1998) denominado *Rational Random Surfer*.

2.1.3.3 Indexado de SWD

A la hora de indexar los SWD, Swoogle emplea técnicas desarrolladas en el ámbito de la búsqueda textual, por dos principales motivos:

- Al ser técnicas que han sido investigadas ampliamente, proporcionan un rendimiento adecuado.

- No sólo se quiere indexar el contenido semántico de los SWD, si no, el texto plano también.

Estas técnicas están basadas en palabras clave o n-gramas (cadenas de texto de n caracteres). Para la aplicación de estas técnicas de indexado, se divide la URI de los recursos en *tokens* (cada uno de los componentes léxicos que forman la URI). Así, al igual que un documento puede ser tratado como un conjunto de palabras, un SWD puede tratarse como un conjunto de *tokens* de las URI de los recursos que describe.

2.1.4 OntoSelect

OntoSelect (Buitelaar et al., 2004) es una librería que recopila ontologías de cualquier dominio a través de la monitorización de la web y el empleo de técnicas de *crawling*. Las ontologías extraídas son analizadas a través de OWL API (Horridge y Bechhofer, 2011), con el objetivo de extraer sus clases y propiedades. Estas ontologías son almacenadas en una base de datos junto con los nombres y etiquetas (`rdfs:label`) de sus clases y propiedades.

Para la exploración de esta base de datos, OntoSelect proporciona al usuario una intefaz en forma de página web, a través de la cual el usuario puede buscar en base al nombre de la ontología, su lenguaje (OWL, DAML o RDFS), sus etiquetas o el número de etiquetas, clases, propiedades u ontologías que importa.

2.1.5 MultiCrawler

MultiCrawler (Harth et al., 2006) es una plataforma cuyo objetivo consiste en permitir la realización de consultas sobre la Web Semántica. Para ello, los autores ven necesario realizar un gran proceso de *crawling* y análisis, que descomponen en las siguientes cinco fases:

- ***Fetch*:** durante esta fase, se descarga el contenido de las URL objetivo y se actualiza su entrada en el gestor de metadatos de MultiCrawler. A continuación, se transfiere dicha URL a la siguiente fase.
- ***Detect:*** esta fase se encarga de detectar el contenido del fichero descargado. Para ello, compara la extensión del fichero con el tipo MIME del fichero. Si ambos resultados coinciden y se encuentran dentro de la lista de tipos que MultiCrawler soporta, se continúa con el proceso.
- ***Transform:*** para transformar los ficheros descargados en RDF, se aplican diferentes métodos, en función del tipo de archivo:
  - Si el contenido no es XML, como es el caso de los ficheros HTML, se convierte en XML a través de herramientas como Tidy[1].
  - Para convertir el contenido XML en RDF se han creado diferentes plantillas XSLT (*Extensible Stylesheet Language Transformations*). MultiCrawler dispone de plantillas para convertir a RDF ficheros RSS 2.0, Atom y XHTML.
- ***Index:*** el objetivo de esta fase consiste en soportar las diferentes consultas realizadas por el usuario de manera eficiente. MultiCrawler dispone de un índice constituido por N-Quads (Carothers, 2014), que además de almacenar

[1] `http://tidy.sourceforge.net/`

sujeto, predicado y objeto, dotan de contexto a la tripleta; y de un segundo índice, denominado *Lexicon*, que asocia cadenas de caracteres con el *quad* en el que aparecen.

- ***Extract:*** en esta fase se extraen las URI que se encuentran almacenadas en el índice, para poder realimentar todo el proceso con nuevas URI. Para ello, se buscan predicados que típicamente tienen URI como objeto, como pueden ser `rdfs:seeAlso` o `rss:link`. Una vez recuperadas, existe la posibilidad de aplicar diferentes filtros a las mismas. Por ejemplo, podría desearse excluir un dominio determinado, solamente recuperar las URI que cumplan una determinada expresión regular, etc.

2.1.6 Watson

El objetivo de Watson (d´Aquin et al., 2007) consiste en habilitar un punto de entrada hacia las ontologías y la información semántica a través de *a*) la recopilación del conocimiento semántico disponible en la Web; *b*) el análisis de dicho contenido con el objetivo de extraer los metadatos útiles para su indexación; *c*) y la implementación de un sistema de consultas para facilitar el acceso a los datos.

Dentro de la arquitectura de Watson pueden encontrarse tres capas principales:

- **Descubrimiento de ontologías:** Watson realiza la captura inicial de datos a partir del repositorio ontológico del editor de ontologías *Protégé* (Noy et al., 2001), a través de la búsqueda en Swoogle (sección 2.1.3) de la lista de palabras más populares del inglés y a través de la extracción de información semántica de los resultados con extensión `.owl` del buscador Google.

- **Generación de metadatos:** en esta capa se generan los metadatos correspondientes a las ontologías recopiladas. Entre estos metadatos pueden encontrarse el lenguaje ontológico empleado (OWL, DAML+OIL, etc.) o las entidades halladas (clases, propiedades, individuales y literales).

  Además, en esta capa se realiza la detección de conocimiento duplicado. Para ello, Watson realiza una primera comparación, puramente sintáctica, seguida

de una segunda comparación en la que se comparan los modelos representados por las diferentes ontologías, independientemente del lenguaje empleado para representarlas.

- **Búsqueda y consulta:** Watson proporciona la búsqueda por palabras clave a través del servicio Apache Lucene[1], permitiendo el empleo de múltiples palabras clave y de operadores lógicos. Esta búsqueda se realiza contra nombres locales de las entidades, etiquetas, comentarios y literales hallados en las ontologías, utilizando varias funciones de búsqueda como la coincidencia exacta, parcial, aproximada, etc.

  A la hora de mostrar la información obtenida al usuario, se pueden mostrar las ontologías en su totalidad, o solamente determinadas clases u otras entidades.

2.1.7 Sindice

Sindice (Tummarello et al., 2007) es un servicio online, cuya misión es la búsqueda de información acerca de un recurso determinado a través de la monitorización de la Web y la indexación de documentos RDF. La diferencia entre Sindice y la mayoría de los trabajos descritos en este estado del arte, reside en que Sindice no almacena los documentos hallados. Al contrario, Sindice almacena la URL del documento en el que se encuentra la información relacionada con una serie de términos. Esto hace que Sindice sea lo más parecido a un buscador tradicional de documentos adaptado para la Web Semántica. Como puntos de entrada, Sindice ofrece tanto una interfaz Web para usuarios como una API para su acceso de manera programática. A través de esta API, Sindice ofrece tres servicios a los clientes:

- $index(url) \Rightarrow nil$: extrae la información del documento RDF o *endpoint* SPARQL al que apunta la URL.

- $lookup(uri) \Rightarrow url[]$: busca un recurso identificado por la URI, devolviendo una lista de fuentes en las que aparezca dicho recurso.

[1] `http://lucene.apache.org/`

- $lookup(ifp, value) \Rightarrow url[\,]$: busca un recurso que esté identificado por la tupla propiedad-valor indicada, devolviendo una lista de fuentes en las que el recurso aparece.

- $lookup(text) \Rightarrow url[\,]$: busca el texto indicado, devolviendo una lista de fuentes en las que el texto aparece.

Por su parte, la arquitectura de Sindice está formada por componentes independientes que se comunican entre si para realizar el *crawling*, el indexado y las consultas pertinentes:

- ***Crawler:*** Su función consiste en extraer información estructurada de las diferentes fuentes de datos, depositándola en la cola de indexación (*indexing queue*).

- ***Indexing queue:*** Recibe las fuentes de datos tanto desde el *crawler* como desde las peticiones explícitas hechas por los administradores de las fuentes de datos a través de la interfaz web. Se comunica con el *gatekeeper* para establecer la prioridad de la cola.

- ***Gatekeeper:*** Se encarga de establecer la prioridad de los elementos de la cola de indexación en base a criterios como la fecha en la que se analizó la fuente anteriormente, la fecha de su última modificación, el resumen de su contenido, etc.

- ***Indexer:*** El indexador extrae las URI, las IFP (*Inverse Functional Property*)[1] y las palabras clave de cada documento, añadiendo cada una de ellas a su índice correspondiente. De esta manera, durante la fase de *lookup* la consulta solamente se envía a su índice correspondiente. Dichos índices están desarrollados bajo la tecnología Solr[2].

[1] Cuando una propiedad se declara como inversa-funcional, significa que el sujeto de la tripleta es el único sujeto que puede estar conectado a el objeto de la tripleta a través de esta propiedad. Por ejemplo, si la propiedad `:biologicalMotherOf` está declarada como IFP, significa que el sujeto de la tripleta `?s1 :biologicalMotherOf ?o` es el único sujeto que puede estar conectado a ese objeto concreto a través de esta propiedad. Por lo tanto, si existiese la tripleta `?s2 :biologicalMotherOf ?o`, se podría decir que `?s1` y `?s2` representan el mismo concepto.

[2] `http://lucene.apache.org/solr/`

- ***Reasoner:*** se encarga de extraer las IFP de cada documento, para que el indexador pueda trabajar con ellas.

2.1.8 SWSE

SWSE (Semantic Web Search Engine) (Harth et al., 2007) es un motor de búsqueda que además de recopilar los datos semánticos de la Web, realiza tareas de integración y consolidación de dichos datos.

Para la recopilación de los datos iniciales, SWSE emplea otra herramienta ya mencionada en este estado del arte, tal y como es MultiCrawler (sección 2.1.5). Una vez recogidos los datos, se realiza un proceso de consolidación, con el objetivo de agrupar toda la información relativa a un recurso. Cuando esta información se encuentra identificada por una URI común, la tarea es sencilla y automática. Cuando las URI difieren o no existen, se trata de consolidar los recursos analizando los valores de las IFP (ver nota al pie número 1 de la página 22). Por otra parte, se realiza el enlazado entre entidades a través del análisis de las propiedades de tipo `rdfs:seeAlso`. Además, SWSE permite que el usuario agregue sus propias fuentes de datos para ser analizadas.

Por último, SWSE ofrece una interfaz de usuario similar a un buscador Web convencional. A partir de una serie de términos introducidos por el usuario, SWSE muestra todos los resultados, pudiéndolos filtrar por el tipo (`rdf:type`) de cada uno. Según el usuario accede a los diferentes enlaces de cada recurso, puede ir descubriendo nueva información relacionada.

2.1.9 Falcons

Falcons (Cheng y Qu, 2009) es un motor de búsqueda de entidades semánticas basado en *keywords* (palabras clave). Para servir estas consultas, Falcons se basa en los denominados *documentos virtuales*, documentos que contienen descripciones textuales de los documentos RDF indexados. A partir de estos documentos virtuales se crean dos índices inversos: el primero permite encontrar una entidad a partir de una serie de palabras clave; el segundo relaciona una entidad con las clases a las que dicha entidad pertenece en las diferentes ontologías con las que es descrita,

dentro de los diferentes recursos RDF en los que aparece. Además, Falcons permite la navegación a través de la jerarquía de clases de las entidades mostradas, permitiendo el filtrado de la búsqueda.

La recogida de datos se realiza a través del ya mencionado proceso de *crawling*. Este *crawler* se alimenta inicialmente a partir de *a*) los resultados de búsqueda obtenidos por Google y Swoogle (sección 2.1.3) al insertar una serie de términos de las tres categorías principales de DMOZ; *b*) otros repositorios semánticos como los ya mencionados Schema Web o Ping the Semantic Web; e *c*) insertando manualmente algunas URI de conjuntos de datos de la LOD Cloud. Para facilitar la tarea del *crawler* únicamente se acepta el tipo de contenido HTTP `application/rdf+xml`.

Una vez recopilados los documentos RDF, Falcons construye los llamados *documentos virtuales*, en los que basa su funcionamiento. Un documento virtual es una descripción textual de una entidad, creado de manera que permita la implementación de un índice invertido que apunte desde los términos de dicho documento virtual a la entidad correspondiente. Dados los principios de *Linked Data*, una entidad debe estar relacionada con otras, por lo que se puede extraer información acerca de una entidad desde otra entidad diferente. Para poder incluir esta información externa dentro del documento virtual de una entidad, Falcons emplea el concepto de *sentencia RDF* (Zhang et al., 2007). Una vez construido el documento virtual, este es procesado por Apache Lucene[1]. Al ser un documento de texto, Lucene tiene la capacidad de procesarlo y crear el índice correspondiente.

A la hora de realizar la búsqueda, Falcons clasifica los resultados en función de dos factores: la **relevancia del resultado respecto a la consulta** y la **popularidad del resultado**. En el primer caso se refiere a la similitud entre la consulta realizada por el usuario y cada uno de los objetos que forman parte del resultado. Para un objeto o, siendo $VDoc(o)$ su documento virtual, y para una consulta q, siendo $Vector(q)$ su vector de palabras clave, la similitud por coseno de ambos vectores será:

$$TextSim(o,q) = \frac{VDoc(o) \cdot Vector(q)}{|VDoc(o) \cdot Vector(q)|} \tag{2.2}$$

[1] `http://lucene.apache.org`

Para calcular la popularidad de un resultado, Falcons tiene en cuenta el número de documentos en los que aparece un objeto. Para un objeto o, $Docs(o)$ será el número de documentos en el que aparece mencionado, calculando la popularidad de la siguiente manera:

$$Popularity(o) = \log(|Docs(o)| + 1) \tag{2.3}$$

Por último, la clasificación final de un objeto o respecto a una consulta q se define como:

$$ObjectScore(o, q) = TextSim(o, q) \cdot Popularity(o) \tag{2.4}$$

Una vez realizada la búsqueda, el principal objetivo de Falcons es mostrar los resultados de manera que sean útiles para el usuario. Para ello se ha elegido el modelo de *snippet*. Un *snippet* es un fragmento de la información resultante de una consulta, en el cual se muestra la información con mayor relación con dicha consulta. Para construir este *snippet*, Falcons deberá elegir el conjunto de tripletas RDF que más se ajusten a la consulta realizada por el usuario. Además, tal y como se ha explicado al inicio de esta sección, Falcons permite refinar una determinada búsqueda filtrando sus resultados por la clase de una ontología a la que pertenecen las entidades mostradas, construyendo un segundo índice que asocia entidades con las clases a las que pertenece.

2.1.10 Sig.ma

Sig.ma (Tummarello et al., 2010) es un servicio cuyo objetivo reside en la agregación de información de la Web Semántica. Para ello, Sig.ma explora la Web Semántica para poder construir una vista agregada de toda la información que se disponga al realizar una búsqueda textual o sobre un determinado recurso. Al contrario que otros trabajos, centrados en mostrar al usuario las fuentes de datos o recursos relacionados con la búsqueda realizada, Sig.ma, al estar situada en una capa por encima de Sindice (sección 2.1.7) se encarga de construir una **vista** con la información extraída de todas las fuentes de datos y recursos relacionados con dicha búsqueda. Como método de acceso al servicio, Sig.ma proporciona tanto una API como una interfaz web.

Sig.ma implementa las siguientes fases para conseguir completar su tarea:

- **Creación del plan de consulta:** para la creación del plan de consulta, Sig.ma toma como entrada una serie de palabras clave, una serie de URL o una serie de URI. La diferencia entre una URL y una URI consiste en que la primera apunta a un documento web que puede alojar varios recursos, y la URI identifica un recurso único, independientemente de que dicha URI pueda ser accedida a través de la Web.

- **Selección de las fuentes de datos:** en primer lugar Sig.ma consulta en su índice las fuentes de datos en las cuales aparece la búsqueda realizada por el usuario, además de realizar dicha búsqueda a través de Yahoo! BOSS API[1]. Dicha API permite realizar búsquedas textuales en Yahoo!, de cuyos resultados Sig.ma extrae el conocimiento estructurado. Una vez obtenido tanto las fuentes del índice como las proporcionadas por Yahoo! se realiza un *ranking* en función del grado de aparición de la búsqueda en dichas fuentes. En el caso de que la lista de fuentes sea muy extensa, se recorta a 25 fuentes, pudiendo el usuario solicitar más fuentes en el caso de que la información mostrada le resulte insuficiente.

- **Recolección de datos en paralelo:** Una vez que se ha extraído la información estructurada de las fuentes, se almacenan en un repositorio local, con el objetivo de proporcionar un acceso rápido a la misma. Ya que la rapidez de acceso a la información es uno de los principales factores que afectan al rendimiento de Sig.ma, el sistema dispone de tres capas para almacenar la información: un servidor *memcached* (Fitzpatrick, 2004), en el que se almacenan todas las búsquedas recientes; el mencionado repositorio local; y la propia URL de los recursos. Si la URL dispone de información semántica, se envía a la cola de Sig.ma para ser analizada posteriormente.

- **Extracción y alineamiento de subgrafos relacionados:** cada documento RDF extraído se divide en los diferentes recursos que describen. Una vez

[1] `http://developer.yahoo.com/boss/search/`

divididos, estos recursos se puntúan en función de las apariciones de las palabras clave de la búsqueda en los literales y también, en la propia URI del recurso.

- **Consolidación:** una vez extraído el conocimiento, el siguiente paso consiste en agrupar todo el conocimiento de las diferentes fuentes de datos relacionado con un mismo recurso. Además, se formatean ciertas propiedades para que sean más legibles para el usuario.

- **Refinamiento de los resultados:** una vez que el resultado es presentado al usuario, éste puede eliminar, validar o añadir fuentes de datos relacionadas con la búsqueda realizada.

2.1.11 Análisis crítico

Como ha podido observarse, el objetivo de los trabajos presentados bajo esta clasificación consiste en, a partir de un término o conjunto de términos determinados, encontrar ontologías o instancias de las mismas en las que aparezca dicho término, o exista algún tipo de relación. Además de la búsqueda directa de los términos, algunos de estos trabajos desarrollan búsquedas más complejas. Buen ejemplo de ello son OntoSelect (sección 2.1.4) o Falcons (2.1.9), ya que permiten refinar la búsqueda a través del filtrado de los resultados. Otros trabajos, como OntoKhoj (2.1.1) o Swoogle (2.1.3), emplean algoritmos como PageRank (Page et al., 1998) o derivados para clasificar los resultados de búsqueda en función de su relevancia.

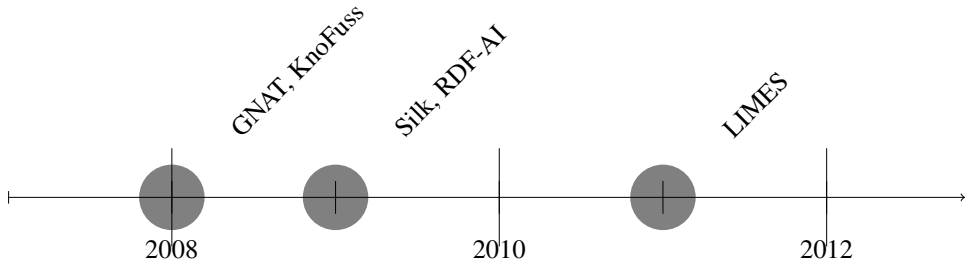
Independientemente de la complejidad de cada uno de los trabajos analizados en esta sección, todos comparten la característica de que para poder encontrar ontologías o instancias de ontologías es necesario introducir una serie de términos extraídos del conjunto de datos origen. La extracción de estos términos o la selección de los términos más relevantes para la correcta caracterización del conjunto de datos, son tareas que deben realizarse de manera independiente ya que ninguna de estas soluciones aporta las herramientas necesarias para su desarrollo.

Por otra parte, estos trabajos se centran en la búsqueda de instancias o clases concretas. Esto puede plantear la situación de que, efectivamente, una instancia

esté relacionada con el término de búsqueda empleado, pero que el resto del conjunto de datos del cual proviene dicha instancia tenga poca o ninguna relación con el conjunto de datos original. La solución planteada en esta tesis doctoral pretende ofrecer, entre otras cosas, una adecuada síntesis del conjunto de datos, con el objetivo de que esta síntesis sirva para la correcta búsqueda de posibles conjuntos de datos relacionados.

2.2 Enlazado de conjuntos de datos estructurados

En esta sección se muestran los trabajos relacionados con el enlazado de conjuntos de datos estructurados. El objetivo de estos trabajos consiste en, dados dos conjuntos de datos estructurados, intentar establecer relaciones entre los recursos de ambos conjuntos en base a unas características establecidas por el usuario. En el siguiente diagrama se representan los trabajos analizados:



2.2.1 GNAT

GNAT (Raimond et al., 2008) es una herramienta para el enlazado de instancias de diferentes conjuntos de datos, orientado hacia el dominio de la música. GNAT basa su funcionamiento en la comparación de literales y en calcular la similitud entre los grafos que forman dichas instancias. Para ello, a partir de dos tripletas de diferentes conjuntos que compartan el mismo predicado, calcula la distancia entre los objetos literales de ambas tripletas utilizando los métodos descritos por Winkler (1994, sección 2). Una vez calculada la distancia entre todos los literales de dos recursos diferentes, GNAT representa una serie de emparejamientos para cada uno de los dos grafos, en base a la distancia entre cada uno de estos literales. Estos emparejamientos son puntuados para elegir el más adecuado. Esta puntuación

se realiza sumando las similitudes entre los literales emparejados y dividiendo esta suma entre el número de emparejamientos de literales realizados. Si la puntuación total supera un determinado umbral, se podrá decir que ambos recursos representan el mismo objeto.

2.2.2 KnoFuss

KnoFuss (Nikolov et al., 2008) es una arquitectura cuyo objetivo consiste en la integración de instancias de una misma ontología OWL. Sus tres principales tareas consisten en identificar recursos idénticos, detectar violaciones de las restricciones de la ontología dentro de cada recurso y resolver inconsistencias. En este estado del arte se analiza la primera de las tareas, la relacionada con la identificación de recursos idénticos.

Para la identificación de recursos idénticos, KnoFuss cuenta con una "librería de métodos de resolución de problemas", que a su vez, se encuentra descrita a través de una "ontología de fusión". En esta librería, como su propio nombre indica, se encuentran los diferentes métodos disponibles para solucionar tareas atómicas, como puede ser el cálculo de la distancia entre dos literales a través de la distancia de Jaro-Winkler. Al estar descrita a través de una ontología desarrollada para tal efecto, a través de sentencias SPARQL se puede obtener el mejor método para realizar una tarea en concreto, teniendo en cuenta las entradas, salidas y diferentes parámetros del método.

Para realizar el ajuste de estos parámetros, KnoFuss propone el empleo de técnicas de *machine learning*. Su enfoque consiste en preparar un juego de ensayo con emparejamientos de diferentes instancias. Ya que entrenar todas las clases de la ontología supondría la creación de un número excesivo de ejemplos, KnoFuss propone emplear los mismos ejemplos para todas las subclases que pertenezcan a una determinada superclase y emplear los ejemplos de una determinada subclase para entrenar todas sus superclases. De esta manera, se reduciría el juego de ensayo notablemente.

Una vez seleccionado el método y sus parámetros, se ejecuta para conseguir el emparejamiento de instancias.

2.2.3 Silk

Silk (Volz et al., 2009b) define un lenguaje de especificación de enlaces denominado *Silk - Link Specification Language* (Silk-LSL) que permite al usuario determinar todos los parámetros necesarios para encontrar enlaces entre dos conjuntos de datos. En primer lugar, el usuario debe especificar cuales serán los dos conjuntos a enlazar y el *endpoint* SPARQL de cada uno de ellos. En segundo lugar, se debe especificar cuál será la relación entre estos dos conjuntos de datos (por ejemplo, `owl:sameAs`) y las clases a las que pertenecen las instancias que se desean enlazar (por ejemplo, si se desean enlazar las ciudades que aparecen en DBpedia con las que aparecen en Geonames[1], habrá que especificar las diferentes clases a las que pertenecen las instancias que representan ciudades tanto en un conjunto como en el otro). Una vez seleccionadas las clases cuyas instancias se van a intentar enlazar, hay que determinar cómo se va a establecer la relación entre ellas. Para realizar esta tarea, Silk ofrece una serie de algoritmos de cálculo de proximidad para aplicar a las diferentes propiedades de cada clase. Un ejemplo sencillo sería, por ejemplo, calcular la distancia existente entre los literales que sirven de objeto para la propiedad `rdfs:label` de cada una de las instancias. O, siguiendo con el ejemplo de las ciudades, determinar qué dos instancias se refieren a la misma ubicación si las coordenadas de las mismas se encuentran en un radio determinado. También podría darse el caso en el que el usuario quisiera emplear dos métricas diferentes. Esto es posible ya que Silk permite agregar el resultado de las diferentes métricas, y seleccionar el más adecuado en base a diferentes funciones como el valor máximo, la media aritmética, etc.

Por último, Silk permite generar un fichero de salida con las nuevas tripletas creadas, además de permitir el establecimiento de un umbral para descartar enlaces cuya fortaleza no cumpla unos requisitos mínimos.

A lo largo del tiempo y gracias al trabajo de investigación de sus creadores, Silk ha ido incrementando su potencial a la hora de hallar enlaces. Algunas de estas mejoras son, por ejemplo, la capacidad para refinar los enlaces hallados a través de un pequeño conjunto de enlaces creados a mano (Volz et al., 2009a) y la programación genética (Isele y Bizer, 2012) o desplegar Silk en un clúster de

[1] `http://www.geonames.org/`

servidores a través de Hadoop para incrementar su rendimiento (Jentzsch et al., 2010).

2.2.4 RDF-AI

RDF-AI (Scharffe et al., 2009) presenta una arquitectura para la integración de conjuntos de datos enlazados, generando un único y nuevo conjunto a partir de dos conjuntos de datos. RDF-AI cuenta con los siguientes cinco módulos:

- ***Preprocessing:*** el módulo de pre-procesamiento realiza una serie de tareas previas a la integración de los conjuntos de datos. Estas tareas consisten en **comprobar** si los conjuntos de datos de entrada son consistentes con sus respectivas ontologías, **materializar** las propiedades inversas o transitivas, **traducir** las propiedades a un mismo lenguaje, adaptar los conjuntos de datos en el caso de que alguna de las dos ontologías sea una **evolución** de la otra y **transformar** ciertas propiedades a un mismo formato.
- ***Matching:*** este módulo se encarga de realizar el emparejado entre las ontologías de ambos conjuntos de datos. Para ello, se emplean las técnicas desarrolladas en Euzenat y Shvaiko (2007).
- ***Interlinking:*** el módulo de enlazado toma como entrada un alineamiento entre dos conjuntos de datos y genera una serie de enlaces `owl:sameAs`.
- ***Fusion:*** por su parte, el módulo de fusión genera un nuevo conjunto de datos a partir de la fusión de los recursos alineados de los conjuntos de datos de entrada.
- ***Post processing:*** por último, el módulo de post-procesado comprueba si se han generado inconsistencias durante el proceso de fusión de ambos conjuntos de datos.

2.2.5 LIMES

LIMES (Link Discovery Framework for Metric Spaces) (Ngomo y Auer, 2011) optimiza el enlazado entre conjuntos de datos a través de la teoría de espacios

métricos. LIMES permite optimizar las distancias empleadas para la búsqueda de similitudes entre recursos de diferentes conjuntos de datos a través del filtrado previo de los recursos. A la hora de buscar enlaces entre el conjunto origen S y el conjunto destino T, LIMES calcula una serie de "ejemplares". Estos ejemplares son una serie de recursos aleatorios extraídos de T, cuyo objetivo consiste en representar T de la manera más homogénea posible. Una vez extraído el subconjunto de ejemplares E, se calcula la distancia de cada uno de ellos a cada recurso de T, asociando cada recurso al ejemplar más cercano o similar. A continuación se calcula la distancia entre los recursos de S y los ejemplares, asociando cada recurso a al ejemplar más cercano. De esta manera se filtran los recursos de T que no están asociados a ese ejemplar, por lo que las comparaciones a realizar entre S y T son menores.

En la figura 2.1 puede observarse la división del espacio métrico conformado por los recursos de T (cuadrados, triángulos y círculos punteados) y los ejemplares (círculos azules con borde negro). Como puede observarse, el espacio métrico queda dividido de manera que cada uno de los recursos de T esté asociado a uno de los ejemplares. A la hora de buscar recursos similares, los recursos de S se asocian al ejemplar más similar, por lo que en este caso se descartarían los recursos asociados a los otros dos ejemplares, reduciendo drásticamente el número de veces a aplicar la técnica de similitud seleccionada.

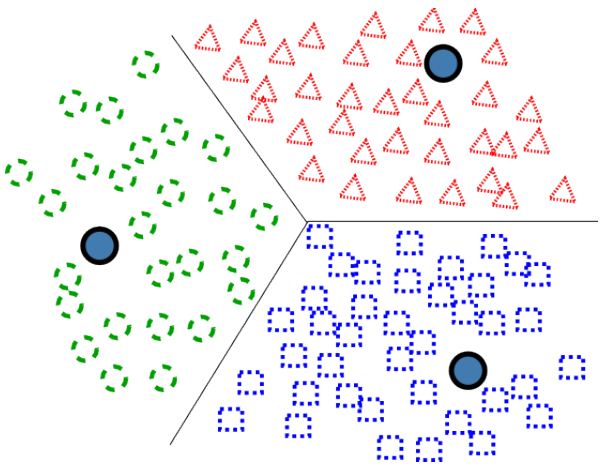


Figura 2.1: Representación de un espacio métrico.

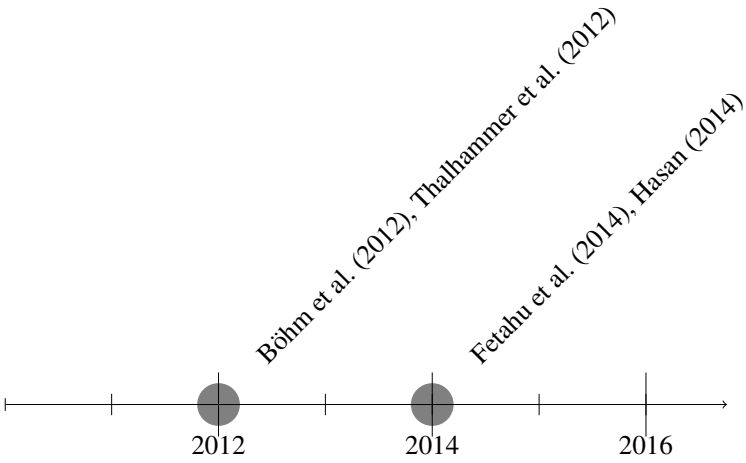
2.2.6 Análisis crítico

Bajo esta categoría se han clasificado aquellos trabajos cuyo objetivo consiste en, dada una pareja de conjuntos de datos estructurados, establecer enlaces entre las instancias de cada uno de ellos. La mayoría de estos trabajos emplean los literales de los conjuntos de datos para encontrar similitudes entre las instancias de ambos conjuntos de datos y establecer enlaces del tipo *owl:sameAs* entre ellos. En el caso de GNAT (sección 2.2.1), se calculan las distancias entre los literales de los conjuntos de datos, determinando, así, si los grafos formados por unas determinadas instancias son equivalentes. Por su parte, SILK (2.2.3) permite aplicar múltiples operaciones y métricas sobre diferentes elementos de los conjuntos de datos, posteriormente agregando estas métricas con el objetivo de encontrar instancias equivalentes en conjuntos de datos similares. Otros, como RDF-AI (2.2.4) que aplica técnicas de *ontology matching* para emparejar clases de diferentes conjuntos de datos, añaden una mayor complejidad al proceso de pareado de instancias.

Desde el punto de vista planteado por esta tesis doctoral, la mayor carencia de este tipo de soluciones consiste en que es el usuario el encargado de determinar cuáles son los conjuntos de datos a enlazar. La principal contribución de esta tesis doctoral consiste, precisamente, en dados un conjunto de datos origen y una colección de conjuntos de datos estructurados, determinar una serie de candidatos a ser enlazados, liberando al usuario de realizar esta tarea. A continuación, estos conjuntos candidatos a ser enlazados podrían servir como entrada a los diferentes trabajos analizados durante esta sección. Puede decirse que el sistema desarrollado para demostrar la hipótesis planteada en esta tesis doctoral podría ser el paso previo a los trabajos presentados en este apartado.

2.3 Síntesis de conjuntos de datos estructurados

En esta sección se citan los trabajos más relevantes dentro del ámbito de la síntesis de conjuntos de datos enlazados. Como puede observarse en el siguiente diagrama, las recientes fechas de publicación de los trabajos analizados sugieren que es un ámbito "joven" en el cual hay todavía mucho terreno inexplorado.



2.3.1 Böhm et al. (2012)

El trabajo de Böhm et al. presenta un enfoque para extraer temas de los datos estructurados en forma de grafo basándose solamente en la estructura que dichos grafos forman, rompiendo, así, la dependencia de las etiquetas textuales. Para ello, se plantean dos alternativas:

- **Patrones de *motifs* conceptuales:** se construye un grafo en el que aparecen tanto las instancias de los recursos como sus clases correspondientes con el objetivo de buscar *motifs*, subgrafos o patrones que se repiten a lo largo de un grafo.

- **Patrones de información mutua:** ya que la creación de los patrones de *motifs* conceptuales demanda un coste computacional elevado, mediante esta alternativa se generan "grafos abstractos". Dichos grafos disponen las clases de las instancias en sus vértices y las relaciones existentes entre estas instancias en sus aristas. De esta manera, las URI de las instancias no aparecen en el grafo y se reduce el número de vértices y aristas.

Una vez extraídos estos patrones, se extraen las etiquetas de las clases y/o instancias. Estas etiquetas definen los temas que describen cada conjunto de datos formando, así, el perfil del conjunto de datos.

2.3.2 Thalhammer et al. (2012)

Thalhammer et al. (2012) plantean una solución para sintetizar las instancias de los conjuntos de datos enlazados del ámbito del cine. En concreto emplean instancias extraídas del conjunto MovieLens2k[1] y de Freebase[2]. Su hipótesis se basa en que las características (pares objeto-valor) compartidas por los K-vecinos más cercanos de una instancia son más representativas para describir un determinado tipo de instancia que las características compartidas con otras instancias que no se encuentran entre los K-vecinos más cercanos.

Para calcular los K-vecinos más cercanos de una instancia, en primer lugar crean la matriz usuario-ítem a partir de las puntuaciones otorgadas por los usuarios de MovieLens[3] a las películas. A partir de esta matriz calculan los K-vecinos más cercanos en base a los coeficientes de probabilidad (*log likelihood-ratio score*) (Dunning, 1993). Una vez seleccionados 20 vecinos más cercanos, se extraen las características de cada una de estas instancias y se crea un *ranking* con las más frecuentes. Además, se extraen todas las características de todas las instancias del conjunto de datos y se añaden al *ranking*. Para refinar el resultado, se aplica la técnica TF-IDF *term frequency - inverse document frequency*, popular dentro de los sistemas de recuperación de información. Esta técnica permite eliminar del *ranking* las características que, a pesar de ser populares a lo largo de todas las instancias, no lo son tanto entre los K-vecinos más cercanos. Por último, se seleccionan las n características más relevantes según el *ranking* establecido para crear el resumen o síntesis de las instancias.

2.3.3 Fetahu et al. (2014)

El trabajo desarrollado por Fetahu et al. (2014) tiene como objetivo la creación de perfiles de conjuntos de datos enlazados. Un perfil de un conjunto de datos consiste en metadatos que describen los temas (o *topics*) relacionados con dicho conjunto. Dichos *topics* son categorías de la DBpedia que tienen relación con el conjunto de datos en cuestión. Para asignar estos temas a los conjuntos de datos, los autores

[1] `http://grouplens.org/datasets/movielens/`
[2] `http://www.freebase.com/`
[3] `https://movielens.org/`

extraen los literales de los recursos y aplican técnicas de NER (*Named Entity Recognition*) sobre ellos, con el objetivo de encontrar las entidades correspondientes en la DBpedia. Una vez extraídas estas entidades, se construye el llamado "grafo de perfil". Dicho grafo dispone los temas y los conjuntos de datos en sus vértices, siendo las aristas del grafo las relaciones entre los conjuntos de datos y los temas. Las aristas del grafo están ponderadas de manera que los temas que mayor relación tengan con un conjunto de datos mayor puntuación tendrán. Para realizar esta ponderación, los autores han planteado tres alternativas:

- ***Normalised Topic Relevance (NTR)*:** para el cálculo de este modelo, se toman en cuenta *i*) el número de entidades $\Phi(t, D)$ asignadas a un tema t dentro del conjunto de datos D y el número de entidades $\Phi(t, \cdot)$ asignadas a ese mismo tema a lo largo de todos los conjuntos de datos y *ii*) el número de entidades $\Phi(\cdot, D)$ asignadas a un conjunto de datos y el número total de entidades $\Phi(\cdot, \cdot)$ a lo largo de todos los conjuntos de datos. Los temas se filtran si su ponderación, calculada de la siguiente manera, no supera un determinado umbral:

$$NTR(t, D) = \frac{\Phi(\cdot, D)}{\Phi(t, D)} + \frac{\Phi(\cdot, \cdot)}{\Phi(t, \cdot)}, \forall t \in \mathbf{T}, D \in \mathbf{D} \tag{2.5}$$

- ***PageRank with Priors*:** es una variación del algoritmo PageRank (Page et al., 1998) que dado el grafo de perfil calcula los pesos de las aristas en función de la conectividad de los vértices. Para el recurso analizado $r \in R_k$ se le asigna la probabilidad inicial $1/|R_k|$, mientras que para el resto de vértices se les asigna una probabilidad de cero.

$$\pi(t)^{(i+1)} = (1 - \beta) \left(\sum_{u=1}^{d_{in}(t)} p(t|u)\pi^{(i)}(u) \right) + \beta p_t \tag{2.6}$$

- ***HITS with Priors*:** al igual que en el caso anterior, presenta una variación sobre el algoritmo HITS (Kleinberg, 1999). En este caso, el camino a seguir se basa en un valor binario generado aleatoriamente: si el valor es cero, si el paso actual es par se caminará por un enlace interno del recurso, si es impar

se seguirá un enlace hacia un vértice externo al recurso. Si el valor no es cero, se visitará uno de los vértices del recurso R_k. Ya que no hay manera de diferenciar entre temas $t \in \mathbf{T}$ que actúen como *hubs* o vértices *acreditados*, el proceso se simplifica ignorando el tipo de vértice.

$$a^{(i+1)}(t) = (1-\beta)\left(\sum_{u=1}^{d_{in}(t)} \frac{h^{(t)(u)}}{\sum\limits_{t\in\mathbf{T}}\sum\limits_{u=1}^{d_{in}(t)} h^{(i)}(t)}\right) \tag{2.7}$$

- ***K-Step Markov*:** los enfoques anteriores representan cadenas de Markov en las que el número de pasos a ejecutar es estocástico. A través del Markov de K-pasos, el itinerario comienza en uno de los temas $t \in \mathbf{T}$ y se detiene despues de K pasos. La principal ventaja de este enfoque es su escalabilidad frente a conjuntos de datos grandes.

2.3.4 Hasan (2014)

En su trabajo, Hasan presenta una solución para el problema del exceso de información a la hora de explicar los razonamientos realizados sobre conjuntos de datos en RDF. Para ello, a partir de la sentencia RDF que se quiere explicar, se extraen nuevas sentencias, navegando hasta una profundidad predefinida. Esta sentencia original se denomina "sentencia raíz" (*root statement* o *rs*). El conjunto de sentencias RDF a partir de las cuales es inferida *rs* se denominan "sentencias de conocimiento" (*knowledge statements* o *KST*). Se denomina grafo de conocimiento (*knowledge graph* o *KG*) al grafo formado por la unión de *rs* y *KST*: $KG = RDFGraph(KST \cup rs)$.

Para seleccionar qué sentencias de *KST* deben ser añadidas al resumen de la explicación, se elabora un ranking. Para la elaboración de este ranking, se emplean tres medidas diferentes para calcular la relevancia de cada sentencia RDF:

- **Sentencias RDF relevantes:** la relevancia de una sentencia RDF muestra la importancia de dicha sentencia dentro del grafo. Para calcular la relevancia $S_{SL}(i)$ de una sentencia RDF (i) se utiliza la "centralidad de grado normalizada" $C_{DN}(v)$:

$$S_{SL}(i) = \theta_1 \times C_{DN}(subjectOf(i)) + \theta_2 \times C_{DN}(objectOf(i)) \quad (2.8)$$

siendo los pesos $\sum_i \theta_i = 1$ y $\forall_i : \theta_i \geq 0$. Las funciones $subjectOf(i)$ y $objectOf(i)$ devuelven, respectivamente, el recurso sujeto y el recurso objecto de la sentencia RDF i.

- **Sentencias RDF similares:** los usuarios de la aplicación pueden especificar una serie de clases FL como criterio de filtrado a la hora de elaborar el ranking, donde $FL \subseteq SC$ y SC es el conjunto de todas las clases que se emplean para describir KG. A través de este método se extraen las sentencias más similares de acuerdo al criterio de filtrado establecido por los usuarios. Para calcular esta similitud, emplean la característica de resolución de consultas similares de Corese[1] (Corby et al., 2006). Para una sentencia i y un conjunto de clases como criterio de filtrado FL, se calcula la similitud $S_{SM}(i, FL)$:

$$\begin{aligned} S_{SM}(i, FL) = \theta_1 &\times similarity_{node}(subjectOf(i), FL) \\ +\theta_2 &\times similarity_{node}(predicateOf(i), FL) \\ +\theta_3 &\times similarity_{node}(objectOf(i), FL) \end{aligned} \quad (2.9)$$

La función $predicateOf(i)$ devuelve el predicado de la sentencia i. La función $similarity_{node}(j, FL)$ se calcula de la siguiente manera:

$$similarity_{node}(j, FL) = \begin{cases} similarity_{type}(\{j\}, FL) & \text{if } j \in SC \\ similarity_{type}(typesOf(j), FL) & \text{if } j \notin SC \end{cases} \quad (2.10)$$

Por último, la función $similarity_{type}$ se calcula de la siguiente manera:

$$similarity_{type}(TP, FL) = \frac{\sum_{m \in FL} maxSimilarity_{type}(m, TP)}{|FL|} \quad (2.11)$$

[1] http://wimmics.inria.fr/corese

$$similarity_{type}(m, TP) = \max_{n \in TP}(similarity_{corese}(m, n)) \tag{2.12}$$

- **Sentencias abstractas:** el autor considera que una sentencia cercana a rs dentro del árbol de evidencias es más abstracta que una sentencia lejana. El nivel de abstracción $S_{AB}(i$ de una sentencia $i \in KST$ se calcula de la siguiente manera:

$$S_{AB}(i) = \frac{1}{level(i)} \tag{2.13}$$

Una vez calculada la puntuación de cada una de las sentencias, se lanza un proceso de re-ranking, con el objetivo de mejorar la clasificación original. Para llevar a cabo este proceso, el autor plantea dos alternativas:

- **Peso de la rama en el árbol de evidencias:** para una rama del árbol de evidencias cuya raíz es la sentencia i, se calcula el peso de dicha rama a través del cálculo de la puntuación media de todas las sentencias de dicha rama:

$$score_{ST}(i) = \frac{\sum_{j \in subtree(i)} score(j)}{|subtree(i)|} \tag{2.14}$$

  La función $score(j)$ representa a una de las funciones explicadas anteriormente.

- **Coherencia:** el autor considera que una sentencia x es coherente con una sentencia y si x deriva directamente de y. Dada una lista ordenada (ranking) RL; el conjunto S de sentencias seleccionadas para formar parte del ranking; y la próxima sentencia a ser elegida en S, i, se reordena RL seleccionando la siguiente i ejecutando un número de $|RL|$ veces la siguiente función:

$$i = \underset{j \in RL}{\operatorname{argmax}}(\lambda_1 \times score(j) + \lambda_2 \times reward(j, S)) \tag{2.15}$$

tal que $\sum_i \lambda_i = 1$ y $\forall_i \geq 0$. Al igual que en la alternativa anterior, $score(j)$ representa a una de las funciones de similitud explicadas previamente, mientras $reward(j, S)$ se calcula de la siguiente manera:

$$reward(j, S) = 1 - \frac{coherent(S)}{coherent(S \cup j)} \tag{2.16}$$

Como puede observarse, la función $reward$ calcula la proporción entre el número de sentencias coherentes de S y el número de sentencias coherentes si j se añadiese a S.

2.3.5 Análisis crítico

Bajo este apartado se han descrito los trabajos cuyo objetivo consiste en sintetizar o resumir conjuntos de datos estructurados, facilitando, así la comprensión de los mismos por parte de los usuarios. Dado que el primer paso a la hora de encontrar conjuntos de datos candidatos a ser enlazados es la síntesis de los mismos, se han incluido los trabajos más representativos de esta área.

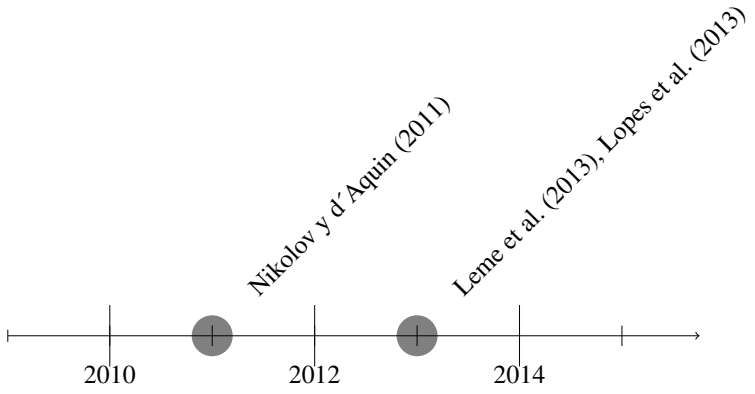
Dentro de esta área existen trabajos como Böhm et al. (2012) o Fetahu et al. (2014), cuyo objetivo consiste en la extracción de una serie de temas (o *topics*) que describan los conjuntos de datos estructurados. En el caso de Böhm et al. (2012), esto se consigue a través del análisis de la estructura de los grafos que forman los conjuntos de datos estructurados. Una vez encontrados los patrones más comunes del grafo, se extraen las etiquetas de las clases e instancias que forman estos patrones, formando a través de estas etiquetas la descripción del grafo. Por su parte, Fetahu et al. (2014) aplica técnicas de NER (*Named Entity Recognition*) sobre los literales de los conjuntos de datos con el objetivo de encontrar las entidades correspondientes en la DBpedia. Una vez halladas dichas entidades se extraen sus categorías que, una vez ponderadas, formarán parte del resumen del conjunto de datos. Desde otro punto de vista, el objetivo de Hasan (2014) consiste en explicar razonamientos semánticos de manera resumida.

A pesar de que, como se expone en el capítulo 3, en esta tesis doctoral se ha desarrollado un mecanismo de síntesis de conjuntos de datos estructurados propio, dicho mecanismo se basa en ideas extraídas a partir de alguno de estos trabajos.

El desarrollo de este mecanismo propio responde a la necesidad de disponer de las subestructuras más frecuentes de cada grafo, no solamente de los temas extraídos por estos trabajos.

2.4 Identificación de fuentes para el enlazado de conjuntos de datos estructurados

En esta sección se presentan los trabajos que tienen mayor relación con la temática de esta tesis doctoral. En estos trabajos se exponen diferentes técnicas que, dado un conjunto de datos estructurados intentan encontrar candidatos a ser enlazados con dicho conjunto en un repositorio de conjuntos de datos enlazados. En el siguiente diagrama se representan los trabajos analizados.



2.4.1 Nikolov y d´Aquin (2011)

Nikolov y d´Aquin proponen una solución para que, dado un conjunto de datos D_s, y un repositorio de conjuntos de datos $\{D_1, \ldots D_n\}$, encontrar qué conjuntos de datos podrían ser enlazados con el conjunto de datos fuente. Para ello, tratan de encontrar un subconjunto de instancias del conjunto de datos origen en el conjunto de datos destino. Esta búsqueda se realiza a través de la introducción en Sig.ma (sección 2.1.10) de los literales hallados en etiquetas como `rdfs:label`, `foaf:name` o `dc:title`. El proceso consta de los siguientes pasos:

1. Para reducir el número de consultas a realizar, se selecciona un conjunto finito de instancias al azar de D_s (conjunto de datos origen).

2. Se realiza la búsqueda de las etiquetas previamente mencionadas de estas instancias en Sig.ma.

3. Se agrupan los resultados devueltos por Sig.ma en función de la fuente de datos de cada uno.

4. Las fuentes de datos se clasifican en función del número de instancias encontradas en cada una de ellas.

Esta clasificación no es suficiente, ya que como explican los autores, no evita la inclusión de resultados irrelevantes. Para filtrar estos resultados, emplean técnicas de emparejamiento de ontologías (*ontology matching*) con el objetivo de determinar si las clases de las instancias potencialmente similares mantienen alguna relación entre ellas. Además, utilizan las relaciones `owl:sameAs` del conjunto de datos publicado en la *Billion Triple Challenge 2009*[1] para corroborar dichas relaciones.

En Nikolov et al. (2011), los autores presentaron una extensión del trabajo previo, aplicando una solución para encontrar las clases más relevantes de cada conjunto de datos, con el objetivo de refinar los resultados.

2.4.2 Leme et al. (2013)

Los autores de Leme et al. (2013) proponen una técnica que se basa en la utilización de clasificadores probabilísticos para encontrar conjuntos de datos candidatos a ser enlazados. Concretamente, los autores emplean un clasificador bayesiano ingenuo (*naive Bayes*) para calcular la probabilidad de que existan enlaces desde los conjuntos de datos del repositorio T hacia el conjunto de origen S:

$$score(T_i, S) = \left(\sum_{j=1..n} log(P(f_j|T_i)) \right) + log(P(T_i)) \tag{2.17}$$

siendo f cada una de las características de S (en este caso, los enlaces existentes en S) y pudiendo calcular $P(f_j|T_i)$ y $P(T_i)$ como:

$$P(f_j|T_i) = \frac{count(f_j, T_i)}{\sum_{j=1}^{n} count(f_j, T_i)} \tag{2.18}$$

[1] `http://vmlion25.deri.ie/`

$$P(T_i) = \frac{count(T_i)}{\sum_{i=1}^{m} count(T_i)} \tag{2.19}$$

donde $count(f_j, T_i)$ es el número de conjuntos de datos del conjunto de entrenamiento que contienen la característica f_j que están conectados al conjunto T_i; y $count(T_i)$ el número de conjuntos de datos enlazados con T_i, excluyendo el conjunto de características.

Para evaluar esta técnica, los autores toman como evidencia la correlación entre los diferentes conjuntos de datos. Por ejemplo, si un conjunto A está relacionado con los conjuntos $\{B, C\}$ con un alto grado de correlación y a su vez, estos conjuntos $\{B, C\}$ están relacionados con los conjuntos $\{D, E, F, G\}$ que no están relacionados con A, se sugiere que A es relevante para los conjuntos $\{D, E, F, G\}$. Se remarca este aspecto ya esta evidencia es una de las bases para elaborar el "gold standard" empleado durante la evaluación del trabajo desarrollado en esta tesis doctoral.

2.4.3 Lopes et al. (2013)

En Lopes et al. (2013), sus autores emplean estrategias propias de las redes sociales para recomendar conjuntos de datos a enlazar. Para ello definen el término "contexto" como el conjunto de enlaces que un conjunto de datos ya dispone. Cuanto más similares sean los contextos de dos conjuntos de datos, mayor probabilidad de que puedan ser enlazados entre ellos. Para calcular esta similitud emplean tanto el coeficiente de Jaccard (ecuación 2.20) como el coeficiente de Adamic-Adar (ecuación 2.21).

$$jc(t, u) = \frac{|C_t \cap C_u|}{|C_t \cup C_u|} \tag{2.20}$$

$$aa(t, u) = \sum_{w \in C_t \cap C_u} \frac{1}{log|C'_w|} \tag{2.21}$$

Estas ecuaciones se explican de la siguiente manera:

- Una red de Linked Data se define como un grafo $G = (S, C)$ donde S es un repositorio de conjuntos de datos enlazados y C el conjunto de aristas (t, u) que definen la conectividad entre dos conjuntos de datos $t \in S$ y $u \in S$.
- Se identifica como contexto de un conjunto de datos, el conjunto de las relaciones ya existentes para un conjunto de datos. Siendo u un conjunto de datos enlazados perteneciente a S, se define su contexto como C_u, siendo este contexto el conjunto de todo $v \in S$ de manera que $(u, v) \in C$. La inversa del contexto de u se define como C'_u, siendo el conjunto de todo $v \in S$ de manera que $(v, u) \in C$.
- t define el conjunto de datos a enlazar, entendiendo que $t \notin S$.
- $|C_t \cap C_u|$ es la cardinalidad de la intersección de los contextos de t y u.
- $|C_t \cup C_u|$ es la cardinalidad de la unión de los contextos de t y u.
- $|C'_w|$ es la cardinalidad de la inversa del contexto de w (la popularidad de w).

Una vez calculado el *ranking*, dejan en mano del usuario seleccionar los conjuntos de datos pertinentes de la lista.

2.4.4 Análisis crítico

En esta sección se han descrito los trabajos que comparten finalidad con el trabajo desarrollado en esta tesis doctoral: dado un conjunto de datos y una colección de conjuntos de datos estructurados, encontrar conjuntos de datos candidatos a ser enlazados con el conjunto de datos origen. El primero de estos trabajos, Nikolov y d´Aquin (2011), extrae los literales de las tripletas con el predicado *rdfs:label*, *foaf:name* o *dc:title*. Se realiza una búsqueda en Sig.ma a partir de estos literales agrupando los resultados en función de su conjunto de datos origen. En función del número de instancias de cada recurso, este tendrá más posibilidades de ser enlazado con el conjunto de datos origen.

Por otra parte, Leme et al. (2013) emplean un clasificador bayesiano ingenuo para calcular la probabilidad de que dos conjuntos de datos sean enlazados. Una vez calculadas estas probabilidades, se ordenan los conjuntos de datos en un *ranking*.

Según los autores, siguiendo este *ranking* es posible reducir el espacio de búsqueda hasta en un 85 %.

El tercero de estos trabajos (Lopes et al., 2013) se basa en la intuición de que si dos conjuntos de datos están enlazados con conjuntos en común, existe la posibilidad de que ellos también sean enlazados. Para ello, plantean un sistema que, dado el contexto de cada uno de los conjuntos de datos, determinan si podrían ser enlazados entre si. Dicho contexto está formado por las relaciones ya existentes de cada uno de los conjuntos de datos. Para calcular la probabilidad de establecer enlaces entre dos conjuntos de datos emplean tanto el coeficiente de Jaccard como el coeficiente de Adamic-Adar. Los autores afirman conseguir una exhaustividad comprendida entre el 75 % y el 90 %.

A pesar de que los objetivos de estos trabajos son similares a los del trabajo desarrollado durante esta tesis doctoral, cabe remarcar una serie de debilidades detectadas durante el estudio de los mismos. En primer lugar, el trabajo realizado por Nikolov y d´Aquin basa su funcionamiento en Sig.ma (a su vez basado en Sindice), servicios que han dejado de ser soportados y que han dejado de recopilar nueva información[1]. Además, tal y como afirman Böhm et al., conviene evitar el uso de etiquetas textuales, ya que son un bien escaso en los conjuntos de datos predominantes.

Por otra parte, en Leme et al. (2013) solamente se evalúa la reducción realizada sobre el espacio de búsqueda. Para calcular esta reducción se toma en cuenta el último resultado válido del *ranking* generado, pero no se contempla que en posiciones superiores a este último resultado puedan existir resultados irrelevantes o erróneos. La inclusión de resultados erróneos puede implicar que, a pesar de haberse reducido el espacio de búsqueda, el usuario malgaste tiempo intentando enlazar el conjunto de datos origen con uno de estos resultados.

Por último, a pesar de que los autores de Lopes et al. (2013) consigan unos muy buenos resultados, el hecho de necesitar el contexto de los conjuntos de datos hace que esta solución no pueda hacer frente a un problema de arranque en frío. Es decir, sería incapaz de funcionar si ninguno de los conjuntos de datos conocidos tuviese enlaces ya establecidos, problemática que podría ser extendida a las otras

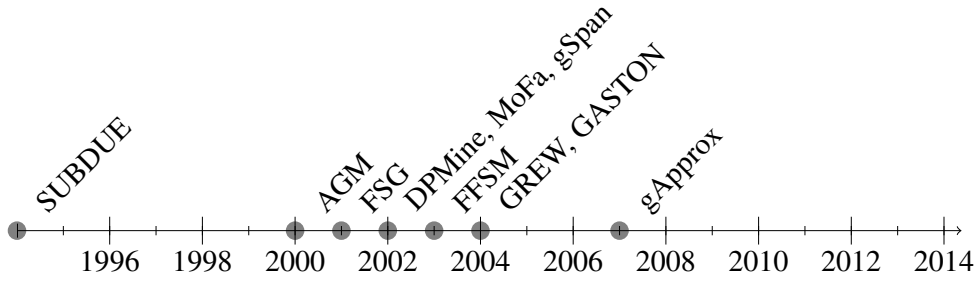
[1] `http://bit.ly/1s6Ujk0`

dos soluciones analizadas en esta sección. El trabajo desarrollado durante esta tesis doctoral pretende abordar esta problemática, entre otras.

2.5 Extracción de subgrafos más frecuentes

Tal y como se puede observar en la sección 1.3, la hipótesis planteada en esta tesis doctoral emplea el análisis de la estructura de los grafos formados por los conjuntos de datos enlazados. Concretamente, se realiza la extracción de los subgrafos más frecuentes dentro de cada grafo, con el objetivo de conseguir una muestra sintetizada y manejable de los grandes conjuntos de datos.

Según Jiang et al., existen dos variantes diferentes del problema de extracción de subgrafos más frecuentes o *frequent subgraph mining (FSM)*: una de ellas basada en transacciones (colección de grafos de mediano tamaño denominados "transacción") y otra basada en grafos individuales (un único grafo de gran tamaño). Dadas las características de los grafos formados por los conjuntos de datos de *Linked Data*, a lo largo de esta sección se analizan los trabajos más relevantes en FSM, tal y como puede observarse en el siguiente cronograma:



Ya que el objetivo de esta sección consiste en realizar una breve introducción para justificar la elección de algoritmos de búsqueda de subgrafos más frecuentes, solamente se describen las características necesarias para poder diferenciar entre un algoritmo y otro. Podrán encontrarse más detalles sobre los mismos en las publicaciones citadas a lo largo de esta sección.

2.5.1 SUBDUE

SUBDUE (Holder et al., 1994) es un sistema que, basado en el principio de la longitud mínima de descripción o *minimum description length (MDL)*, descubre subestructuras cuyo reemplazo comprime el grafo original al máximo. El principio MDL plantea que la mejor teoría para describir un conjunto de datos es aquella que minimiza la longitud de la descripción de la totalidad del conjunto de datos (Rissanen, 1978). De acuerdo con este principio, el objectivo de SUBDUE consiste en minimizar $I(S) + I(G|S)$, siendo S la nueva subestructura descubierta, G el grafo de entrada, $I(S)$ el número de bits necesarios para codificar la subestructura e $I(G|S)$ el número de bits necesarios para codificar el grafo de entrada sustituyendo la subestructura S. Otra de las características de SUBDUE es la capacidad de realizar búsquedas inexactas: con el objetivo de no omitir subestructuras que a pesar de no ser exactamente iguales tienen una gran relación, SUBDUE asigna un coste a cada una de las transformaciones necesarias para convertir una subestructura en otra, siendo estas transformaciones la eliminación, inserción o sustitución de aristas o vértices. El usuario puede determinar el peso de cada una de estas transformaciones, así como establecer un umbral de similitud entre las diferentes subestructuras.

El formato de entrada de SUBDUE consiste en un único multigrafo (dos vértices pueden estar conectados por varias aristas) dirigido y etiquetado (tanto los vértices como las aristas), mientras que el algoritmo de búsqueda empleado se clasifica dentro de los algoritmos de búsqueda en haz (Reddy, 1976). Este tipo de algoritmos, al igual que los algoritmos de *primero el mejor*, expanden la búsqueda al nodo que aparentemente mejores resultados va a dar, con la diferencia de que solamente mantienen en memoria un conjunto limitado de mejores soluciones. Siguiendo este algoritmo, SUBDUE comienza la búsqueda a partir de un único nodo, expandiendo el subgrafo en cada iteración, manteniendo una lista de las mejores n subestructuras. El algoritmo finaliza cuando se analizan todas las subestructuras posibles o se superan unos límites de computación definidos por el usuario.

2.5.2 AGM

El objetivo de AGM (*Apriori-based Graph Mining*) (Inokuchi et al., 2000) consiste en realizar búsquedas de subestructuras frecuentes dentro de grafos. Para ello, se apoya en la matriz de adyacencia –representación matemática de los grafos– y en la extensión del algoritmo del tipo "apriori" denominado *nivel prudente de búsqueda* o *levelwise search* (Agrawal et al., 1994). Ya que AGM admite como entrada grafos etiquetados, la matriz de adyacencia difiere levemente de la matriz tradicional empleada para la representación de grafos: siendo los elementos i, j de la matriz un par de vértices del grafo, se insertará un 0 en la matriz en el caso de que no exista ninguna arista entre dichos vértices, o un número positivo en el caso de que existiera, siendo este un número asignado aleatoriamente para cada una de las etiquetas diferentes de las aristas del grafo. A partir de estas matrices, AGM identifica los subgrafos frecuentes a través del análisis de afinidad.

2.5.3 FSG

Kuramochi y Karypis presentan FSG, un nuevo algoritmo para la detección de subgrafos frecuentes dentro de grandes conjuntos de grafos etiquetados y no dirigidos. Basado, al igual que AGM (sección 2.5.2), en algoritmos *apriori*, FSG cuenta entre sus contribuciones con *a*) la representación de los grafos de manera dispersa, minimizando el espacio de almacenamiento y el tiempo de computación necesario; *b*) la generación eficiente de los candidatos a subestructuras más frecuentes; *c*) el empleo de algoritmos simples de búsqueda de isomorfismo y etiquetado canónico de grafos para mayor eficiencia con grafos de menor tamaño; y *d*) la optimización de la generación de candidatos respecto a los trabajos posteriores, permitiendo mayor escalabilidad del sistema.

2.5.4 DPMine

DPMine (Vanetik et al., 2002) presenta un algoritmo para la detección de subgrafos más frecuentes dentro de un conjunto de grafos o transacciones. Al igual que AGM

y FSG (secciones 2.5.2 y 2.5.3 respectivamente) basa su funcionamiento en algoritmos *apriori*, siendo su mayor aportación la reducción de candidatos a formar la lista de subgrafos más frecuentes.

2.5.5 MoFA

Orientado hacia el dominio de la bioingenería, MoFA (*Molecular Fragment Miner*) (Borgelt y Berthold, 2002) se decanta por el algoritmo Eclat (Zaki et al., 1997) para la selección de subestructuras candidatas. Tomando como entrada una serie de moléculas representadas como grafos etiquetados no dirigidos, MoFA toma como punto de inicio un núcleo común a todas las moléculas a analizar. Una vez establecido este núcleo, MoFA añade un átomo a este núcleo, barajando todas las posibilidades que aparecen en las moléculas analizadas. A lo largo del proceso MoFA construye un árbol en el cual etiqueta en qué moléculas aparece cada subestructura. Una vez finalizado el proceso, se cuentan las apariciones de cada subestructura.

2.5.6 gSpan

La principal novedad de gSpan (Yan y Han, 2002) respecto a las soluciones analizadas hasta este momento, consiste en la omisión del paso de generación de candidatos. Para ello, gSpan combina en un único paso la fase de incremento de las subestructuras y la comprobación de la frecuencia de éstas, además de ser el primer algoritmo en aplicar la búsqueda en profundidad (DFS) (Cormen et al., 2001) en el campo de la búsqueda de subestructuras más frecuentes. Para la aplicación de este algoritmo DFS, gSpan presenta la técnica de *minimum DFS code* para codificar los grafos y el *DFS lexicographic order* para ordenar dichos códigos.

2.5.7 FFSM

El algoritmo FFSM (*Fast Frequent Subgraph Mining*) (Huan et al., 2003) incrementa la eficiencia respecto al algoritmo gSpan (sección 2.5.6) a través de la introducción de un nuevo método para la representación canónica del grafo. Este nuevo método consiste en representar el grafo a través de una matriz, con la diferencia

respecto a sus antecesores de que las etiquetas de los nodos se ubican en la diagonal de la matriz, en lugar de ubicarse en los laterales. De la misma manera que en el resto de modelos, en la intersección entre dos nodos se coloca la etiqueta de la arista que sirve de unión entre ellos, colocando un 0 si no existiese arista alguna. Además, introducen una serie de mejoras respecto al algoritmo gSpan a través de las operaciones *FFSM-Join* y *FFSM-Extension*.

2.5.8 GREW

GREW es un algoritmo de búsqueda de subgrafos más frecuentes que toma como entrada grafos etiquetados no-dirigidos (Kuramochi y Karypis, 2004). Según explican los autores, GREW descubre patrones de mayor longitud en menor tiempo. A cambio, GREW no garantiza que encuentre absolutamente todos los patrones, puediendo pasar por alto alguno de ellos. Al contrario que otros trabajos de este dominio, GREW no hace uso de matrices para representar los grafos, sino que emplea la técnica denominada "grafo aumentado" (*augmented graph*). El grafo aumentado está diseñado de manera que contenga toda la información necesaria para poder recuperar el grafo original tras aplicar las transformaciones necesarias para encontrar los patrones más frecuentes.

2.5.9 Gaston

Uno de los mayores problemas a la hora de encontrar los subgrafos más comunes es el de la búsqueda de isomorfismo entre dos grafos, ya que dada la complejidad de esta tarea no existe un algoritmo en el que el tiempo de ejecución pueda calcularse de manera polinomial. Para solventar esta problemática, Gaston (*GrAph/Sequence/Tree extratiON*) (Nijssen y Kok, 2004) divide el grafo en árboles y rutas, estructuras para las cuales sí que existen algoritmos eficientes de búsqueda de polimorfismo. Para ello, Gaston analiza primero las rutas; a través de las diferentes transformaciones esta ruta puede convertirse en un "árbol libre" (*free tree*), para pasar, por último, a formar un grafo.

2.5.10 gApprox

Al igual que otras soluciones planteadas a lo largo en este capítulo, como Subdue (sección 2.5.1), gApprox realiza búsquedas aproximadas para encontrar los grafos más frecuentes, por lo que se permite que las ocurrencias encontradas no sean exactamente iguales (Chen et al., 2007). Para ello emplean un error de tolerancia establecido por el usuario. Al contrario de soluciones anteriores, las redes que emplea gApprox como entrada no son etiquetadas, para poder soportar dicho error de tolerancia.

2.5.11 hSiGraM y vSiGraM

Kuramochi y Karypis (2005) presentan dos algoritmos para la búsqueda de subgrafos más frecuentes detro de un único gran grafo. Al igual que otras alternativas presentadas a lo largo de esta sección, hSiGraM y vSiGraM trabajan con grafos no-dirigidos y etiquetados, con la diferencia de que cada vértice o arista del grafo no tiene por qué tener una única etiqueta. Entre ambos algoritmos también existen diferencias: mientras hSiGraM sigue un enfoque horizontal y emplea el algoritmo de búsqueda en anchura, vSiGraM, por su parte, emplea el algoritmo de búsqueda en profundidad, siguiendo un enfoque vertical.

2.5.12 Selección de la herramienta de minería de grafos

Para llevar a cabo la investigación planteada en el marco de esta tesis doctoral, tal y como ha descrito en la sección 1.3.1, existe la necesidad de realizar tareas de minería de grafos. Ya que el objetivo de esta tesis doctoral no es plantear un nuevo algoritmo de extracción de subgrafos más frecuentes, se ha elegido una de las herramientas descritas en la sección 2.5. Para la elección de esta herramienta, se ha tomado en cuenta que las características de los grafos con los que trabaja concuerden con las características de los grafos RDF, es decir, grafos etiquetados (tanto en aristas como vértices) y dirigidos.

Otra de las características que se tienen en cuenta es el tipo de entrada soportada por los algoritmos de minería de grafos. En la solución planteada se extraen los

Solución	Transacc./Individuales	Dirigidos	Etiquetado de vértices	Etiquetado de aristas	Implementación
SUBDUE	Individuales	✔	✔	✔	✔
AGM	Transacciones	✔	✘	✔	✘
FSG	Transacciones	✘	✔	✔	✔
DPMine	Transacciones	✔	✔	✔	✔
MoFA	Transacciones	✘	✔	✘	✔
gSpan	Transacciones	✘	✔	✔	✔
FFSM	Transacciones	✘	✔	✔	✔
GREW	Individuales	✘	✔	✔	✘
Gaston	Individuales	✘	✔	✔	✔
gApprox	Transacciones	✘	✘	✘	✘
(h/v)SiGraM	Individuales	✘	✔	✔	✔

Cuadro 2.1: Comparativa de las diferentes soluciones para la extracción de subgrafos más frecuentes.

subgrafos más frecuentes desde grafos individuales, en lugar de desde transacciones (conjuntos de grafos). Por último, se ha tenido en cuenta la existencia de una implementación funcional del algoritmo, y no solamente la existencia del planteamiento teórico del mismo.

Para facilitar la selección de la herramienta adecuada, en el cuadro 2.1 se muestran las herramientas analizadas junto con sus características más relevantes. Una vez realizada dicha síntesis, no es difícil determinar qué herramientas cumplen con los requisitos especificados. Como puede observarse, la direccionalidad de los grafos supone un inconveniente en la mayoría de las soluciones, ya que añade un alto grado de complejidad a la problemática de la extracción de subgrafos más frecuentes. Filtrando las herramientas a través de esta única característica, solamente quedan SUBDUE, DPMine y AGM como alternativas factibles. De estas tres candidatas, AGM solamente admite etiquetas en las aristas de los grafos, por lo que las únicas habilitadas para ser empleadas con grafos RDF son DPMine y SUBDUE.

Dado que en la solución planteada en esta tesis los subgrafos se extraen desde grafos individuales, **la herramienta adecuada para este fin es SUBDUE**. En el capítulo 3 se dan más detalles acerca del uso dado a la herramienta.

2.6 Conclusión

A lo largo de este capítulo se han presentado las principales soluciones existentes en los campos de investigación relacionados con esta tesis doctoral. Como se ha podido observar, a pesar de que existen múltiples trabajos relacionados con la búsqueda, recomendación y enlazado de conjuntos de datos estructurados, y de diversa naturaleza, ninguno de ellos cumple con las premisas de partida de la hipótesis de esta tesis doctoral, descrita en la sección 1.3.3. Además, en este capítulo se justifica la elección de la herramienta SUBDUE para realizar la minería de grafos requerida por esta tesis. En el próximo capítulo se describe la solución desarrollada para solventar la problemática planteada a lo largo de este documento.

Every honest researcher I know admits he's just a professional amateur.

Charles Franklin Kettering

CAPÍTULO

3

Modelo para la recomendación de conjuntos de datos enlazados

Este capítulo tiene como objetivo presentar la solución inicial planteada para la búsqueda de conjuntos de datos candidatos a ser enlazados. Esta solución propone emplear el modelo basado en grafos implementado por RDF para analizar los subgrafos más frecuentes de un conjunto de datos, y así, poder extraer un resumen o síntesis del mismo. Una vez extraídos los subgrafos más frecuentes de cada conjunto de datos, se pasará a la comparación entre dichos subgrafos, con el objetivo de hallar similitudes entre ellos.

Durante este capítulo, además de presentar el modelo desarrollado, en la sección 3.1 se exponen los conceptos necesarios para la correcta comprensión de la solución desarrollada.

3.1 Antecedentes

En esta sección se presentan el modelo de datos implementado por RDF y las particularidades de la herramienta de minería de datos seleccionada.

3.1.1 Resource Description Framework

RDF es un lenguaje desarrollado por el W3C para representar datos en la Web. Para ello, este lenguaje define una sintaxis cuyo elemento principal son las **tripletas**, cada una de ellas formada por un **sujeto**, un **predicado** y un **objeto**. En el listado 3.1[1] puede verse un ejemplo de dos tripletas que describen que el sujeto representado por la URI `<http://dbpedia.org/resource/Tim_Berners-Lee>` es de tipo "persona" (representado a través de la propiedad *type* de la sintaxis RDF y de la clase *Person* de la ontología *Friend of a Friend* o FOAF[2]) y tiene como nombre (`foaf:name`) el literal "Tim Berners-Lee".

```
1  @prefix :     <http://dbpedia.org/resource/> .
2  @prefix foaf: <http://xmlns.com/foaf/0.1/> .
3  @prefix rdf:  <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
4
5  :Tim_Berners-Lee rdf:type  foaf:Person ;
6                   foaf:name "Tim Berners-Lee" .
```

Listado 3.1: Ejemplo de tripletas.

El conjunto de estas tripletas se denomina *grafo RDF*, y puede ser visualizado como una serie de nodos o vértices **etiquetados** (sujetos y objetos) enlazados a través de una o más aristas **etiquetadas** y **dirigidas** (predicado). La representación en forma de grafo de las tripletas del listado 3.1 puede observarse en la figura 3.1.

Los grafos derivados de los conjuntos de datos enlazados pueden contener un gran número de vértices y aristas. Por poner un ejemplo, el conjunto de datos que describe las publicaciones de la *Association of Computer Machinery* (ACM) contiene más de doce millones de tripletas, de las cuales 1.625.437 son sujetos. Esto supone que el grafo derivado de este conjunto de datos tendrá, al menos, 1.625.437 vértices, a los que hay que sumar un vértice por cada objeto del conjunto de datos. Dado que el coste computacional a la hora de buscar similitudes entre grafos de esta magnitud es demasiado alto, se ha buscado la manera de sintetizar o resumir estos grafos, buscando un subgrafo mínimo que sirva para representar todo el grafo en su conjunto.

[1] Por motivos de legibilidad, todo el contenido semántico que aparece en esta tesis está escrito utilizando la notación Turtle.

[2] `http://www.foaf-project.org/`

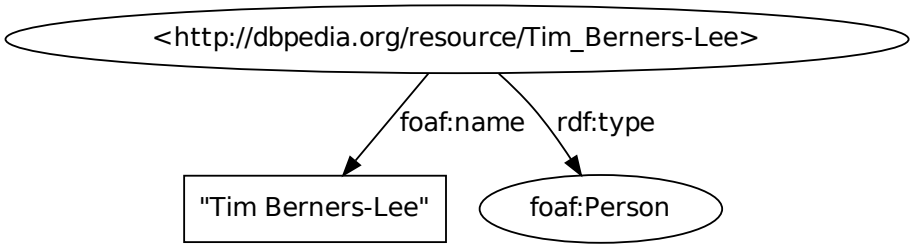


Figura 3.1: Ejemplo de una tripleta. Durante toda esta tesis, se utilizarán elipses para representar las URI y rectángulos para representar literales.

3.1.2 Minería de subgrafos más frecuentes

Para conseguir resumir estos grafos, se ha empleado una técnica de minería de grafos, denominada "minería de subgrafos más frecuentes" (*frequent subgraph mining*). Esta técnica analiza un grafo con el objetivo de encontrar el subgrafo que más ocurrencias tenga dentro de dicho grafo. Según la definición de grafo y subgrafo de Jiang et al. (2013):

- **Grafo etiquetado:** un grafo etiquetado G puede ser representado como una tupla $(V, E, L_V, L_E, \varphi)$ donde V es el conjunto de vértices, $E \subseteq V \times V$ el conjunto de aristas (del inglés *edge*), L_V el conjunto de etiquetas (*label*) de los vértices, L_E el conjunto de etiquetas de las aristas y φ la función que determina las correspondencias $V \rightarrow L_V$ y $E \rightarrow L_E$. G será un grafo dirigido si se cumple que $\forall e \in E$, e es un par **ordenado** de vértices.

- **Subgrafo:** dados dos grafos $G_1(V_1, E_1, L_{V_1}, L_{E_1}, \varphi_1)$ y $G_2(V_2, E_2, L_{V_2}, L_{E_2}, \varphi_2)$ G_1 será subgrafo de G_2 siempre y cuando se cumpla que i) $V_1 \subseteq V_2$, y $\forall v \in V_1, \varphi_1(v) = \varphi_2(v)$ ii) $E_1 \subseteq E_2$, y $\forall (u, v) \in E_1, \varphi_1(u, v) = \varphi_2(u, v)$.

La hipótesis planteada mantiene que a través del análisis de las características estructurales de los conjuntos de datos es posible hallar conjuntos de datos candidatos a ser enlazados. Ya que analizar la estructura de un conjunto de datos en su totalidad resulta extremadamente costoso en términos computacionales, se plantea

la extracción del subgrafo más frecuente de un conjunto de datos dada, con el objetivo de lograr una muestra representativa del conjunto de datos. Se tomó la decisión de emplear esta técnica dada su popularidad dentro de las técnicas de minería de grafos, tal y como puede observarse en el estado del arte de esta tesis doctoral (sección 2.5, página 46). Una vez extraído el subgrafo más frecuente de los conjuntos de datos a comparar, la similitud entre estos subgrafos reflejará la similitud entre los conjuntos de datos en su totalidad.

Para la extracción de estos subgrafos más frecuentes se ha seleccionado la herramienta SUBDUE, por los motivos expuestos en la sección 2.5.12. La herramienta SUBDUE extrae, a partir de un grafo etiquetado y dirigido, los subgrafos más frecuentes del mismo. SUBDUE establece que el subgrafo más frecuente es aquel que, una vez sustituido en el grafo original por un único vértice, es el que más comprime el tamaño de dicho grafo original. Siendo G el grafo analizado, S el subgrafo a evaluar, $size(G)$ y $size(S)$ el tamaño de G y S, y $size(G|S)$ el tamaño de G al sustituir S por un único vértice, se puede calcular el el ratio de compresión resultante ($value$):

$$value(S, G) = \frac{size(G)}{(size(S) + size(G|S))} \tag{3.1}$$

en donde:

$$size(G) = (|V| + |E|) \tag{3.2}$$

Para ello, SUBDUE necesita que el grafo de entrada sea descrito con una sintaxis específica. En el listado 3.2, se muestra el fichero de entrada de SUBDUE resultante del grafo de la figura 3.1. La estructura de este fichero es sencilla. Las líneas uno, dos y tres representan a los vértices del grafo. Cada una de estas líneas contiene el código que indica que dicha línea describe un vértice (`v`), el identificador numérico único de dicho vértice y su correspondiente etiqueta. Por su parte, las aristas (líneas cuatro y cinco) son descritas a través del código `e`, el identificador del vértice origen de la arista, el identificador del vértice destino y la etiqueta de la arista.

```
1  v 1 <http://dbpedia.org/resource/Tim_Berners-Lee>
2  v 2 "Tim Berners-Lee"
3  v 3 foaf:Person
4  e 1 2 foaf:name
5  e 1 3 rdf:type
```

Listado 3.2: Ejemplo de fichero de entrada de SUBDUE.

Además del fichero de entrada, SUBDUE cuenta con una serie de parámetros que varían su funcionamiento. A continuación se describen algunos de los parámetros empleados para la parametrización de SUBDUE durante esta tesis. La parametrización aplicada se describe en la sección 3.4:

- **inc:** este parámetro indica a SUBDUE que debe analizar el grafo de manera incremental. Para ello, el grafo de entrada se divide en múltiples ficheros de entrada.
- **limit**: el número de subestructuras que toma SUBDUE en consideración en cada iteración.
- **prune**: poda el grafo descartando subestructuras cuyo valor sea menor que el de su subestructura padre.

Una vez descrito el grafo en la sintaxis adecuada y establecidos los parámetros de entrada, SUBDUE buscará los subgrafos más frecuentes de dicho grafo.

3.2 Modelo de recomendación de datos enlazados

A lo largo de esta sección se expone el modelo de recomendación de datos enlazados desarrollado en esta tesis doctoral. Para el desarrollo de este modelo se ha generado un modelo similar a lo que Böhm et al. (2012) denominan "grafo abstracto" (*abstract graph*). Este grafo abstracto dispone las clases a las que pertenecen las instancias del conjunto de datos en sus vértices, y las relaciones entre dichas clases en sus aristas. Adicionalmente, mientras que Böhm et al. (2012) descartan las etiquetas textuales y los enlaces externos, en el presente modelo han sido incluidos. Para la correcta comprensión del modelo desarrollado para la recomendación de datos enlazados, en el listado 3.3 (página 60) se muestra un conjunto de datos

RDF, mientras que en la figura 3.2 se muestra su representación en forma de grafo. Como puede observarse, en este ejemplo se muestran tres recursos que describen a los tres autores (`:Tim_Berners-Lee`, `:James_Hendler` y `:Ora_Lassila`) de la publicación `:pub1`.

```
@prefix :         <http://example.org/resource/> .
@prefix foaf:     <http://xmlns.com/foaf/0.1/> .
@prefix rdf:      <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
@prefix dcterms: <http://purl.org/dc/terms/> .
@prefix aktors:  <http://www.aktors.org/ontology/portal> .

:Tim_Berners-Lee rdf:type             foaf:Person ;
                 foaf:name            "Tim Berners-Lee" ;
                 foaf:mbox            "timbl@w3.org" ;
                 foaf:homepage        <http://www.w3.org/People/
                                      Berners-Lee/> ;
                 foaf:knows           :James_Hendler ;
                 foaf:knows           :Ora_Lassila .

:James_Hendler   rdf:type             foaf:Person ;
                 foaf:name            "James Hendler" ;
                 foaf:mbox            "hendler@cs.rpi.edu" ;
                 foaf:homepage        <http://www.cs.rpi.edu/~hendler/
                                      contact.html> ;
                 foaf:knows           :Tim_Berners-Lee ;
                 foaf:knows           :Ora_Lassila .

:Ora_Lassila     rdf:type             foaf:Person ;
                 foaf:name            "Ora Lassila" ;
                 foaf:mbox            "webmaster@lassila.org" ;
                 foaf:homepage        <http://www.lassila.org/> ;
                 foaf:knows           :Tim_Berners-Lee ;
                 foaf:knows           :James_Hendler .

:pub1            rdf:type             aktors:Article-Reference ;
                 aktors:has-title     "The Semantic Web" ;
                 aktors:published-by "Scientific American" ;
                 dcterms:creator      :Tim_Berners-Lee ;
                 dcterms:creator      :James_Hendler ;
                 dcterms:creator      :Ora_Lassila .
```

Listado 3.3: Conjunto de datos RDF en el que se describe una publicación junto sus tres autores.

El primer paso para adaptar este grafo RDF al modelo de recomendación de datos enlazados planteado, consiste en eliminar las URI únicas que identifican cada uno de estos recursos. Esta sustitución se debe a que como estas URI son únicas,

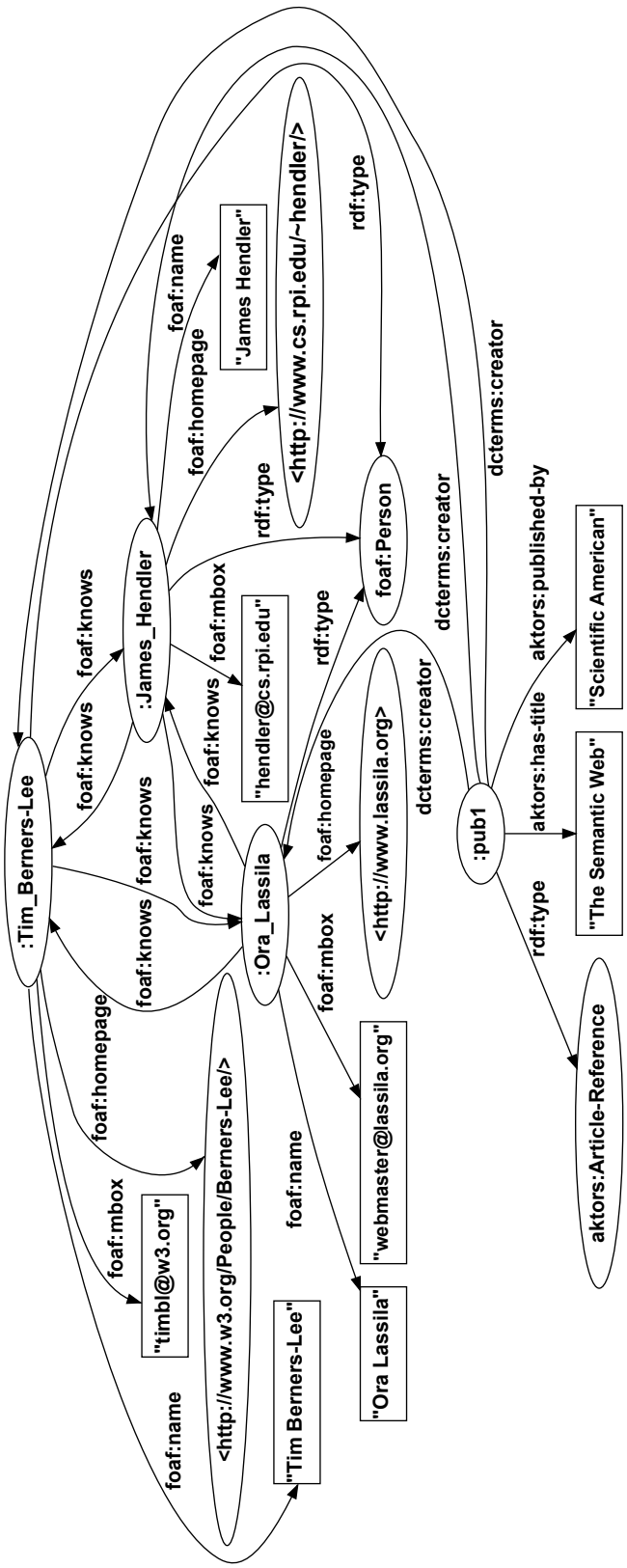


Figura 3.2: Representación en forma de grafo de las tripletas del listado 3.3.

se generarían tantos nodos únicos como sujetos existentes en el conjunto de datos. Estos nodos únicos no podrían formar parte de ninguna subestructura candidata ya que entorpecerían el trabajo de hallar los subgrafos más frecuentes.

Debido al enfoque tomado en esta tesis doctoral, dichas URI se han sustituido en el grafo por la clase ontológica a la que pertenece cada una de las instancias que representan. De esta manera, el modelo representará relaciones del tipo "una persona conoce a otra persona" o "una persona es autora de una publicación". Este tipo de simplificación del grafo permite que la estructura del conjunto de datos quede representada de una manera generalizada, facilitando su posterior síntesis. En la figura 3.3 puede verse el estado del grafo tras el reemplazo de las URI por la clase ontológica a la que pertenecen.

Tal y como se ha visto en el capítulo 1, una de las características de los conjuntos de datos enlazados consiste en entrelazar los datos de diferentes conjuntos, con el objetivo de enriquecer los datos originales y ofrecer la posibilidad de inferir nuevo conocimiento. Por esta razón, muchos de los conjuntos de datos empleados para la validación ya contenían enlaces externos. Siguiendo el enfoque adoptado, los enlaces externos cuyo destino fuese un recurso RDF deberían tratarse al igual que los sujetos del conjunto de datos, sustituyendo las URI de dichos recursos por las clases ontológicas a las que pertenecen. Sin embargo, además de la complejidad que añade el hecho de tener que recuperar estos recursos externos a través de la red, ya que uno de los objetivos de esta tesis es solucionar el problema de arranque en frío que tienen otras soluciones relacionadas, estos enlaces han sido eliminados en el modelo de recomendación. En la figura 3.4 puede observarse el estado del grafo tras la supresión de las URI externas.

Respecto a los literales, a pesar de que se han mantenido dentro del modelo, ninguna de las subestructuras más comunes extraídas de los conjuntos de datos empleados durante la validación de la hipótesis contiene ningún literal. La explicación de este fenómeno es similar a la de las URI: al haber tanta variedad de literales diferentes, la probabilidad de que formen parte de una subestructura candidata a formar el subgrafo más frecuente es mínima.

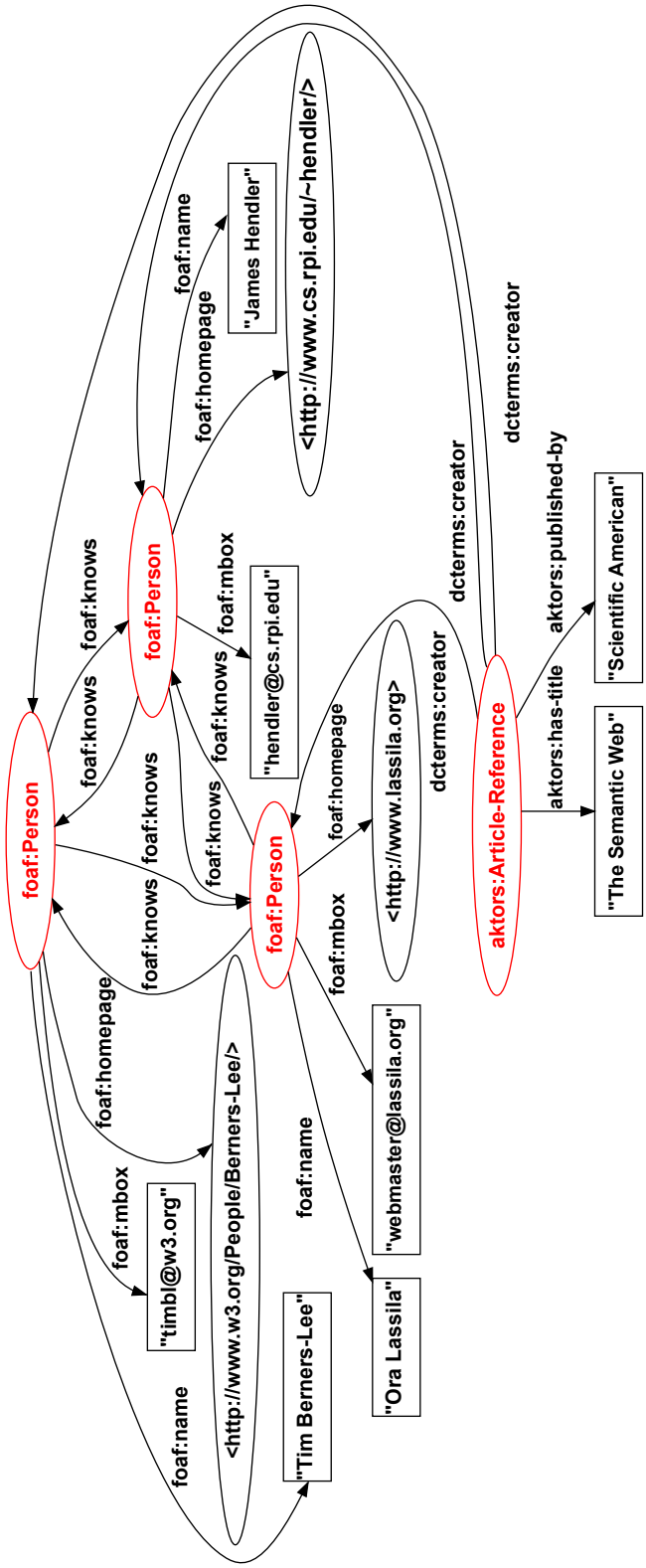


Figura 3.3: Primera transformación del grafo original: sustitución de las URI por la clase a la que pertenece la instancia representada. En rojo los nodos de las URI que han sido sustituidos por la clase a la que pertenecen.

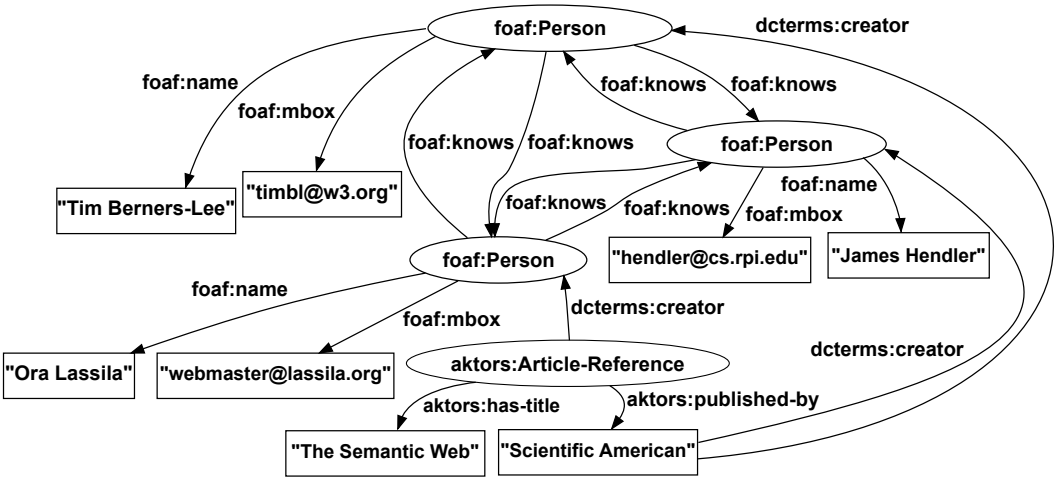


Figura 3.4: Estado del grafo RDF de ejemplo tras la supresión de las URI externas.

3.3 Algoritmo de modelado: rdf2subdue

En esta sección se describe el algoritmo desarrollado para realizar la conversión entre el modelo RDF y el modelo de recomendación de conjuntos de datos estructurados planteado en la sección 3.2. Este algoritmo, denominado **rdf2subdue** consta de tres pasos: la asignación de identificadores a los vértices, el establecimiento de las aristas y la escritura de los ficheros. Con el objetivo de poder extraer la información de manera adecuada de los conjuntos de datos, éstos son previamente almacenados en un repositorio RDF. Como salida de este algoritmo de modelado, se genera el fichero de SUBDUE correspondiente, de la misma manera que se ha mostrado en el listado 3.2 (página 59). El algoritmo, al igual que todos los algoritmos de este capítulo, está descrito en pseudocódigo con sintaxis al estilo del lenguaje Python. En la figura 3.5 puede observarse la representación gráfica de la sucesión de los diferentes algoritmos descritos en esta sección y en las posteriores (3.4, 3.5 y 3.6), para mejor compresión de los mismos.

3.3.1 Asignación de identificadores a los vértices y generación de aristas

Durante este paso se generan los identificativos numéricos únicos para cada uno de los vértices del grafo, además de reemplazar las URI de los recursos por su

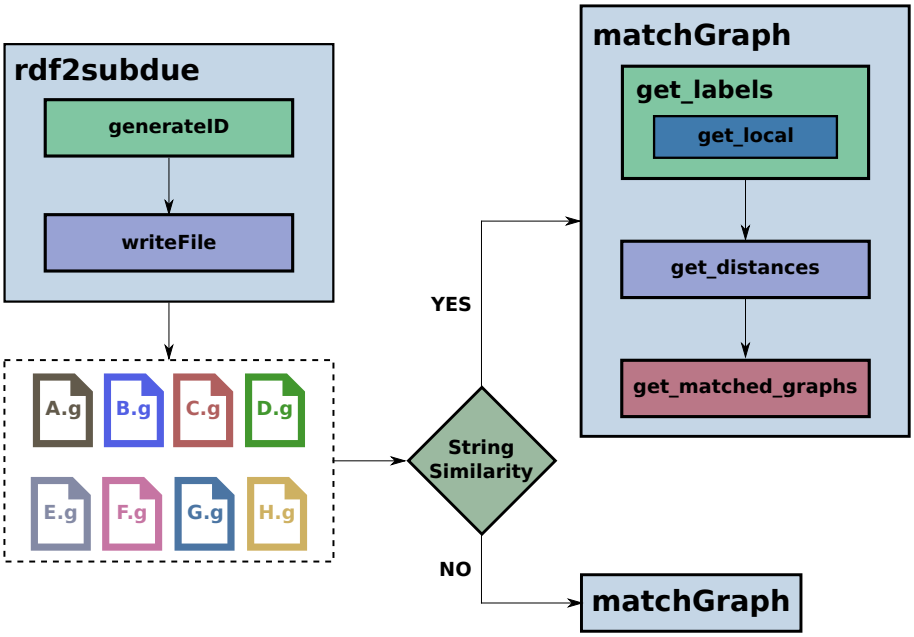


Figura 3.5: Representación de la sucesión de los diferentes algoritmos descritos en este capítulo.

clase ontológica correspondiente y de generar las aristas, tal y como puede verse en el listado 3.4. Esta función se llama una única vez por cada conjunto de datos. Siendo V_{id} el identificador de un vértice y $\mathbf{V}$ el conjunto de todos los vértices de un conjunto de datos, SUBDUE establece que $\forall V \in \mathbf{V}, V_{id} \geq 1 \wedge V_n = V_{n-1} + 1$, siendo $n \in [0 \ldots |\mathbf{V}| - 1]$.

```
1  def generateID(dataset):
2      _id = 1
3      literalID = 0
4
5      # Extraccion de sujetos
6      query1 = '''SELECT DISTINCT ?s ?class WHERE {?s a ?class}'''
7      triples = virtuoso_query.create(query, dataset)
8      for s, clazz in triples:
9          cache.push('%s:%s' % (dataset, s), _id)
10         bd.store(_id=_id, label=clazz, type_='vertex')
11         _id += 1
12     # Extraccion de objetos
13     query2 = '''SELECT DISTINCT ?o WHERE { ?s ?p ?o .
14                 FILTER EXISTS { ?s a ?class } .
15                 FILTER NOT EXISTS { ?o ?p2 ?o2 } }'''
16     triples = virtuoso_query.create(query2, dataset)
```

```
17      for o in triples:
18          if not cache.exists('%s:%s' % (dataset, o)):
19              if o.is_literal():
20                  cache.push('%s:%s' % (dataset, o), _id)
21                  bd.store(_id=_id, label=get_hash(o), type_='vertex')
22                  _id += 1
23                  literalID += 1
24      # Generacion de aristas
25      query3 = '''SELECT DISTINCT ?s ?p ?o WHERE { ?s ?p ?o .
26                  FILTER EXISTS { ?s a ?class } }'''
27      triples = virtuoso_query.create(query3, dataset)
28      for s, p, o in triples:
29          if p != RDF.type:
30              source__idlist = cache.get_list('%s:%s' % (dataset, s))
31              target__idlist = cache.get_list('%s:%s' % (dataset, o))
32              for source_id in source__idlist:
33                  for target_id in target__idlist:
34                      bd.store(source=source_id, target=target_id,
35                               label=predicate, type_='edge')
```

Listado 3.4: Generación de vértices y aristas.

El primer paso del procedimiento `generateID` consiste en extraer todos los recursos que pertenezcan a una o más clases ontológicas, como puede verse entre las líneas 5 y 12. La sentencia SPARQL `query1` extrae todos los sujetos y clases diferentes del conjunto de datos. A continuación, se asigna un identificador numérico único a cada uno de los sujetos y se almacenan en la memoria *caché*, para garantizar su rápido acceso. En la memoria caché se almacenan los identificadores en forma de lista ya que un recurso puede pertenecer a varias clases ontológicas. En la base de datos, en cambio, se almacena este identificador junto con la clase ontológica a la que pertenece el recurso analizado.

El código situado entre las líneas 12 y 24 se encarga de extraer los objetos del conjunto de datos. La sentencia SPARQL `query2` extrae los objectos que pertenezcan a una tripleta cuyo sujeto pertenezca al menos a una clase ontológica (ver restricciones en la sección 1.3.3) y que a su vez, dichos objetos no sean sujetos en otra tripleta. De esta manera, se asegura que todos los resultados son *a*) literales o *b*) enlaces externos, ya que al no ser sujeto de otra tripleta se puede afirmar que no pertenecen al conjunto de datos. A continuación, por cada objeto devuelto por la consulta se comprueba si es literal, ya que las URI externas son descartadas en este modelo. Al igual que con los sujetos, a cada literal se le asigna un identifica-

dor único y se almacenan tanto en la memoria caché como en la base de datos. En este punto existe una pequeña diferencia: ya que algunos literales pueden contener caracteres "extraños" (que no pertenecen a UTF-8, por ejemplo), cada literal se sustituye por el identificador `literalID`. Actualmente no se almacena la correlación entre el literal original y el identificador, ya que los literales no son utilizados en este modelo. Sin embargo, de esta manera, si en un futuro se pretendiese emplear los literales, podría extraerse el valor de cada uno a través de dicho identificador.

Por último, a partir de la línea 24 comienza la extracción de aristas. La sentencia SPARQL recorre todas las tripletas del grafo siempre y cuando cada recurso analizado pertenezca a una clase ontológica. Ya con los identificadores de los vértices generados, lo único que queda es extraer los predicados para establecer las aristas que conectan los vértices entre sí.

3.3.2 Generación del fichero SUBDUE

Una vez que se han generado los vértices y aristas del grafo, el siguiente paso consiste en generar los ficheros de entrada para SUBDUE. Esto podría realizarse en el momento en el que se generan los vértices y las aristas evitando, así, la necesidad de almacenarlos en la base de datos. Sin embargo, dado el tamaño de muchos de los conjuntos de datos analizados, ha sido necesario utilizar la capacidad de análisis incremental de SUBDUE. Esto permite dividir el grafo en múltiples ficheros con el objetivo de analizarlo incrementalmente, siempre y cuando se cumplan dos condiciones:

- **Vértices ordenados:** los vértices tienen que estar ordenados en orden ascendente en base a su identificativo numérico.
- **Aristas con vértices conocidos:** en un fichero determinado solamente pueden aparecer aristas entre vértices que aparezcan en ese mismo fichero o que ya hayan aparecido en ficheros anteriores.

Debido a esta restricciones resulta imposible generar los ficheros en el mismo momento en el que se generan los vértices y las aristas, por lo que se hace necesario un segundo procedimiento que escriba los ficheros. En este caso, los grafos se

han dividido en ficheros de mil vértices cada uno. En el listado 3.5, se muestra el pseudocódigo del algoritmo.

```
1  def writeFile(dataset):
2      FIXED_LIMIT = 1000
3
4      end = False
5      bottom_limit = 0
6      upper_limit = 0
7      count = 1
8
9      while not end:
10         upper_limit = count * FIXED_LIMIT
11         # Extraccion de vertices
12         vertex_list = bd.query('''SELECT * FROM %s WHERE id > %s
13                                   AND id <= %s AND type = "vertex"
14                                   ORDER BY id''' %
15                                   (dataset, bottom_limit, upper_limit))
16         if len(vertex_list) == 0:
17             end = True
18             continue
19
20         output_file = open('%s_%s.g' %
21                            (dataset, count), 'w')
22         # Escritura de vertices
23         for vertex in vertex_list:
24             output_file.write('v %s %s\n' % (vertex.id, vertex.label))
25         # Extraccion de aristas
26         edge_list = bd.query('''SELECT * FROM %s WHERE
27                                 (source <= %s AND target <=%s )
28                                 AND (source > %s OR target > %s) ''' %
29                                 (dataset, upper_limit, upper_limit,
30                                  bottom_limit, bottom_limit))
31         for edge in edge_list:
32             output_file.write('d %s %s %s\n' % (source, target, label))
33
34         output_file.close()
35         count += 1
36         bottom_limit += FIXED_LIMIT
```

Listado 3.5: Generación de ficheros de SUBDUE.

Como puede observarse, el algoritmo es muy intuitivo. A partir de la línea 11 se ejecuta la consulta para extraer los vértices de la base de datos cumpliendo con las restricciones planteadas. En este caso, se asegura que el identificador de los vértices se encuentre entre los límites inferior y superior para el fichero que se está generando en ese momento. Una vez se han extraído los vértices se escriben

en el fichero de salida (línea 22 en adelante).

Con las aristas ocurre algo similar. A partir de la línea 25 puede observarse la sentencia SQL para la extracción de aristas. Esta sentencia cuenta con dos condiciones que deben cumplir las aristas para poder incluirse en el fichero en curso:

- $source <= upper\_limit \wedge target <= upper\_limit$: esta condición asegura que tanto el vértice origen como el vértice destino de la arista no sean un vértice que todavía no ha aparecido en ninguno de los ficheros generados.

- $source > bottom\_limit \vee target > bottom\_limit$: esta condición asegura que al menos uno de los dos vértices (origen o destino) es mayor que el límite inferior. Esto significaría que la arista no ha aparecido en ninguno de los ficheros anteriores evitando, así, la duplicidad de aristas.

Una vez extraídas las aristas, se escriben en el fichero de salida. El algoritmo finalizará cuando no existan más vértices en el grafo.

3.4 Extracción de subgrafos: SUBDUE

Tal y como se ha justificado en la sección 2.5.12, la herramienta seleccionada para extraer los subgrafos más frecuentes de un grafo es la herramienta SUBDUE. Una vez generados los ficheros de entrada, esta herramienta extrae los subgrafos más frecuentes de cada grafo. Tal y como se avanzaba en la sección 3.1.2, a la hora de lanzar SUBDUE ha habido que modificar algunos de sus parámetros por defecto. Esta parametrización de SUBDUE ha sido necesaria dado el gran tamaño de los grafos a analizar. A continuación se justifican los valores asignados a cada uno de los diferentes parámetros:

- **inc:** este parámetro activa la característica de análisis incremental de SUBDUE. Dado el tamaño de los grafos analizados, el tamaño del fichero de SUBDUE correspondiente consume una cantidad excesiva de memoria, tanto al ser generado como al ser leído por SUBDUE. Al fragmentar la salida en ficheros de menor tamaño, se solventan ambos problemas.

- **prune:** al activar el parámetro `prune` SUBDUE descarta las subestructuras cuyo valor (ecuación 3.1.2, página 58) es menor que el de su subestructura padre. Gracias a este parámetro, el espacio de búsqueda se reduce notablemente, mejorando la eficiencia a la hora de analizar grandes grafos. Esto se explica ya que usualmente, según va aumentando el tamaño de una subestructura, la longitud de su descripción (principio en el que basa SUBDUE su funcionamiento) no desciende (Cook y Holder, 1993).

- **limit:** SUBDUE funciona iterando sobre el grafo, comenzando desde la subestructura mínima, es decir, subestructuras con un único vértice. A partir de ahí, ordena las subestructuras en función de su valor y las amplía añadiendo un vértice a la subestructura en cada iteración. Por defecto, en cada iteración SUBDUE selecciona las n mejores subestructuras, siendo $n = \frac{|E|}{2}$, donde $|E|$ es el número de aristas. Dadas las características de los grafos analizados, el valor de n puede resultar demasiado grande en muchos casos provocando que SUBDUE consuma toda la memoria del sistema. El parámetro `limit` permite limitar este valor. Tras una serie de pruebas empíricas, se ha seleccionado limitar el número de subestructuras por cada iteración a cinco, ya que este límite funciona con todos los grafos analizados.

Una vez extraídos los subgrafos más frecuentes de cada uno de los grafos, estos subgrafos son almacenados en la base de datos para su posterior análisis.

3.5 Emparejamiento de subgrafos

Una vez extraídos los subgrafos más frecuentes de cada conjunto de datos, el siguiente paso consiste en encontrar similitudes entre dichos subgrafos. Para ello, se ha empleado la herramienta *graph matcher* (gm) de SUBDUE. A partir de una pareja de grafos, esta herramienta calcula el coste de transformar el más grande de los dos en el más pequeño. Por defecto SUBDUE asigna el mismo coste para todas las transformaciones realizadas. Estas transformaciones comprenden acciones como insertar o eliminar vértices o aristas, sustituir etiquetas de vértices o aristas, insertar o eliminar aristas junto a vértices o cambiar la direccionalidad de una arista.

Como puede intuirse, el coste de transformación no es un valor absoluto, ya que varía en función del tamaño de los grafos a analizar. En este trabajo, la normalización del coste de transformación se realiza de la siguiente manera:

$$costeTransNormalizado(G_1, G_2) = \frac{costeTransformacion}{max(|V_1| + |E_1|, |V_2| + |E_2|)} \quad (3.3)$$

De la misma manera, se establece la similitud entre grafos como:

$$sim(G_1, G_2) = 1 - costeTransNormalizado(G_1, G_2) \quad (3.4)$$

A la hora de encontrar conjuntos de datos candidatos a ser enlazados con un conjunto determinado y dar solución a la problemática planteada en la sección 1.3.1, se ejecuta el algoritmo descrito en el listado 3.6 (página 72). El algoritmo presentado es, una vez más, muy intuitivo. En primer lugar, se extrae de la base de datos la subestructura más frecuente del grafo que se desea enlazar y se escriben sus vértices y aristas formando un fichero de entrada de SUBDUE (líneas 2 - 12). A continuación, se extraen todos los subgrafos más frecuentes del resto de los conjuntos y se realiza la misma operación con cada uno de ellos (líneas 12 - 26). Al final de cada iteración, se lanza el comando `gm` para obtener el coste de transformación de cada uno de ellos (líneas 26 - 31) y se normaliza siguiendo lo planteado en la ecuación 3.5 (líneas 31 - 38). Por último, si la similitud (ecuación 3.5) supera el umbral asignado, el resultado se añade a la lista que será devuelta como resultado de todo el proceso (línea 38 en adelante). Este umbral puede ser definido por el usuario en función de los conjuntos de datos a enlazar. Durante la evaluación (capítulo 4) se han empleado diferentes umbrales con el objetivo de encontrar el umbral que proporcione mejores resultados.

En el capítulo 4 se muestran los resultados obtenidos al evaluar esta técnica a la hora de encontrar conjuntos de datos candidatos a ser enlazados.

```
def matchGraph(source_graph_name, threshold):
    #
    source_graph = bd.query('''SELECT * FROM SUBGRAPHS WHERE name == "%s"'''
                            % (source_graph))
    source_file = tempfile.TemporaryFile('w')
    for vertex in source_graph.vertexes:
        source_file.write('v %s %s\n' % (vertex.id, vertex.label))
    for edge in source_graph.edges:
        source_file.write('d %s %s %s\n' %
                          (edge.source.id, edge.target.id, edge.label))
    source_file.flush()
    #
    graph_list = bd.query('SELECT * FROM GRAPHS')
    match_list = []
    for graph in graph_list:
        if graph != source_graph:
            graph_file =  tempfile.TemporaryFile('w')
            for vertex in graph.vertexes:
                graph_file.write('v %s %s\n' % (vertex.id, vertex.label))
            for edge in edge.vertexes:
                graph_file.write('d %s %s %s\n' %
                                 (edge.source.id,
                                  edge.target.id,
                                  edge.label))
            graph_file.flush()
            #
            cost = os.system('%s/bin/gm %s %s' %
                             (SUBDUE_PATH,
                              source_file.name,
                              graph_file.name))
            #
            max_size = max(len(source_graph.vertexes) +
                           len(source_graph.edges),
                           len(graph.vertexes) +
                           len(graph.edges))

            normalized_cost = cost / max_size
            #
            sim = 1 - normalized_cost
            if normalized_cost > threshold:
                match_list.append(normalized_cost)

    return match_list
```

Listado 3.6: Búsqueda de conjuntos candidatos a ser enlazados.

3.6 Búsqueda de similitudes entre cadenas de caracteres

A pesar de los prometedores resultados obtenidos durante la evaluación del sistema (sección 4.3.1, página 92), se hallaron deficiencias a la hora de calcular la similitud entre conjuntos de datos del mismo dominio pero descritos por diferentes ontologías. Para ilustrar esta problemática se pueden observar los dos grafos de la figura 3.6. En esta figura se muestras los subgrafos más frecuentes de los conjuntos RISKS[1] y Hedatuz[2]. Aparentemente y a juzgar por sus respectivos subgrafos más frecuentes, ambos conjuntos de datos describen publicaciones. Sin embargo, debido a su planteamiento, el sistema es incapaz de detectar ningún tipo de similitud entre ellos, a pesar de la similitud existente entre los términos `foaf:Person` y `aktors:Person` por un lado y `bibo:Article` y `aktors:Article-Reference` por el otro.

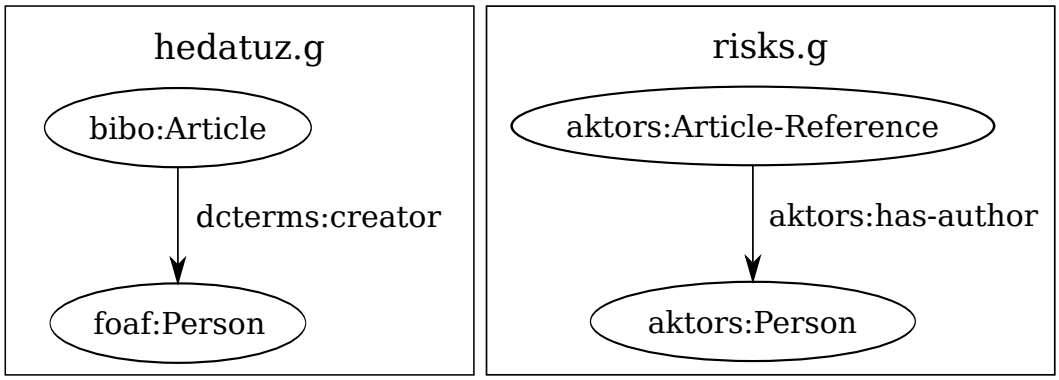


Figura 3.6: Subestructuras más frecuentes de los conjuntos de datos RISKS y Hedatuz.

Para tratar de solventar este problema se plantea la aplicación de técnicas de búsqueda de similitudes entre cadenas de caracteres, con el objetivo de reemplazar las etiquetas de vértices o aristas que superen un umbral de similitud determinado por etiquetas comunes. Estas técnicas, aplicadas en el mundo del emparejamiento de ontologías (*ontology matching*), permiten calcular la distancia existente entre

[1] `http://datahub.io/dataset/rkb-explorer-risks`
[2] `http://datahub.io/dataset/hedatuz`

dos cadenas de caracteres (Euzenat y Shvaiko, 2007). De las diferentes técnicas existentes, se han empleado aquellas que, siguiendo la clasificación de Euzenat y Shvaiko para las técnicas de emparejado de ontologías, pueden ser denominadas como "técnicas basadas en cadenas de caracteres" y "técnicas basadas en el lenguaje". Para la definición de cada una de ellas se va a emplear la notación seguida por los autores de dicha clasificación, descrita a continuación.

Siguiendo esta notación, $\mathbb{S}$ representa el conjunto de cadenas de caracteres, esto es, secuencias de letras de cualquier longitud sobre un alfabeto $\mathbb{L} : \mathbb{S} = \mathbb{L}*$. $\forall s, t \in \mathbb{S}, s + t$ representa la concatenación de las cadenas s y t, mientras $|s|$ representa la longitud de la cadena s, es decir, el número de caracteres que contiene. $s[i]$ para $i \in [1 \dots |s|]$ representa la letra en la posición i de la cadena s.

Dentro de las técnicas basadas en cadenas de caracteres se han utilizado los siguientes métodos:

- **Similitud de cadenas de caracteres:** este método devuelve 0 si las cadenas de caracteres no son idénticas y 1 si lo son. Se puede definir como una similitud $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0, 1]$ de manera que $\forall x, y \in \mathbb{S}, \sigma(x, x) = 1$ y siempre que $x \neq y, \sigma(x, y) = 0$.

- **Similitud de subcadenas:** este método calcula la distancia entre dos cadenas basándose en el tamaño de la subcadena más larga que tengan en común. Se puede definir como una similitud $\sigma : \mathbb{S} \times \mathbb{S} \rightarrow [0 \dots 1]$ de manera que $\forall x, y \in \mathbb{S}$, t será la subcadena común más larga de x e y:

$$\sigma(x, y) = \frac{2|t|}{|x| + |y|} \tag{3.5}$$

- **Distancia de edición**: la distancia de edición es el coste mínimo de operaciones necesarias para transformar una cadena de caracteres en otra. Dado un conjunto Op de operaciones de cadenas de caracteres ($Op : \mathbb{S} \rightarrow \mathbb{S}$), y una función de coste $w : Op \rightarrow \mathbb{R}$, de manera que para toda pareja de cadenas exista una secuencia de operaciones que transformen la primera en la segunda (y viceversa), la distancia de edición es la disimilitud $\delta : \mathbb{S} \times \mathbb{S} \rightarrow [0 \dots 1]$ cuando $\delta(s, t)$ es el coste de la secuencia de operaciones menos costosa para transformar s en t:

$$\delta(s,t) = \min_{(op_i)_I; op_n(...op_1(s))=t} (\sum_{i \in I} w_{op_i}) \tag{3.6}$$

En este caso, se ha tomado la distancia de Levenshtein como función de coste. La distancia de Levenshtein (Levenshtein, 1966) calcula el número mínimo de ediciones que hacen falta para convertir una cadena en otra. Estas ediciones comprenden operaciones como la inserción, eliminación o sustitución de caracteres. Este tipo de distancia de edición establece que todas las operaciones de edición tiene un coste igual a 1.

- **Distancia SMOA:** la distancia SMOA (*a String Metric for Ontology Alignment*) (Stoilos et al., 2005) calcula la distancia entre dos cadenas basándose tanto en sus diferencias como en sus similitudes. Al igual que distancias como la ya mencionada de Levenshtein calculan el coste de edición en función a las operaciones de transformación entre una cadena y otra, esta métrica establece que el hecho de no realizar ediciones debería bonificar disminuyendo la distancia entre ambas cadenas. SMOA calcula la similitud entre dos cadenas de la siguiente manera:

$$Sim(s_1, s_2) = Comm(s_1, s_2) - Diff(s_1, s_2) + winkler(s_1, s_2) \tag{3.7}$$

La función $Comm$ calcula las subcadenas en común entre las cadenas s_1 y s_2. Una vez encuentra la subcadena en común más larga, la elimina de ambas cadenas y busca la siguiente subcadena en común más larga. Este proceso se repite hasta que no existen más cadenas en común. Una vez extraídas todas estas subcadenas, se suman sus longitudes y se normaliza el resultado:

$$Comm(s_1, s_2) = \frac{2 * \sum_i length(maxComSubString_i)}{length(s_1) + length(s_2)} \tag{3.8}$$

Por su parte, la función $Diff$ se basa en las subcadenas que no han podido ser emparejadas por la funcción $Comm$. Ya que los autores de esta métrica consideran que las diferencias deben tener menor peso que las similitudes, para calcularlo han utilizado el producto de Hamacher (Hamacher et al., 1978), que es una norma triangular paramétrica:

$$Diff(s_1, s_2) = \frac{uLen_{s_1} * uLen_{s_2}}{p + (1-p) * (uLen_{s_1} + uLen_{s_2} - uLen_{s_1} * uLen_{s_2})} \quad (3.9)$$

donde $p \in [0, \infty)$, y $uLen_{s_1}$ y $uLen_{s_2}$ representan la longitud normalizada de la subcadena que no ha podido ser emparejada de las cadenas s_1 y s_2 respectivamente. p es una variable que determina la importancia del factor de diferencia.

Por último, la función $winkler$ representa al método empleado por Winkler para mejorar el resultado. Dada esta similitud SMOA, en este caso se calcula la diferencia de la siguiente manera:

$$smoaDiff(s_1, s_2) = 1 - Sim(s_1, s_2) \quad (3.10)$$

Por parte de las técnicas basadas en el lenguaje, se ha empleado un tesauro extrínseco, en este caso WordNet (Miller, 1995). En concreto, WordNet se utiliza para calcular la "distancia de sinonimia básica". Inspirada en el sistema Ontosim[1], dados dos términos, este método calcula la distancia existente entre los sinónimos de uno de los términos y el otro término. Si la distancia de subcadena entre los dos términos originales es menor que la distancia de uno de los términos a cualquier sinónimo del otro, se devuelve esta distancia. En el caso de que la distancia entre uno de los términos y alguno de los sinónimos del otro término sea menor, se devuelve la distancia entre dicho sinónimo y el término.

Estas técnicas se emplean para tratar de encontrar similitudes entre la parte local de las URI que representan las clases de las diferentes ontologías. Por ejemplo, dadas las URI `<http://purl.org/ontology/bibo/Article>` y `<http://www.aktors.org/ontology/portal#Article-Reference>` se calculará la distancia entre la parte local de ambas URI, es decir, entre los términos *Article* y *Article-Reference*. Una vez calculadas las distancias empleando las diferentes técnicas descritas, estas se agregan a través de la media aritmética de todas ellas. En la figura 3.7 se muestra un ejemplo de cómo quedarían las subestructuras de la figura 3.6 tras sustituir las etiquetas similares por etiquetas comunes.

[1] http://ontosim.gforge.inria.fr/

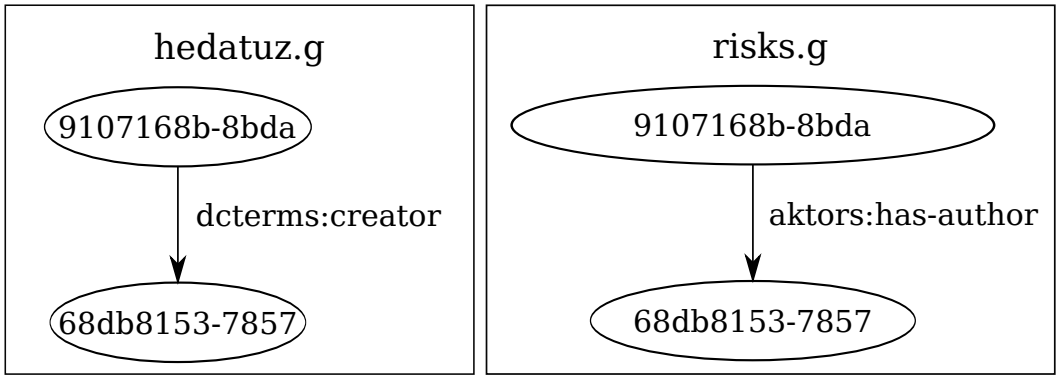


Figura 3.7: Subestructuras más frecuentes de los conjuntos de datos RISKS y Hedatuz tras reemplazar los términos similares.

En el listado 3.7 se muestra el algoritmo empleado para encontrar conjuntos de datos candidatos a ser enlazados empleando este tipo de técnicas. Esta versión del método `matchGraph` contiene muchas similitudes con el método descrito anteriormente en el listado 3.6. El primer paso de este nuevo método consiste en extraer de la base de datos tanto el conjunto de datos que se desea enlazar como el resto de conjuntos del repositorio (líneas 3 - 7). Como puede observarse a partir de la línea 7, el algoritmo itera a través de todos conjuntos de datos del repositorio. En primer lugar se extraen las etiquetas tanto del grafo fuente como del grafo objetivo a través del método `get_labels`, descrito en el listado 3.8. Este método extrae las etiquetas de cada grafo, clasificando por una parte las etiquetas de los vértices y por otra las de las aristas. A estas etiquetas se les elimina su espacio de nombres (*namespace*) a través del método `get_local` (listado 3.9).

```
1  def matchGraph(source_graph_name, string_threshold,
2                 threshold):
3      #
4      source_graph = bd.query('''SELECT * FROM SUBGRAPHS WHERE name == "%s"'''
5                              % (source_graph))
6      graph_list = bd.query('SELECT * FROM GRAPHS')
7      #
8      match_list = []
9      for graph in graph_list:
10         if graph != source_graph:
11           vertex_label_set, edge_label_set = get_labels(source_graph, graph)
12           #
13           distance_map = {}
14
```

```
15            distance_map.update(get_distances(vertex_label_set))
16            distance_map.update(get_distances(edge_label_set))
17            #
18            replace_map = {}
19
20            for label in distance_map.keys():
21              diff_map = distance_map[label]
22              min_distance = 2
23              min_label = ''
24              for key in diff_map.keys()
25                if diff_map[key] < min_distance:
26                  min_distance = diff_map[key]
27                  min_label = key
28
29              if 1 - min_distance > string_threshold:
30                uuid = uuid.UUID()
31                replace_map[label] = uuid
32                replace_map[min_label] = uuid
33            #
34            source_matched_graph = get_matched_graph(source_graph,
35                                                     replace_map)
36            target_matched_graph =
37                    get_matched_graph(graph, replace_map)
38
39            source_file = tempfile.TemporaryFile('w')
40            for vertex in source_matched_graph.vertexes:
41                source_file.write('v %s %s\n' % (vertex.id, vertex.label))
42            for edge in source_matched_graph.edges:
43                source_file.write('d %s %s %s\n' %
44                                  (edge.source.id,
45                                   edge.target.id,
46                                   edge.label))
47            source_file.flush()
48
49            graph_file =  tempfile.TemporaryFile('w')
50            for vertex in graph.vertexes:
51                graph_file.write('v %s %s\n' % (vertex.id, vertex.label))
52            for edge in edge.vertexes:
53                graph_file.write('d %s %s %s\n' %
54                                 (edge.source.id,
55                                 edge.target.id,
56                                 edge.label))
57            graph_file.flush()
58
59            cost = os.system('%s/bin/gm %s %s' %
60                             (SUBDUE_PATH,
61                              source_file.name,
62                              graph_file.name))
63            max_size = max(len(source_graph.vertexes) +
64                           len(source_graph.edges),
65                           len(graph.vertexes) +
```

```
66                          len(graph.edges))
67
68            normalized_cost = cost / max_size
69            sim = 1 - normalized_cost
70            if normalized_cost > threshold:
71                match_list.append(normalized_cost)
72
73      return match_list
```

Listado 3.7: Búsqueda de conjuntos candidatos a ser enlazados empleando técnicas de similitud entre cadenas de caracteres.

```
1  def get_labels(source_graph, graph):
2    vertex_label_set = Set()
3    edge_label_set = Set()
4    for v in source_graph.vertexes():
5      vertex_label_set.add(get_local(v.label))
6    for v in graph.vertexes():
7      vertex_label_set.add(get_local(v.label))
8    for e in source_graph.edges():
9      edge_label_set.add(get_local(e.label))
10   for e in graph.edges():
11     edge_label_set.add(get_local(e.label))
12
13   return vertex_label_set, edge_label_set
```

Listado 3.8: Extracción de etiquetas de un grafo.

```
1  def get_local(uri):
2    if '#' in uri:
3      return uri.split('#')[1]
4    else:
5      suri = uri.split('/')
6      return suri[:len(suri) - 1]
```

Listado 3.9: Extracción de la parte local de una URI.

El siguiente paso consiste en extraer la distancia entre las diferentes etiquetas de los vértices y aristas (líneas 12 - 17). Tal y como muestra el método `get_distances` (listado 3.10), la distancia entre los diferentes términos se extrae de una base de datos en la cual se han almacenado las distancias previamente calculadas para todas las parejas de términos de todos los conjuntos de datos a analizar.

A continuación (líneas 17 - 33), por cada término, se busca el término respecto al cual la distancia es menor. Si dicha distancia supera el umbral asignado (*string_threshold*), se considerará que los términos son similares, por lo que se sustituirán por un identificador común.

```
1  def get_distances(label_set):
2    distance_map = {}
3    permutations = itertools.permutations(label_set)
4
5    for perm in permutations:
6      if perm[0] != perm[1]:
7        diff_list = db.query('''SELECT value FROM diff WHERE
8                                source = "%s" AND target = "%s"'''
9                                % (perm[0], perm[1]))
10       accum = 0
11       count = 0
12       for value in diff_list:
13         accum += value
14         count += 1
15       diff = accum / count
16       if perm[0] not in distance_map.keys():
17         distance_map[perm[0]] = {}
18       distance_map[perm[0]][perm[1]] = diff
19     else:
20       distance_map[perm[0]][perm[1]] = 0
21
22     return distance_map
```

Listado 3.10: Obtención de distancias entre etiquetas.

Una vez detectados los términos similares, se generarán nuevos grafos con dichos términos reemplazados por etiquetas en común, tal y como puede observarse en el método `get_matched_graph` (listado 3.11). Este método crea una copia del grafo original sustituyendo las etiquetas de los vértices y aristas por las etiquetas comunes previamente calculadas. El hecho de reemplazar los términos similares por etiquetas en común hará que la herramienta `gm` de SUBDUE detecte los vértices o aristas como iguales.

```
1  def get_matched_graph(graph, replace_map):
2    matched_graph = Graph(graph.label())
3    for vertex in graph.vertexes():
4      id = vertex.id
5      label = vertex.label
6      if label in replace_map.keys():
```

```
7         label = replace_map[label]
8
9       new_vertex = Vertex(label, id)
10      matched_graph.add_vertex(new_vertex)
11
12    for vertex in graph.vertexes():
13      for edge in vertex.edges():
14        matched_source_vertex = matched_graph.get_vertex(vertex.id)
15        matched_target_vertex = matched_graph.get_vertex(edge.get_target.id)
16        label = edge.label
17        if label in replace_map.keys():
18          label = replace_map[label]
19        matched_edge = Edge(label, matched_target_vertex)
20        matched_source_vertex.add_edge(matched_edge)
21        matched_graph.update_vertex(matched_source_vertex)
22
23    return matched_graph
```

Listado 3.11: Generación de los nuevos grafos con las etiquetas en común.

Por último, a partir de la línea 33, al igual que en la anterior versión del método `matchGraph`, se escriben los ficheros de SUBDUE con el objetivo de ser procesados y calcular el coste de transformación entre ambas subestructuras.

3.7 Conclusión

En este capítulo se ha presentado la solución de la problemática planteada. Estas solución consiste en extraer los subgrafos más frecuentes de los conjuntos de datos, con el objetivo de que estos subgrafos sirvan como síntesis del conjunto de datos completo (sección 3.1.2). Una vez extraídos estos subgrafos, se buscan similitudes entre ellos para poder realizar el emparejado correspondiente 3.5. Ya que al evaluar esta solución se detectó una importante debilidad, se incluyeron técnicas para la similitud de cadenas de caracteres y basadas en el lenguaje con el objetivo de mejorar los resultados obtenidos 3.6. En el capítulo siguiente se detalla la evaluación llevada a cabo con cada una de las alternativas desarrolladas.

Ez pentsa ezerren beharrik daukazula heltzeko hire helmugara.

Delirium Tremens

CAPÍTULO

4

Evaluación

A lo largo de este capítulo se muestran los experimentos realizados para la evaluación de la hipótesis de investigación propuesta, teniendo en cuenta la solución implementada. En primer lugar se explica el proceso realizado para la elaboración del estándar de referencia. A continuación se detalla la evaluación realizada, para finalizar mostrando y analizando los resultados obtenidos.

4.1 Estándar de referencia

El "estándar de referencia" (*gold standard*) es un concepto desarrollado en las ciencias médicas y farmacéuticas que establece, en una serie de condiciones estándar, los mejores resultados del mejor diagnóstico existente hasta el momento. A partir de la comparación con este estándar, se determina la eficacia de las nuevas técnicas de diagnóstico desarrolladas. Para evaluar el trabajo desarrollado durante esta tesis, se ha establecido un estándar de referencia para la evaluación de los conjuntos de datos candidatos a ser enlazados. Este estándar ha sido extraído de dos fuentes de datos u orígenes, tal y como se muestra a continuación.

4.1.1 The Datahub

Inspirado por Leme et al. (2013), la primera de estas fuentes son los propios enlaces externos ya existentes en los conjuntos de datos. Para extraer estos enlaces externos, se ha consultado el repositorio de datos The Datahub[1]. The Datahub es un repositorio de datos que permite a los usuarios publicar sus conjuntos de datos, así como sus respectivos puntos de acceso SPARQL, recursos de ejemplo, etc. Para la inclusión de un conjunto de datos en la LOD Cloud, el primer requisito consiste en publicar dicho conjunto en The Datahub, por lo que todos los conjuntos de datos empleados para la evaluación disponen de su respectiva entrada en este repositorio. Además, otra de las condiciones es que en la entrada correspondiente a cada conjunto de datos aparezcan los conjuntos de datos hacia los que tienen enlaces y el número de los mismos. Esto se indica a través de la propiedad `links:<id_dataset_destino>`. Esta información se ha extraído a través de la API de The Datahub, recolectando los enlaces existentes entre los diferentes conjuntos de datos empleados para la evaluación.

Además de los enlaces declarados explícitamente en The Datahub, existen multitud de enlaces que, por diversos motivos, los administradores no han incluido en sus conjuntos de datos. Algunos de estos motivos pueden ser que algunos de los conjuntos candidatos a ser enlazados hayan sido creados posteriormente y que el administrador no los haya revisado para incluir nuevos enlaces en su conjunto de datos, o simplemente, que el administrador desconociese la existencia de estos conjuntos. La ausencia de estos enlaces entre conjuntos candidatos a ser enlazados podría generar una situación en la que el sistema de recomendación desarrollado recomendase enlaces que, efectivamente, sí fuesen candidatos a ser enlazados, pero al no estar así indicado en The Datahub se tomasen como falsos positivos.

4.1.2 Consultando a los expertos

Para determinar los posibles enlaces entre conjuntos de datos no declarados en The Datahub, se planteó consultar esta cuestión con investigadores expertos en la Web

[1] `http://datahub.io/`

Semántica. Para ello, se ha desarrollado un sistema de encuestas[1] en el que se presentan una serie de parejas de conjuntos de datos. Este sistema ha sido desarrollado bajo el *framework* web Django. Además de almacenar los resultados de las encuestas realizadas, se han desarrollado diferentes métodos para ayudar a comprender los resultados obtenidos. Cada evaluador determina para cada pareja, en base al título, la descripción y otros recursos de los conjuntos de datos extraídos de The Datahub, si dicha pareja puede ser enlazada o no. En el caso de no poder determinarse por no disponer de suficiente información, el evaluador puede indicarlo a través de una tercera opción. Para disponer de diferentes opiniones, cada pareja de conjuntos ha sido evaluada por tres evaluadores diferentes calculando, después, el índice de acuerdo entre los diferentes evaluadores que tomaron parte en esta encuesta.

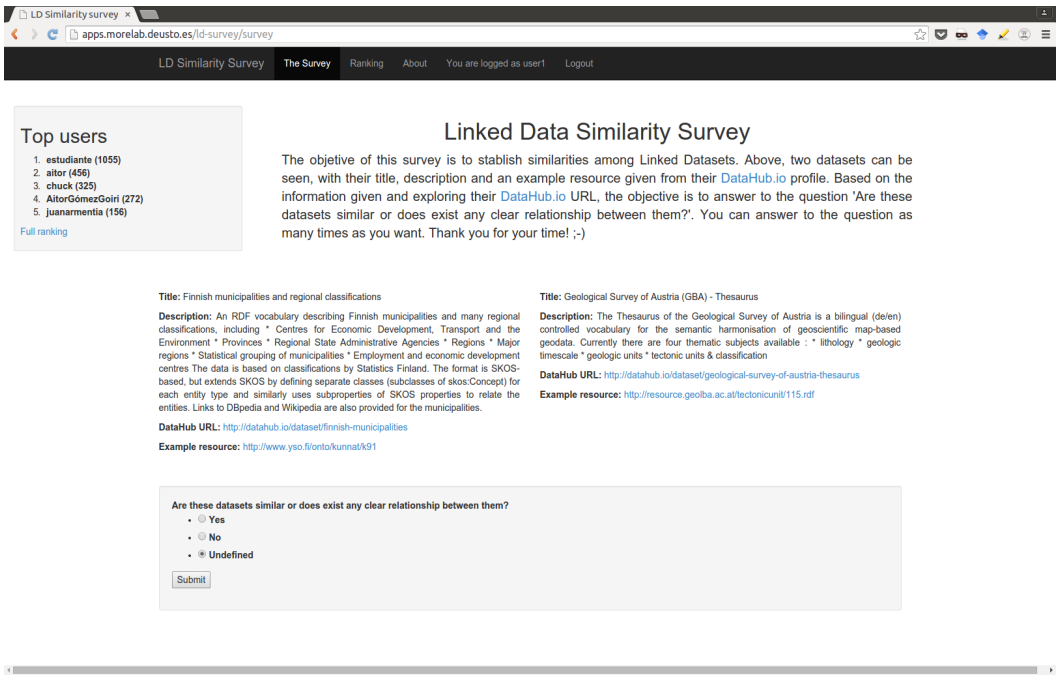


Figura 4.1: Sistema de encuestas a través del cual se consultó a los expertos.

Este planteamiento lanza una nueva problemática. Tal y como puede observarse en el cuadro A.1 del apéndice A, el número de conjuntos de datos empleados para la evaluación asciende a 69, por lo que las combinaciones de dos elementos posibles ascenderían a $\binom{69}{2} = 2346$ combinaciones. Teniendo en cuenta que se tomó la

[1] `https://github.com/memaldi/ld-similarity-survey/`

decisión de que cada pareja de conjuntos de datos debía ser evaluada por tres evaluadores diferentes, el número total de evaluaciones a realizar ascendería a 7038. Al considerar que este número de evaluaciones era demasiado alto como para ser completado por los expertos seleccionados, se planteó la reducción del número de parejas de conjuntos de datos a evaluar.

Para reducir el número de evaluaciones a realizar, se ha partido de la siguiente evidencia, planteada en Fetahu et al. (2014). Fetahu et al. consideran que si los conjuntos de datos conectados con los conjuntos $\{A, B, C\}$ están también conectados con el conjunto D, los conjuntos $\{A, B, C\}$ podrían estar relacionados también con el conjunto D. Este concepto se puede ver representado en la figura 4.2.

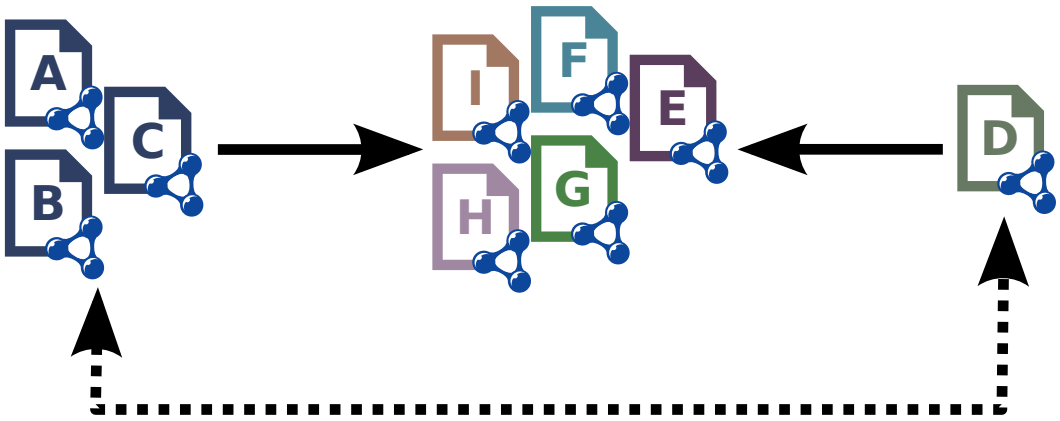


Figura 4.2: Representación gráfica de la evidencia planteada por Fetahu et al. Si la proporción de conjuntos enlazados compartidos por los conjuntos $\{A, B, C\}$ y el conjunto D fuera alta, podrían existir enlaces entre $\{A, B, C\}$ y D.

Partiendo de esta evidencia, solamente se evaluó la relación entre los conjuntos que tuviesen enlaces con al menos un conjunto de datos en común, reduciendo, así, el número de parejas a evaluar a 198, cada una de ellas por tres revisores diferentes.

4.1.2.1 **Resultados**

Una vez realizada la encuesta, se ha calculado el índice de acuerdo entre los evaluadores a través del coeficiente Kappa de Fleiss (Fleiss, 1971). El coeficiente Kappa de Fleiss permite calcular el índice de acuerdo entre un número fijo de evaluadores a la hora de asignar puntuaciones discretas. Se diferencia del coeficiente de Kappa

de Cohen (Cohen, 1960) en que este último solamente permite calcular el índice de acuerdo entre dos evaluadores, mientras el de Fleiss permite que existan multiples evaluadores y que todos los evaluadores no tengan que haber evaluado todos los sujetos de estudio. El cálculo de este coeficiente revela un acuerdo del 41 % entre los evaluadores, lo que según Landis y Koch supone un acuerdo moderado. Cabe remarcar que no existe un criterio unificado que establezca si una pareja de conjuntos de datos deba ser enlazada o no, por lo que la decisión tomada por los expertos es puramente objetiva y basada en su experiencia.
En la figura 4.3 se puede observar el índice de acuerdo entre los diferentes evaluadores según el coeficiente Kappa de Cohen. A través de este índice se puede observar el acuerdo existente entre las diferentes parejas de evaluadores.

4.1.3 Estándar de referencia

Una vez realizada la consulta a los expertos, el estándar de referencia se formó a partir de *a*) los enlaces establecidos en The Datahub y *b*) aquellas parejas de conjuntos de datos en los que al menos dos de los tres revisores estuviesen de acuerdo en la posibilidad de crear un enlace entre ellas, siempre y cuando no entrasen en conflicto con lo establecido en The Datahub.

4.2 Soluciones de referencia

Las soluciones de referencia son sencillas implementaciones que pretenden dar una solución básica al problema planteado. Estas soluciones de referencia tienen como objetivo marcar una línea de base cuyos resultados pretende mejorar la nueva solución planteada. Dentro de la evaluación de la solución desarrollada se han implementado tres soluciones de referencia diferentes.

4.2.1 Comparación de espacios de nombres de ontologías

Esta solución pretende encontrar conjuntos de datos candidatos a ser enlazados a través de la comparación de las ontologías empleadas para la descripción de dichos conjuntos de datos. La evidencia detrás de esta solución de referencia consiste en que cuantas más ontologías en común tengan dos conjuntos de datos,

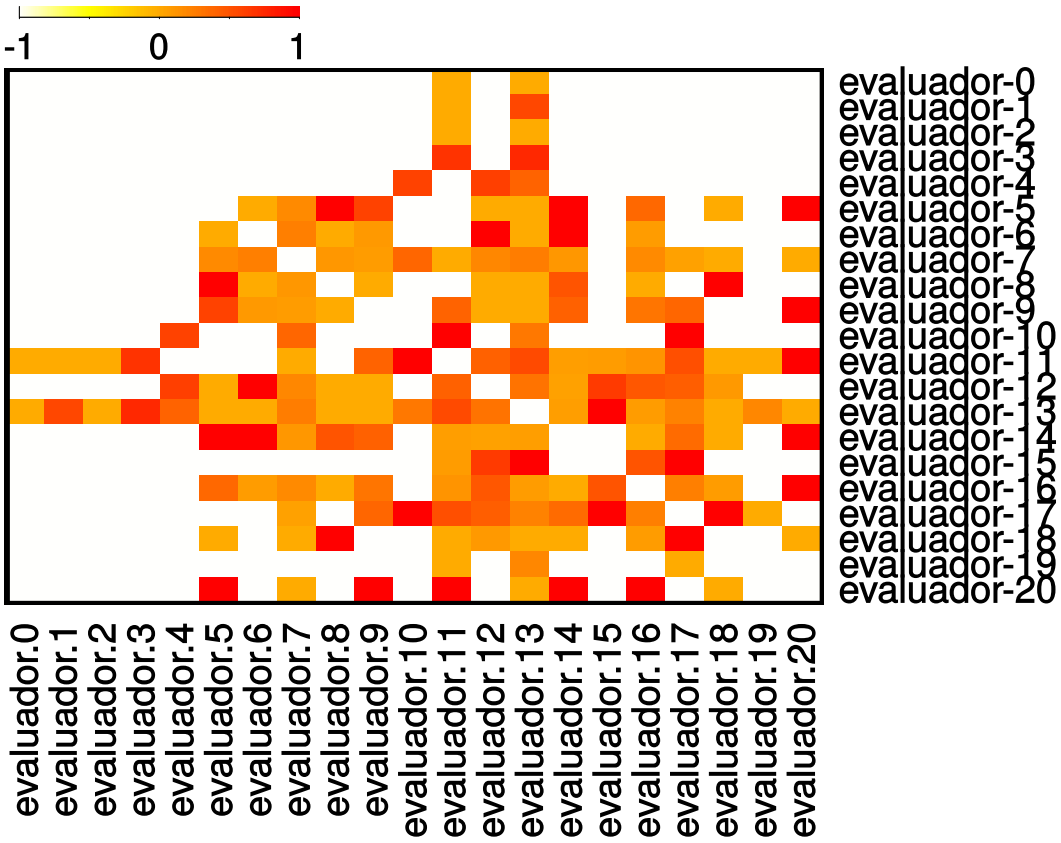


Figura 4.3: Índice de acuerdo entre los evaluadores calculado según el coeficiente Kappa de Cohen, donde el valor 0 representa ningún acuerdo y 1 total acuerdo entre los evaluadores. Un índice de acuerdo de -1 indica que esa pareja de evaluadores no ha evaluado ninguna pareja de conjuntos de datos común.

existen más posibilidades de poder ser enlazados. Siendo N el conjunto de ontologías empleadas para describir un conjunto de datos D, se calcula la puntuación ($score \in [0 \dots 1]$) entre diferentes conjuntos de la siguiente manera:

$$score(D_1, D_2) = \frac{N_1 \cap N_2}{max(|N_1|, |N_2|)} \tag{4.1}$$

4.2.2 Ranking de uso de ontologías

De manera similar a la solución anterior, esta solución de referencia toma en cuenta las ontologías empleadas para describir los conjuntos de datos con la diferencia de que crea un *ranking* en función del porcentaje de uso de las clases y propiedades de una ontología dentro de cada conjunto de datos. Una vez creados los rankings para cada conjunto de datos, calcula la distancia entre estos rankings con el objetivo de calcular la afinidad entre dichos conjuntos. La distancia empleada es la distancia Tau de Kendall (Kendall, 1938). Esta distancia se calcula contando las veces en las que los valores del primer ranking se encuentran en orden inverso en el segundo ranking. Siendo τ_1 y τ_2 la pareja de rankings a comparar, la distancia Tau de Kendall se calcula de la siguiente manera:

$$K(\tau_1, \tau_2) = \sum_{i,j \in P} \bar{K}_{i,j}(\tau_1, \tau_2) \tag{4.2}$$

siendo $\bar{K}_{i,j}(\tau_1, \tau_2) = 0$ si los elementos i y j están en el mismo orden en las ambos rankings y $\bar{K}_{i,j}(\tau_1, \tau_2) = 1$ si están en orden diferente. Esta distancia tau de Kendall se ha normalizado de la siguiente manera:

$$K_{normalized}(\tau_1, \tau_2) = \frac{K(\tau_1, \tau_2)}{n(n-1)/2} \tag{4.3}$$

siendo n el número de elementos del ranking. En el caso de que uno de los rankings tenga más elementos que el otro, se rellenará el ranking menor con los elementos que falten, estableciendo el porcentaje de uso de estos elementos a 0.

4.2.3 Tripletas en común

Esta solución de referencia calcula la afinidad entre dos conjuntos de datos en función del porcentaje de tripletas en común entre esos dos conjuntos. Ya que los sujetos de las tripletas varían en función del espacio de nombres del conjunto de datos, solamente se comparan los predicados y los objetos de la tripleta. La afinidad entre conjuntos de datos se calcula a través de la distancia de Jaccard, siendo T_1 y T_2 el conjunto de tripletas de los conjuntos de datos a comparar:

$$d_J(T_1, T_2) = \frac{|T_1 \cup T_2| - |T_1 \cap T_2|}{|T_1 \cup T_2|} \tag{4.4}$$

4.3 Resultados

En esta sección se muestran los resultados obtenidos tras la evaluación del sistema, en los siguientes términos:

- **Precisión:** es el ratio de verdaderos positivos respecto al total de verdaderos y falsos positivos. En este contexto, los verdaderos positivos son los candidatos a ser enlazados recomendados por el sistema que según el "gold standard", efectivamente, son candidatos a ser enlazados; los falsos positivos son los candidatos a ser enlazados recomendados por el sistema que según el "gold standard" no son candidatos a ser enlazados. Se calcula de la siguiente manera:

$$Precisión = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ positivos} \tag{4.5}$$

- **Exhaustividad:** es el ratio de verdaderos positivos respecto al total de verdaderos positivos y falsos negativos. En el contexto de esta tesis se refiere al número de candidatos a ser enlazados recomendados respecto al número total de candidatos que deberían haber sido recomendados según el "gold standard".

$$Exhaustividad = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ negativos} \tag{4.6}$$

- **Valor-f:** representa la media armónica entre la precisión y la exhaustividad:

$$Valor\text{–}f = 2 * \frac{Precisión * Exhaustividad}{Precisión + Exhaustividad} \tag{4.7}$$

- **Exactitud:** a través de la exactitud se calcula lo cercano que está el resultado obtenido respecto al resultado real, teniendo en cuenta tanto el número de verdaderos positivos como de verdaderos negativos:

$$Exactitud = \frac{vp + vn}{vp + vn + fp + fn} \tag{4.8}$$

Para ello, en primer lugar se muestran los resultados de la ejecución de la solución sin aplicar las técnicas de similitud de cadenas de caracteres. Después, se muestran los resultados tras la ejecución de la solución aplicando las técnicas de similitud de cadenas de caracteres y se comparan con los resultados anteriores. A continuación, tras reflexionar sobre los resultados obtenidos, se comparan los resultados de la mejor de las técnicas empleados con los resultados obtenidos por las soluciones de referencia. Por último, se analizan todos estos resultados.

4.3.1 Solución propuesta

En la figura 4.4 pueden observarse los resultados obtenidos tras la aplicación de la solución planteada sin técnicas de similitud de caracteres. Como puede observarse, la precisión obtenida supera un valor de 0.8 a partir de un umbral de 0.4, llegando a alcanzar una precisión de 0.86. Por su parte, el valor máximo de la exhaustividad se sitúa en torno a un valor de 0.51, decayendo a partir de un umbral de 0.3. Esto se debe a que cuanto mayor es el umbral, el grado de similitud necesario para que dos conjuntos de datos sean considerados como candidatos a ser enlazados es mayor, por lo que existen candidatos, que a pesar de estar relacionados según el "golden standard", el grado de similitud resultante no es lo suficientemente alto. Estos resultados se pueden traducir en que, las recomendaciones de conjuntos de datos realizadas por la solución planteada son válidas en un alto porcentaje de ocasiones (un número muy bajo de falsos positivos), aunque existen muchos conjuntos de datos que realmente son candidatos a ser enlazados pero que la solución planteada omite (falsos negativos). En términos de exactitud, se puede observar que la proporción entre los verdaderos positivos y los verdaderos negativos es alta.

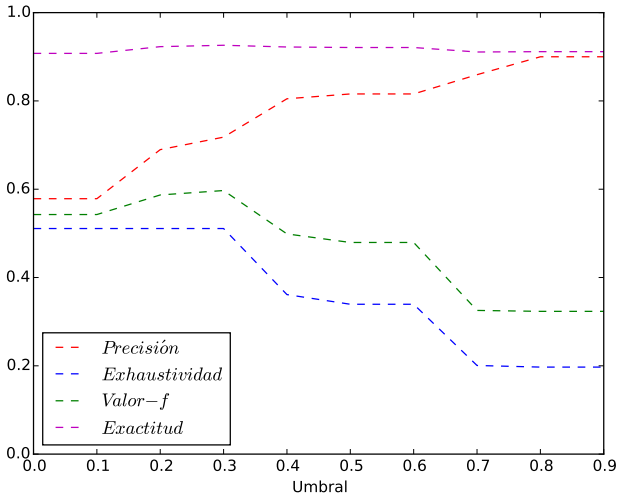


Figura 4.4: Resultados obtenidos a partir de la aplicación de la solución propuesta sin aplicar técnicas de similitud de cadenas de caracteres.

4.3.1.1 Similitud de cadenas de caracteres

A continuación, se ha evaluado la aplicación de las técnicas de similitud de caracteres presentadas en la sección 3.6 tanto por separado como a través del cálculo de la media de todas ellas. Esta evaluación muestra dos parámetros diferentes: por una parte, el umbral a partir del cual se determina que dos subgrafos son candidatos a ser emparejados (referido en las gráficas como "Umbral"); y por otra parte, el umbral que determina que dos cadenas de caracteres representan el mismo concepto (referido como "Umbral de cadena de caracteres"). En la figura 4.5 pueden observarse los resultados obtenidos tras el cálculo de la **similitud de cadenas de caracteres**. Como puede observarse, la precisión resulta ser algo inferior en los umbrales más bajos, pero al llegar a 0.4 se establece en niveles superiores a 0.8. En cuanto a la exhaustividad, a pesar de que en umbrales inferiores se obtienen levemente superiores (en torno a 0.55), a medida que el umbral crece la exhaustividad decae de la misma manera que en la solución anterior. Respecto al valor-f, el mejor resultado ronda un valor de 0.58, alcanzando la precisión un valor de 0.63 y la exhaustividad un valor de 0.54. Los resultados obtenidos a través de la aplicación de esta técnica no son muy diferentes a los anteriores ya que esta técnica requiere que la parte local de las URI que representan las propiedades o clases de los subgrafos

sean exactamente iguales, situación que no ocurre frecuentemente.

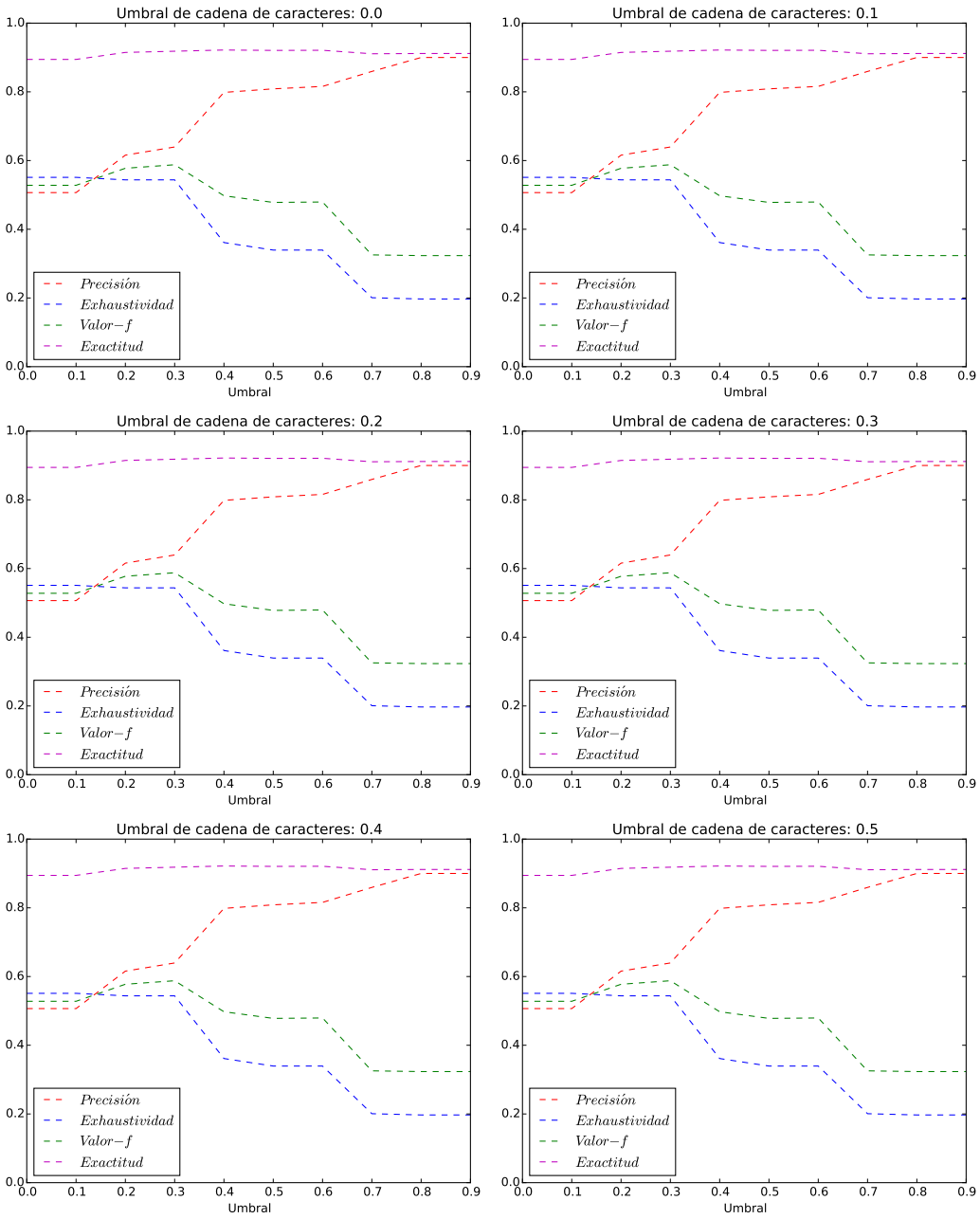


Figura 4.5: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la similitud de cadenas de caracteres.

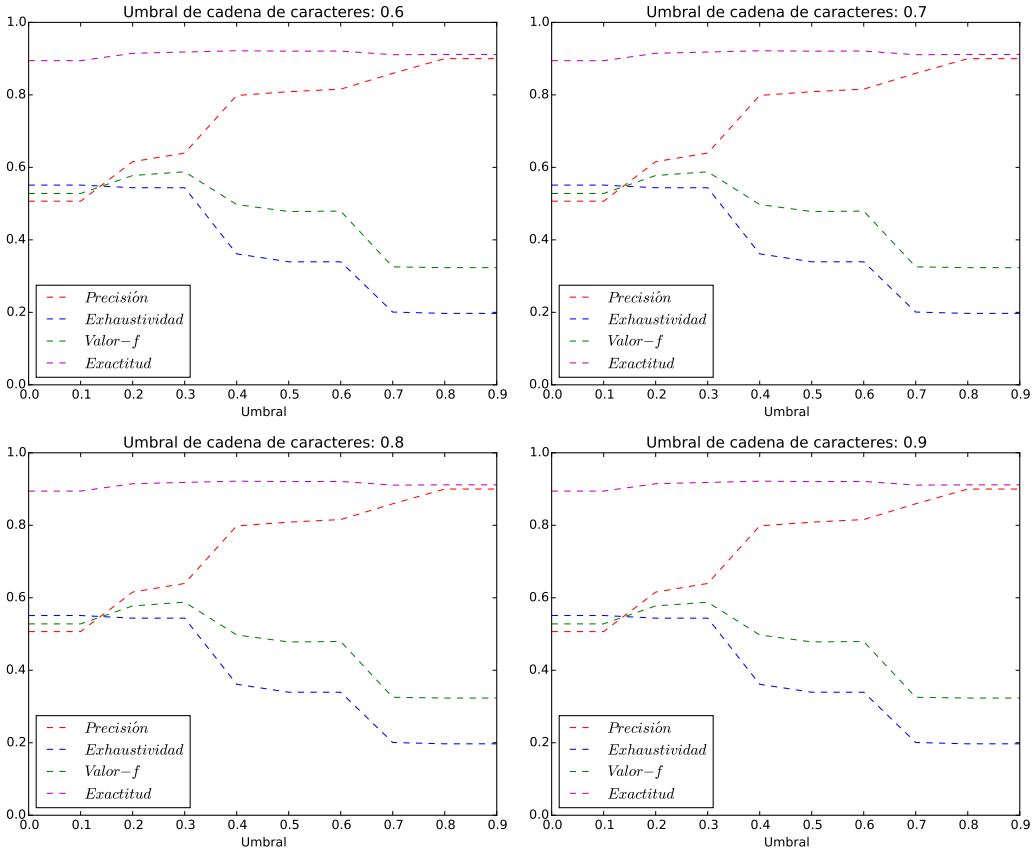


Figura 4.5: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la similitud de cadenas de caracteres (continuación).

4.3.1.2 Cálculo de similitud entre subcadenas

A continuación, en la figura 4.6 se muestran los resultados obtenidos tras aplicar el **cálculo de similitud entre subcadenas**. Como puede observarse, la variación de los resultados respecto a la primera solución planteada es mayor debido a que con esta técnica, al contrario que con el cálculo de similitud entre cadenas de caracteres, las cadenas no tienen por qué ser exactamente iguales para que se consideren similares (con un fragmento basta, cuanto más mayor es este fragmento mayor similitud), por lo que aparecen más conjuntos candidatos a ser recomendados. En general, la exhaustividad parte de valores más elevados a pesar de decaer a niveles similares a los anteriores. Esto implica que la precisión arranca desde valores in-

feriores aunque, de la misma manera, llega a alcanzar valores superiores al 0.8, al igual que en resultados previos. En el mejor valor-f, 0.63, la precisión alcanza un valor de 0.68 y la exhaustividad 0.59.

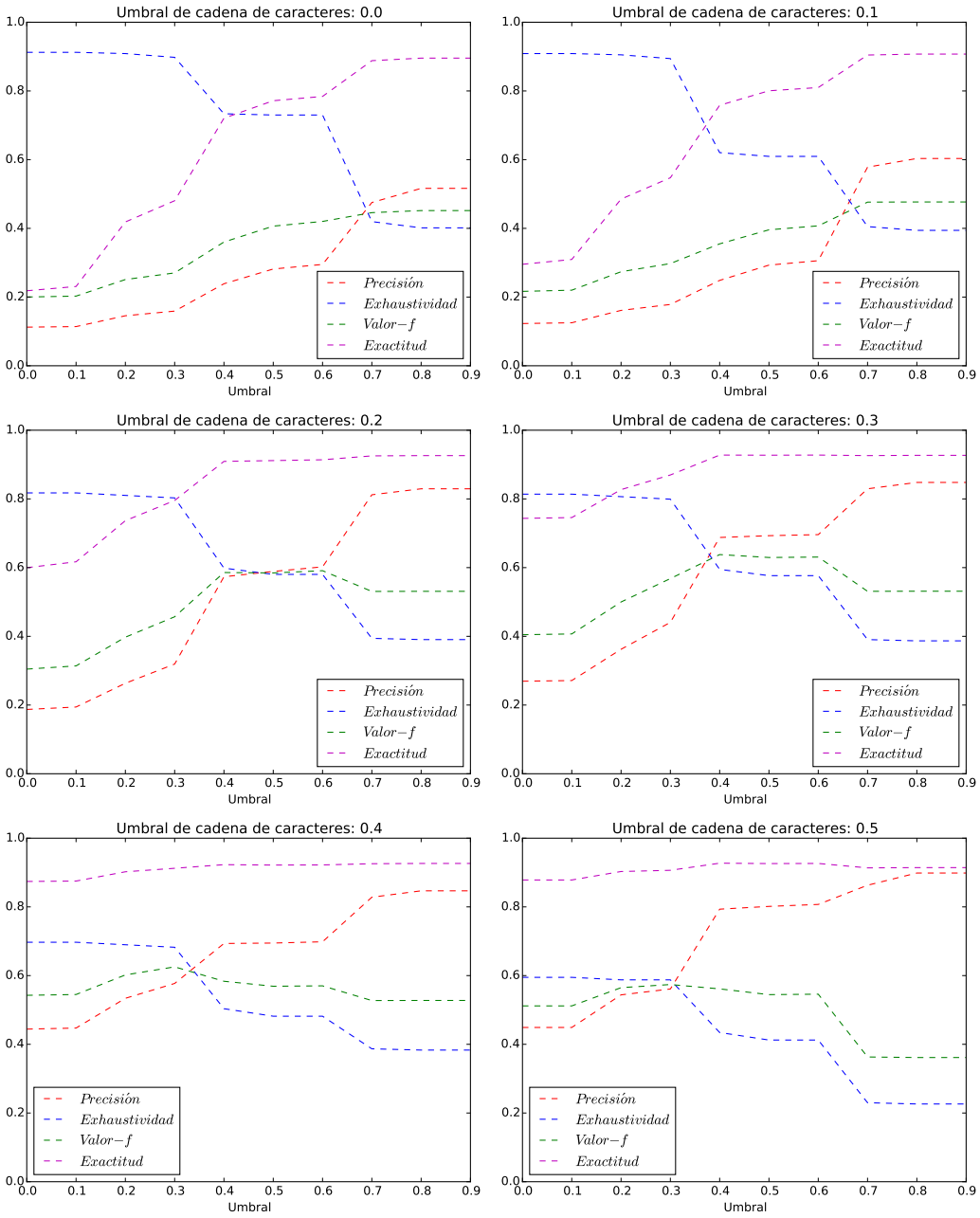


Figura 4.6: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la similitud de subcadenas.

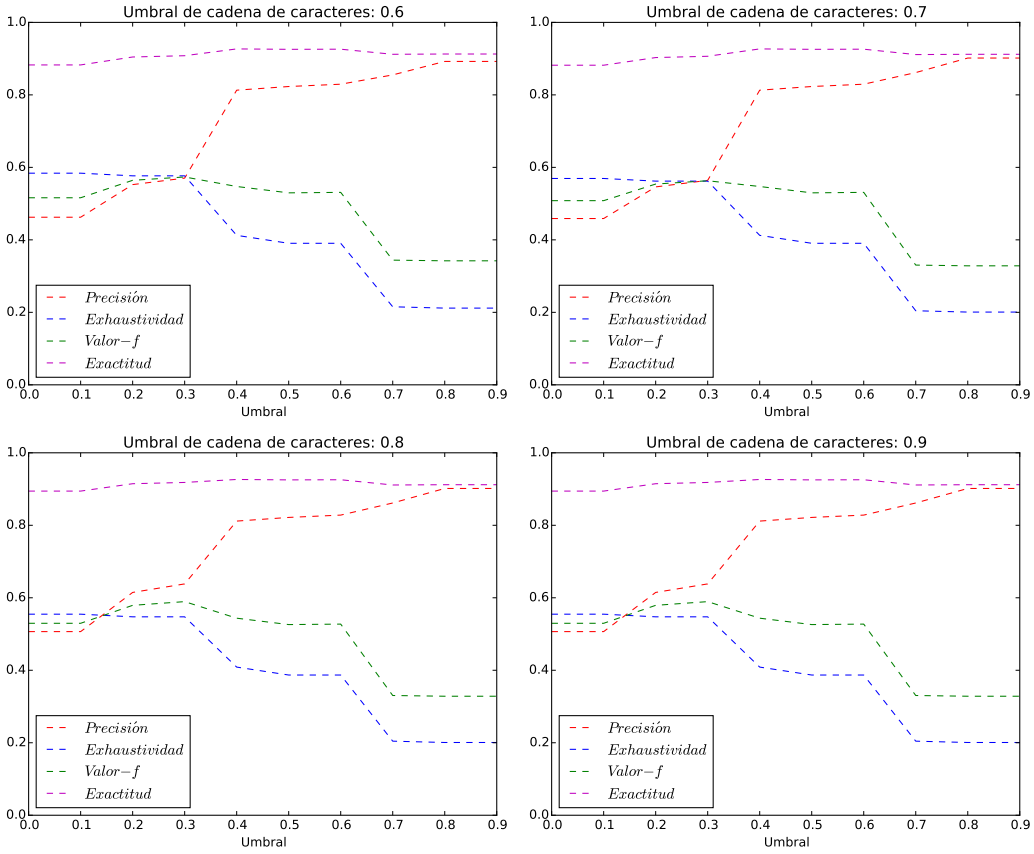


Figura 4.6: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la similitud de subcadenas (continuación).

4.3.1.3 Distancia de Levenshtein

Por su parte, en la figura 4.7 se muestran los resultados obtenidos tras la aplicación de la **distancia de Levenshtein** a la hora de comparar las cadenas de caracteres. Este método, como se podrá observar más adelante, es el que mejor valor-f logra. Esto se debe mayormente a que con esta técnica, que calcula la distancia de edición entre dos cadenas de caracteres, las cadenas no tienen por qué ser exactamente iguales o compartir un fragmento exactamente igual, siendo más flexible a la hora de determinar la similitud entre las mismas, por lo que la exhaustividad alcanza valores mayores. El valor-f máximo (0.67) se consigue en el con un umbral de cadena de caracteres de 0.4. En este punto, la precisión alcanza un valor de 0.59 y

la exhaustividad un 0.78.

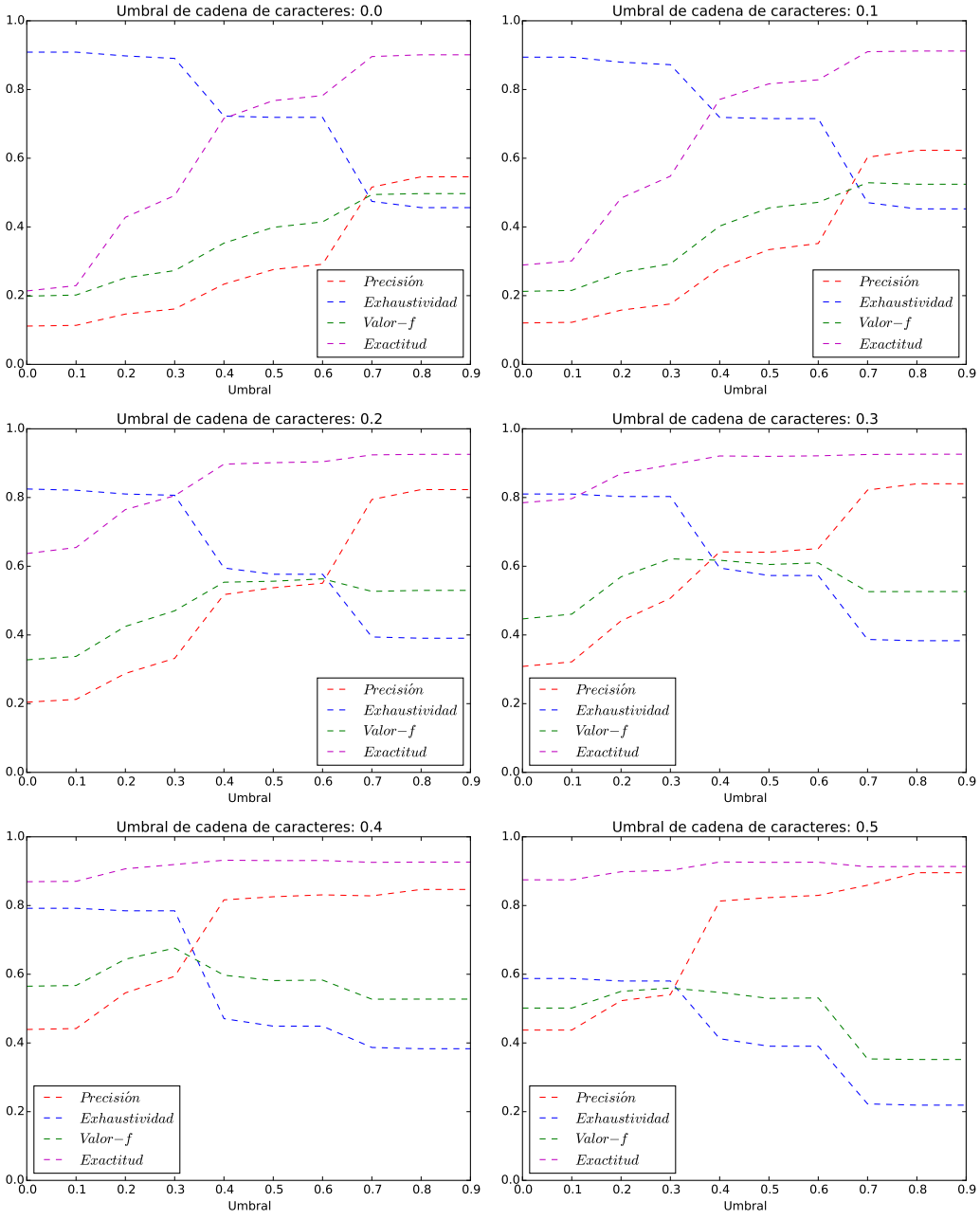


Figura 4.7: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la distancia de Levenshtein.

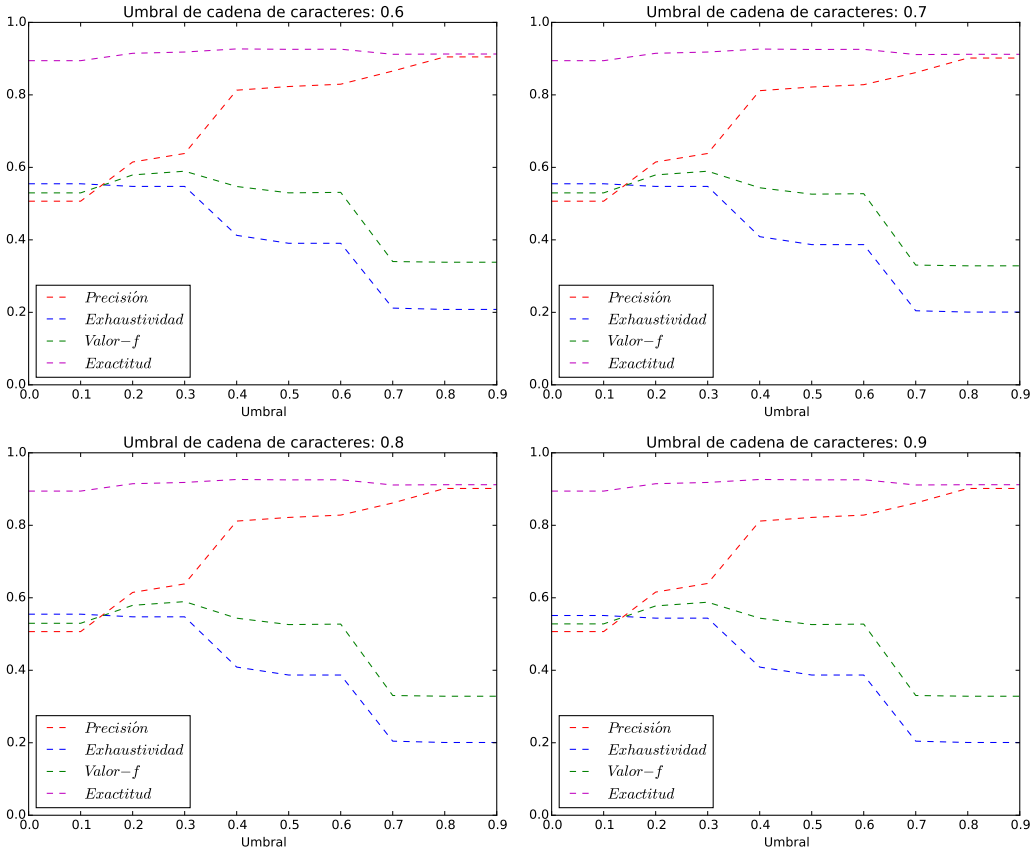


Figura 4.7: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la distancia de Levenshtein (continuación).

4.3.1.4 **Distancia SMOA**

En la figura 4.8 se muestran los resultados obtenidos tras la aplicación de la **distancia SMOA**. Los resultados obtenidos a través de esta técnica son similares a los obtenidos tras la aplicación de la similitud de subcadenas: exhaustividad por encima de 0.8 que se ve reducida a medida que se incrementan los umbrales a la vez que la precisión crece hasta superar un valor de 0.8. Esta técnica, al igual que la distancia de Levenshtein, ofrece mayor flexibilidad a la hora de de comparar cadenas, por lo que un número mayor de conjuntos de datos son seleccionados como candidatos. Por el contrario, esta flexibilidad penaliza la precisión, ya que el número de conjuntos de datos seleccionados que no son candidatos válidos según

el “gold standard” es mayor que con otras técnicas. En el mejor valor-f (0.64) la exhaustividad se sitúa en 0.8 mientras que la precisión se sitúa en 0.54. La aplicación de esta técnica incrementa la exhaustividad en un 0.02 respecto a la distancia de Levenshtein, pero sacrifica un 0.05 de precisión aproximadamente.

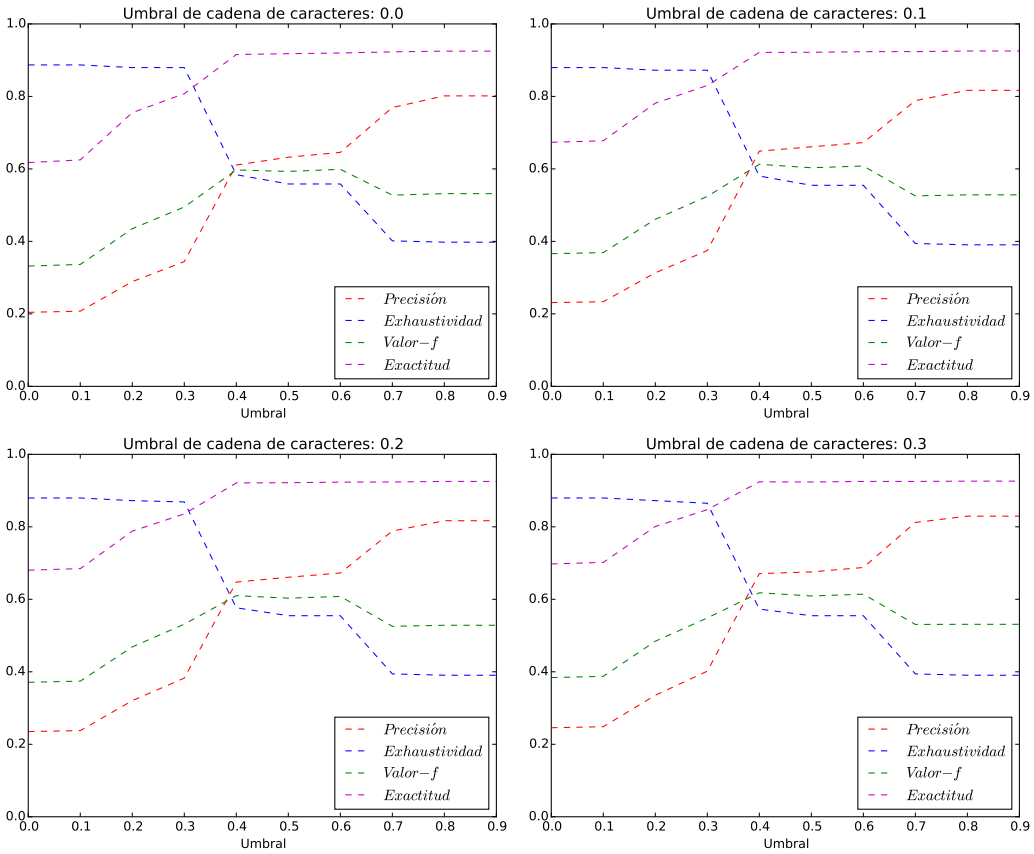


Figura 4.8: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la distancia SMOA.

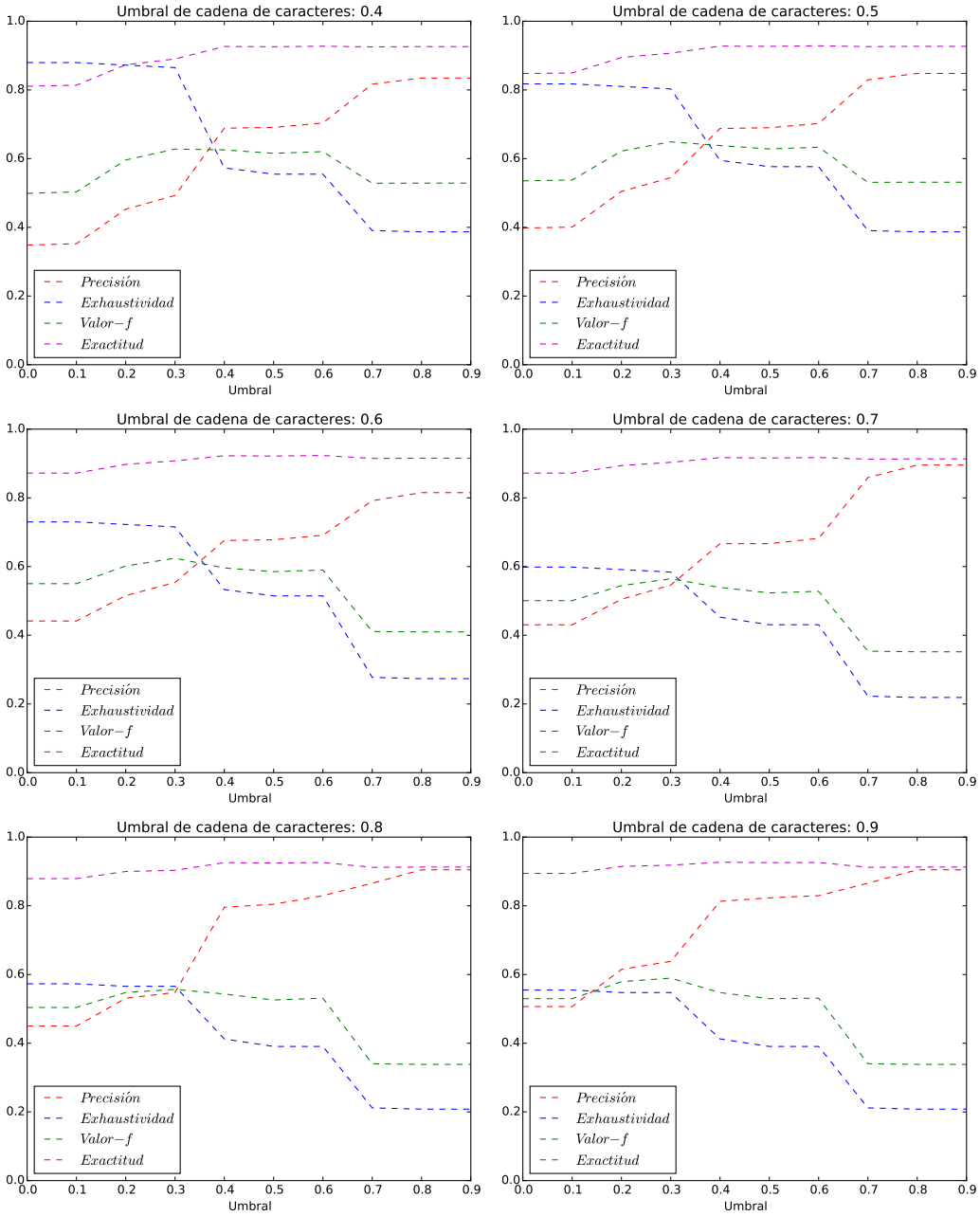


Figura 4.8: Resultados obtenidos tras la aplicación de la técnica basada en el cálculo de la distancia SMOA (continuación).

4.3.1.5 Distancia de sinonimia básica

En lo que respecta a la técnica basada en el lenguaje empleada, en este caso la **distancia de sinonimia básica**, tal y como puede observarse en la figura 4.9, se consiguen muy buenos resultados con umbrales bajos que van decayendo a medida que se aumenta el valor de dichos umbrales. El máximo valor-f es similar al obtenido tras la aplicación de la distancia de Levenshtein (0.67). Sin embargo, en este mejor valor-f, la aplicación de esta última técnica permite lograr una precisión superior en un 0.06 (0.65), sacrificando la exhaustividad en un 0.09 (0.69). Como puede observarse, los resultados obtenidos por esta técnica no destacan respecto a los obtenidos con el resto de técnicas. Sin embargo, existen otras técnicas basadas en el lenguaje más avanzadas como la cosinonimia (*Cosynonymy*) o la similitud semántica de Resnik, a través de las cuales podrían obtenerse mejores resultados.

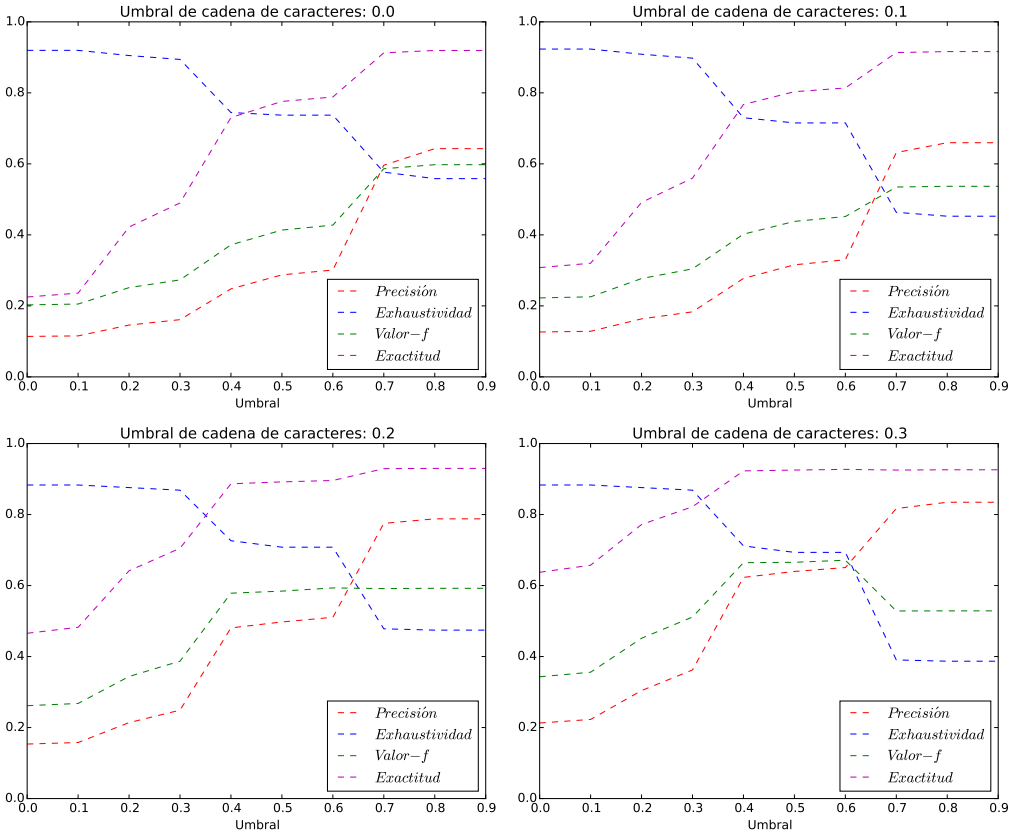


Figura 4.9: Resultados obtenidos tras la aplicación de la técnica basada en la sinonimia básica.

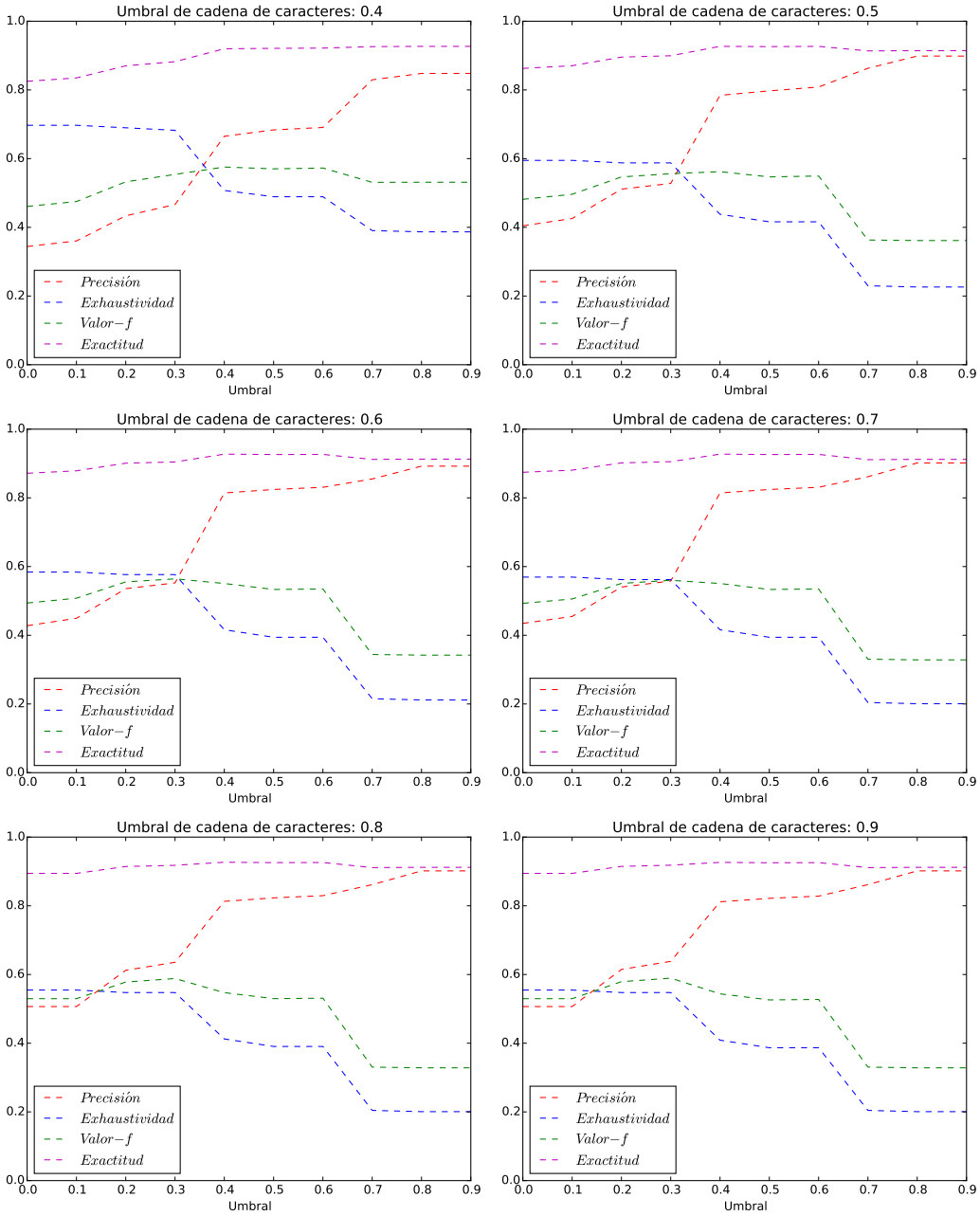


Figura 4.9: Resultados obtenidos tras la aplicación de la técnica basada en la sinonimia básica (continuación).

4.3.1.6 Media geométrica

Una vez evaluados los resultados obtenidos por las diferentes técnicas de búsqueda de similitudes entre cadenas de caracteres, se ha calculado la media geométrica resultante de la aplicación de todas las técnicas anteriores. En esta media geométrica se ve la tendencia general de todos los resultados: exhaustividad superior a 0.8 en los umbrales inferiores y precisión por encima de 0.80 en los superiores. El mejor valor-f es de 0.65, en el que la precisión alcanza un valor de 0.55 y la exhaustividad un valor de 0.79. Tal y como se habla en el capítulo 5, dentro del trabajo futuro se pretende sustituir esta media por una media ponderada, asignando diferentes pesos a cada una de las técnicas. Para seleccionar los pesos más adecuados se emplearán técnicas de regularización, al igual que en los sistemas de aprendizaje automático.

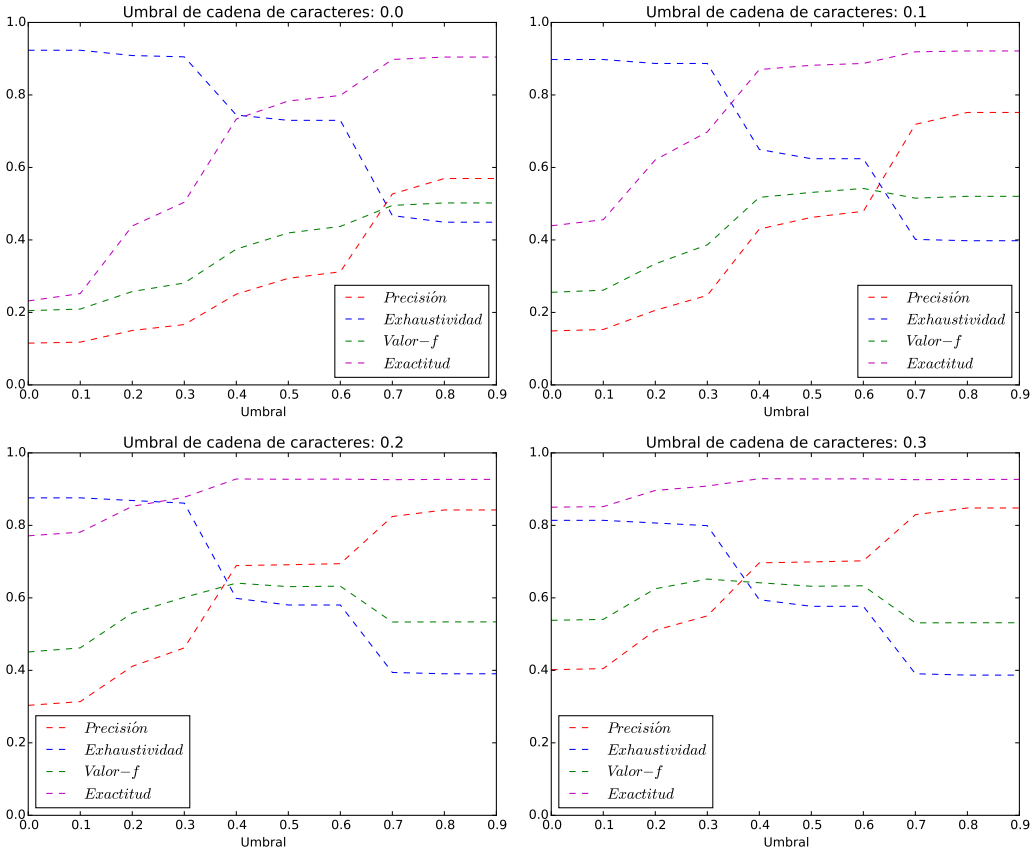


Figura 4.10: Resultados resultantes del cálculo de la media geométrica a partir de los resultados anteriores.

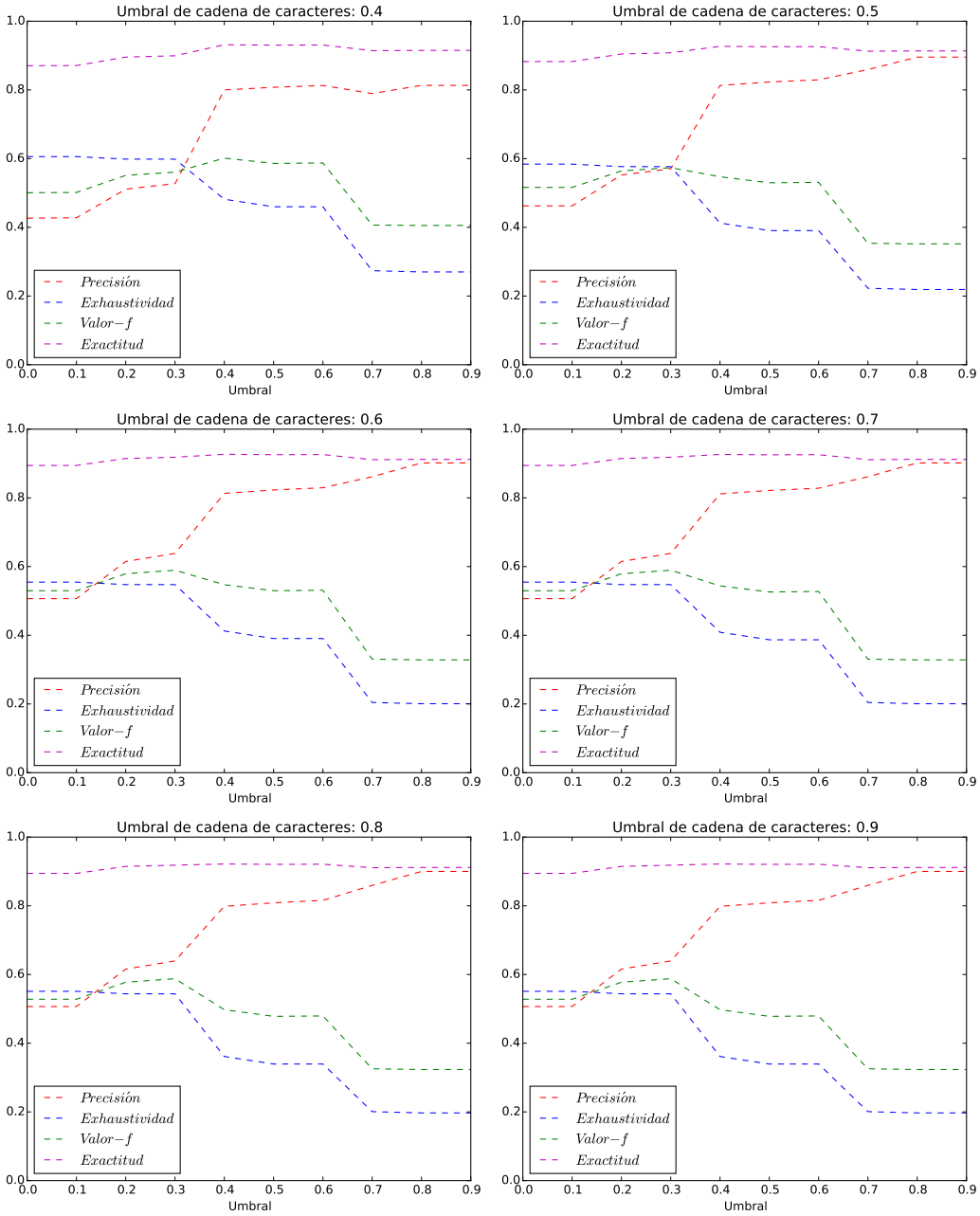


Figura 4.10: Resultados resultantes del cálculo de la media geométrica a partir de los resultados anteriores (continuación).

4.3.1.7 **Resultados**

En el cuadro 4.1 se muestran los mejores resultados obtenidos por cada técnica de cálculo de similitudes entre cadenas de caracteres. Como puede observarse, el mejor valor-f se logra a través de la distancia de Levenshtein, por lo que este resultado será tomado como referencia a la hora de comparar esta solución con la primera solución planteada, en la que no se aplicaban estas técnicas de cálculo de similitudes entre cadenas de caracteres.

Técnica	Valor-f	Precisión	Exhaustividad
Similitud de cadenas	0.58	0.64	0.54
Similitud de subcadenas	0.63	0.68	0.54
Distancia de edición (Levenhstein)	0.676	0.59	0.78
Distancia SMOA	0.64	0.54	0.80
Sinonimia básica	0.671	0.65	0.69
Media geométrica	0.65	0.55	0.79

Cuadro 4.1: Resumen de los resultados obtenidos tras realizar la evaluación aplicando técnicas de cálculo de similitudes entre cadenas de caracteres.

A continuación se muestra la comparativa entre los resultados obtenidos por la solución planteada sin aplicar técnicas de similitud de caracteres y aplicando la distancia de Levenshtein. En la figura 4.11 puede observarse la comparativa entre la precisión, la exhaustividad, el valor-f y la exactitud. Como puede observarse, la solución inicial supera levemente a la solución que aplica la distancia de Levenhstein en términos de precisión. En cambio, esta última solución representa una mejora considerable, de casi un 20 % respecto a la solución inicial en términos de exhaustividad. De la misma manera, los valores-f de la solución que aplica la distancia de Levenshtein son considerablemente superiores. Por parte de la exactitud, a pesar de que en umbrales inferiores la solución inicial obtiene mejores valores, a partir de un umbral de 0.4 la solución que aplica la distancia de Levenshtein obtiene mejores resultados.

En conclusión, en vista de los resultados obtenidos, se podría decir que efectivamente, y tal como se había planteado, **las técnicas de comparación de cadenas de**

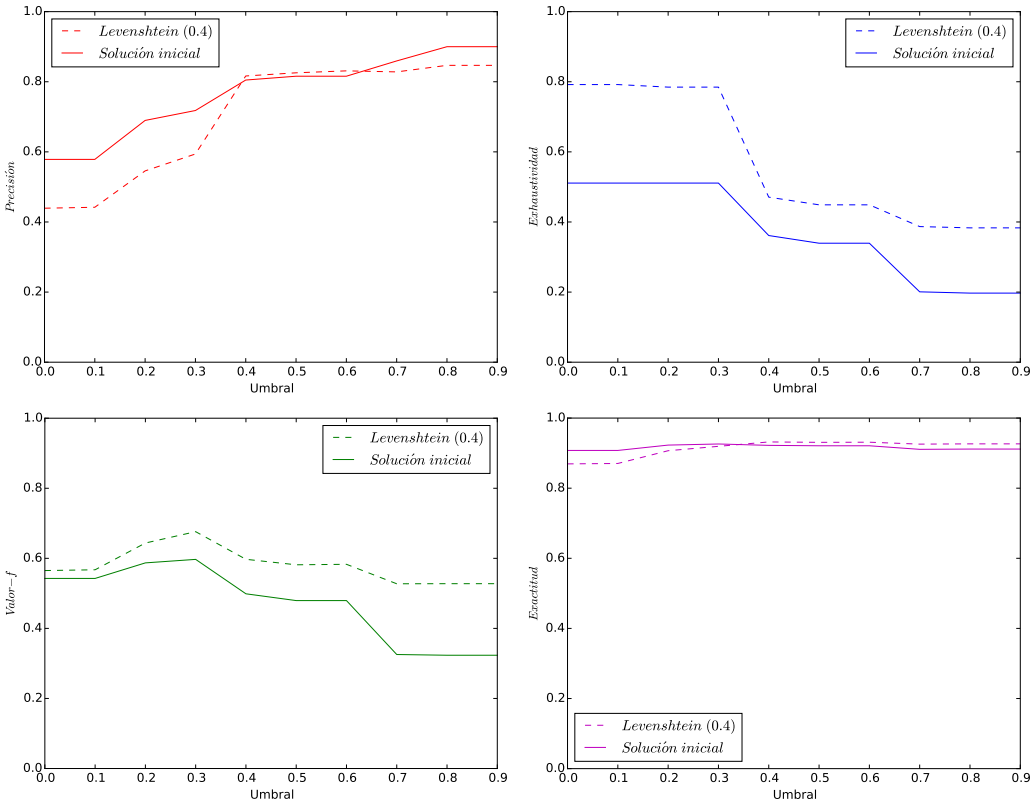


Figura 4.11: Comparativa entre la precisión, la exhaustividad, el valor-f y la exactitud de las dos soluciones planteadas.

caracteres contribuyen positivamente en la tarea de la recomendación de conjuntos de datos candidatos a ser enlazados. En la siguiente sección, se analizan los resultados de la evaluación de las soluciones de referencia planteadas.

4.3.2 Soluciones de referencia

A lo largo de esta sección se muestran los resultados obtenidos por las tres soluciones de referencia planteadas en la sección 4.2 en comparación con los resultados obtenidos por la solución planteada en esta tesis doctoral. En la figura 4.12 pueden observarse los resultados obtenidos en términos de precisión, exhaustividad, valor-f y exactitud.

Como puede observarse, la solución planteada logra buenos resultados en lo que a precisión respecta, a pesar de que la exhaustividad no es tan buena como se podría

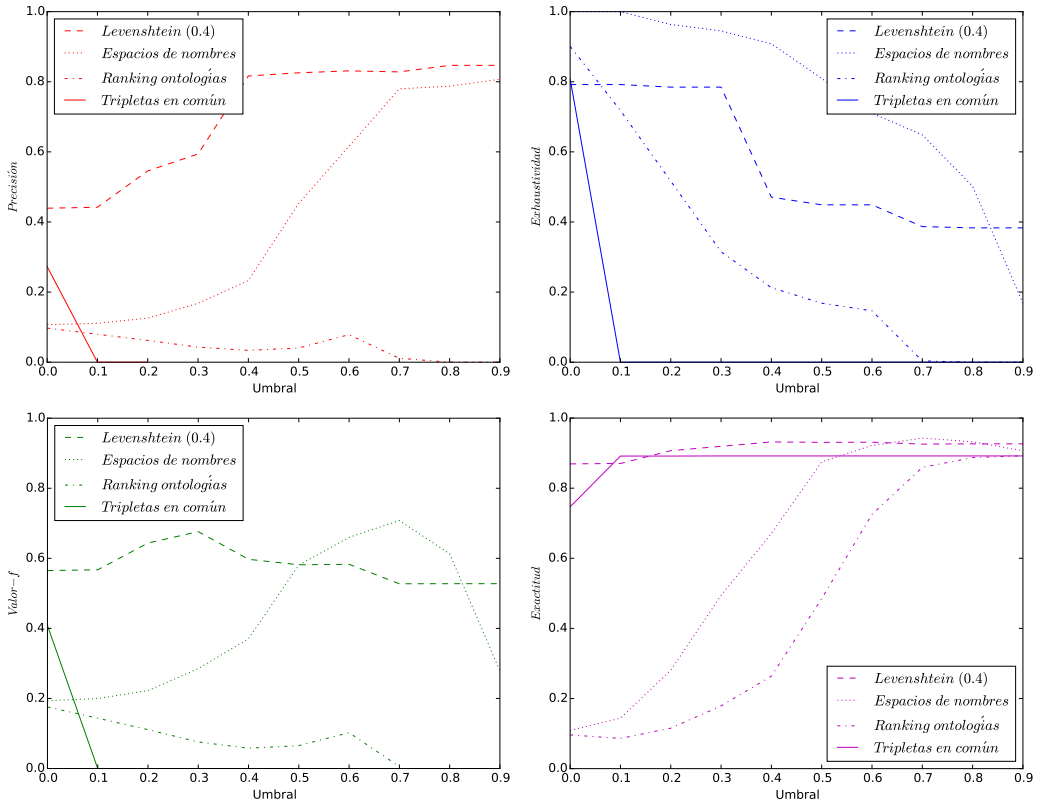


Figura 4.12: Comparación de los resultados obtenidos por las soluciones de referencia y la solución propuesta.

desear, viéndose superada por la solución de referencia basada en los espacios de nombres en común. Respecto al valor-f, se puede observar que la solución propuesta logra un adecuado equilibrio entre la precisión y la exhaustividad, a pesar de verse superada con los umbrales más altos por la solución de referencia basada en los espacios de nombres en común.

Sin embargo, a pesar de que la solución de referencia basada en los espacios de nombres en común muestre, en ocasiones, mejores resultados que la solución desarrollada durante esta tesis doctoral, esto se debe a una situación relacionada con los conjuntos de datos disponibles en la Linked Open Data Cloud y su evolución. Tal y como puede observarse en el apéndice A, un gran número de conjuntos de datos son los publicados a través de la plataforma RKB Explorer (Glaser y Millard,

2007), desarrollada dentro del proyecto ReSIST[1]. Los conjuntos de datos publicados por esta plataforma, relacionados con el mundo académico y de investigación, han sido creados a través de los mismos procesos, metodologías e incluso personas, por lo que todos comparten los mismos espacios de nombres. Este hecho ocasiona que los resultados obtenidos por la solución de referencia basada en los espacios de nombres en común logre mejores resultados que los esperados. Sin embargo, en la figura 4.13 se puede observar el resultado tanto de las soluciones de referencia como de la solución desarrollada durante esta tesis doctoral cuando se excluyen los conjuntos de datos de la plataforma RKB Explorer. Como puede observarse, a pesar de que la exhaustividad se mantiene algo por debajo, la precisión de los resultados de la solución propuesta queda muy por encima de la de los resultados de la solución de referencia basada en los espacios de nombres en común. El resultado conjunto de la precisión y la exhaustividad puede verse en el valor-f, en el que la solución propuesta queda muy por encima de las soluciones de referencia para la mayoría de umbrales. En las conclusiones expuestas en el capítulo 5 se reflexiona acerca de los conjuntos de datos empleados para la evaluación y la LOD Cloud.

4.4 Conclusión

A lo largo de este capítulo se han presentado los resultados obtenidos tras evaluar la solución planteada en sus dos variantes: tanto aplicando técnicas de similitud de cadenas de caracteres como sin aplicarlas. Para ello, en primer lugar se ha presentado el estándar de referencia empleado para realizar dicha evaluación, además de presentar el proceso a través del cual se ha obtenido este estándar de referencia. A continuación, se han presentado las soluciones de referencia, sencillas soluciones básicas cuyos resultados se intentan mejorar, y se han comparado los resultados de la solución planteada con los resultados de estas soluciones de referencia. Por último, en el capítulo 5 se presentan tanto las conclusiones derivadas de esta evaluación como las extraídas durante el desarrollo de la tesis doctoral en su totalidad. Quizá el lector haya echado en falta una evaluación sobre el rendimiento del sistema desarrollado. Esta evaluación no se ha realizado ya que la solución propuesta fue concebida para ser ejecutada como un proceso *batch* que no nece-

[1] http://www.resist-noe.org/

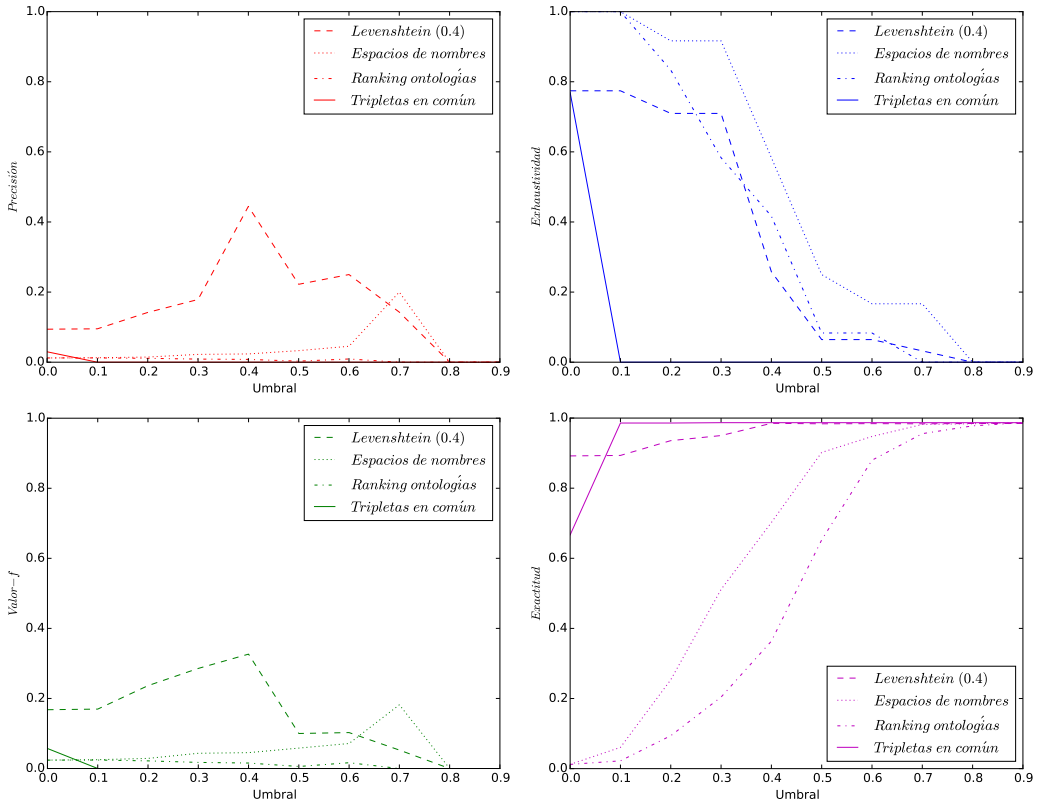


Figura 4.13: Comparación de los resultados obtenidos por las soluciones de referencia y la solución propuesta excluyendo los conjuntos de datos de la plataforma RKB Explorer.

sitaba la interacción del usuario para su finalización. Por esta razón, se consideró que evaluar el rendimiento de la solución desarrollada era irrelevante. Sin embargo, a modo de ilustración, se puede decir que en el caso del conjunto de datos `rkb-explorer-acm`, uno de los más voluminosos con más de 12 millones de tripletas[1] (aunque su gran número de tripletas no es el habitual en el resto de los conjuntos de datos empleados para la evaluación), el proceso de conversión de RDF a fichero de SUBDUE llegó a tardar alrededor de 48 horas. Al contrario, la duración del resto de algoritmos alcanzaba unos pocos minutos en el peor de los casos.

[1] `http://datahub.io/dataset/rkb-explorer-acm`

Por fin llegó el final de la función...
por esta noche.

M.C.D.

CAPÍTULO

5

Conclusiones y trabajo futuro

Antes de abordar este capítulo final, realizaremos un rápido repaso por toda la tesis en su conjunto. En el primer capítulo se ha contextualizado la tesis, presentando al lector la Web Semántica y el paradigma Linked Data. Además, en este capítulo se detallaba la problemática detectada y se planteaba la hipótesis a validar. A continuación, en el segundo capítulo se analizaba el estado del arte más destacado dentro de los campos de la búsqueda, enlazado y síntesis de conjuntos de datos, identificación de fuentes para el enlazado de conjuntos de datos y extracción de subgrafos más frecuentes. Junto con la descripción de los trabajos más relevantes en cada una de las áreas, se encuentra un análisis crítico en el que se resaltan las fortalezas y carencias de cada uno de ellos. Una vez analizado el estado del arte, en el capítulo 3 se ha expuesto la solución desarrollada para la problemática planteada durante el primer capítulo. Esta solución es evaluada y contrastada durante el capítulo 4, a través del estándar de referencia y de las soluciones de referencia desarrolladas.
Por último, una vez desarrollada la evaluación corresponde analizar los resultados obtenidos. De la evaluación realizada se pueden extraer dos conclusiones principales:

1. Los resultados plasmados en la figura 4.4 demuestran que, tal y como declaraba la hipótesis planteada (sección 1.3.3), **a través del análisis de las características estructurales de los propios conjuntos de datos, es posible hallar conjuntos de datos candidatos a ser enlazados**. Como puede observarse, a través de esta técnica se han logrado unos resultados muy favorables en términos de precisión, alrededor del 80 %. Esto significa que en torno al 80 % de los conjuntos de datos sugeridos por la solución desarrollada son efectivamente candidatos a ser enlazados. Por otra parte, la exhaustividad, rozando el 40 % en el mejor de los casos, es mejorable. Esto significa que de todas las parejas de conjuntos de datos a emparejar, la solución desarrollada solamente recomienda el 40 % del total. Teniendo en cuenta los recursos existentes actualmente a la hora de elegir conjuntos candidatos a ser enlazados, consideramos que el dato de la exhaustividad no es del todo importante ya que lo principal es reducir el espacio de búsqueda de conjuntos candidatos, aunque está claro que siempre es conveniente su mejora.

2. En la sección 3.6 se planteaba la problemática que surgía cuando el mismo concepto aparecía descrito a través de términos de ontologías diferentes en los subgrafos más frecuentes de los conjuntos de datos. Esto provocaba que la solución planteada no identificase estos conceptos similares y surgiesen falsos negativos. Para paliar esta carencia se propuso el empleo de técnicas de similitud de caracteres con el objetivo de identificar cuando dos términos de ontologías diferentes se referían al mismo concepto. Tras evaluar la aplicación de diferentes técnicas de búsqueda de similitudes entre cadenas de caracteres sobre la solución desarrollada, los resultados varían sensiblemente, tal y cómo puede observarse en la figura 4.11 (página 106). A pesar de que con la aplicación de las técnicas de búsqueda de similitudes entre cadenas de caracteres la precisión desciende levemente, la exhaustividad aumenta considerablemente, solucionando la problemática planteada en la primera de las conclusiones. Por lo tanto **puede afirmarse que las técnicas de búsqueda de similitudes entre cadenas de caracteres contribuyen de manera positiva con los resultados obtenidos**.

Por lo tanto, se confirma que tal y como enunciaba la hipótesis planteada en esta tesis doctoral, *"a partir de un conjunto de datos estructurado, se pueden hallar, de manera automática, otros conjuntos de datos idóneos para su enlazado solamente a través del análisis de las características estructurales de los propios conjuntos de datos"*. Pero, además de validar esta hipótesis, durante esta tesis doctoral se ha demostrado que el empleo de técnicas de similitud de cadenas de caracteres puede mejorar los resultados a la hora de comparar conjuntos de datos descritos a través de ontologías diferentes. La validación de esta hipótesis demuestra que **la herramienta desarrollada puede ser un valioso recurso para generar la entrada de herramientas de enlazado como SILK, ya que facilita la elección de los conjuntos de datos entre los que buscar enlaces**.

Respecto a las soluciones de referencia, en la sección 4.3.2 se muestra la comparativa entre dichas soluciones y la solución desarrollada. Como ya se menciona en dicha sección, una de las soluciones de referencia logra buenos resultados debido a que un gran número de los conjuntos de datos empleados para la evaluación proceden del mismo proyecto. La solución óptima para la correcta evaluación de la solución planteada hubiese sido emplear un conjunto de conjuntos de datos más heterogéneo. Esto no ha sido posible dado el estado en el que se encuentran muchos de los conjuntos de datos de la LOD Cloud en la actualidad. Ya que para extraer los subgrafos más frecuentes era necesario disponer del conjunto de datos en su totalidad, en un principio se optó por descargar los volcados de los conjuntos de datos y en los casos en los que no existiese dicho volcado se realizarían consultas sobre su punto de acceso SPARQL. Sin embargo, esto no fue posible ya que muchos puntos de acceso no respondían o no eran capaces de responder a consultas complejas. En el momento de la escritura de esta tesis doctoral, según la web SPARQL Endpoint Status[1], de 535 puntos de acceso SPARQL monitorizados solamente estaban operativos 266 (49.72 %), algunos de los cuales, además, habían tenido problemas de conexión en las últimas 24 horas. Muchos de ellos no eran capaces de responder a consultas que devolviesen un gran número de resultados, por lo que se intentó limitar los resultados devueltos a través de las cláusulas *OFFSET* y *LIMIT* del lenguaje SPARQL, sin éxito en la mayoría de los casos. Además, algunos conjuntos de datos no disponían de ningún tipo de recurso accesible.

[1] `http://sparqles.ai.wu.ac.at/availability`

Esta situación incita a reflexionar sobre a) la razón por la cual tantos conjuntos de datos han dejado de ser operativos y b) la razón por la cual dichos conjuntos no han sido retirados del grupo LOD Cloud en el registro DataHub. Puede que todavía el proceso de publicación, consumo y mantenimiento de los datos como Linked Data siguiendo todas las directrices del W3C sea demasiado complicado para el beneficio obtenido, y esto haga que las organizaciones que administraban estos conjuntos de datos hayan abandonado estos proyectos. Sea cual sea el motivo, existe la necesidad de crear mecanismos y herramientas que colaboren tanto en la publicación de nuevos conjuntos de datos como en el mantenimiento de los existentes, además de mantener registros y catálogos como DataHub actualizados. Creemos que **el trabajo desarrollado durante esta tesis doctoral contribuirá de manera positiva en el proceso de generación y publicación de datos enlazados**.
Recapitulando, esta tesis doctoral ha aportado las siguientes contribuciones al estado del arte:

1. Desarrollo de un método que permite la síntesis o resumen de la estructura de los grandes conjuntos de datos, de cara a su posterior análisis.

2. Desarrollo de una herramienta para la conversión de grafos RDF a un formato procesable por la herramienta de minería de grafos SUBDUE.

3. Desarrollo de un sistema que dada una serie de conjuntos de datos enlazados, como podría ser la LOD Cloud, recomienda conjuntos de datos candidatos a ser enlazados.

4. Desarrollo de un "gold standard" que permite la evaluación tanto del sistema desarrollado en esta tesis como de los sistemas de recomendación de Linked Data desarrollados en el futuro.

5. Desarrollo de una aplicación web para la evaluación a través del *crowdsourcing* de los conjuntos candidatos a ser enlazados.

Sin embargo, se han detectado una serie de carencias en la solución desarrollada en esta tesis que deberían solucionarse durante el trabajo futuro. Una de estas carencias está relacionada con los enlaces externos. Por motivos técnicos, durante

el desarrollo de la solución actual no se contempló el uso de los enlaces externos de los conjuntos de datos por varios motivos. Uno de ellos está relacionado con la excesiva carga en la red que provoca el hecho de consultar la gran cantidad de enlaces externos que existen en algunos conjuntos de datos. Durante las pruebas de concepto realizadas con enlaces externos, se detectó que esta carga de red ralentizaba mucho la generación de ficheros de SUBDUE. Por otra parte, muchos puntos de acceso SPARQL quedaban inoperativos debido al gran volumen de consultas realizadas, imposibilitando la realización de la tarea. Una manera de resolver esta carencia podría ser realizando las consultas pertinentes sobre el punto de acceso SPARQL de la LOD Cloud caché[1].

Otra de las carencias de la solución propuesta está relacionada con la aplicación de técnicas de similitud de cadenas de caracteres. Tal y cómo se ha demostrado durante la evaluación (capítulo 4), este tipo de técnicas pueden contribuir en la búsqueda de conjuntos de datos candidatos a ser enlazados. Aún así, existen algunas carencias en la aplicación de estas técnicas durante estas tesis doctoral. Por una parte hay que resaltar la sencillez de estas técnicas frente a otras de mayor complejidad y que podrían producir mejores resultados, como las técnicas de Procesado de Lenguaje Natural (PLN). Por otra parte, a la hora de evaluar la efectividad de estas técnicas aplicadas a la búsqueda de conjuntos de datos candidatos a ser enlazados, además de evaluar cada técnica por separado se calculó la media geométrica, cuando el cálculo de una media ponderada asignando diferentes pesos a cada una de las técnicas empleadas hubiese sido más apropiado. Para asignar los pesos correspondientes a cada una de las técnicas, se plantea el empleo de técnicas de regularización similares a las empleadas en el ámbito de la inteligencia artificial.

Por otra parte, a pesar de los buenos resultados obtenidos, la solución propuesta es mejorable. Para finalizar, dentro del trabajo futuro se plantea llevar a cabo las siguientes acciones:

1. **Analizar las últimas herramientas para el análisis de grafos como Spark GraphX[2] o Apache Giraph[3].** A pesar de que se encuentra fuera de los objetivos de esta tesis doctoral, la inclusión de estas librerías en sustitución de la

[1] `http://lod.openlinksw.com/sparql`
[2] `http://spark.apache.org/graphx/`
[3] `http://giraph.apache.org/`

herramienta SUBDUE incrementarían las capacidades de la fase de extracción de subgrafos más frecuentes, ya que están preparadas para su ejecución en entornos de computación en paralelo y permite realizar operaciones sobre grafos de manera sencilla.

2. **Encontrar el tamaño óptimo de los subgrafos más frecuentes.** Actualmente, a la hora de encontrar los subgrafos más frecuentes se prioriza el subgrafo más frecuente de menor tamaño. Establecer un tamaño mínimo para estos subgrafos podría (o no) mejorar los resultados obtenidos.

3. **Incluir enlaces externos a la hora de generar los ficheros de SUBDUE.** La solución planteada ignora los enlaces externos. El análisis de los enlaces externos ya existentes enriquecería el grafo generado.

4. **Incluir técnicas de búsqueda de similitudes entre cadenas de caracteres más avanzadas.** Tal y como puede observarse durante la sección 3.6 (página 73), las técnicas de búsqueda de similitudes entre cadenas de caracteres empleadas son muy básicas. Esto se debe a que el uso de estas técnicas dentro de esta tesis doctoral se empleó solamente para demostrar que a través de ellas se podrían mejorar los resultados obtenidos. Una vez demostrado que así es, se analizarán técnicas más avanzadas con el objetivo de incluirlas en la solución desarrollada. Dentro de estas técnicas se contemplan las de Procesado de Lenguaje Natural (PLN) o incluso el uso del *crowdsourcing* para determinar cuándo dos términos representan el mismo concepto. De la misma manera, a la hora de evaluar los resultados obtenidos por estas técnicas, se sustituirá la actual media geométrica por la media ponderada, asignando los pesos adecuados a cada una de las técnicas.

5. **Incrementar el número y variedad de los conjuntos de datos empleados en la evaluación.** Tal y como se ha expuesto anteriormente en este capítulo, el número y variedad de los conjuntos de datos empleados en la evaluación se ha visto limitado por cuestiones técnicas. En el futuro se intentará superar estos impedimentos técnicos con el objetivo de incluir nuevos conjuntos de datos en la evaluación e incluso agregar conjuntos de datos ajenos a la LOD Cloud.

APÉNDICE

A

Conjuntos de datos empleados durante la evaluación

A continuación, en el cuadro A.1 se indican la URL de la entrada en la plataforma DataHub de cada uno de los conjuntos de datos empleados.

URL	
archiveshub-linkeddata	bfs-linked-data
bio2rdf-interpro	bio2rdf-sgd
bio2rdf-sider	clean-energy-data-reegle
colinda	courts-thesaurus
data-cnr-it	dbtune-john-peel-sessions
dcs-sheffield	dewey_decimal_-classification
education-data-gov-uk	environmental-applications-reference-thesaurus
eu-institutions	fao-linked-data
finnish-municipalities	fu-berlin-medicare
gemet	geolinkeddata
geological-survey-of-austria-thesaurus	gesis-thesoz

hellenic-fire-brigade	hellenic-police
iserve	jamendo-dbtune
klappstuhlclub	linked-open-vocabularies-lov
lista-encabezamientos-materia	national-diet-library-subject-headings
ntnusc	nvd
nytimes-linked-open-data	osm-semantic-network
pleiades	rkb-explorer-acm
rkb-explorer-budapest	rkb-explorer-cordis
rkb-explorer-courseware	rkb-explorer-curriculum
rkb-explorer-darmstadt	rkb-explorer-dblp
rkb-explorer-deepblue	rkb-explorer-deploy
rkb-explorer-eurecom	rkb-explorer-ft
rkb-explorer-ibm	rkb-explorer-ieee
rkb-explorer-irit	rkb-explorer-jisc
rkb-explorer-kisti	rkb-explorer-laas
rkb-explorer-newcastle	rkb-explorer-os
rkb-explorer-pisa	rkb-explorer-rae2001
rkb-explorer-risks	rkb-explorer-roma
rkb-explorer-southampton	rkb-explorer-ulm
rkb-explorer-unlocode	rkb-explorer-wiki
semantic-web-dog-food	stw-thesaurus-for-economics
telegraphis	temple-ov-thee-lemur-datasets
thesesfr	transparency-linked-data
yovisto	

Cuadro A.1: Conjuntos de datos empleados durante la evaluación de esta tesis doctoral. Por motivos de legibilidad se ha omitido el prefijo de la URL `http://datahub.io/dataset/`.

Bibliografía

AGRAWAL, R., SRIKANT, R. ET AL. Fast algorithms for mining association rules. En *Proceedings of 20th International Conference on Very Large Data Bases*, vol. 1215, página 487–499. 1994. ISBN 1-55860-153-8. 48

AUER, S., BIZER, C., KOBILAROV, G., LEHMANN, J., CYGANIAK, R. y IVES, Z. Dbpedia: A nucleus for a Web of Open Data. En *The Semantic Web. 6th International Semantic Web Conference, 2nd Asian Semantic Web Conference*, páginas 722–735. Springer, 2007. ISBN 978-3-540-76298-0. 3

BECKETT, D., BERNERS-LEE, T., PRUD'HOMMEAUX, E. y CAROTHERS, G. Turtle: Terse RDF triple language. `http://www.w3.org/TR/turtle/`, 2013. 4

BECKETT, D. y CAROTHERS, G. N-triples: A line-based syntax for an RDF graph. `http://www.w3.org/TR/n-triples/`, 2013. 4

BECKETT, D. y MCBRIDE, B. RDF/XML syntax specification (revised). `http://www.w3.org/TR/REC-rdf-syntax/`, 2004. 4

BERNERS-LEE, T. Hypertext style: Cool URIs don't change. `http://www.w3.org/Provider/Style/URI.html`, 1998. 5

BERNERS-LEE, T. Linked Data - Design issues. `http://www.w3.org/DesignIssues/LinkedData.html`, 2006. iii, v, vii, 2

BERNERS-LEE, T., CHEN, Y., CHILTON, L., CONNOLLY, D., DHANARAJ, R., HOLLENBACH, J., LERER, A. y SHEETS, D. Tabulator: Exploring and analyzing Linked Data on the Semantic Web. En *Proceedings of the 3rd International Semantic Web User Interaction Workshop, collocated with the 5th International Semantic Web Conference*. 2006. 3

BERNERS-LEE, T. y CONNOLLY, D. Notation3 (N3): A readable RDF syntax. `http://www.w3.org/TeamSubmission/n3/`, 2011. 4

BERNERS-LEE, T., HENDLER, J. y LASSILA, O. The Semantic Web. *Scientific american*, vol. 284(5), páginas 28–37, 2001. 1

BÖHM, C., KASNECI, G. y NAUMANN, F. Latent topics in graph-structured data. En *Proceedings of the 21st ACM international conference on information and knowledge management*, páginas 2663–2666. ACM, 2012. ISBN 978-1-4503-1156-4. xii, 34, 40, 45, 59

BORGELT, C. y BERTHOLD, M. R. Mining molecular fragments: finding relevant substructures of molecules. En *Proceedings of IEEE International Conference on Data Mining*, páginas 51–58. 2002. ISBN 0-7695-1754-4. 49

BOX, D., EHNEBUSKE, D., KAKIVAYA, G., LAYMAN, A., MENDELSOHN, N., NIELSEN, H. F., THATTE, S. y WINER, D. Simple object access protocol (SOAP) 1.1. `http://www.w3.org/TR/SOAP/`, 2000. 16

BRICKLEY, D., GUHA, R. V. y MCBRIDE, B. RDF vocabulary description language 1.0: RDF schema. `http://www.w3.org/TR/rdf-schema/`, 2004. 1

BUITELAAR, P., EIGNER, T. y DECLERCK, T. OntoSelect: a dynamic ontology library with support for ontology selection. En *In Proceedings of the Demo Session at the 13th International Semantic Web Conference*. 2004. 18

CAROTHERS, G. RDF 1.1 N-Quads. `http://www.w3.org/TR/n-quads/`, 2014. 19

CHEN, C., YAN, X., ZHU, F. y HAN, J. gApprox: Mining frequent approximate patterns from a massive network. En *Proceedings of the 7th IEEE International Conference on Data Mining*, páginas 445–450. 2007. ISBN 978-0-7695-3018-5. 51

CHENG, G. y QU, Y. Searching linked objects with Falcons. *International Journal on Semantic Web and Information Systems*, vol. 5(3), páginas 49–70, 2009. ISSN 1552-6283. 23

COHEN, J. A coefficient of agreement for nominal scales. *Journal on Educational and Psychological Measurement*, vol. 20, páginas 37–46, 1960. 87

COOK, D. J. y HOLDER, L. B. Substructure discovery using minimum description length and background knowledge. *Journal of Artificial Intelligence Research*, vol. 1(1), páginas 231–255, 1993. 70

CORBY, O., DIENG-KUNTZ, R., GANDON, F. y FARON-ZUCKER, C. Searching the Semantic Web: Approximate query processing based on ontologies. *Intelligent Systems, IEEE*, vol. 21(1), páginas 20–27, 2006. ISSN 1541-1672. 38

CORMEN, T. H., LEISERSON, C. E., RIVEST, R. L. y STEIN, C. *Introduction to algorithms*. The MIT press, 2001. ISBN 978-0262033848. 49

DING, L., FININ, T., JOSHI, A., PAN, R., COST, R., PENG, Y., REDDIVARI, P., DOSHI, V. y SACHS, J. Swoogle: A Semantic Web search and metadata engine. En *Proceedings of the 13th ACM Conference on Information and Knowledge Management*, páginas 652–659. 2004. ISBN 1-58113-874-1. 16

DUNNING, T. Accurate methods for the statistics of surprise and coincidence. *Computational linguistics*, vol. 19(1), páginas 61–74, 1993. 35

D´AQUIN, M., SABOU, M., DZBOR, M., BALDASSARRE, C., GRIDINOC, L., ANGELETOU, S. y MOTTA, E. Watson: a gateway for the Semantic Web. *4th European Semantic Web Conference*, 2007. 5, 20

EUZENAT, J. y SHVAIKO, P. *Ontology matching*. Springer Heidelberg, 2007. ISBN 978-3-642-38720-3. 31, 74

FETAHU, B., DIETZE, S., NUNES, B. P., CASANOVA, M. A., TAIBI, D. y NEJDL, W. A scalable approach for efficiently generating structured dataset topic profiles. En *The Semantic Web: Trends and Challenges*, páginas 519–534. Springer, 2014. ISBN 978-3-319-07442-9. xii, xvi, 34, 35, 40, 86

FITZPATRICK, B. Distributed caching with memcached. *Linux Journal*, vol. 2004(124), página 5, 2004. ISSN 1075-3583. 26

FLEISS, J. L. Measuring nominal scale agreement among many raters. *Psychological bulletin*, vol. 76(5), página 378, 1971. 86

GLASER, H. y MILLARD, I. RKB Explorer: Application and infrastructure. En *Proceedings of the Semantic Web Challenge, in conjunction with the 6th International Semantic Web Conference*. 2007. 107

GUHA, R., MCCOOL, R. y MILLER, E. Semantic search. En *Proceedings of the 12th international conference on World Wide Web*, páginas 700–709. 2003. ISBN 1-58113-680-3. 15

HAMACHER, H., LEBERLING, H. y ZIMMERMANN, H.-J. Sensitivity analysis in fuzzy linear programming. *Fuzzy sets and systems*, vol. 1(4), páginas 269–281, 1978. 75

HARTH, A., HOGAN, A., DELBRU, R., UMBRICH, J., O'RIAIN, S. y DECKER, S. SWSE: Answers before links! En *Proceedings of the Semantic Web Challenge, in conjunction with the 6th International Semantic Web Conference*. 2007. 23

HARTH, A., UMBRICH, J. y DECKER, S. Multicrawler: A pipelined architecture for crawling and indexing Semantic Web data. En *Proceedings of the 5th International Semantic Web Conference*, páginas 258–271. Springer, 2006. 19

HASAN, R. Generating and summarizing explanations for Linked Data. En *The Semantic Web: Trends and Challenges*, páginas 473–487. Springer, 2014. ISBN 978-3-319-07442-9. xii, 34, 37, 40

HOLDER, L. B., COOK, D. J. y DJOKO, S. Substructure discovery in the SUBDUE system. En *Proceedings of the AAAI Workshop on Knowledge Discovery in Databases*, páginas 169–180. 1994. 47

HORRIDGE, M. y BECHHOFER, S. The OWL API: A Java API for OWL ontologies. *Semantic Web*, vol. 2(1), páginas 11–21, 2011. ISSN 1570-0844. 18

HUAN, J., WANG, W. y PRINS, J. Efficient mining of frequent subgraphs in the presence of isomorphism. En *Proceedings of the 3rd IEEE International Conference on Data Mining*, páginas 549–552. 2003. ISBN 0-7695-1978-4. 49

INOKUCHI, A., WASHIO, T. y MOTODA, H. An apriori-based algorithm for mining frequent substructures from graph data. En *Principles of Data Mining and Knowledge Discovery*, página 13–23. Springer, 2000. ISBN 978-3-540-41066-9. 48

ISELE, R. y BIZER, C. Learning linkage rules using genetic programming. En *Proceedings of the Very Large Database Endowment*, vol. 5, páginas 1638–1649. 2012. ISSN 2150-8097. 30

JENTZSCH, A., ISELE, R. y BIZER, C. Silk - Generating RDF links while publishing or consuming Linked Data. En *Proceedings of the poster and demo session of the 9th International Semantic Web Conference*. 2010. 31

JIANG, C., COENEN, F. y ZITO, M. A survey of frequent subgraph mining algorithms. *The Knowledge Engineering Review*, vol. 28(1), páginas 75–105, 2013. ISSN 1469-8005. 46, 57

KENDALL, M. G. A new measure of rank correlation. *Biometrika*, páginas 81–93, 1938. 89

KIFER, M. y LAUSEN, G. F-logic: a higher-order language for reasoning about objects, inheritance, and scheme. En *Proceedings of the 1989 ACM SIGMOD international conference on Management of data*, vol. 18, páginas 134–146. 1989. ISBN 0-89791-317-5. 15

KLEINBERG, J. M. Authoritative sources in a hyperlinked environment. *Journal of the ACM*, vol. 46(5), página 604–632, 1999. ISSN 0004-5411. 36

KURAMOCHI, M. y KARYPIS, G. Frequent subgraph discovery. En *Proceedings of the 1st IEEE International Conference on Data Mining*, páginas 313–320. 2001. ISBN 0-7695-1119-8. 48

KURAMOCHI, M. y KARYPIS, G. GREW: a scalable frequent subgraph discovery algorithm. En *Proceedings of the 4th IEEE International Conference on Data Mining*, páginas 439–442. 2004. ISBN 0-7695-2142-8. 50

KURAMOCHI, M. y KARYPIS, G. Finding frequent patterns in a large sparse graph. *Data mining and knowledge discovery*, vol. 11(3), páginas 243–271, 2005. ISSN 1384-5810. 51

LANDIS, J. R. y KOCH, G. G. The measurement of observer agreement for categorical data. *Biometrics*, vol. 33(1), páginas 159–174, 1977. 87

LEME, L. A. P. P., LOPES, G. R., NUNES, B. P., CASANOVA, M. A. y DIETZE, S. Identifying candidate datasets for data interlinking. En *Proceedings of the 13th International Conference on Web Engineering*, páginas 354–366. Springer, 2013. ISBN 978-3-642-39199-6. xii, 8, 41, 42, 44, 45, 84

LEVENSHTEIN, V. I. Binary codes capable of correcting deletions, insertions and reversals. *Soviet physics doklady*, vol. 10(8), páginas 707–710, 1966. 75

LOPES, G. R., LEME, L. A. P. P., NUNES, B. P., CASANOVA, M. A. y DIETZE, S. Recommending tripleset interlinking through a social network approach. En *Proceedings of the 14th international conference on Web Information Systems Engineering*, páginas 149–161. Springer, 2013. ISBN 978-3-642-41229-5. xii, 41, 43, 45

MAALI, F., CYGANIAK, R. y PERISTERAS, V. Re-using cool URIs: entity reconciliation against LOD hubs. En *Proceedings of the Linked Data on the Web workshop in conjunction with the 20th international World Wide Web conference*, vol. 813. CEUR, 2011. ISSN 1613-0073. 5

MILLER, G. A. Wordnet: a lexical database for English. *Communications of the ACM*, vol. 38(11), páginas 39–41, 1995. ISSN 0001-0782. 15, 76

NGOMO, A. y AUER, S. LIMES: a time-efficient approach for large-scale link discovery on the web of data. En *Proceedings of the 22nd International Joint Conference on Artificial Intelligence*, páginas 2312–2317. 2011. ISBN 978-1-57735-515-1. 31

NIJSSEN, S. y KOK, J. N. A quickstart in frequent structure mining can make a difference. En *Proceedings of the 10th ACM SIGKDD international conference on Knowledge Discovery and Data mining*, páginas 647–652. ACM, 2004. ISBN 1-58113-888-1. 50

NIKOLOV, A. y D´AQUIN, M. Identifying relevant sources for data linking using a Semantic Web index. En *Proceedings of the Linked Data on the Web workshop in conjunction with the 20th international World Wide Web conference*, vol. 813. 2011. ISSN 1613-0073. xii, 41, 44, 45

NIKOLOV, A., D'AQUIN, M. y MOTTA, E. What should I link to? Identifying relevant sources and classes for data linking. En *Proceedings of the Joint International Semantic Technology Conference*, vol. 7185, páginas 284–299. Springer, 2011. ISBN 978-3-642-29922-3. 42

NIKOLOV, A., UREN, V., MOTTA, E. y DE ROECK, A. Integration of semantically annotated data by the KnoFuss architecture. En *Knowledge Engineering: Practice and Patterns*, vol. 5268, páginas 265–274. Springer, 2008. ISBN 978-3-540-87695-3. 29

NOY, N. F., SINTEK, M., DECKER, S., CRUBÉZY, M., FERGERSON, R. W. y MUSEN, M. A. Creating Semantic Web contents with Protege-2000. *Intelligent Systems, IEEE*, vol. 16(2), páginas 60–71, 2001. ISSN 1541-1672. 20

PAGE, L., BRIN, S., MOTWANI, R. y WINOGRAD, T. The PageRank citation ranking: Bringing order to the Web. Technical report, Stanford University, 1998. 14, 18, 27, 36

PATEL, C., SUPEKAR, K., LEE, Y. y PARK, E. K. OntoKhoj: a Semantic Web portal for ontology searching, ranking and classification. En *Proceedings of the 5th ACM International Workshop on Web Information and Data Management*, páginas 58–61. 2003. 14

PRUD'HOMMEAUX, E. y SEABORNE, A. SPARQL query language for RDF. `http://www.w3.org/TR/rdf-sparql-query/`, 2008. 2

RAIMOND, Y., SUTTON, C. y SANDLER, M. B. Automatic interlinking of music datasets on the Semantic Web. En *Proceedings of the Linked Data on the Web workshop in conjunction with the 17th international World Wide Web conference*, vol. 369. CEUR, 2008. ISSN 1613-0073. 28

REDDY, D. R. Speech recognition by machine: A review. *Proceedings of the IEEE*, vol. 64(4), páginas 501–531, 1976. ISSN 0018-9219. 47

RISSANEN, J. Modeling by shortest data description. *Automatica*, vol. 14(5), páginas 465 – 471, 1978. ISSN 0005-1098. 47

SAUERMANN, L., CYGANIAK, R. y VÖLKEL, M. Cool URIs for the Semantic Web. `http://www.dfki.uni-kl.de/~sauermann/2007/01/semweburisdraft/uricrisis.pdf`, 2007. 5

SCHARFFE, F., LIU, Y. y ZHOU, C. RDF-AI: an architecture for RDF datasets matching, fusion and interlink. En *Proceedings of Identity and Reference in web-based Knowledge Representation Workshop in conjunction with the International Joint Conferences on Artificial Intelligence*. 2009. 31

SEABORNE, A. RDQL - A query language for RDF. `http://www.w3.org/Submission/RDQL/`, 2004. 15

STOILOS, G., STAMOU, G. y KOLLIAS, S. A string metric for ontology alignment. En *Proceedings of the 4th International Semantic Web Conference*, páginas 624–637. Springer, 2005. ISBN 978-3-540-29754-3. 75

THALHAMMER, A., TOMA, I., ROA-VALVERDE, A. y FENSEL, D. Leveraging usage data for Linked Data movie entity summarization. En *Proceedings of the 2nd International Workshop on Usage Analysis and the Web of Data in conjunction with the 21st International World Wide Web Conference*. 2012. xii, 34, 35

TUMMARELLO, G., CYGANIAK, R., CATASTA, M., DANIELCZYK, S., DELBRU, R. y DECKER, S. Sig.ma: Live views on the web of data. *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 8(4), páginas 355–364, 2010. ISSN 1570-8268. 25

TUMMARELLO, G., DELBRU, R. y OREN, E. Sindice.com: Weaving the Open Linked Data. En *Proceedings of the 6th International Semantic Web Conference and the 2nd Asian Semantic Web Conference*, páginas 552–565. 2007. ISBN 978-3-540-76297-3. 21

VANETIK, N., GUDES, E. y SHIMONY, S. E. Computing frequent graph patterns from semistructured data. En *Proceedings of the 2002 IEEE International Conference on Data Mining*, páginas 458–465. 2002. ISBN 0-7695-1754-4. 48

VOLZ, J., BIZER, C., GAEDKE, M. y KOBILAROV, G. Discovering and maintaining links on the web of data. En *Proceedings of the 8th International Semantic Web Conference*, páginas 650–665. Springer, 2009a. ISBN 978-3-642-04929-3. 30

VOLZ, J., BIZER, C., GAEDKE, M. y KOBILAROV, G. Silk: A link discovery framework for the web of data. En *Proceedings of the Linked Data on the Web workshop in conjuntion with the 18th international World Wide Web conference*, vol. 583. 2009b. ISSN 1613-0073. 30

WINKLER, W. E. Advanced methods for record linkage. *Proceedings of the Section on Survey Research Methods*, páginas 467–472, 1994. 28, 76

YAN, X. y HAN, J. gSpan: Graph-based substructure pattern mining. En *Proceedings of the 2002 IEEE International Conference on Data Mining*, páginas 721–724. 2002. ISBN 0-7695-1754-4. 49

ZAKI, M. J., PARTHASARATHY, S., OGIHARA, M., LI, W. ET AL. New algorithms for fast discovery of association rules. En *Proceedings of the 3rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, vol. 97, páginas 283–296. 1997. 49

ZHANG, X., CHENG, G. y QU, Y. Ontology summarization based on RDF sentence graph. En *Proceedings of the 16th international conference on World Wide Web*, páginas 707–716. 2007. ISBN 978-1-59593-654-7. 24

Esta tesis doctoral fue finalizada en Bilbao el 2 de octubre de 2015