Universidad Nacional
de Educación a Distancia

Escuela Técnica Superior de Ingeniería Informática

Grafos de Conocimiento como apoyo en la Docencia

Nidia Medea Nieto Fernández

Director: José Luis Fernández Vindel

Codirector: Jorge Pérez Martín

Trabajo de Fin de Máster Universitario
en Investigación en Inteligencia Artificial

Febrero 2022

Resumen

Este trabajo parte de la existencia de una cantidad masiva de datos estructurados en la Web Semántica y de otras fuentes de datos semiestructuradas disponibles, así como del estado avanzado de las tecnologías semánticas para el diseño y explotación de grafos de conocimiento locales al servicio de aplicaciones, en este caso como apoyo a la docencia en línea.

Sobre estas premisas, el trabajo describe un proceso de exploración de estrategias de aplicación de estos datos masivos y de estos grafos de conocimiento locales como soporte para la generación de actividades. En cada punto de esta exploración se han buscado siempre los procedimientos y herramientas que tuvieran una aplicabilidad más transversal, con la intención de que pudieran servir como guía de un soporte escalable de asignaturas impartidas en la UNED.

Abstract

This research study makes use of the existence of a massive amount of structured data in the Semantic Web and other semi-estructured data sources available, as well as the advanced state of semantic technologies for the design an explotation of local knowledge graphs at the service of applications, in this case as support for online teaching.

Based on these premises, this paper describes a process of exploring strategies for applying these massive data and these local knowledge graphs as support for the generetation of activities. At each point of this exploration, procedures and tools that have a more transversal applicability have always been sought, with the intention that they could serve as a guide for a scalable support of subjects taught at the UNED.

Palabras clave: knowledge graph, education knowledge graph, ontology, semantic web, semantic technologies

Índice

Capítulo 1

Motivación y Objetivos

1.1. Contexto y motivación

Existen sistemas de tutoría inteligente (*Intelligent Tutoring Systems*, ITSs) con los que interactúan estudiantes como guía durante el proceso de aprendizaje. Los ITSs se dividen en cuatro modelos (división establecida por *Nwana* [23], *Freedman et al.* [12] y *Nkambou et al.* [22]): de dominio, de estudiante, de tutorización y de interfaz de interacción.

La producción del modelo de dominio se realizaba desde cero a partir de entrevistas a expertos en el área, en este caso relacionado con el aprendizaje, a equipos docentes (un proceso similar al del diagnóstico en un área concreta en medicina).

Por otro lado, la existencia de la Web de Datos, junto con las existentes tecnologías semánticas, se puede usar como herramienta alternativa de extracción de información, en lugar de extraerla de entrevistas con el experto o de tecnologías de procesamiento de lenguaje natural, porque ya proporciona información estructurada.

El concepto de **Web de Datos** o **Web Semántica** surgió desde la creación de la Web (*World Wide Web*) por Berners-Lee[1] en 1989 [4], que es un repositorio enorme de información de fácil lectura para humanos, pero esta codificación de la

[1] https://www.w3.org/People/Berners-Lee/

información no resulta comprensible para las máquinas. Por ello, Lee, tuvo la idea de incluir información semántica desde el principio, con el objetivo de que las máquinas también fueran capaces de procesar estos datos. Pero no lo llevó a cabo hasta 2001 [5] a través del *World Wide Web Consorcium* (W3C), organización de estándares web que fundó en 1994, bajo el nombre de Web Semántica[2].

La Web Semántica se entiende entonces como una extensión de la Web que se basa en añadir metadatos semánticos y ontológicos para establecer relaciones entre los datos y facilitar la manipulación automática mediante lógica y motores de inferencia.

Desde el lanzamiento de Wikipedia[3] en 2001, el grafo de conocimiento de Google [27] y la base de conocimiento Wikidata[4] en 2012, bases de datos que actualmente cuentan con millones de elementos, los grafos de conocimiento (*knowledge graph, KG*) (término acuñado por Google) son cada vez más populares, no solo como una implementación particular de modelos de dominio para un ITS, sino como un procedimiento distinto de recopilar y enlazar datos y conceptos en una ontología de dominio.

Se refiere a un KG como un tipo de base de conocimiento con estructura de grafo, donde a su vez, una **base de conocimiento** (*knowledge base*, KB) es una base de datos de un conjunto de definiciones con forma de reglas lógicas para describir un dominio, cuya estructura ayuda a inferir otros hechos que no estaban previamente definidos en la base.

Se puede construir un KG a partir de otros públicos o mediante Tecnologías del Lenguaje que permiten rastrear masivamente documentos escritos por expertos del dominio y extraer y clasificar términos de estos documentos como candidatos a ser entidades, atributos y propiedades de la ontología de dominio que se está construyendo de esta forma semiautomática.

[2]https://www.w3.org/standards/semanticweb/
[3]https://en.wikipedia.org/wiki/Wikipedia:About
[4]https://www.wikidata.org/

La principal motivación en este trabajo, es servir de apoyo al equipo docente universitario, que suelen contar con un número alto de alumnos y deben realizar una gran cantidad de tareas, no solo docentes, y esto conlleva a disponer poco tiempo para dedicar una atención personalizada a cada alumno.

Esta motivación surgió inicialmente del contrato como colaboradora del grupo de innovación docente ISLearning de la UNED, donde se pretendía explorar las tecnologías semánticas al soporte de asignaturas en esta universidad de la manera más escalable posible, es decir, que fuera eficiente y se pudiera aplicar al mayor número de actividades.

Por otro lado, siguiendo una experiencia previa: el TFM de Omar Khalil [13] sobre un KG para la docencia de Psicofarmacología, donde se vió un problema en la utilidad de este tipo de grafos, la sobre-conceptualización y la sobre-población. Por ello y por la carga docente que tiene un profesor, se pretende explorar bases de conocimiento específicas para cada asignatura como principal fuente de datos para mantener pocas entrevistas con el equipo docente.

Mencionar que debido a la exploración de uso de estas tecnologías, a lo largo del proceso, han ido surgiendo otras motivaciones que han sido decisivas para seguir por un camino y que se detallan al inicio de cada etapa del estudio.

1.2. Hipótesis y objetivos

La hipótesis de partida de este trabajo se puede desglosar en cuatro supuestos:

1. Es posible construir un grafo previo de conocimiento concreto para una asignatura con una interacción inicial muy escasa con su equipo docente: a partir de los datos estructurados de la Web de Datos y a partir de textos recomendados del área y otros recursos docentes, como un repositorio de exámenes o guías.

2. Es posible establecer un procedimiento de refinamiento y confirmación de este KG de dominio mediante interacciones con el equipo docente, que pueden alterar tanto el contenido del KG actual como los filtros aplicables a las fuentes.

3. Es posible ampliar esta ontología de dominio a otros modelos sobre usuario y formas de tutorización, más transversales y de los que hay mucha experiencia previa. E implementar esos modelos como ampliación dentro de la propia estructura de grafo.

4. Se esperan beneficios en la interacción de esos KGs con el equipo docente o con los estudiantes.

Teniendo en cuenta las anteriores hipótesis, los objetivos iniciales de este trabajo han sido:

1. Explorar métodos, lo más transversales y eficientes posibles, para la detección de las fuentes de datos adecuadas para el soporte escalable de asignaturas diversas, así como la selección de herramientas y procedimientos de consulta que permitan poner a disposición de cada asignatura masivamente datos para el diseño de sus actividades.

2. Construir un grafo de conocimiento inicial sin contacto con el experto y posterior revisión del equipo docente para una asignatura dada, a partir de un repositorio de exámenes y de extracción de información de la bibliografía básica o de un glosario.

3. Contactar con el equipo docente de cada asignatura para refinar y confirmar el KG de dominio mediante interacciones.

4. Estudiar si es generalizable la construcción de estos KGs y valorar en qué medida se depende de la participación del experto.

5. Utilizar estos grafos de conocimiento para la generación de actividades o preguntas de tipo test de la asignatura.

1.3. Revisión de la bibliografía

El uso de los KGs, como se menciona en *Hogan et al.* [17], se ha extendido desde el lanzamiento del grafo de conocimiento de Google junto con este concepto, en

empresas conocidas como Amazon, eBay, IBM, LinkedIn, Microsoft y redes sociales como Facebook, entre otras cosas, para ayudar a mejorar sus búsquedas.

El KG de Google [27], cuenta con unos 500 billones de hechos sobre 5 billones de entidades, y entre sus utilidades, ayuda a generar paneles de conocimiento (*knowledge panels*) que recogen la información más buscada y se muestran, cuando se realiza una búsqueda sobre una entidad.

Existen otro KGs de acceso libre como: YAGO[5] desde 2018 que recoge datos de Wikipedia y de WordNet[6]; DBpedia[7] fundado en 2014 con más de 228 millones de entidades extraídas de Wikipedia; y Wikidata[8] desde el 2012 que recoge más de 90 millones de elementos añadidos por unos 24000 usuarios actualmente.

Se pueden encontrar grafos especializados en un área en la ***Linked Open Data Cloud*** (LOD Cloud) mantenida por *John P. McCrae* [1] y que almacena actualmente unos 1561 KGs.

Los KGs no solo son útiles para mejorar la rapidez de una búsqueda, como generadores de fichas con información básica o como fuente de datos estructurada para almacenar información no estructurada, si no que también se pueden usar como herramienta para generar actividades.

Para la construcción, estructura y poblamiento, de los grafos, se pueden usar diferentes técnicas y herramientas para conseguir datos de una fuente, dependiendo de cómo esté estructurada la información o si la información procede exclusivamente de uno o varios expertos sobre un área concreta, donde será necesario hacer a mano el KG o una parte. En *Heist et al.* [21] se hace un estudio de este tipo.

[5]https://yago-knowledge.org/
[6]https://wordnet.princeton.edu/
[7]https://www.dbpedia.org/
[8]https://www.wikidata.org/

Estudios de KGs para la educación

La información puede proceder del dominio de la educación como libros de texto, tutoriales de asignaturas o guías de estudio como se estudia en *Chen et al.* [8] donde se presenta un sistema llamado *KnowEdu* para automatizar la construcción de un KG para la enseñanza, extrayendo conceptos con herramientas como OCR y *Speech To Text* que expertos etiquetarán usando *BIOTagging* y extrayendo relaciones a partir de resultados de exámenes, y mencionan que resulta relativamente fácil para asignaturas como Matemáticas o de Ciencias en general.

En *Sun et al.* [29] se extrae información de un corpus formado por ejercicios digitalizados por colegios y ejercicios online gratuitos para aprender inglés, donde primero hacen un preprocesado para separar palabras, PoS e identificar entidades con nombre, después aplican SVM para clasificar y crear una jerarquía y crear un KG.

En *Quin et al.* [25] se utilizan preguntas de tipo Q&A extraídas de la Web para aprender Bases de Datos y usan también los etiquetados *BIO* como en *Chen et al.*, construyen un *knowledge card* como resultado de una búsqueda a partir de su KG, como hace Google.

Por otro lado, la información puede proceder de datos estructurados, como de otros grafos de conocimiento como en el estudio de *Omar Khalil* [13], que usa Wikidata y el *Unified Medical Language System* (UMLS) como fuentes de datos.

Existe un grafo de conocimiento llamado AI-KG (siglas de *Artificial Intelligence Knowledge Graph*) creado por creado por la *Open University* por *Dessì et al.* [9] a partir de un corpus de publicaciones extraídas del *Microsoft Academic Graph* (MAG) y que se actualiza con datos extraídos del mismo MAG, de Wikidata y de la *Computer Science Ontology* (CSO).

Tecnologías y herramientas

Entre las herramientas utilizadas para construir grafos existe el software T-Mitocar y Tarql utilizados en *Meissner y Köbis* [20] para crear redes de asociación (*association nets*) a partir de textos (información no estructurada) y convertir a RDF mediante Tarql.

En *Qin et al.* [25] se utiliza **Neo4j** para la construcción de los KGs, llamados *property graphs*, cuyo lenguaje de consulta es **Cypher**.

En este trabajo se ha usado el modelo **RDF** (siglas de *Resource Description Framework*) [26] para definir los grafos, en lugar de los *property graphs*, por tanto el lenguaje de consulta para éstos es **SPARQL** [14] en lugar de Cypher. Se han usado otras tecnologías semánticas, como **RDF Schema** [7], **RDFS-Plus** [2], que extiende RDF con un subconjunto de OWL, RDF extendido con vocabularios como **SKOS** [18] (*Simple Knowledge Organization System*) y **PROV** y **OWL2** [15] (*Web Ontology Language*).

Capítulo 2

Metodología y Herramientas

En este capítulo se introduce la exposición de las líneas de investigación que han resultado del objetivo principal como colaboradora del grupo ISLearning de la UNED, exploración de tecnologías semánticas al soporte de asignaturas. Estas líneas de investigación han sido cuatro que se han producido de forma secuencial debido a las limitaciones encontradas en la búsqueda de construir y mantener un KG a disposición del equipo docente:

1. **Viabilidad del poblamiento de actividades desde la Web de Datos**: conocer la Web de Datos o Web Semántica, las tecnologías semánticas y la búsqueda de posibles fuentes de información afines a un área concreta para alcanzar los objetivos del trabajo y ver las limitaciones de estas fuentes y tecnologías.

2. **Fuentes y métodos de propuesta de grafos locales de dominio**: centrar la construcción de un grafo en la de un grafo de dominio específico en un área.

3. **Desde los grafos de gestión de actividades hacia los grafos de dominio**: debido a la particularidad de la línea anterior, centrar la atención en crear un grafo de dominio para una sola asignatura a partir de un grafo de preguntas junto con el equipo docente.

4. **Propuestas escalables de aplicación para el soporte de asignaturas**: generalizar el proceso llevado en la línea anterior para posponer el contacto del experto lo máximo posible.

En la primera sección de este capítulo se resume el proceso de cada línea de investigación y en la segunda sección se detallan las principales herramientas que se han usado en cada etapa.

2.1. Metodología

El desarrollo de cada línea de investigación seguida ha mantenido la idea de generalizar el proceso de generación de un grafo de un área concreta sin la intervención de un experto, y por eso se ha visto necesario cambiar de camino cuando el proceso acababa siendo muy específico de un área o resultaba ser poco funcional. En las siguientes secciones se visualiza de forma general la metodología del trabajo y en el capítulo siguiente se detalla cada fase.

Primera línea

El primer objetivo fue conocer los conceptos, las herramientas y las fuentes de datos actuales de donde poder extraer información para la generación de un grafo de conocimiento. Para evitar fuentes de datos de información no estructurada se hizo una primera búsqueda en LOD Cloud para encontrar una conjunto de datos libre y con información estructurada.

Debido a la poca eficiencia y generalización de esta búsqueda, a partir de este punto, se utilizó Wikidata. En un principio se pensó se esperaba usarla para buscar otras fuentes de datos debido a su flexibilidad de búsqueda, pero finalmente se utilizó como proveedora de datos, por la gran cantidad de información que almacena y está sujeta a crecimiento y a revisión constante.

Se utilizó como caso de estudio el área de Historia de la Arte para conocer la estructura general de Wikidata y qué jerarquías se han tenido en cuenta para definir los conceptos de este área.

Por un lado, fue necesario crear estrategias de búsqueda por limitación de tiempo en las consultas y porque los resultados extraídos excedían la información pedida. Por otro lado, estos resultados se devolvían en formato tabla, y de esta forma no se mantienen todas las relaciones entre los términos.

Segunda línea

Siguiendo la línea anterior y con el objetivo de solucionar las limitaciones generadas, se orientó el estudio a crear un grafo de dominio sobre el área de la Inteligencia Artificial, por un lado con la motivación de poner a disposición del departamento de IA de la UNED un grafo de conocimiento con conceptos y relaciones de este área, y por otro lado, porque se pueden estructurar los términos en pocas clases generales (como ocurre en las materias de Ciencias en general), a diferencia del caso anterior estudiado.

La primera aproximación se hizo a Wikidata, pero la taxonomía generada no fue la esperada, así que se buscó en Wikipedia, encontrando una tabla en el artículo principal dedicado a la IA que engloba los conceptos en seis grupos. La combinación de ambas fuentes podría haber generado un grafo de conocimiento, pero se descubrió el AI-KG.

El AI-KG es un grafo de conocimiento creado específicamente para este área con una clasificación general de los conceptos como se buscaba. Junto con este grafo, en paralelo, se extrajo información de la tabla de contenidos y del glosario de dos libros específicos en el área, creando un grafo por cada uno. De esta forma, el objetivo era generar un grafo de conocimiento por asignatura relacionada con la IA uniendo información de las tres fuentes de datos.

Este proceso resultó ser muy extenso en tiempo y muy poco generalizable para otras asignaturas.

Tercera línea

Siguiendo con la línea anterior de crear un grafo de dominio para una asignatura concreta y manteniendo la idea de intentar generalizar lo máximo posible este proceso, se propuso un cambio de fuente inicial de datos, teniendo en cuenta que una de las aplicaciones de uso de un grafo es la de generar actividades evaluables.

La UNED cuenta con un depósito de exámenes de acceso público para docentes y alumnos, una fuente con la que se puede contar para extraer información de cualquier asignatura de la Universidad. Partir de preguntas de tipo test como fuente de datos fue el cambio en esta línea de investigación por su estructura compacta que da información tanto del enunciado como del posible resultado.

Primero se construyó una ontología, una estructura para agrupar conceptos de un dominio concreto, para el grafo de preguntas con la idea de usarlo para crear posteriormente el grafo de dominio.

Los casos de estudio en esta línea, para saber cómo de generalizable es este proceso, fueron dos asignaturas de las que son docentes los directores de este trabajo (*Lógica* y *Deep Learning*), por lo que en este caso se contó con la intervención de un experto en cada asignatura.

Cuarta línea

Siguiendo con la construcción de grafos de dominio a partir de grafos de gestión de actividades, en esta línea se estudió cómo de generalizable fue el proceso anterior para el caso de dos asignaturas sin contar con el equipo docente (*Psicología de las diferencias individuales* e *Historia económica mundial*) y que además son diferentes a las dos estudiadas antes, por estar en el ámbito de las Ciencias.

2.2. Herramientas

Las fuentes de datos, herramientas y tecnologías utilizadas en el trabajo se explican a continuación según su utilidad y separadas por línea de investigación.

Primera línea

Las herramientas **LOD Cloud**[1] y **LOV**[2] se utilizaron como primera aproximación a una búsqueda de fuentes de datos, aunque ambas fueron descartadas por **Wikidata**, Anexo A, que se utilizó como fuente de datos en lugar de indexadora de otros conjuntos de datos, como se ha explicado en 2.1

Existen puntos de consulta propios ofrecidos por la misma base de datos, como el Wikidata Query Service, pero también se pueden lanzar consultas desde un script de Python usando `SPARQLWrapper`, una librería que permite conectarse con el servidor de la base de datos que se quiera consultar.

Existen también dos herramientas que sirven para consultar a Wikidata: `wdtaxonomy`, para conocer la taxonomía de un elemento sobre una propiedad, y `MediaWiki API`, que también sirve para consultar a Wikipedia sobre términos que aparecen en el artículo.

El modelo de datos estándar que se ha utilizado es el de **RDF** que se basa en convertir enunciados en expresiones con forma de `sujeto-predicado-objeto` conocidas como tripletas. Un KG que sigue este modelo se llama **grafo RDF**, donde el `sujeto` y el `objeto` representan los nodos del grafo y el `predicado` la relación entre ellos (por ejemplo, `Velázquez pintor de Las Meninas`). Por tanto, se puede definir un KG como un conjunto de tripletas.

[1] https://lod-cloud.net/
[2] https://lov.linkeddata.es/dataset/lov

El vocabulario RDF se identifica con el URI (*Uniform Resource Identifier*):

`http://www.w3.org/1999/02/22-rdf-syntax-ns#`,

que se suele abreviar con el prefijo `rdf`. Un ejemplo de tripleta es:

`rdf:subject rdf:type rdf:Property`

que se interpreta como que el término `sujeto` definido por `rdf:subject` es una propiedad. Conviene mencionar que `rdf:type` se suele abreviar por el token `a`, debido a su frecuente uso. Este ejemplo muestra uno de los tres tipos de términos que pueden aparecer en las tripletas, los URIs, que representan un concepto

`http://www.w3.org/1999/02/22-rdf-syntax-ns#subject`,

pero también existen los literales y los nodos en blanco (*blank nodes*), donde:

- los literales representan valores: una cadena de caracteres, números, fechas, booleanos, ...
- los nodos en blanco, como los URIs, representan un concepto de forma única, pero sin darle un valor predefinido, quedándose vacío.

Existen otros vocabularios estándares para extender el modelo RDF y que Wikidata utiliza en algunos casos como, **RDF Schema**, **SKOS**, **PROV** y **OWL**.

Para las consultas a Wikidata, o a cualquier base de conocimiento estructurada mediante el modelo RDF, se usa el lenguaje estándar **SPARQL**, para el que existen diferentes tipos de consulta según el comando que se use. En esta primera línea se usaron las consultas `SELECT`, que permiten preguntar por una cantidad indefinida de variables y devuelve como resultado una tabla con todos los posibles resultados de esas variables. Estos resultados se pueden exportar en formato **JSON** o **CSV**.

Segunda línea

Además de usar Wikidata, se utilizó Wikipedia como fuente extra de datos y la librería de Python `BeautifulSoup` para extraer información de páginas web. Esta

librería también se usó para extraer el glosario y la tabla de contenidos del libro *Artificial Intelligence 2E: Foundations of Computational Agents* y transformándolos a formato **JSON** para conservar la estructura de árbol.

Otra fuente de datos fue **AI-KG**. Para conocer más sobre este grafo se puede consultar su web oficial `https://scholkg.kmi.open.ac.uk/`, el artículo que detalla su construcción [9] o al Anexo B donde se ha resumido su estructura.

En esta tarea de construcción de un grafo de dominio, además de utilizar consultas `SELECT` para conocer las fuentes de datos estructuradas, se han utilizado consultas `CONSTRUCT` de SPARQL, que solo permiten buscar por tres variables que forman una tripleta, dando como resultado el conjunto de tripletas buscado. También se han usado consultas `INSERT`, que tienen la misma estructura que las `CONSTRUCT`, pero éstas directamente añaden el resultado al grafo al que se hace referencia.

Para trabajar desde Python, además de la librería `SPARQLWrapper`, también existen `RDFLib` y `OWLReady2`, que permiten construir grafos y hacer inferencia sobre ellos, respectivamente.

Para mayor comodidad en esta línea se ha usado el programa **GraphDB**, que pertenece a la compañía ontotext, que cuenta con diferentes herramientas internas útiles para trabajar con KGs, como un punto de consulta interno con el que se pueden hacer consultas a otros KGs externos (consultas federadas) y **OntoRefine** (similar a OpenRefine, de Google), que permite trabajar con datos tabulares (en formato tabla) para su posterior conversión a RDF.

El vocabulario RDF en esta línea se ha extendido con `RDF Schema` para describir clases, con `SKOS` para describir relaciones entre entidades, con `OWL` para describir similitud con otros términos de un KG externo y con `PROV` de manera indirecta, pues es usado en AI-KG para definir de dónde procede un elemento.

Las consultas `CONSTRUCT` permiten descargar su resultado en Turtle, JSON,

RDF/XML o N3. En este caso se ha usado **Turtle**, sintaxis que facilita la lectura también para una persona donde cada término de RDF se escribe de la siguiente forma:

- los **URIs** o **IRIs** se escriben entre < >, por ejemplo,

  ```
  <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  ```

  donde `type` es el nombre local. Para una mejor visión de la tripleta, se pueden abreviar estos términos etiquetándolos previamente para hacer referencia a ellos con un *prefijo* (*prefixed name*) del siguiente modo:

  ```
  @prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#> .
  ```

  por tanto, el término anterior queda reducido a `rdf:type`.

- los **literales** se escriben entre comillas.

- los **nodos en blanco** se escriben con un guión bajo seguido del nombre local, por ejemplo, `_:velazquez`.

Últimas dos líneas

Para la creación de una ontología del grafo de preguntas se ha usado el editor de ontologías **Protégé**[3], con el que se ha utilizado vocabulario de `OWL2`[15] para definir clases y relaciones y poder realizar inferencia de éstas sin tener que definir cada una de ellas.

En estas líneas además del modelo RDFS-Plus, se planteó la opción de crear algunas tripletas bajo **RDF-Star**, que sirve para definir como sujeto una tripleta que es una pregunta que a su vez puede tener características propias, por ejemplo:

```
<< :enunciado1 :tiene_opcion :opcion1 >> :tiene_valor :verdadero
```

[3]https://protege.stanford.edu/

que se interpreta como que la proposición `enunciado+opcion` es cierta.

En el momento de extraer información del corpus de preguntas a través de Python fue necesario el uso de librerías de NLP como **textttspaCy**, `NLTK` junto con `CountVectorizer` y `TfidfVectorizer`, para contar el número de términos que hay en un documento y cuántos documentos tienen ese término devolviendo la medida TF-IDF que sigue la siguiente fórmula:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \cdot \text{IDF}(t,d) = \text{TF}(t,d) \cdot (log(\frac{n+1}{df(t)+1}) + 1)$$

donde $\text{TF}(t,d)$ es la frecuencia del término t en el documento d, n es el número de documentos, $\text{df}(t)$ es el número de documentos que contiene el término t.

Se usó `LDA`, modelo *Latent Dirichlet Allocation* [6] para encontrar *topics*, para conseguir más información sobre el conjunto de preguntas.

Capítulo 3

Líneas de investigación

3.1. Explotación de la Web Semántica

Existen diversos proveedores de información en la Web, como la Web Semántica de donde se puede extraer datos de forma masiva para ayudar a equipos docentes de la universidad, por ejemplo, la creación automática o semiautomática de actividades y preguntas de tipo test.

Existen diferencias fundamentales en la información de interés que se puede extraer de un área según la orientación de la asignatura. Por ejemplo, en asignaturas de ciencias o relacionadas con la tecnología, llamadas STEM (*Science, Technology, Engineering, Mathematics*) la información de mayor relevancia procederá de las herramientas, métodos o materiales que se usen, mientras que en otras disciplinas como la historia puede ser más relevante el orden cronológico de unos hitos temporales. En *Chen et al.* [8] se menciona brevemente la diferencia de extraer información de asignaturas STEM con las que enseñan idiomas o relacionadas con la literatura.

La primera línea central de estudio se centraba en la construcción de un KG con los contenidos de un área para ponerlo a disposición del equipo docente. El KG resultante debía estar lo más ajustado posible al dominio de dicho área, sin olvidar ningún elemento de dicho dominio, pero también sin incluir elementos que no fuesen

necesarios.

Aunque los grafos de conocimiento se han definido en el capítulo anterior, una forma simplificada de entender su estructura es saber que representan las relaciones de una colección de entidades (que pueden ser cualquier cosa, objetos y eventos del mundo real o conceptos abstractos), de forma que los nodos son las entidades y las aristas entre los nodos son las relaciones que existen entre dichas entidades.

Tanto las entidades como las relaciones se definen de forma semántica para que tanto personas como máquinas puedan interpretar esta información. Este entrelazado forma un grafo dirigido, de ahí su nombre, como se puede observar en la Figura 3.1.

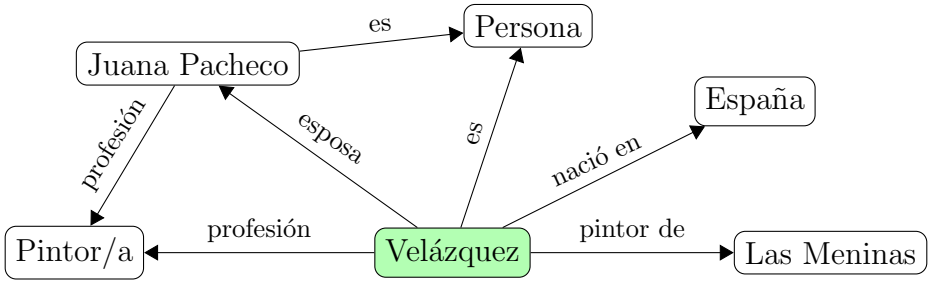

Figura 3.1: *Ejemplo de un grafo de conocimiento, cuya entidad principal es Velázquez y se relaciona con otras entidades a través de cinco atributos (profesión, esposa, nació en, es y pintor de).*

3.1.1. Tareas y métodos

El primer paso para acometer esta línea de estudio consistió en definir y ordenar las tareas o pasos principales para encontrar la información relevante para cualquier asignatura:

1. buscar una fuente de datos adecuada a partir de los términos extraídos de pistas indirectas, es decir, elementos relacionados con el área de estudio como las guías o textos de la asignatura. Principalmente se considerarán aquellas fuentes de acceso libre, para evitar contactar con el equipo docente desde el inicio.

2. analizar el contenido y estructura de la fuente de datos mediante consultas generales

3. proponer un listado de variables por las que hacer consultas de búsqueda, a través de consultas `SELECT`, a partir de una lista de términos clave relacionados con la semántica del área de estudio.

4. establecer los filtros necesarios para ajustar la información que se desee recibir en la respuesta de la consulta.

5. ejecutar y traza de la consulta sobre la fuente de datos.

6. analizar del resultado de la consulta y almacenamiento de los datos.

Para cumplir la primera tarea, primero se intentó buscar una fuente de datos próxima a un área, tomando de ejemplo el grado en *Historia del Arte*, en la LOD Cloud junto con ***Linked Open Vocabularies*** (*LOV*[1]) [24], a partir de una serie de palabras clave del área, extraídas de la guía, concretamente del listado de asignaturas que tiene el grado de la UNED.

Esta aproximación se descartó debido a que la búsqueda no es eficiente por varios motivos: solo se puede realizar a través de un "cuadro de búsqueda" sin más información avanzada; el resultado de la búsqueda es una lista de los posibles *datasets* que pueden contener los términos de búsqueda, pero es necesario consultar dichos *datasets* uno a uno manualmente para saber si alguno es apropiado para construir el grafo; y que no se puede generalizar.

Por este motivo, se decidió usar Wikidata, un nodo de la *LOD Cloud*, con formato abierto y datos enlazados, que permite realizar búsquedas con mayor flexibilidad y facilidad, además de tener un mecanismo de generalización a través del punto de consulta. Además de utilizar esta KB como indexadora de otros *datasets*, se puede explotar como proveedora de información.

En este punto del trabajo, se centró el estudio en Wikidata como posible fuente de datos para una asignatura o curso, estudiando sus características y analizando

[1] https://lov.linkeddata.es/dataset/lov

sus limitaciones. Para ello, la siguiente tarea consistió en explorar esta KB. La exploración de los contenidos se puede hacer desde el mismo punto de consulta que ofrece Wikidata, Wikidata Query Service, mediante consultas `SELECT`.

La exploración del contenido y la estructura de esta KB, se realizó a través de preguntas generales. Una de estas preguntas, que se puede realizar para consultar cualquier grafo RDF, es la siguiente

```
SELECT (COUNT(DISTINCT ?sub) AS ?num_sub)
WHERE { ?sub ?pred ?obj .}
```

que pregunta por la cantidad de elementos distintos que hacen de sujeto, es decir, cuántos elementos están enlazados a otros desde ellos. Siguiendo este procedimiento para consultar cuántos predicados y objetos se han definido. Pero estas preguntas a Wikidata, exceden el tiempo de consulta, debido a que Wikidata es un conjunto de datos muy extenso. Por lo que hay que seguir otra estrategia de exploración.

Wikidata dispone de un vocabulario predefinido que ayuda para entender qué predicados pueden usarse en las tripletas, prefijos de Wikidata[2]. Este vocabulario se compone del estándar predefinido por la W3C, como `rdf`, `owl`, `skos` y `xsd` entre otros, y del lenguaje propio para esta base para definir sus elementos, utilizando los URIs: `wd` para las entidades; `wdt`, `p`, `ps`, entre otros, para las propiedades; `wds` para las declaraciones; y `wikibase` para definir su ontología. (Para más detalle, ver en A).

A través de su lista de propiedades[3], Wikidata permite definir sus elementos, en lugar de usar vocabularios estandarizados como RDF u OWL. La siguiente consulta también devuelve el listado de las aproximadamente 9625 propiedades:

```
SELECT (COUNT(?prop) AS ?n_prop)
WHERE { ?prop rdf:type wikibase:Property.}
```

Para conocer más sobre estas propiedades, en la lista de propiedades se puede ver el número de veces que se usa cada propiedad, actualizado. Las siguientes consultas

[2]https://www.wikidata.org/wiki/EntitySchema:E49
[3]https://www.wikidata.org/wiki/Special:ListProperties

también muestran algunas de las características interesantes a tener en cuenta de las propiedades. Todas redirigen al Wikidata Query Service:

. consulta para conocer el nombre, los alias, y la descripción de cada propiedad.

· consulta para saber el tipo de cada propiedad.

· consulta que devuelve el número de propiedades que hay por tipo.

· consulta para conocer la clasificación de las clases de propiedades.

· consulta para conocer la jerarquía de las clases de propiedades.

· consulta para conocer la jerarquía entre propiedades.

· consulta para conocer ejemplos de uso de propiedades.

Por otro lado, Wikidata no define clases, como otras bases que utilizan los términos `rdf:Class` u `owl:Class`, ni usa un vocabulario propio para ello, como en el caso de las propiedades. Por este motivo, es muy difícil detectar las clases de un elemento y solo se pueden extraer a través de consultas inversas, a partir de las propiedades `P31` (*ser instancia de*), y `P279` (*ser subclase de*). Por ejemplo, se pueden responder preguntas sobre las clases a las que pertenece un elemento y qué subclases tiene utilizando la propiedad `P279`:

con la siguiente consulta se responde a la pregunta de qué clases contienen a la clase `Q5` (*Ser Humano*), dando como resultado 4

```
SELECT ?superclass
WHERE { wd:Q5 wdt:P279 ?superclass }
```

mientras que, con la siguiente consulta se responde a qué clases contienen las clases que contienen a la clase *Ser Humano*, da como resultado 22 clases

```
SELECT ?superclass
WHERE { wd:Q5 wdt:P279* ?superclass }
```

con la siguiente consulta se responde a qué subclases tiene la clase *Ser Humano*, dando como resultado 288 subclases

```
SELECT ?subclass
WHERE { ?subclass wdt:P279 wd:Q5 }
```

mientras que, con la siguiente consulta se responde a qué subclases tiene las subclases de *Ser Humano*, devolviendo 17737 subclases

```
SELECT ?subclass
WHERE { ?subclass wdt:P279* wd:Q5 }
```

Otra forma de conocer las clases que existen alrededor de un elemento es usando la herramienta wdtaxonomy[4]. Para usar esta herramienta es necesario conocer el identificador del término de Wikidata sobre el que se va a consultar y el identificador de la propiedad, por ejemplo, para saber *a qué clases pertenece la clase Ser Humano* se puede utilizar la siguiente consulta:

```
wdtaxonomy Q5 -P 279 -r -c
```

que devuelve el siguiente resultado:

```
human (Q5) •243 ×9624432
├──natural person (Q154954) •54 ×4
├──omnivore (Q164509) •74
├──person (Q215627) •85 ×41
└──Homo sapiens (Q15978631) •68 ×1
```

Una vez completada la segunda tarea, exploración de la fuente de datos, la siguiente tarea es buscar información en ella sobre términos de la asignatura, curso, grado o área de estudio. Para esta tarea es necesario conocer qué términos escoger. Al igual que en el primera tarea, si no se cuenta con el equipo docente desde el inicio, es necesario acudir a fuentes indirectas como las guías docentes para extraer algunas palabras clave que se consideren relevantes. Resaltar de nuevo que Wikidata es una base de conocimiento multilingüe y que por esta razón no importa en qué idioma estén escritos los conceptos a buscar.

Para buscar información de estas palabras en cualquier base de conocimiento, es necesario el uso de la función `CONTAINS`, que filtra la búsqueda por el término buscado, dentro de la cláusula `WHERE` de la siguiente forma:

```
FILTER ( CONTAINS(STR(?variable), 'termino' ))
```

[4]https://wdtaxonomy.readthedocs.io/en/latest/README.html

La ejecución de este tipo de consultas en Wikidata, excede el tiempo de consulta debido a la cantidad de elementos que tiene que comprobar, de nuevo, uno a uno. Por esta limitación, es necesario planificar estrategias de consulta y reducir el tiempo de la búsqueda. La API de MediaWiki es una herramienta que resuelve este problema, permitiendo encontrar tanto elementos relacionados en Wikidata como en Wikipedia. La consulta debe tener la siguiente estructura dentro de la cláusula `WHERE`:

```
SERVICE wikibase:mwapi {
    bd:serviceParam wikibase:endpoint "www.wikidata.org";
                    wikibase:api "EntitySearch";
                    mwapi:search "termino";
                    mwapi:language "es".
    ?item wikibase:apiOutputItem mwapi:item.
    ?num wikibase:apiOrdinal true.
    }
```

donde "termino" es la palabra a buscar. Esta subconsulta devolverá los identificadores de los elementos relacionados, y una vez obtenidos, consultar a Wikidata por ellos en lugar de por su etiqueta reduce el tiempo.

Otra alternativa para evitar la limitación de tiempo por el servidor, es la descarga parcial de datos de Wikidata manteniendo los datos en una caché local. Para este proceso es necesario el uso de librerías de Python como RDFLib y SPARQLWrapper o el uso de consultas `CONSTRUCT` en GraphDB.

Esta descarga de datos se hizo en dos partes. Por un lado, la descarga de todas las propiedades junto con su estructura y con todas sus definiciones. Por otro lado, una descarga que depende del caso de estudio, para descargar solo los elementos junto con sus relaciones que están relacionados con el área.

La primera descarga se puede hacer, o bien a través de una serie de consultas como las definidas antes sobre propiedades, o bien mediante la siguiente consulta que las recoge todas directamente y no da problemas de tiempo:

```
CONSTRUCT { ?prop ?p ?o. }
WHERE {
  ?prop rdf:type wikibase:Property .
  ?prop ?p ?o .}
```

Una vez realizada esta consulta, se pueden utilizar las propiedades para realizar la búsqueda de un dominio. Esta segunda descarga, se realizó para el área del grado en Historia del Arte, pero debido a la gran cantidad de elementos relacionados, se redujo al conjunto de Pinturas.

3.1.2. Caso de estudio

Historia del Arte

Se seleccionó este área con la motivación de crear algo parecido a lo hecho en Google Arts[5] y BNE[6], sin que el usuario tuviese que conocer SPARQL ni el concepto de grafo o base de conocimiento, con búsquedas facetadas a artistas y obras, pero ajustado al grado de Historia del Arte de la UNED.

El objetivo, por tanto, era construir un grafo ajustado para este grado y que fuese de utilidad tanto para docentes como para estudiantes.

Como se ha mencionado en la sección anterior, la aproximación al área se hizo sin contar con ningún equipo docente y por ello fue necesario extraer términos relevantes de la guía del grado, concretamente de la lista de asignaturas que se imparten.

Se realizó una primera aproximación directamente con la palabra *arte*, para conocer la clasificación que se ha formado en torno a este concepto. Se siguieron dos estrategias en paralelo para la búsqueda de esta información: (1) desde arriba, consultas que usan el término "arte" y conocer su taxonomía; (2) desde abajo, consultas sobre términos que podrían pertenecer a clases relacionadas con el arte, por ejemplo, "Las Meninas" y saber a qué clasificación pertenece.

Suponiendo que el término "arte" se corresponde con una clase, se quiere conocer

[5]https://artsandculture.google.com/explore
[6]https://datos.bne.es/inicio.html

la taxonomía de *Arte*. Esta primera idea se puede comprobar rápidamente con la siguiente consulta:

```
SELECT ?subclase ?subclaseLabel
WHERE {
  ?subclase wdt:P279 wd:Q735.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "es,en" }}
```

que devuelve 195 subclases de la clase *Arte*, cuyo identificador es el `Q735`.

Por otro lado, ver qué resultados se obtienen al consultar la clase a la que pertenece la obra de arte "Las Meninas":

```
SELECT ?clase ?claseLabel ?superclase ?superclaseLabel
WHERE {
  wd:Q208758 wdt:P31 ?clase.
  ?clase wdt:P279 ?superclase.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "es,en" }}
```

donde se ve que *Las Meninas* pertenece a la clase *Pintura* con identificador `Q3305213` y que es subclase de otras dos, *Artefacto* y *Two-dimensional visual artwork*.

Para conocer mejor la taxonomía de estas clases, *Arte* y *Pintura*, se utilizó `wdtaxonomy`.

En la Figura 3.2 se muestra un resumen del resultado para el caso de *Arte* resaltando el camino hacia la clase *Pintura* con identificador `Q11629`.

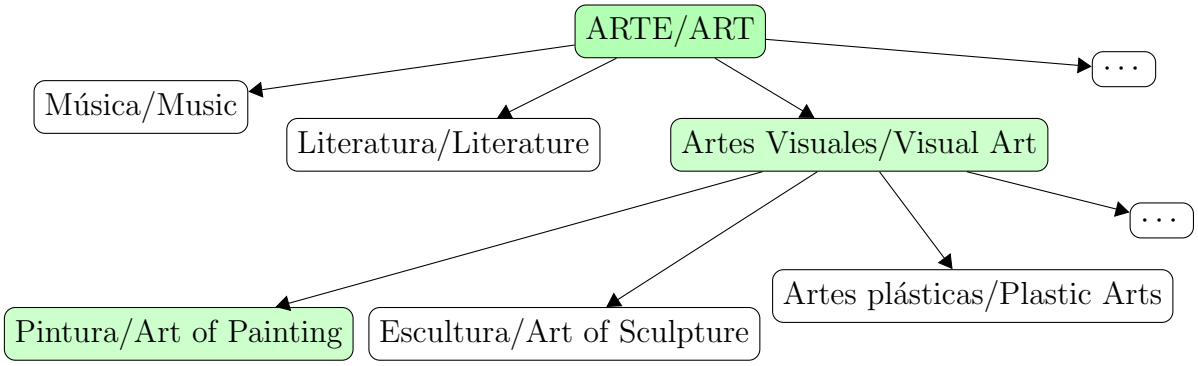

Figura 3.2: *Clasificación extraída desde* `wdtaxonomy` *de la clase Arte de Wikidata*

En la Figura 3.3 se muestra un resumen del resultado para la clase *Pintura* con

identificador `Q3305213` resaltando el camino hacia este nodo.

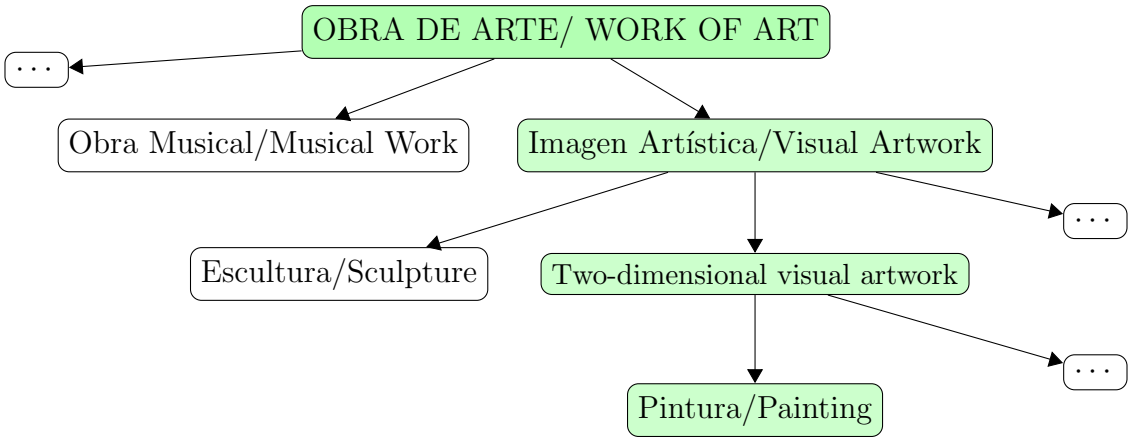

Figura 3.3: *Clasificación extraída desde* ***wdtaxonomy*** *de la clase Pintura de Wikidata*

Resaltar que cuando se estudió por primera vez esta última taxonomía, no existía el nodo `two-dimensional visual artwork`. La clase *Pintura* pertenecía directamente a la clase *Imagen Artística*, de ahí la importancia de mantener un repositorio actualizado, sobre todo si está conectado a una KB dinámica como lo es Wikidata.

Como se ve en los grafos, las dos clases *Pintura* tienen taxonomías diferentes, esto puede deberse a que a la hora de añadir un concepto en este área no se ha tenido en cuenta si ya se había definido o clasificado previamente.

Como ambas clases podrían contener información relevante, se han realizado las dos siguientes consultas, para saber cuántos elementos tiene cada clase de una taxonomía y de la otra y poder elegir de dónde extraer los datos.

1. Consulta que devuelve el número de instancias que tienen las clases `Arte`, `Artes Visuales` y `Pintura`:

```
SELECT ?item_arte ?item_artesvisuales ?item_pintura
WHERE {
  {select (count(distinct ?item) as ?item_arte)
   where {?item wdt:P31/wdt:P279* wd:Q735.}}
```

```
  {select (count(distinct ?item) as ?item_artesvisuales)
   where {?item wdt:P31/wdt:P279* wd:Q36649.}}
  {select (count(distinct ?item) as ?item_pintura)
   where {?item wdt:P31/wdt:P279* wd:Q11629.}}}
```

item_arte	item_artesvisuales	item_pintura
~136753	~46659	~1442

2. Consulta que devuelve el número de instancias que tienen las clases `Obra de Arte`, `Imagen Artística` y `Pintura` (`Q3305213`):

```
SELECT ?item_obrarte ?item_imagenart ?item_pintura
WHERE {
  {select (count(distinct ?item) as ?item_obrarte)
   where {?item wdt:P31/wdt:P279* wd:Q838948.}}
  {select (count(distinct ?item) as ?item_imagenart)
   where {?item wdt:P31/wdt:P279* wd:Q4502142.}}
  {select (count(distinct ?item) as ?item_pintura)
   where {?item wdt:P31/wdt:P279* wd:Q3305213.}}}
```

item_obrarte	item_imagenart	item_pintura
~3334824	~1796837	~569492

Como se ve en los resultados de las anteriores consultas, la taxonomía de la clase *Obra de Arte*, almacena muchos más elementos que la de la clase *Arte*, por ello se ha centrado la búsqueda en la clase *Pintura* con identificador `Q3305213`.

Esta clase tiene 98 subclases, de las cuales, 41 tienen al menos 1 elemento y 57 no tienen ninguno. De las 41 que tienen al menos 1 elemento, la clase *Acuarela* es la que más elementos tiene con 15693 y la siguiente *Pintura al pastel* con 1742. Este resultado se puede ver con la consulta:

```
SELECT (COUNT(?pint) AS ?pints) ?subclase ?subclaseLabel
WHERE {
  ?subclase wdt:P279 wd:Q3305213.
  ?pint wdt:P31 ?subclase.
  SERVICE wikibase:label { bd:serviceParam wikibase:language "es,en". }}
```

```
GROUP BY ?subclase ?subclaseLabel
ORDER BY DESC(?pints)
```

Como se ha visto, la clase *Pintura* llega a tener más de 560000 elementos, y ni el 10 % de éstos se agrupan en las subclases propuestas de *Pintura*.

Se puede ver en la siguiente consulta, que buscando por autor, época o fecha de las pinturas (detalles característicos de éstas) ocurre algo parecido. Esto ocurre porque falta mucha información básica sobre las pinturas almacenadas en Wikidata.

```
SELECT ?pint ?autor ?epoca (YEAR(?fecha) AS ?year)
WHERE {
  ?pint wdt:P31/wdt:P279* wd:Q3305213.
  OPTIONAL {?pint wdt:P170 ?autor}
  OPTIONAL {?pint wdt:P2348 ?epoca}
  OPTIONAL {?pint wdt:P571 ?fecha}}
ORDER BY ?year
```

Siguiendo con la estrategia para evitar la desconexión por exceso de tiempo al consultar por términos a Wikidata (incluso aunque esté restringida la búsqueda a solo la clase *Pintura*, ésta sigue almacenando una cantidad suficientemente alta de elementos para que se exceda el tiempo de consulta al comprobar cada uno de ellos), se ha descargado una caché local de los elementos de *Pintura* junto con sus declaraciones.

Esta descarga se hizo con un script de Python utilizando consultas `CONSTRUCT` para exportar el resultado en un grafo en formato Turtle con las librerías `SPARQLWrapper` y `RDFLib`. Fue necesario planear una estrategia para realizar la descarga, que fue dividir la búsqueda en diez consultas filtrando por el número final del identificador de sus instancias (0, 1, ..., 9). La siguiente consulta muestra el proceso con los identificadores acabados en 0:

```
CONSTRUCT {?s ?p ?o.}
WHERE {
  {SELECT ?p
```

```
WHERE {
  ?prop rdf:type wikibase:Property;
        wikibase:directClaim ?p.}}
?s wdt:P31/wdt:P279* wd:Q3305213. FILTER (STRENDS(str(?s), '0'))
?s ?p ?o.}
```

Se obtuvieron diez grafos separados que había que unir, pero esta unión no se pudo llevar a cabo por el elevado coste computacional desde Python.

3.1.3. Discusión de resultados

Desde el punto de consulta de Wikidata, se puede visualizar el resultado de formas diferentes. El más utilizado es el formato de tabla donde cada columna representa el término buscado, pero también se pueden mostrar en formato de grafo, con un gráfico de burbujas o incluso con imágenes, como puede verse en las figuras 3.4 y 3.5.

La consulta *cuántos elementos tiene cada clase de Pintura* da como resultado la Figura 3.4.

pints	subclase	subclaseLabel
15693	wd:Q18761202	acuarela
1742	wd:Q12043905	pintura al pastel
1031	wd:Q132137	icono
922	wd:Q22669139	fresco
903	wd:Q21281546	aguada
395	wd:Q8362	iluminación
350	wd:Q504073	estarcido
209	wd:Q76317998	Q76317998
202	wd:Q282129	miniatura
152	wd:Q50488927	abanico pintado
142	wd:Q56676227	oil painting
133	wd:Q55439	pintura sobre tabla
107	wd:Q99516640	pintura mural
94	wd:Q19969434	pintura sobre pergamino
80	wd:Q22970505	painted crucifix
32	wd:Q1964917	boceto al óleo
23	wd:Q7282803	Ragamala painting
19	wd:Q2026188	pintura panorámica

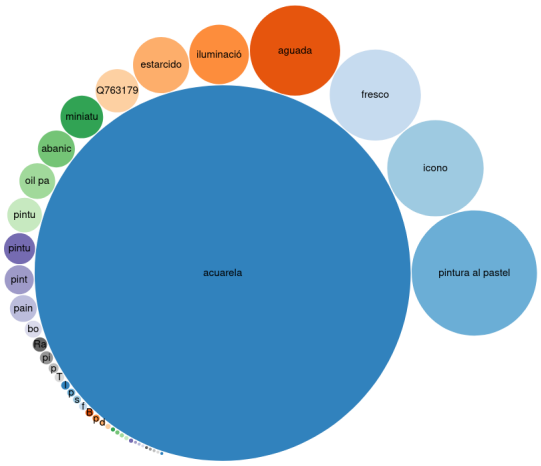

Figura 3.4: *A la izquierda el resultado en formato tabla y a la derecha en formato de gráfico de burbujas (añadiendo al inicio de la consulta* `#defaultView:BubbleChart`*)*

Si se quieren obtener imágenes se puede utilizar el *ImageGrid*, por ejemplo, la consulta *qué pinturas tienen imagen y época marcada* devuelve todas las imágenes

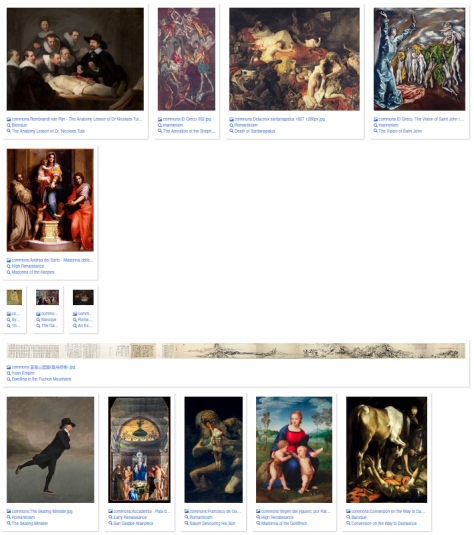

Figura 3.5: *Resultado con imágenes de Wikimedia Commons*

que corresponden con pinturas a las que se le ha añadido la información de la época a la que pertenecen, algunas de éstas se ven en la Figura 3.5.

Las tablas generadas no dan suficiente información entre los términos, solo los posibles resultados de las variables por las que se ha preguntado, y éstas pueden no tener información entre ellas. Para que sea de mayor utilidad para una asignatura o área, es necesaria información semántica entre los elementos de la respuesta, es decir, consultas que relacionen las variables buscadas, como sucede con las consultas de tipo `CONSTRUCT`, que se pregunta directamente por una tripleta, dos entidades enlazadas por una propiedad, y de esta forma devuelve un resultado con estructura de grafo.

Otro problema con el caso de estudio escogido, el grado en Historia del Arte, es que es muy general conteniendo muchos datos de cada tema (pintura, escultura, fotografía, cine, música, literatura). Dividir este subgrafo por asignaturas específicas del grado, conllevaría volver a repetir el proceso por cada asignatura teniendo las mismas limitaciones, como se ha visto con la clase *Pintura*, que no evita la limitación de tiempo, cuando se consulta por un elemento dentro esta clase.

Wikidata, a pesar de sus limitaciones y falta de características básicas en algunos

casos (como lo visto entre las pinturas) sigue siendo una fuente de datos a tener en cuenta porque, por un lado, está en constante crecimiento y los datos almacenados se van actualizando, y por otro lado, la limitación de tiempo se puede evitar mediante estrategias de paralelización.

3.2. Grafos locales de dominio

De los resultados obtenidos en la primera línea de estudio se pueden destacar la falta de información semántica entre los datos de la tabla, generada como resultado, y que para el caso de estudio anterior no fue posible establecer un proceso que se pudiera generalizar y automatizar para diferentes áreas.

Para resolver el problema de la falta de información semántica recogida por tablas, se planteó la construcción de un grafo de conocimiento de dominio, utilizando consultas `CONSTRUCT` en lugar de `SELECT`.

En esta línea, se planteó un cambio de área, que pudiera estar mejor clasificada y un cambio de fuente de datos, una más especializada en el área de estudio y menos general que Wikidata, que abarca muchos y diferentes temas.

El área seleccionada como caso de estudio fue la Inteligencia Artificial (IA) y, la fuente de datos, el grafo de conocimiento generado por la *Open University* sobre este área, llamado AI-KG. La elección de este área surgió de la motivación, por un lado, de ayudar al departamento de IA de la UNED con el nuevo grado sobre este área y, por otro lado, de estudiar un área relacionada con las disciplinas STEM, en contraste con la anterior.

Por tanto, el objetivo de esta línea de estudio consistió en construir un KG sobre conceptos de IA que podría estar a disposición del departamento de Inteligencia Artificial de la ETS de Ingeniería Informática de la UNED. Como objetivo secundario se planteó la posibilidad de utilizar dicho KG como fuente de datos principal para

generar los KGs de las asignaturas del área.

Un grafo de conocimiento se puede construir desde cero, como cualquier base de datos, de forma manual, definiendo las entidades y las relaciones entre ellas, pero este es un proceso muy lento y requiere de conocimiento experto. En esta línea de estudio se llevaron a cabo los siguientes pasos para construir y poblar un grafo:

- extracción de información a partir de otros grafos que ya tienen estructura de tripleta.
- extracción de información a partir de datos semiestructurados, por ejemplo, tablas de contenidos y glosario de libros de texto. Esta información se descargará con una estructura en forma de árbol para luego ser procesada y enlazar los términos de forma semiautomática utilizando herramientas ya existentes como el programa GraphDB o la librería RDFLib de Python.

3.2.1. Tareas y métodos

Al igual que para la primera línea de estudio, primero se buscó información sobre el área de IA en Wikidata, sin embargo, la clasificación generada no se acercó a lo esperado, aunque las declaraciones de cada concepto siguen siendo interesantes para poblar el grafo, por la información que se mantiene de cada una a través de sus declaraciones. Buscando una estructura más acotada, se estudió en paralelo la clasificación realizada en Wikipedia en el artículo sobre IA, con el objetivo de crear una ontología alrededor de estos conceptos.

Tanto Wikidata como Wikipedia son proyectos de edición libre, en los que diferentes usuarios pueden crear contenido. Esto puede conllevar a datos subjetivos, según la opinión de un usuario, y que si el usuario no comprueba la existencia de los datos que se quiere añadir puede derivar a diferentes clasificaciones.

En la Figura 3.6 (izquierda) se muestra el resultado de las subclases directas de la clase IA (`Q11660`), donde se ve que dos de sus clases, *artificial neural network* y *federated learning*, han sido etiquetadas tanto como subclases de la clase principal,

artificial intelligence, como de la subclase, *machine learning*. Esto puede deberse, a que cuando se añadió la información de estos conceptos no se comprobó si ya existían. La Figura 3.6 (derecha) muestra una clasificación de la IA en seis grupos: *Major Goals*, *Approaches*, *Philosophy*, *History*, *Technology* y *Glossary*.

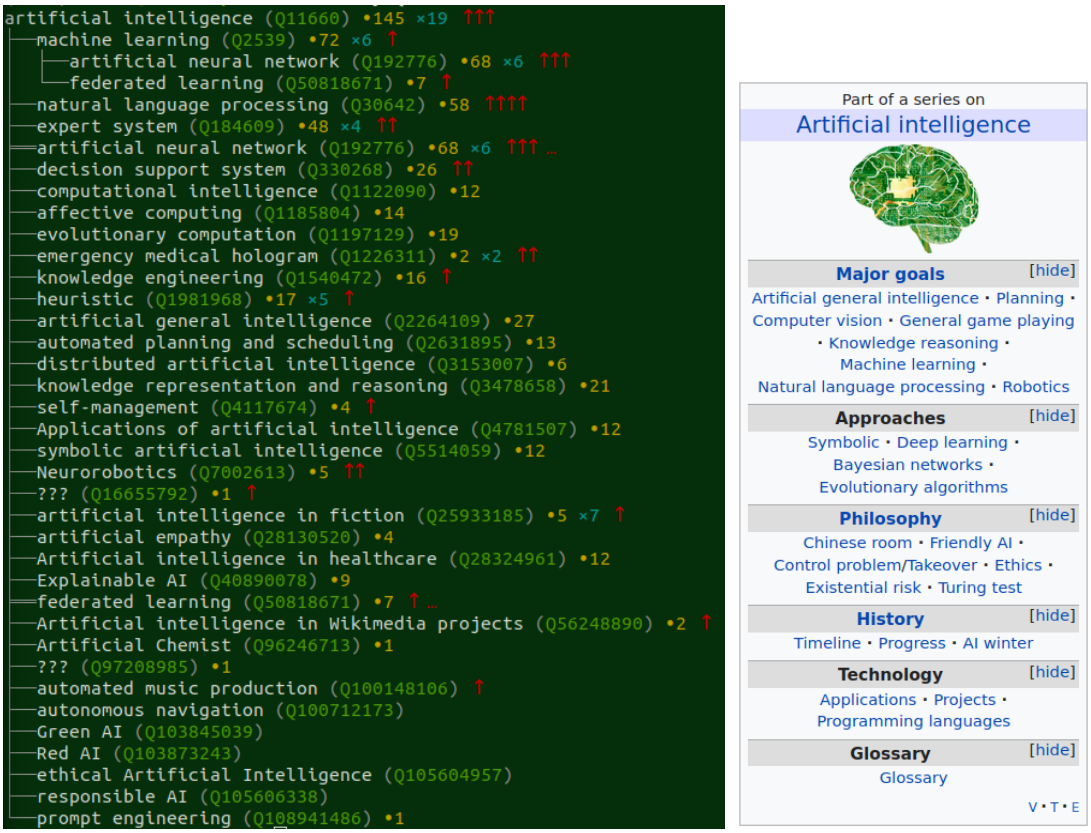

Figura 3.6: *A la izquierda, subclases directas de la clase IA en Wikidata y a la derecha, tabla de partes de IA en Wikipedia, extraída del artículo de IA de Wikipedia*

Como cada término del glosario de IA de la Wikipedia[7] tiene un artículo propio, y esto a su vez contienen referencias al artículo de Wikidata, se puede construir un KG local que siga la clasificación descrita en Wikipedia y contenga los datos de cada término de Wikidata.

Para la construcción de este grafo, fue necesaria la librería *BeautifulSoup* para extraer la información de la página web de Wikipedia y las referencias de cada término que tiene con Wikidata.

[7]https://en.wikipedia.org/wiki/Glossary_of_artificial_intelligence

Esta línea se detuvo debido al descubrimiento de un grafo especializado en el área y cuya ontología se ajusta a la clasificación buscada. Se evaluó AI-KG como fuente de datos principal, utilizando Wikidata como fuente de datos secundaria y de manera indirecta, porque el AI-KG mantiene enlaces con ella.

Siguiendo la tarea 2 de analizar la fuente de datos mediante consultas generales, se sabe que clasifica los conceptos de IA en cinco grandes clases: *Task*, *Method*, *Metric*, *Material* y *OtherEntity*, que éstas a su vez pertenecen a una más general, llamada *ResearchEntity*. Para conocer más detalles sobre este grafo ir al Anexo B.

En paralelo a la fuente de datos de AI-KG, se trabajó en la extracción de información de las tablas de contenidos y los glosarios de dos libros específicos sobre IA: Artificial Intelligence: A Modern Approach 4th edition (en adelante AIMA) y Artificial Intelligence 2E: Foundations of Computational Agents (en adelante ARTINT).

En este punto, se mantendrían tres grafos de conocimiento: un grafo reducido de AI-KG; clasificación y términos de AIMA; y clasificación y términos de ARTINT. De esta forma, si se quisiera generar un KG particular para una asignatura relacionada con IA, se poblaría con datos extraídos de cada uno de estos grafos, a partir de sus términos clave. Los términos clave pueden ser dados por el equipo docente u obtenidos de pistas indirectas, desde guías o textos, como se ha mencionado en la anterior línea de estudio.

A continuación se expone el caso de estudio sobre el máster de IA en la que se detalla cómo se ha construido cada grafo y un caso de ejemplo en el que se utiliza los tres grafos para la generación de un grafo de una asignatura.

3.2.2. Caso de estudio

Para este caso, se construyeron tres grafos de conocimiento: uno con información extraída del AI-KG; otro con información extraída de AIMA; y otro con información

extraída de ARTINT. Cada uno de los grafos se llevó a un repositorio en GraphDB.

AI-KG reducido

La información relevante del AI-KG se extrajo a través del punto de consulta de GraphDB a través de varias consultas `INSERT`:

1. la clase *ResearchEntity*, la clase general que recoge las clases principales del grafo, con sus propiedades (una tripleta)

```
PREFIX aikg-ont: <http://scholkg.kmi.open.ac.uk/aikg/ontology#>
INSERT {?res ?pred ?obj.}
WHERE { SERVICE <https://scholkg.kmi.open.ac.uk/sparqlendpoint> {
        BIND (aikg-ont:ResearchEntity AS ?res) .
        ?res ?pred ?pobj .}}
```

   y las subclases con sus propiedades (15 tripletas) (ser de tipo `Class`, subclases de *ResearchEntity* y su definición con `rdfs:comment`) como se ve en la Figura 3.7

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX aikg-ont: <http://scholkg.kmi.open.ac.uk/aikg/ontology#>
INSERT {?subclase ?pred ?obj}
WHERE { SERVICE <https://scholkg.kmi.open.ac.uk/sparqlendpoint> {
        ?subclase rdfs:subClassOf aikg-ont:ResearchEntity .
        ?subclase ?pred ?obj .}}
```

2. las instancias de cada clase (820.732 tripletas):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX aikg-ont: <http://scholkg.kmi.open.ac.uk/aikg/ontology#>
INSERT {?sub ?tipo ?clase}
WHERE { SERVICE <https://scholkg.kmi.open.ac.uk/sparqlendpoint> {
        ?clase rdfs:subClassOf aikg-ont:ResearchEntity .
        BIND (rdf:type AS ?tipo)
        ?sub ?tipo ?clase.}}
```

Por ejemplo:

`bayesian type Method`

`pattern_recognition type Task`

3. el dominio y el rango de cada clase (52 tripletas):

```
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX aikg-ont: <http://scholkg.kmi.open.ac.uk/aikg/ontology#>
INSERT {?sub ?pred ?clase}
WHERE { SERVICE <https://scholkg.kmi.open.ac.uk/sparqlendpoint> {
        ?clase rdfs:subClassOf aikg-ont:ResearchEntity .
        VALUES ?pred {rdfs:domain rdfs:range}
        ?sub ?pred ?clase.}}
```

Por ejemplo:

`usesMaterial range Material`

`materialIncludedBy domain Material`

4. forma general de añadir las propiedades de los términos, pero, como son más de 2 millones de tripletas, ha sido necesario descomponer la consulta en tres consultas, según el valor del predicado: dominio de clase, rango de clase y *broader* (propiedad para definir que un elemento menos general que otro) y *narrower* (propiedad inversa a *broader*, define que un elemento es más general que otro):

```
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
INSERT {?sub ?pred ?obj}
WHERE { SERVICE <http://192.168.0.14:7200/repositories/AIKG-Completo> {
        ?statement rdf:subject ?sub ;
                rdf:predicate ?pred ;
                rdf:object ?obj .}}
```

Por ejemplo:

`bayesian_network methodUsedBy ai_task`

`bayesian_network broader ai_approach`

`ai_approach narrower bayesian_network`

5. añadir los términos enlazados a Wikidata y a CSO (*Computer Science Ontology*)[8] una ontología sobre áreas de investigación del campo de la ciencia de la computación:

```
PREFIX owl: <http://www.w3.org/2002/07/owl#>
INSERT {?sub ?pred ?obj}
WHERE { SERVICE <http://scholkg.kmi.open.ac.uk/sparqlendpoint> {
        BIND (owl:sameAs AS ?pred)
        ?sub ?pred ?obj .}}
```

   Mencionar que solo unos 26.185 términos de los 820.732 que hay, están enlazados con términos de Wikidata o de CSO.

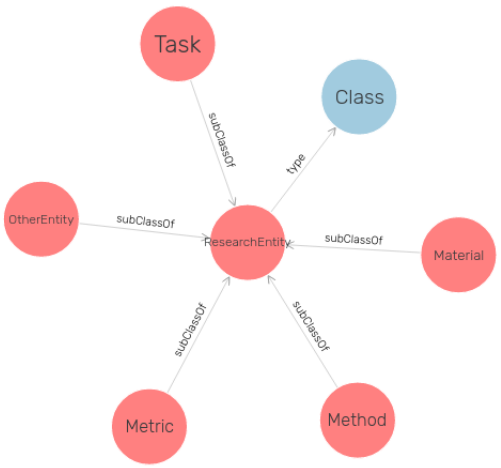

Figura 3.7: *Clases del AI-KG que son subclases de la clase general* ***ResearchEntity****. (Imagen extraída de GraphDB)*

Por último, para que resultara más fácil buscar por el nombre del término entre los tres grafos, a cada elemento de este grafo se le añadió una etiqueta con el nombre del término, en aquellos con más de una palabra, se sustituyó el carácter "_" que separaba las palabras por espacios, por ejemplo, el término `aikg:bayesian_network` tiene por etiqueta "`bayesian network`". De esta forma se normaliza los nombres de las etiquetas.

[8]https://cso.kmi.open.ac.uk/home

Es importante mencionar que, en este punto, los enunciados declarados en el AI-KG se pierden por no ser necesarias algunas de las propiedades que definían a esos enunciados, solo se mantiene la relación entre `rdf:subject`, `rdf:predicate` y `rdf:object` de las declaraciones.

AIMA y ARTINT

Para generar el grafo de AIMA, ha sido necesario crear un `script` de Python para extraer la información desde un pdf, tanto del glosario como de la tabla de contenidos. Estos datos se han transformado en un archivo JSON, en el que se mantiene una estructura de árbol del libro. En dicho árbol los términos están anidados en función del capítulo, sección y subsección al que pertenecen como se ve en la Figura 3.8, clasificación que se sigue para crear la ontología de este grafo.

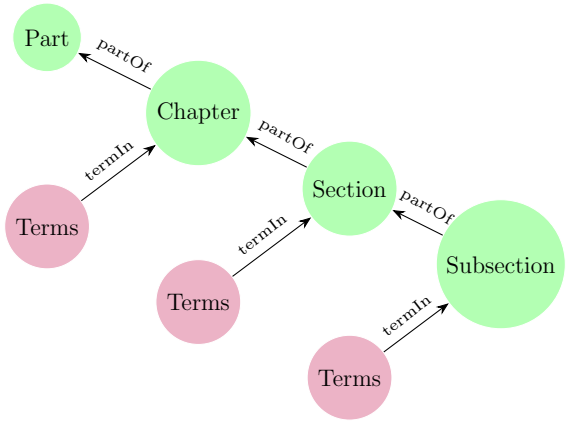

Figura 3.8: *Estructura de secciones del libro AI: A Modern Approach*

Mencionar que cada parte del libro trata sobre una rama específica de la IA, por lo que para extraer información para una asignatura, se podría filtrar por la parte indicada.

Para el caso de ARTINT, la extracción de datos se ha hecho desde su página web utilizando BeautifulSoup. El glosario consiste en una lista terminológica en la que se indica la página de cada término, con lo que utilizando la tabla de contenidos, se puede asociar un término a una sección. La Figura 3.9 muestra la estructura del

libro y que se siguió para construir la ontología de este grafo.

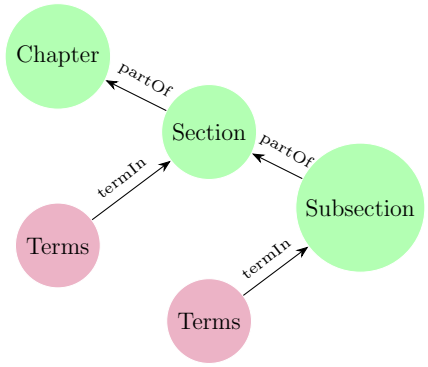

Figura 3.9: *Estructura de secciones del libro AI: Foundations of Computational Agents*

En este caso, hay varios capítulos que hacen referencia a un mismo tema de la IA, por lo que habrá que tenerlos en cuenta a la hora de extraer información para una asignatura.

Una vez descargada la estructura junto a los términos en JSON, de ambos libros, se ha procesado esta información mediante la herramienta OntoRefine interna de GraphDB. Esta herramienta procesa los datos a través de una tabla que permite editar los datos para eliminar, modificar o añadir nuevos. Esta edición se puede realizar con la ayuda de expresiones regulares y funcionalidades propias de la herramienta, como dividir una columna en dos o crear una columna de identificadores de los términos aplicando `md5(value)`.

Una vez depurada la información en las tablas, en el mismo GraphDB, se puede realizar un mapeo para convertir estos datos en tripletas, con relaciones realizadas a mano, a través de *mapping RDF*. La Figura 3.10 muestra un ejemplo del procedimiento de creación de las tripletas para el caso de AIMA (procedimiento similar para ARTINT, sin la sección Parte ni los términos que se enlazan a los capítulos).

Una vez terminado el mapeo a RDF se pueden descargar las tripletas en formato Turtle para su posterior importación en GraphDB.

Mencionar que, para el grafo de ARTINT, hubo problemas con el procesamiento

Figura 3.10: *Tripletas que describen cada uno de los capítulos del libro. De tipo "heading", sus etiquetas, qué páginas corresponden a esta sección, a qué parte pertenece y qué nivel de encabezado es. Para las secciones y las subsecciones se sigue el mismo procedimiento.*

de datos desde OntoRefine, debido a la cantidad de trabajo manual que se requería.

Estos KGs construidos servirán, entonces, para generar uno propio a una asignatura, como se ve en el caso de uso siguiente, sobre la asignatura Métodos Probabilistas. Se explica también el proceso a seguir para extraer información del grafo de ARTINT si se hubiera generado.

Caso de uso: Métodos Probabilistas

El grafo ajustado para la asignatura de Métodos Probabilistas se construyó a través de GraphDB, mediante consultas a los tres repositorios de los grafos construidos anteriormente (debido al problema mencionado antes, el grafo ARTINT estaba incompleto).

Para el grafo de AIMA, se puede restringir la búsqueda a la parte IV, *Uncertain knowledge and reasoning*, que es la que está relacionada con el área. Se ven los capítulos de esta parte en la Figura 3.11.

Para el grafo de ARTINT, se podría restringir la búsqueda a los capítulos 8, *Reasoning with Uncertainty*, 9, *Planning with Uncertainty*, y 10, *Learning with Uncertainty* que son los que se relacionan con esta asignatura.

Para el grafo de AI-KG reducido, se hicieron búsquedas a partir de cinco términos

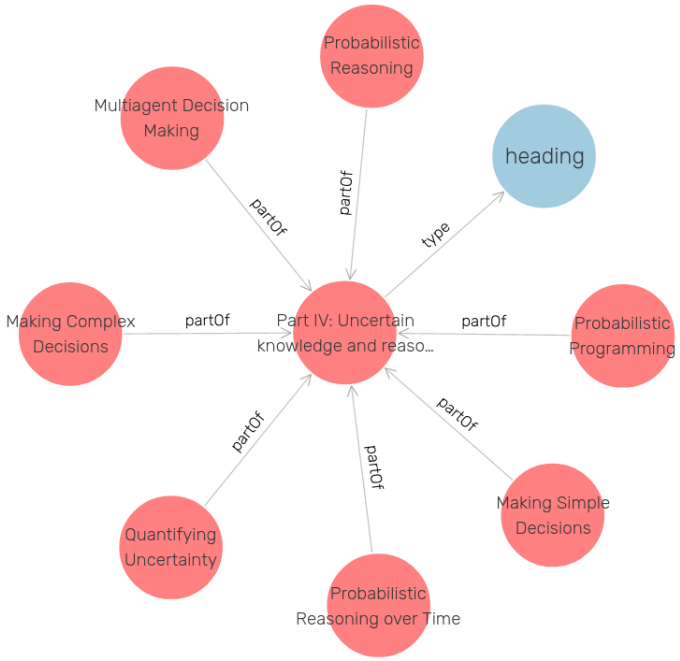

Figura 3.11: *Grafo de secciones de la parte IV del libro AI: A Modern Approach*

clave estudiados en la asignatura. Aunque se hizo una aproximación con los términos de la sección de AIMA relacionada con el área, la búsqueda devolvía muchos conceptos que no se estudian en la asignatura.

Para visualizar algunas de las consultas que se tuvieron en cuenta para tener conocimiento de los grafos construidos, se ha creado un servidor en `http://62.204.199.55:7200/` con dos repositorios, uno para AIMA y otro para el de AI-KG reducido, llamados `aima` y `aikg-terms` respectivamente. Éstas son:

1. Consulta para saber qué términos de AIMA coinciden exactamente con términos de AI-KG y además saber de qué tipo son (es necesario lanzarla desde el repositorio `aima`).

2. Consulta para conocer todos los términos de AIMA que contienen las palabras clave: *"bayesian net"*, *"influence diagram"*, *"naive bayes"*, *"decision tree"* y *"graphical model"* (lanzarla desde el repositorio `aima`).
   Salen 57 resultados, indicando en qué sección están los términos.

3. Consulta para conocer todos los términos de AI-KG que contienen las palabras clave: *"bayesian net"*, *"influence diagram"*, *"naive bayes"*, *"decision tree"* y *"graphical model"* (lanzarla desde el respositorio `aikg-terms`).
   Salen 1227 resultados, indicando a qué tipo de clase pertenecen los términos y cuáles están enlazados a Wikidata y a CSO.
   Ejemplo: `bayesian network model Method cso:bayesian_network_model`

4. Consulta para mostrar los términos filtrados por *"bayesian network"* en AIMA con propiedad *narrower*.
   Ejemplo: `bayesian_network narrower markov_bayesian_network`

5. Consulta para mostrar los términos filtrados por *"bayesian network"* en AIMA con propiedad *broader*.
   Ejemplo: `bayesian_network broader directed_model`

6. Consulta para mostrar los términos filtrados por *"bayesian network"* en AIMA que son evaluados, mejorados, incluídos y usados por otros términos.
   Ejemplos:
   `bayesian_network methodEvaluatedBy complexity`
   `bayesian_network methodImprovedBy artificial_intelligence`
   `bayesian_network methodIncludedBy data_analysis_method`
   `bayesian_network methodUsedBy bayesian_classification_model`

Por último, mencionar que los términos mantienen una referencia de la fuente de datos de la que fueron extraídos. Así se puede consultar al grafo original desde ese término. Como existen términos similares entre los tres grafos, éstos se relacionan con la propiedad `owl:sameAs`.

3.2.3. Discusión de resultados

Este proceso de construcción de un KG para una asignatura dentro un área general ha sido largo dada la cantidad de información extraída de diferentes fuentes de datos, la necesidad de conocimientos previos de éstas para saber qué extraer y la

construcción de varios KGs para el área.

Para los grafos construidos a partir de fuentes no estructuradas, han existido otras dificultades a destacar. El glosario no sigue una estructura común entre los libros. En un libro, este glosario anota directamente el número de la sección a la que pertenece cada término (esta información ayuda a conocer la jerarquía que sigue el término entre las secciones). En el otro libro, este glosario anota la página o páginas donde se encuentra el término.

Esta última anotación obliga a tener que conocer las páginas de inicio y final de cada sección, surgiendo un problema añadido, que es que hay secciones que acaban en la misma página donde comienza otra. Por lo que no se sabe a priori si el término pertenece a una o a otra con solo la información del glosario, teniendo que acudir al libro y anotarlo a mano.

Por último destacar que parte del procesado en OntoRefine se ha hecho de forma manual de forma muy específica para este área y no se podría generalizar el proceso a otras áreas de conocimiento.

3.3. Grafos de preguntas para construir grafos de dominio

Siguiendo la idea de crear un grafo de dominio para una asignatura, se planteó otra línea de estudio, contar con la interacción de expertos en la materia desde el principio. Aprovechando que los dos directores de este trabajo son docentes de asignaturas relacionadas con el área estudiada en la anterior sección (Lógica y *Deep Learning*).

Además, debido a las dificultades encontradas en las dos líneas de investigación anteriores por la falta de generalización del proceso de extracción de información, de creación del KG y la falta de funcionalidad en algunos puntos del proceso, se

consideró cambiar el punto de partida. Se consideró utilizar preguntas de tipo test de exámenes previos de una asignatura, como fuente de datos, por dos motivos principales. Uno fue, el de conocer las preguntas de evaluación que se generarían a partir de un KG de dominio. El otro, el de saber que existe un depósito de exámenes de la UNED, de donde poder extraer preguntas de exámenes sin tener que contar con el equipo docente desde el principio.

3.3.1. Tareas y métodos

Debido a que la fuente de datos es un conjunto de preguntas de tipo test procedentes de exámenes previos, es necesario antes conocer cómo se pueden clasificar las preguntas según su estructura, por si además se pudiera extraer información de otro tipo de preguntas que no sean necesariamente de tipo test. Después construir una ontología que recoja esta información de la forma más general posible y que se pueda realizar inferencia, para no tener que definir todas las relaciones.

Una clasificación general para agrupar las preguntas puede dividirse en tres familias: de confirmación, de búsqueda o de desarrollo donde:

- las de **confirmación** o de **decisión** son aquellas con estructura típica de test, donde dado un enunciado se da la respuesta o respuestas entre diferentes opciones. A su vez, éstas se pueden dividir según el tipo de respuesta, desde elegir entre dos opciones, verdadero y falso, hasta marcar todas las opciones correctas.

- las de **búsqueda** son aquellas donde hay que relacionar de forma correcta una lista de enunciados con una lista de opciones. Esta relación puede ser inyectiva (dos opciones no pueden tener el mismo enunciado, pero alguna opción puede no tener un enunciado), sobreyectiva (todas las opciones deben tener al menos un enunciado) o biyectiva (todos los enunciados y opciones están relacionados uno a uno)

- las de **desarrollo** son aquellas donde solo se da el enunciado, el alumno deberá dar el resultado con la única información de conocer el enunciado.

A través de la **lógica descriptiva** [11] se pueden definir las preguntas formalmente.

- una pregunta siempre tiene un enunciado

$$\texttt{Pregunta} \sqsubseteq \exists \texttt{preg\_enunciado}\,.\,\texttt{Enunciado}$$

- una pregunta de confirmación es una pregunta que tiene un enunciado y diferentes opciones, al menos una

$$\texttt{Conf} \sqsubseteq \texttt{Pregunta} \sqcap \exists \texttt{preg\_opcion}\,.\,\texttt{Opcion}$$

- una pregunta de búsqueda es una pregunta que tiene diferentes enunciados y diferentes opciones

$$\texttt{Busq} \sqsubseteq \texttt{Pregunta} \sqcap \exists 2\texttt{preg\_enunciado}\,.\,\texttt{Enunciado} \sqcap \exists 2\texttt{preg\_opcion}\,.\,\texttt{Opcion}$$

- una pregunta de desarrollo es una pregunta que solo tiene un enunciado.

$$\texttt{Des} \sqsubseteq \texttt{Pregunta}$$

- una pregunta sólo puede pertenecer a una familia, es decir, las preguntas de confirmación, búsqueda y desarrollo son disjuntas entre ellas.

$$\texttt{Conf} \sqcap \texttt{Busq} \sqcap \texttt{Des} \sqsubseteq \bot$$

Ontología para la gestión de preguntas

El diseño de esta ontología se ha producido en cuatro etapas, respectiva e incrementalmente con los siguientes objetivos:

1. **Facilitar el poblamiento más usual de la ontología**, con los tipos de preguntas habituales. Para ello hay que determinar las clases y propiedades básicas para este alojamiento.

2. **Facilitar clasificaciones dinámicas de preguntas en la clase Pregunta**, mediante el uso de patrones ontológicos que permitan definir jerarquías independientes de subclases respecto a criterios distintos. Algunos de estos criterios taxonómicos serán tan transversales como, por ejemplo, el tipo de presentación de la pregunta; pero otros pueden particularizarse a cada asignatura.

3. **Facilitar la descomposición de preguntas 'de tipo test' en sus proposiciones individuales**, donde cada proposición_k se compone del enunciado común y de la respectiva opción_k de la pregunta. Esta clase adicional de Proposiciones consta ahora de sentencias concretas sobre el dominio y resultará ser una vía para una próxima modelización del dominio

4. **Facilitar clasificaciones dinámicas de proposiciones en la clase Proposición**, de forma similar a la clasificación de las preguntas. Y de nuevo, respecto a algunos criterios transversales, como la agrupación de proposiciones por su valor de verdad; pero con la posibilidad de añadir otros criterios particulares de dominio, usando los mismos patrones ontológicos.

La clase primitiva básica es `Pregunta`. Las preguntas tienen al menos un enunciado y de hecho se fija como condición necesaria que así sea. Se opta por crear una clase de individuos de tipo `Enunciado` en vez de considerarlos de entrada como meros literales. Estos individuos de tipo `Enunciado` pueden contener tanto texto como referencias externas (como a imágenes o a datos). La declaración de estas primeras propiedades y clases, en formato Turtle, se encuentra en el Cuadro 3.1

Estas dos clases permiten el poblamiento de preguntas de desarrollo. El otro tipo de preguntas habituales lo constituye toda la familia de preguntas de test diversas. Cada una de estas preguntas requiere, además de un enunciado, varias opciones. Opciones que además pueden tener asignadas cada una un valor de verdad.

De forma similar a las declaraciones anteriores, se declaran las clases `Opcion` y `ValorVerdad`. Para un individuo de la clase `Pregunta` se puede usar la propiedad `preg_opc` para asignarle un individuo de la clase `Opcion`. Estas opciones pueden

```
:Enunciado rdf:type owl:Class .
:Pregunta rdf:type owl:Class ;
    rdfs:subClassOf [
        rdf:type owl:Restriction ;
        owl:onProperty :preg_enunc ;
        owl:someValuesFrom :Enunciado ] .

:preg_enunc rdf:type owl:ObjectProperty ;
    rdfs:domain :Pregunta ;
    rdfs:range :Enunciado .
:enunc_texto rdf:type owl:DatatypeProperty ;
    rdfs:domain :Enunciado .
:enunc_ref rdf:type owl:ObjectProperty ;
    rdfs:domain :Enunciado .
```

Cuadro 3.1: *Clases* `Pregunta` *y* `Enunciado`

presentar tanto texto como referencias externas (como a imágenes o a datos) a través de las propiedades `opc_texto` y `opc_ref`. Y cada opción puede tener asignada un valor de verdad mediante la propiedad `opc_valor`, con rango en el conjunto enumerado de individuos `ValorVerdad` $\equiv$ {`verdadero, falso, indefinido`}.

Hay otros tipos de preguntas, además de las preguntas de desarrollo o de selección. Tan sólo se considerarán de momento las preguntas de tipo 'matching', las de tipo correspondencia o las de tipo ordenación (reducibles a las anteriores). En estas preguntas el enunciado propone encontrar el emparejamiento correcto entre dos conjuntos. Si se quiere gestionar este tipo de preguntas hay que facilitar la enumeración de los elementos del dominio y de los elementos del rango que considera la pregunta. Para ello se propone una clase adicional `ItemRelacionable` y una propiedad `preg_item_dominio` y otra `preg_item_rango` desde individuos de la clase `Pregunta` a sus componentes de dominio y rango, todos ellos en la clase `ItemRelacionable`.

La visualización de la ontología que ofrece Protégé (mediante su plugin de Ontograph) hasta este punto es la que se observa en la Figura 3.12.

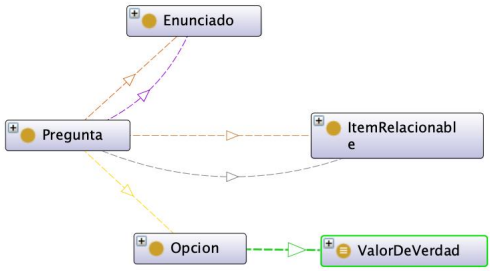

Figura 3.12: *Clases básicas en la primera etapa de diseño*

La implementación adecuada de algunas facilidades en ontologías se ve beneficiada por la existencia de patrones ontológicos aplicables a tareas abstractas muy repetidas. Un indexador de estos patrones se encuentra en `http://ontologydesignpatterns.org/wiki/Main_Page`.

Para esta tarea en concreto se va a utilizar el Esquema de Clasificación Facetada (`http://ontologydesignpatterns.org/wiki/Submissions:Faceted_Classification_Scheme`), por su versatilidad.

Para producir una clasificación de individuos en la clase `Pregunta` se declara una clase 'espejo' `PregCriterio`, como contenedor de los diversos criterios de clasificación aplicables. Inicialmente se pretende la clasificación de las preguntas por su tipo funcional de presentación, y por ello se declara una subclase `CriterioFuncional` en `PregCriterio`. Esta clase `CriterioFuncional` puede incluir toda la jerarquía de subclases que se necesite para desarrollar esta perspectiva de clasificación. En principio, solo se considerará una subclase de `CriterioFuncional`, que denominaremos `CritDesarrollo`.

Cada una de estas clases, que van desgranando criterios, deben contener al menos

un individuo, que representará a la clase y con el cual se podrá relacionar una pregunta para declararse como de ese tipo, dentro de ese criterio. Para representar a la clase `CritDesarrollo` se declara un individuo de esa clase, que denominamos, por ejemplo, `critDesarrollo`.

En la clase `Pregunta`, cuyos individuos se desean clasificar, se replica esa estructura: se declara una subclase `PorTipoFuncional` y una subclase de esta última como `TipoDesarrollo`. Con estos contenedores dispuestos, se supone que se define un individuo pregunta (preg1) y se enlaza con el individuo que le caracteriza en el espacio de criterios: `preg1 preg_crit critDesarrollo`. Este individuo pregunta resultará (tras aplicación del razonador) dinámicamente clasificado en la subclase `TipoDesarrollo` porque esta clase se definirá equivalente a "la de las cosas que están relacionadas por `preg_crit` con algún elemento de la clase `CritDesarrollo`".

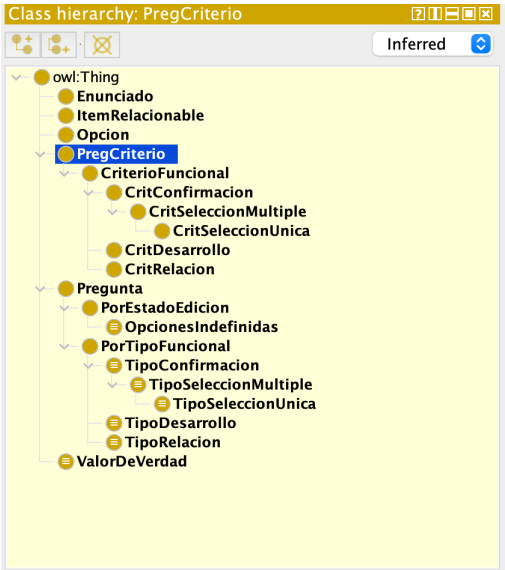

Figura 3.13: *Clasificaciones multicriterio en la clase* `Pregunta`

La Figura 3.13 muestra la aplicación del patrón anterior para la definición de un par de jerarquías independientes de subclases dentro de la clase `Pregunta`. Para una de ellas, por el tipo de pregunta, se usa el espacio de criterios `CriterioFuncional`, implementado como una jerarquía de subclases dentro de la clase `CriterioFuncional`.

Si esta última contuviera otras jerarquías de criterios, se podría usar adicionalmente de forma independiente sobre los individuos de `Pregunta`. Adicionalmente, también se pueden usar otras clases externas a `Pregunta` para caracterizar nuevas líneas de clasificación de los individuos de `Pregunta`.

Es el caso del segundo eje de clasificación en `Pregunta (PorEstadoEdicion)` que produce una clasificación independiente de las preguntas (de confirmación, de test) según contengan o no alguna opción aún en estado de borrador (con valor 'indefinida').

Para probar este marco de clasificación se definen tres individuos distintos de la clase `Pregunta`:

- `pregConfirmacionVFI`: con relación `preg_crit` con el individuo `critSeleccionMultiple` de la clase `critSeleccionMultiple`. Y adicionalmente, con enlace a las 3 opciones de esa pregunta de tests: `preg_opc opc1, preg_opc opc2, preg_opc opc3`. Donde estas opciones son individuos de la clase `Opcion`, relacionadas respectivamente cada una con los valores falso, verdadero e indefinido de ValorDeVerdad.

- `pregDesarrollo`: con relación `preg_crit` con el individuo `critDesarrollo` de la clase `CritDesarrollo`.

- `pregRelacion`: con relación `preg_crit` con el individuo `critRelacion` de la clase `CritRelacion`.

La Figura 3.14 muestra dos justificaciones que ofrece Protégé para respaldar la inferencia sobre la clasificación de ese individuo: primero como miembro de la clase `TipoSeleccionMultiple` y después como miembro de la clase `OpcionesIndefinidas`.

Conviene resaltar que si bien uno de los ejes de clasificación de Preguntas requiere la caracterización explícita de cada pregunta (vía relación con el individuo adecuado

en la clase `PregCriterio`), el otro eje de clasificación surge de la utilización indirecta de los valores de verdad de cada opción de una pregunta. En esta línea, se trabajará para construir clasificaciones de interés (generalistas o específicas) para el banco de preguntas mantenido por esta ontología en cada asignatura.

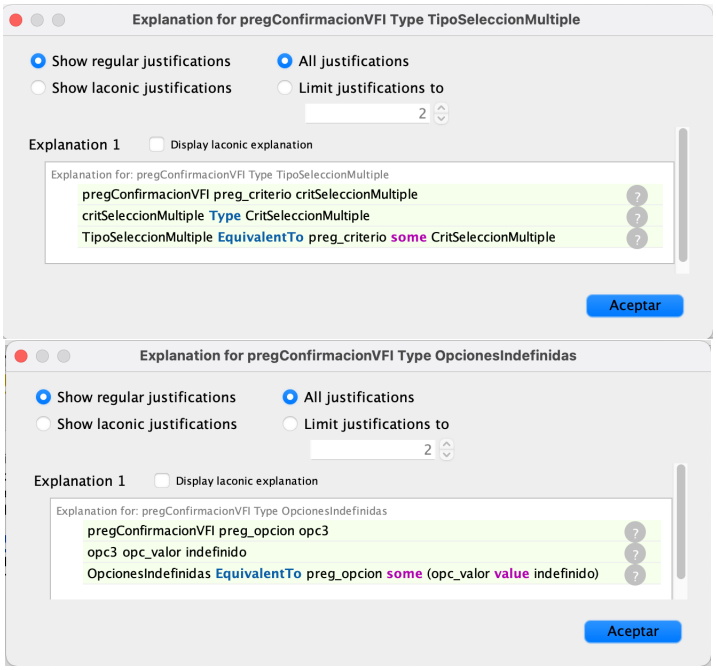

Figura 3.14: *Clasificación automática del individuo* `pregConfirmacionVFI`

En este punto se consideró de interés declarar una nueva clase `Proposicion` para albergar las diversas proposiciones en que puede descomponerse una pregunta de test. Una pregunta así, con un enunciado y K opciones, resulta en tantos problemas independientes de decisión, de confirmación, como opciones tenga la pregunta. Cada una de las proposiciones consta del enunciado común y de una opción y la tarea requerida al usuario es confirmar (`v`/`f`) cada una de estas proposiciones independientes.

Desde el punto de vista funcional, el enunciado contiene tanto instrucciones como datos del dominio, así como cada opción, y una proposición se puede considerar una sentencia relacional $R(d1, d2, ..., dn)$ donde las d_k son los datos que aparecen tanto en el enunciado como en la opción.

En el proceso de poblamiento de la ontología, por cada pregunta de test se crean tantos individuos en la clase `Proposicion` como opciones tenga esa pregunta. La relación entre un individuo pregunta y las K proposiciones generadas se implementa mediante la relación `preg_opc`. Adicionalmente, se relaciona cada individuo proposición tanto con su opción (en la clase `Opcion`) como con su enunciado (en la clase `Enunciado`) correspondiente. El enlace de cada proposición hacia su opción es explícito pero el enlace hacia su enunciado se produce de forma automática.

La siguiente justificación de Protégé confirma cómo se produce el enlace de la proposición (prop1) hacia su enunciado correspondiente (`enuncConfirmacion1`), sabiendo que esa proposición lo es de una pregunta con ese enunciado y de que el predicado `prop_enunc` se declara por composición de otros dos predicados.

- `prop_preg o preg_enunc SubPropertyOf: prop_enunc`
- `preg_prop InverseOf prop_preg`
- `pregConfirmacionVFI preg_prop prop1`
- `pregConfirmacionVFI preg_enunc enuncConfirmacion1`

Mediante una justificación similar, se explica por qué el razonador genera automáticamente la relación entre la proposición (prop1) y su valor de verdad, que es la de su opción correspondiente (`falso`). Así, hasta el momento, las nuevas clases y propiedades creadas permiten poblar la clase `Proposicion` a partir de la información ya almacenada en la ontología de una forma semi-automatizada.

Otra tarea distinta consiste en facilitar clasificaciones superpuestas sobre la clase `Proposicion`. De forma análoga a como en la clase `Pregunta`, ahora es posible definir subclases de la clase `Proposicion` con la intención de que agrupen automáticamente (es decir, por inferencia) todas las proposiciones con valor `verdadero`, `falso` o `indefinido` respectivamente. En concreto, la clase `PropFalsa` se define como equivalente

a "`Proposicion and prop_valor value falso`". La siguiente justificación muestra por qué el razonador asigna el individuo (prop1) a la clase `PropFalsa`:

- `prop1 Type Proposicion`
- `prop1 prop_opc opc1`
- `prop_opc o opc_valor SubPropertyOf: prop_valor`
- `opc1 opc_valor falso`
- `PropFalsa EquivalentTo Proposicion and (prop_valor value falso)`

Manteniendo estos esquemas de clasificación facetada, por clasificación explícita o inferida, se pueden superponer sobre todo el conjunto de individuos de la clase `Proposicion` diversos ejes independiente de clasificación. Desde los muy transversales, como la agrupación de todas las opciones verdaderas, hasta los más específicos de la asignatura.

Conviene resaltar, en este punto, algunos aspectos sobre el interés de este tipo de soporte ontológico en el banco de actividades de una asignatura, especialmente en el caso de preguntas de test:

+ tanto algunos enunciados como algunas opciones pueden repetirse, formar parte de más de una pregunta; en todo caso, estos individuos enunciado e individuos opción se registran y gestionan sin réplicas, permitiendo esa reutilización

+ ante lo que cabe posicionarse sobre su valor de verdad es ante una proposición (ante cada composición 'enunciado'+'opción'), y la descomposición de preguntas en proposiciones aísla lógicamente este comportamiento y facilita su gestión

+ clases como `PropVerdadera` incluyen dinámicamente todas las proposiciones con valor de verdad 'verdadero', no importa de qué pregunta provenga, es decir, este tipo de clases agrupa por un lado enunciados verdaderos muy directamente cercanos al dominio de la asignatura y por otro (en clases como `PropFalsa`) todos los distractores utilizados en las preguntas

+ la clasificación inducida en `Proposiciones` respecto a su valor de verdad se puede ver complementada con otras, que se pueden implementar con el uso de una clase instrumental `PropCriterios` donde se encapsulen esos criterios adicionales. Y eso permite clasificar las proposiciones atendiendo al tipo de relación (de dominio) y de entidades (de dominio) utilizadas. Así, en el caso de la asignatura de Lógica, aparecen propiedades como 'satisfacibilidad' o 'validez' o relaciones como 'equivalencia', 'satisfacibilidad' y 'consecuencia' que se pueden utilizar para agrupar las proposiciones en los que aparecen esas relaciones.

Conclusión

A partir de esta estructura genérica se puede clasificar cualquier colección de preguntas de tipo test, y crear un grafo de preguntas automáticamente.

De esta forma, cualquier información extra que se quiera añadir de una asignatura particular sobre una pregunta, como el profesor que la propuso; el año o años en las que se preguntó en un examen; o a qué tema pertenece, pueden añadirse como atributos a la clase `Pregunta`.

Por otro lado, haber creado la clase `Proposicion`, ayuda a relacionar este grafo de preguntas con el de dominio, a través de entidades y relaciones que aparecen en una pregunta y que se pueden extraer de cada proposición.

3.3.2. Casos de estudio

Esta estructura se aplicó en dos asignaturas que imparten los dos directores de este trabajo para contar con ellos como expertos a lo largo de la construcción de los grafos de conocimiento. Para la construcción del grafo de preguntas, se contó con un conjunto de preguntas de tipo test, unas realizadas en exámenes de años anteriores y otras como posible propuesta para futuros exámenes, todas con solución.

Caso de estudio: Lógica

Para este caso particular, se añadió también a cada pregunta el tema al que pertenece dentro de la asignatura, concretamente se indicó si representan si pertenecen a la *lógica de predicados* o a la *lógica de proposiciones*. Como se ha indicado, la ontología propuesta para el grafo de preguntas permitió añadir dicha información sin necesidad de hacer cambios estructurales.

Los términos más representativos de la asignatura que aparecen en las preguntas, son fórmulas en su mayoría, y éstas son difíciles de clasificar. Por ello, fue necesario el experto, para la construcción del grafo de dominio, siendo los elementos muy específicos por su terminología matemática. A su vez, se pueden relacionar estos términos entre ellos mediante inferencias como la equivalencia, si estos elementos son fórmulas.

Caso de estudio: *Deep Learning*

Este caso se ha utilizado para planificar un flujo de trabajo que ayude a generalizar el proceso de construcción de un grafo de dominio a partir de un grafo de preguntas sin que sea necesario un experto. Partiendo de la situación ideal de que se cuenta con un experto y éste, por un lado, va a facilitar la fuente de datos necesaria para comenzar y, a lo largo del proceso, va a evaluar los resultados generados y compararlos con los esperados.

Dado el conjunto de preguntas, facilitado por el docente, se creó el grafo de preguntas, utilizando la ontología descrita, sin ningún atributo particular añadido, es decir, se poblaron las clases `Pregunta`, `Enunciado`, `Opcion`, `Proposicion` y `ValorVerdad`, siendo estos últimos solo dos (`verdadero` y `falso`).

En paralelo se crearon dos corpus de preguntas, que consisten es una lista de proposiciones, uno que reúne todas las proposiciones y otro que solo tiene en cuenta las ciertas. Éstos, se generaron para extraer términos, según el número de aparición

de cada término en cada pregunta y según la medida TF-IDF, y una vez obtenidos se compararon con el glosario de términos facilitado por el docente.

La búsqueda de términos clave del corpus, se centró en los 25 términos primeros términos con la medida más alta de TF-IDF, sobre su número de aparición en una pregunta frente al número de preguntas que lo contiene.

Debido a que los términos pueden constar de varias preguntas se dividió en unigramas, bigramas y n-gramas con $n > 2$. Y la aplicación de *stopwords* necesariamente para los dos primeros, debido a que salen términos muy genéricos como las preposiciones o términos compuestos por un determinando y un sustantivo (caso de los bigramas). Para los n-gramas se tuvo en cuenta tanto con *stopwords* como sin ellos.

Destacar que la lista predefinida de *stopwords* en castellano de NLTK contiene 313 y de spaCy 551. Se probó con ambas y no daban resultados muy distintos, así que para hacer todas las combinaciones anteriores solo se utilizó spaCy, por simplificar los resultados. Para este caso, se usan palabras inglesas con frecuencia, así que quizá haya alguna palabra no eliminada.

TdfidfVectorizer fue la primera función que se utilizó, pero devuelve pocos términos y no son clave de la asignatura ("eliminar", "reducir tamaño", "representación normalizar pesos"), por eso se tuvo más en cuenta CountVectorizer, que devuelve términos algo más representativos como "aprendizaje supervisado" o "red neuronal", pero la mayoría son muy genéricos.

Para conocer más sobre la información que pueden dar los términos, se usó LDA, pero en este caso, los términos de esta asignatura están muy relacionados entre ellos y muchos aparecen en varias preguntas, evitando así que salgan tópics con cierta relevancia. Aún así, el número más alto posible detectado es 6.

Por último, con esta lista de términos afinada y ajustada, acudir a una fuente de datos estructurada siguiendo la lista de tareas descrita en la primera sección de este capítulo y generar el grafo de dominio a partir de información ya estructurada.

Se buscó la entidad de Wikidata y el artículo de Wikipedia asociado a cada término del glosario, mediante un script de Python que lanza consultas mediante MWAPI.

Hay palabras que por su traducción al castellano, no están registradas por ese nombre ni en Wikidata ni en Wikipedia, como el término "abandonar", y devuelve términos similares, pero que no tienen nada que ver con este área. En este caso se devuelve "salir" como elemento de Wikidata y el artículo "Abandono de sí" de Wikipedia.

Hay otras con las que acierta sin ningún problema, como el caso del término "análisis de sentimiento" que se enlaza con "sentiment analysis" cuyo identificador en Wikidata es `Q2271421` y el artículo homónimo de Wikipedia.

Hay otros términos, como "arquitecturas de red", "AWS Machine Learning" o incluso "capa convolucional" que no dan ningún enlace a Wikidata, esto se debe a que busca el nombre de la etiqueta completo y no se ha etiquetado ninguna entidad por esos nombres. Pero, en Wikipedia, estos tres términos se corresponden con el artículo "Aprendizaje profundo", que tiene que ver con la asignatura.

3.3.3. Discusión de resultados

Por un lado, el grafo de preguntas a través de la ontología creada previamente, queda bien clasificado.

Por otro lado, la búsqueda de términos clave a través de CountVectorizer o TfidfVectorizer, no resultó muy próxima al área, quedando pendiente usar estrategias para afinar esta extracción, como mediante el uso de técnicas PoS Tagging, para detectar sustantivos, adjetivos o verbos.

El uso de LDA, en estos dos casos, no devuelve tópicos diferentes, debido a que los términos utilizados en cada pregunta se parecen o se repiten con una frecuencia alta en otras, o en el caso de fórmulas que solo aparezca una vez en una pregunta, ocurriendo con varios términos y no terminan teniendo un peso alto.

Los resultados de las búsquedas, por los términos clave de una asignatura, a Wikidata y Wikipedia, en general, no resultaron estar desconectados del área. Pero para términos muy generales, como la palabra "clasificación" o "prueba" o términos muy concretos determinados por siglas como "SeLU" o "SGD" los resultados se alejaban mucho de la asignatura. Por esto último, hay que eliminar antes estos términos, para no añadir información acerca de ellos al grafo de conocimiento, aquí es necesario la intervención de un experto.

El grafo de dominio para Lógica no sirve de ejemplo para generalizar otras asignaturas. Es demasiado específico usando terminología propia.

3.4. Escalabilidad en la construcción de grafos de dominio

En paralelo a la creación del grafo de dominio para la asignatura de Lógica y conjuntamente con la línea de estudio de generalizar el proceso de creación de un grafo de dominio a través de *Deep Learning* como ejemplo, se propusieron dos asignaturas como casos de estudio donde comprobar este flujo de trabajo y ver hasta dónde es posible llegar sin contar con los expertos en el área.

Por un lado, se sabe de la línea anterior, que para crear el grafo con conocimiento experto, caso de Lógica, se requiere demasiado tiempo y esfuerzo del equipo docente y que no es viable institucionalmente.

Por otro lado, es viable la generación de un grafo de dominio a través del flujo de trabajo propuesto por encima en el caso de *Deep Learning*, aunque fuera necesaria la interacción de los expertos en algún momento, para refinar la información extraída.

El proceso propuesto en el caso anterior, se refinará aquí, debido a que surgen nuevas tareas a seguir si se pospone el contacto con el experto.

Existen dos depósitos de exámenes a disposición de alumnos y profesores de la UNED, de todas las asignaturas de todos los grados y másteres que se imparten en la universidad. Éstos son el de Calatayud[9] y el de Barbastro[10] donde, en algunos casos, también hay soluciones.

3.4.1. Tareas y métodos

La descarga de exámenes se hizo desde el depósito de Calatayud, que contiene más que el de Barbastro, al menos en las dos asignaturas de estudio que se han seleccionado. Mediante un *script* en Python se automatizó la descarga local de todos los exámenes disponibles estas asignaturas.

Los exámenes, tanto de Calatayud como de Barbastro, no están en su formato digital original, sino que se guarda una copia escaneada de cada uno. Esto genera la necesidad de hacer un preprocesamiento para convertir PDFs de imágenes a texto plano. Esta conversión se ha realizado con un *script* de Python, utilizando la librería `pytesseract` para realizar el OCR y corrigiendo los errores encontrados manualmente, debido a las peculiaridades de presentación que tiene cada asignatura e incluso los distintos modelos a través de los años dentro de la misma.

Una vez extraído el texto de las preguntas del examen, se han separado los enunciados de las opciones para cargarlos en el grafo y trabajar a partir de aquí como se ha hecho en la línea de estudio anterior.

Los exámenes disponibles en estos repositorios, no cuentan, en su mayoría, con soluciones. Por este motivo, al no conocer si son ciertas o falsas las opciones se consideró el valor `indefinido` en la ontología, visto en la sección anterior.

En los casos en los que no se conozca la solución, solo se puede generar un corpus

[9] http://www.calatayud.uned.es/examenes/examenes_step_0.asp
[10] https://www.unedbarbastro.es/Default.aspx?id_servicio=14

de preguntas por asignatura consistente en todas las proposiciones, independientemente de si son ciertas o falsas. A partir de este corpus, se ha seguido la misma extracción de términos con CountVectorizer.

A partir de estos términos, se han realizado consultas con MWAPI, para buscar más información, tanto en Wikidata como en Wikipedia. Esta información ha servido para desambiguar conceptos; descartar aquellos que no están relacionados con el área; y para la descarga parcial de las clases existentes alrededor del término para la construcción del grafo de dominio.

Conclusión

El flujo general resultante del proceso para crear el grafo de dominio de una asignatura es el siguiente:

1. Descarga de exámenes del depósito.

2. Procesado previo de estos exámenes, debido a que están escaneados y éstos se guardan en formato pdf. Convertir imágenes a texto.

3. Convertir texto a formato excel o CSV, para dividir enunciados y opciones de las preguntas.

4. Construir el grafo de preguntas utilizando la ontología descrita en la sección anterior.

5. Generar uno o dos corpus de preguntas, según la información disponible (soluciones o no).

6. Extraer términos clave del corpus utilizando técnicas de NLP.

7. Extraer información de otras fuentes de datos estructuradas utilizando los términos.

3.4.2. Casos de estudio

La elección de estas asignaturas, fue motivada en parte por generalizar este proceso sobre cualquier tipo de área, siendo *Psicología de las Diferencias Individuales* e *Historia Económica Mundial* áreas ajenas a lo estudiado previamente, además de que sus docentes están interesados en mejorar la calidad docente de la UNED y este procedimiento puede ser de gran utilidad.

Caso de estudio: Psicología de las Diferencias Individuales

Para esta asignatura se han analizado 9 exámenes, de los 53 que hay en el depósito de Calatayud. Cada examen tiene aproximadamente unas 40 preguntas y 3 opciones por cada pregunta.

Debido a que existen diferentes modelos de examen, en este caso, hay tres: los divididos en dos columnas; los no divididos, cada pregunta es una línea; y los mixtos, se ha centrado en convertir a texto solo los de dos columnas para generalizar este caso.

Los 25 unigramas con mayor frecuencia, calculados con CountVectorizer, representan el área, pero de forma muy general, como "inteligencia", "análisis", "individuo", "modelo" o "rasgos".

Los 25 bigramas resultantes se acercan más a ser términos clave: "componentes cognitivos", "diferencias individuales", "consistencia transituacional", "psicología diferencial" o "procesos mentales".

Los 25 trigramas sin *stopwords* resultantes se alejan de ser términos clave. Con *stopwords* tampoco se aproximan, exceso de determinantes y preposiciones delante de un sustantivo, como "de la conducta" o "una de las".

En este punto, es necesario la introducción del experto, para eliminar términos

que no se corresponden o que son muy generales en la asignatura. O mejorar el proceso de extracción teniendo en cuenta otras técnicas de NLP.

Caso de estudio: Historia Económica Mundial

Para esta asignatura se han analizado 11 exámenes, de los 72 que hay en el depósito de Calatayud. Cada examen tiene aproximadamente 10 preguntas de tipo test con 4 opciones cada una, además de otras preguntas de desarrollo, pero solo se ha centrado el estudio en las de tipo test.

Como en el caso anterior, también existen diferentes modelos de examen, en este caso, sobre las preguntas de tipo test, cambia la estructura de presentar las opciones, y a la hora de generalizar un proceso se ha escogido solo un modelo.

Entre los 25 unigramas resultantes de aplicar CountVectorizer, aparecen "agricultura", "comercio", "europa", "francia", "población" y "xviii". Términos relacionados con la asignatura, pero no se sabe hasta qué punto son clave.

Entre los 25 bigramas aparecen "siglo xviii" y "años 20", datos característicos de asignaturas relacionadas con la historia.

Entre los 25 trigramas con y sin *stopwords* ocurre algo parecido con el anterior caso. Destacar que entre los 25 términos con *stopwords* aparecen dos interesantes "mano de obra" y "primera guerra mundial", por lo que pueden tener una gran relevancia en la asignatura por sí mismas.

Al igual que en el caso anterior, en este punto es necesario utilizar otras técnicas de extracción de entidades, debido en parte a que no han aparecido apenas términos clave, aunque sí estén muy próximos al área.

3.4.3. Discusión de resultados

El preprocesado, de convertir imágenes a texto, lleva mucho tiempo, también debido al cambio de modelo de cada año, en la forma de presentar las preguntas. Estos modelos se deben tener en cuenta, tanto dentro de una asignatura como entre ellas. Por esto, este proceso concreto no se puede generalizar.

El grafo de preguntas de cada asignatura se adapta perfectamente a la estructura de la ontología creada y explicada en la línea anterior.

Como lo comentado en la sección anterior, es necesario aplicar técnicas de extracción de información que mejoren la extracción de entidades lo más próximas al área.

Capítulo 4

Discusión de resultados

Parte del objetivo de este trabajo es explorar las técnicas semánticas para ponerlas al servicio de la UNED. Con la idea de ayudar a los equipos docentes de la universidad.

4.1. Resultados y evaluación

La búsqueda de la fuente de datos de donde extraer información para la construcción del grafo de conocimiento ajustado a una asignatura, ha dado diferentes resultados en cada línea de investigación. **Wikidata**, una que recoge una gran cantidad de datos, de cualquier tema, con la que poder contar para poblar los grafos de conocimiento. **AI-KG**, un grafo de conocimiento específico en el área de la IA, y por eso, usar esta fuente para asignaturas relacionadas con el área. **Libros de texto**, también específicos en un área, de los que poder extraer información de sus tablas de contenidos y terminología del tema. Exámenes con **preguntas de tipo test** de la asignatura, que dan información compacta en forma de enunciados.

Por otro lado, la exploración realizada hacia las tecnologías semánticas útiles para el propósito de construir la construcción del grafo, resultó estar relacionado con RDF, consultas SPARQL, sintaxis Turtle e inferencia a través de OWL para modelar y estructurar los datos y el grafo de forma cómoda y con lenguaje entendido tanto para máquinas como por humanos.

El uso de estas tecnologías junto a la Web de Datos, ha ayudado a no tener que utilizar en todo el proceso técnicas de NLP, que requieren más estrategias para extraer información relevante y ajustada sobre un tema, como se ha visto en las dos últimas líneas de investigación, en la búsqueda de entidades relacionadas con un área.

El resultado de la primera aproximación de búsqueda de fuentes de datos, fue Wikidata, que resultó ser de utilidad a la lo largo de todo el trabajo.

Los resultados de Wikidata, como principal fuente de datos, en la primera línea de investigación, no generaban suficiente información semántica entre los datos de respuestas en forma de tablas. En la segunda línea, Wikidata se usó solo como una fuente de información extra, conectada a través de algunos de los términos que podía tener similares con el AI-KG. En la tercera y cuarta línea, resultó ser la fuente de donde extraer información para poblar el grafo de conocimiento, junto con Wikipedia.

Los resultados de usar como fuente de datos principal a AI-KG, junto con los dos libros de texto sobre la IA (AIMA y ARTINT), fueron tres grafos de conocimiento sobre cada una de las fuentes de los que extraer información de nuevo para construir uno solo para una asignatura. Proceso que resultó ser muy poco funcional para el objetivo de generalizarlo para cualquier asignatura.

El resultado de haber creado una ontología para los grafos de preguntas, tanto de desarrollo como de confirmación, es que se ajusta perfectamente a cualquier conjunto de preguntas dado, dando así una fuente de información estructurada para cada asignatura.

Por último, para la construcción de los grafos de dominio a partir de los grafos de preguntas, es necesario la aplicación de técnicas de NLP para extraer información de cada pregunta (proposición) y las utilizadas en este trabajo daban como resultado

términos que no eran clave para la asignatura y dejando otros que sí lo son.

La evaluación de los resultados fue subjetiva bajo el análisis de expertos. Por un lado estos expertos fueron miembros del grupo ISLearning y por otro lado, docentes de algunas de las asignaturas con las que se tuvo este contacto, los mismos directores de este trabajo, y uno de la asignatura de Métodos Probabilistas.

4.2. Análisis de implicaciones éticas y sociales y de aspectos de género

Implicaciones éticas y sociales

Por un lado, el uso actual de los grafos construidos no será absolutamente autónomo, sino que estará supervisado por un docente o equipo docente, por lo que la perspectiva ética recaerá sobre los usuarios.

Por otro lado, el sesgo de dominio puede darse a la hora de recopilar información de la asignatura si, por ejemplo, hablando de Historia del Arte occidental, no se incorpora información del arte de otra región, pero esto no es sino reflejo del sesgo existente en el contenido de la asignatura.

Por último, a los grafos no se le añade información personal del equipo docente ni de los estudiantes, este último caso si se quisiera tener información de las respuestas de éstos, solo sería qué han respondido no quién ni cuándo, por lo que no existiría ningún problema con respecto al tratamiento de datos personales.

Aspectos de género

Durante la construcción del grafo, la responsabilidad del aspecto de género recaería en la fuente de extracción de información, ya que el sesgo de dominio se

hereda, como se ha mencionado antes.

Para las fuentes relacionadas con IA, no se ha extraído información sobre las personas, solo de los conceptos relacionados con el tema.

Para la fuente de preguntas, depende del equipo docente y del contenido que imparten.

Para Wikidata, depende de los datos añadidos por los usuarios, pero si en el área de información de estos datos no se cumple los aspectos de género, habría que considerar incorporar otras fuentes.

4.3. Conclusiones y trabajo futuro

A lo largo de la exploración de la Web de Datos y de las tecnologías semánticas, se han encontrado fuentes de datos con una gran información, que se puede utilizar para apoyar contenidos en asignaturas, a través de las tecnologías semánticas mencionadas a lo largo del trabajo, que han resultado ser de gran utilidad en diferentes puntos como, para la extracción y para la construcción. De esta forma, queda alcanzado el primer objetivo.

Debido a las limitaciones que presentaban cada fuente de datos escogida y la necesidad de conocerlas previamente para saber cuánto de útiles pueden ser, desencadenó estas diferentes líneas de estudio.

Por un lado la limitación de Wikidata por mantener una gran cantidad de datos y que hay que saber cómo conseguir acceder a ellos para una descarga masiva de información.

Por otro lado, la información particular que requiere un área e incluso una asignatura, conlleva a separarse, a veces, muy pronto, del proceso general de la construcción del grafo.

Por estos motivos, se crearon diferentes grafos de conocimiento previos a cualquier refinamiento posterior. De esta forma, se alcanzó el segundo objetivo.

El contacto con el equipo docente, en dos asignaturas de las estudiadas, ayudó en el refinamiento de estos grafos y a la generalización del proceso, a través de la construcción de una ontología para grafos de preguntas. De esta forma se han alcanzado el tercer y cuarto objetivo.

El quinto objetivo no se alcanzó debido a que ha llevado mucho tiempo el proceso de exploración y construcción de los grafos quedando estos últimos aún muy poco ajustados a una asignatura.

Debido a los problemas surgidos y el trabajo pendiente, algunos de los posibles trabajos futuros considerados a lo largo del trabajo son:

1. Evitar la limitación de tiempo de Wikidata utilizando estrategias de paralelización en las búsquedas y descarga de información para mantener una caché local y que además se pueda actualizar sobre un área.
2. Mejorar la extracción de términos utilizando estrategias de búsqueda por sustantivos, adjetivos y/o verbos para determinar entidades de una asignatura.
3. Utilizar los grafos de conocimiento de dominio para generar preguntas de tipo test.

Anexo A

Wikidata

Wikidata es una base de conocimiento multilingüe y multitemática que se edita en colaboración por unos 24000 usuarios activos[1] y mantenida por la fundación Wikimedia[2], recogiendo más de 90 millones de elementos.

Cada artículo de Wikidata trata sobre el concepto de una entidad independientemente del idioma, a diferencia de otros proyectos de la misma fundación. Éste es uno de los motivos junto a mantener en un solo lugar datos genéricos, como fechas y lugares, para dar soporte al resto de proyectos de Wikimedia que almacenen estos datos, ya que una actualización de estos datos en un idioma estará disponible en el resto y en todos los proyectos, proceso que por ejemplo en Wikipedia no sucede, porque existen diferentes artículos de un mismo concepto para diferentes idiomas.

La edición en Wikidata es algo más estricta que en Wikipedia, porque se debe seguir una estructura fija establecida. Esta estructura mantiene una cabecera, donde se hace referencia a la **etiqueta** del elemento (título del artículo), al **identificador** propio de la base (llamado QID), a la **descripción** y a otros posibles nombres por los que es conocido, **alias** (ver Figura A.1); mantiene un cuerpo de **declaraciones** o enunciados para describir al elemento con una propiedad y un valor (ver Figura A.1);

[1] https://www.wikidata.org/wiki/Wikidata:Statistics

[2] https://wikimediafoundation.org/

y por último se reserva un apartado para las **referencias** a otras bases de datos (ver Figura A.2) y otro para los propios proyectos de Wikimedia (ver Figura A.3).

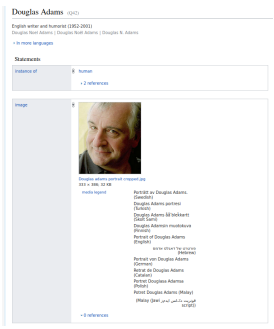

Figura A.1: *Cabecera y declaraciones de un artículo*

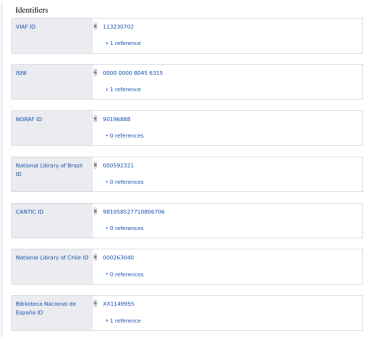

Figura A.2: *Referencias a otras bases de datos de la entidad*

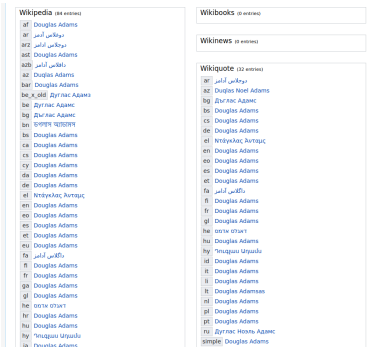

Figura A.3: *Referencias a proyectos de Wikimedia de la entidad*

Las anteriores figuras se han extraído de la entidad `Douglas Adams (Q42)`[3], ejemplo usado también por la misma Wikidata.

Un artículo de un elemento concreto se puede descargar en formato Turtle a través de su término URI con la extensión `.ttl` directamente desde el navegador, por ejemplo para `Douglas Adams`:

$$http://www.wikidata.org/entity/Q42.ttl$$

Los elementos se distinguen por un identificador único de Wikidata, un número natural con el prefijo `Q` (por ejemplo, Q42 que coincide con la persona *Douglas*

[3]https://www.wikidata.org/wiki/Q42

Adams), de ahí que reciban el nombre de QIDs, y las propiedades se identifican por un número natural con el prefijo `P` (por ejemplo, `P279` que es *subclase de*).

Existe libertad de edición para añadir y modificar estos elementos mientras que para las propiedades no. Para el caso de querer añadir, modificar o eliminar una propiedad se sugiere una posible y pasa por una revisión previa para evitar que haya duplicados o que el concepto se refiera a otro ya definido. De esta forma, se clasifica y se limita la gran cantidad de datos que mantiene la base. Actualmente existen unas 9600 propiedades[4].

Wikidata está formada por tripletas, ya que las declaraciones siguen la estructura `elemento-propiedad-valor` de la siguiente forma:

$$\texttt{Q11660} \xrightarrow{\texttt{P31}} \texttt{Q11862829}$$

que significa que *Inteligencia Artificial* (`Q11660`) es *instancia de* (`P31`) *Disciplina Académica* (`Q11862829`), y bajo el modelo de grafo RDF la tripleta se ve de la siguiente forma:

```
wd:Q11660 wdt:P31 wd:Q11862829
```

Usa vocabulario estándar de RDF, RDFs, OWL y SKOS entre otros, para definir entidades, como la etiqueta (`rdfs:label` y `skos:prefLabel`) y propiedades (`owl:ObjectProperty`), pero principalmente usa vocabulario propio definido para la base.

Las entidades se agrupan bajo el URI `<http://www.wikidata.org/entity/>`, como en el ejemplo anterior, `wd:Q11660`, siendo `wd` el prefijo del URI, y las propiedades se agrupan bajo diferentes URIs dependiendo de su uso, siendo el más usado por usuarios el `<http://www.wikidata.org/prop/direct/>`, como en el ejemplo anterior, `wdt:P31`, donde `wdt` es su prefijo.

Mencionar que existen usos parecidos entre el vocabulario propio de Wikidata con el usual, por ejemplo, el uso de la propiedad `P31` (*ser instancia de*) equivale a

[4]https://www.wikidata.org/wiki/Wikidata:List_of_properties

`rdf:type` (*ser de tipo*) y `P279` (*subclase de*) equivale a `rdfs:class` (*ser clase de*).

Como cada declaración puede tener sus propias propiedades, se ha definido un vocabulario propio para ello bajo el URI `<http://www.wikidata.org/entity/statement/>` y abreviado bajo el prefijo `wds`. Por ejemplo, la declaración

```
wds:Q42-F078E5B3-F9A8-480E-B7AC-D97778CBBEF9
```

hace referencia al enunciado *"Douglas Adams es ser humano"*

```
wd:Q42 wdt:P31 wd:Q5
```

De esta forma, para saber qué referencias se han usado para definir una declaración, se puede preguntar por ella directamente. Siguiendo el ejemplo anterior, la declaración tiene dos referencias, resultado que se puede ver aquí.

Lista de URIs para agrupar las propiedades:

- `p`, `ps`, `psn` y `psv` se utilizan para las declaraciones,

```
wd:Q42 p:P31 ?statement

?statement ps:P31 wd:Q5
```

- `pr`, `prn`, `prv` se utilizan para las referencias de las declaraciones.
- `pq`, `pqn`, `pqv` se utilizan para los calificadores.
- `wdt` se utiliza para definir las propiedades directas, dentro de las declaraciones.

En el siguiente código se muestra un extracto de la definición de la propiedad `P50` (*autor*) del archivo Turtle de la entidad `Q42`, donde se puede ver de qué tipo es y cómo se puede definir según su URI:

```
wd:P50 a wikibase:Property ;
        rdfs:label "author"@en ;
        skos:prefLabel "author"@en ;
        schema:name "author"@en ;
```

```
        schema:description "main creator(s) of a written work (use on works, not humans);
                        use P2093 when Wikidata item is unknown or does not exist"@en ;
        wikibase:propertyType <http://wikiba.se/ontology#WikibaseItem> ;
        wikibase:directClaim wdt:P50 ;
        wikibase:claim p:P50 ;
        wikibase:statementProperty ps:P50 ;
        wikibase:statementValue psv:P50 ;
        wikibase:qualifier pq:P50 ;
        wikibase:qualifierValue pqv:P50 ;
        wikibase:reference pr:P50 ;
        wikibase:referenceValue prv:P50 ;
        wikibase:novalue wdno:P50 .

p:P50 a owl:ObjectProperty .
psv:P50 a owl:ObjectProperty .
pqv:P50 a owl:ObjectProperty .
prv:P50 a owl:ObjectProperty .
wdt:P50 a owl:ObjectProperty .
ps:P50 a owl:ObjectProperty .
pq:P50 a owl:ObjectProperty .
pr:P50 a owl:ObjectProperty .
wdno:P50 a owl:Class ;
        owl:complementOf _:b031acc4c46aca4c83feb5878a7c4a6a .
_:b031acc4c46aca4c83feb5878a7c4a6a a owl:Restriction ;
        owl:onProperty wdt:P50 ;
        owl:someValuesFrom owl:Thing .
```

Se puede consultar la información de esta base a través del punto de consulta ofrecido por Wikidata Wikidata Query Service[5]. El lenguaje para realizar estas consultas es SPARQL.

Como a priori no se conocen los identificadores de los elementos por los que se quiere preguntar, se puede usar el atajo de recomendación usado normalmente `ctrl+space` sobre el prefijo de los URIs.

[5]https://query.wikidata.org/

Anexo B

AI-KG

AI-KG[1] es un grafo de conocimiento creado por la *Open University* por *Dessì et al.* [9] que consta de unos 14 millones de tripletas RDF que describen un conjunto de 1,2 millones de declaraciones y unas 820 mil entidades extraídas de una colección de 333.609 publicaciones de IA entre 1989 y 2018.

La versión actual se generó a través de un *pipeline* que añade (y actualiza cada cierto tiempo) datos de *Microsoft Academic Graph*, de *Computer Science Ontology* (CSO) y de *Wikidata*.

La ontología de AI-KG[2] describe cinco tipos de entidades (*Task*, *Method*, *Metric*, *Material* y *OtherEntity*), definidas como clases, que a su vez pertenecen a la clase general llamada *ResearchEntity*, como se ve en la Figura 3.7. Para crear esta ontología ha sido necesario un vocabulario propio y otro basado en los modelos SKOS, PROV-O[3]y OWL para definir entidades y las relaciones entre ellas.

El grafo está compuesto por enunciados (*statements*) con forma de tripleta estándar para describir la relación entre dos entidades y un número de metadatos relevantes: *rdf:subject*, *rdf:predicate* y *rdf:object*. El siguiente ejemplo, extraído del artículo [9], muestra la información interesante de un enunciado de AI-KG, (*statement_11053*), y que se puede ver visualmente en la Figura B.1.

[1] https://scholkg.kmi.open.ac.uk/

[2] https://scholkg.kmi.open.ac.uk/aikg/ontology

[3] https://www.w3.org/TR/prov-o/

```
aikg:statement_110533 a aikg-ont:Statement, provo:Entity ;
        aikg-ont:hasSupport 4 ;
        aikg-ont:isInferredByTransitivity false ;
        aikg-ont:isInverse false ;
        rdf:subject aikg:learning_algorithm ;
        rdf:predicate aikg-ont:usesMethod ;
        rdf:object aikg:gradient_descent ;
        provo:wasDerivedFrom aikg:1517004310,
                aikg:1973720487,
                aikg:1996503769,
                aikg:2085159862 ;
        provo:wasGeneratedBy aikg:DyGIE++,
                aikg:OpenIE,
                aikg:pipeline_V1.2 .
```

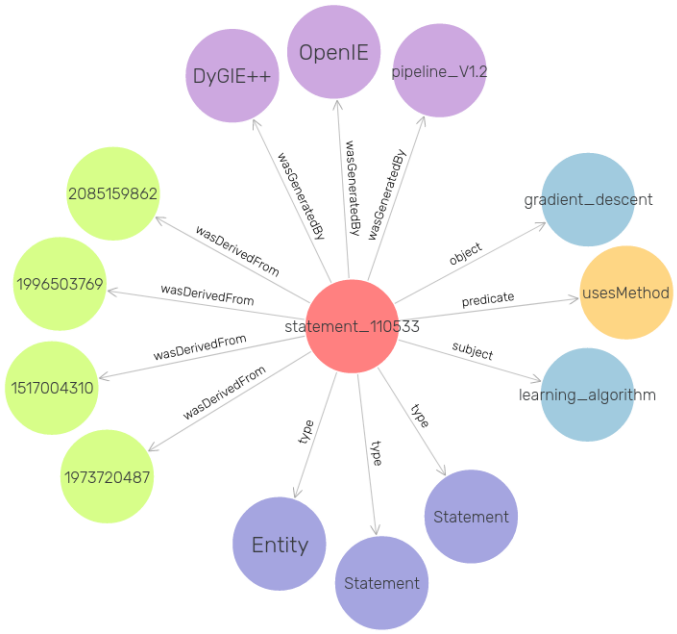

Figura B.1: *Grafo de la declaración 110533 que corresponde con la tripleta* `aikg:learning_algorithm aikg-ont:usesMethos aikg:gradient_descent` (Imagen extraída de GraphDB)

El vocabulario propio que se ha definido para el grafo tiene los siguientes espacios de nombre:

- `aikg`, prefijo del URI `http://scholkg.kmi.open.ac.uk/aikg/resource/`, para identificar los términos que son recursos del grafo, por ejemplo, `aikg:bayesian_network`.

- `aikg-ont`, prefijo del URI `http://scholkg.kmi.open.ac.uk/aikg/ontology#`, para identificar los términos definidos por la ontología, por ejemplo, `aikg-ont:Method` o para definir propiedades propias del grafo como: `aikg-ont:hasSupport`, que da el número de publicaciones donde aparece la declaración; `aikg-ont:isInverse`, que indica si la declaración fue creada por inferir la inversa de una relación extraída del texto; o `aikg-ont:isInferredByTransitivity` que indica si la declaración fue inferida por otras.

Del vocabulario predefinido de RDF que se ha utilizado para definir entidades y propiedades, remarcar algunas URIs de los modelos que se han nombrado:

- Del modelo SKOS, `http://www.w3.org/2004/02/skos/core#`, se han utilizado dos propiedades interesantes, que son inversas entre ellas: `skos:broader`, que relaciona un término con otro más general; y `skos:narrower`, que relaciona un término con otro más conciso o menos general. Por ejemplo,

  ```
  aikg:bayesian_network skos:broader aikg:directed_model
  ```

  se puede traducir como que una red bayesiana es un modelo dirigido y

  ```
  aikg:bayesian_network skos:narrower aikg:markov_bayesian_network
  ```

  se puede traducir como que una red bayesiana de Markov es un tipo de red bayesiana.

- Del modelo PROV-O, http://www.w3.org/ns/prov# se han utilizado propiedades como: `provo:wasDerivedFrom`, que enlaza cada declaración a los IDs de las publicaciones de donde se extrajo la información; y `provo:wasGeneratedBy`, que enlaza cada declaración tanto a las herramientas que se necesitaron como a la versión del *pipeline* que se usó para extraer la información.

- Del modelo OWL, `http://www.w3.org/2002/07/owl#` se han utilizado los URIs:

  - `owl:Class`, para definir clases como las que se ven en la Figura 3.7.

    ```
    aikg:Method a owl:Class
    ```

- `owl:DatatypeProperty`, `owl:ObjectProperty` y `TransitiveProperty`, para definir las propiedades con URI `aikg-ont`.

```
aikg-ont:isInverse a owl:DatatypeProperty
```

- `owl:sameAs`, para enlazar términos similares definidos en un grafo y en otro, como se ha mencionado en los casos de Wikidata y CSO.

```
aikg:graphical_model owl:sameAs wikidata:Q1143367
```

Se centraron en estas cinco clases porque varios puntos de referencia y herramientas de extracción de información dan soporte para trabajar con ellas.

Las relaciones entre las instancias de estas clases se formaron analizando los predicados principales y se seleccionaban aquellos más frecuentes extraído por técnicas de NLP. Por ejemplo, la aparición del verbo *uses* se usó para generar los predicados *usesMethod*, *usesTask*, *usesMetric* y *usesOtherEntity*; y el verbo *is a* se tradujo en la propiedad *skos:broader*.

El resultado fue revisado en varias interacciones por cuatro expertos del dominio, que seleccionaron 27 relaciones derivadas de 9 verbos básicos (*uses, includes, is, evaluates, provides, supports, improves, requires, predicts*) y se definieron otras características como el dominio, el rango y la transitividad.

Por último, mencionar las herramientas que usaron para la construcción: *DyGIE++*; *CSO Classifier*; *OpenIE* y *Pos Tagger of Stanford Core NLP*. Para mayor detalle ir a [9].

Bibliografía

[1] Andrejs Abele, John P McCrae, Paul Buitelaar, Anja Jentzsch, and Richard Cyganiak. Linking open data cloud diagram 2017. URL: http://lod-cloud.net, Insight-Centre, 2017.

[2] Dean Allemang and Jim Hendler. RDFS-Plus, pages 153–185. 12 2011.

[3] David Beckett, Tim Berners-Lee, Eric Prud'hommeaux, and Gavin Carothers. Rdf 1.1 turtle. "`https://www.w3.org/TR/turtle/`", Feb 2014.

[4] Tim Berners-Lee. Information management: A proposal. "`https://www.w3.org/History/1989/proposal.html`", Mar 1989.

[5] Tim Berners-Lee, James Hendler, and Ora Lassila. The semantic web: A new form of web content that is meaningful to computers will unleash a revolution of new possibilities. ScientificAmerican.com, 05 2001.

[6] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. J. Mach. Learn. Res., 3(null):993–1022, mar 2003.

[7] Dan Brickley and R.V. Guha. Rdf schema 1.1. "`https://www.w3.org/TR/rdf-schema/`", Feb 2014.

[8] P. Chen, Y. Lu, V. W. Zheng, X. Chen, and B. Yang. Knowedu: A system to construct knowledge graph for education. IEEE Access, 6:31553–31563, 2018.

[9] Danilo Dessi, Francesco Osborne, Diego Reforgiato Recupero, Davide Buscaldi, Enrico Motta, and Harld Sack. Ai-kg: an automatically generated knowledge graph of artificial intelligence, 2020.

[10] Bob DuCharme. Learning SPARQL (2nd Edition). O'Reilly, 2013.

[11] Werner Nutt Franz Baader. Basic Description Logic, pages 47–100. Cambridge University Press, 2002.

[12] Reva Freedman, Syed S. Ali, and Susan McRoy. Links: What is an intelligent tutoring system? Intelligence, 11(3):15–16, sep 2000.

[13] Omar Khalil Gómez. Estudio de generación automática de distractores desde fuentes estructuradas y no estructuradas: el caso de psicofarmacología, 2020.

[14] Steve Harris and Andy Seaborne. Sparql 1.1 query language. "`https://www.w3.org/TR/sparql11-query/`", Mar 2013.

[15] Pascal Hitzler, Markus Krötzsch, Bijan Parsia, Peter F. Patel-Schneider, and Sebastian Rudolph. Owl 2 web ontology language primer (second edition). "`https://www.w3.org/TR/owl2-primer/`", Dec 2012.

[16] Aidan Hogan. The Web of Data. Springer, Cham, 2020.

[17] Aidan Hogan, Eva Blomqvist, Michael Cochez, Claudia D'amato, Gerard De Melo, Claudio Gutierrez, Sabrina Kirrane, José Emilio Labra Gayo, Roberto Navigli, Sebastian Neumaier, and et al. Knowledge graphs. ACM Computing Surveys, 54(4):1–37, Sep 2021.

[18] Antoine Isaad and Ed Summers. Skos simple knowledge organization system primer. "`https://www.w3.org/TR/skos-primer/`", Aug 2009.

[19] Shaoxiong Ji, Shirui Pan, Erik Cambria, Pekka Marttinen, and Philip S. Yu. A survey on knowledge graphs: Representation, acquisition and applications, 2020.

[20] Roy Meissner and Laura Köbis. Annotated Knowledge Graphs for Teaching in Higher Education: Supporting Mentors and Mentees by Digital Systems, pages 551–555. 06 2020.

[21] Daniel Ringler Nicholas Heist, Sven Hertling and Heiko Paulheim. Knowledge graphs on the web - an overview. Jan 2020.

[22] Roger Nkambou, Jacqueline Bourdeau, and Riichiro Mizoguchi. Advances in Intelligent Tutoring Systems. Springer, Dec 2010.

[23] Hyacinth S. Nwana. Intelligent tutoring systems: an overview. Artificial Intelligence Review, (4):251–277, Dec 1990.

[24] María Poveda-Villalón Pierre-Yves Vandenbussche, Ghislain A. Atemezing and Bernard Vatant. Linked open vocabularies (lov): a gateway to reusable semantic vocabularies on the web. 2014.

[25] Yuehua Qin, Han Cao, and Leyi Xue. Research and application of knowledge graph in teaching: Take the database course as an example. Journal of Physics: Conference Series, 1607:012127, aug 2020.

[26] Guus Schreiber and Yves Raimond. Rdf 1.1 primer. "`https://www.w3.org/TR/rdf11-primer/`", Jun 2014.

[27] Amit Singhal. Introducing the knowledge graph: things, not strings. `https://www.blog.google/products/search/introducing-knowledge-graph-things-not/.`, Jun 2012.

[28] Fabian Suchanek, Jonathan Lajus, Armand Boschin, and Gerhard Weikum. Knowledge Representation and Rule Mining in Entity-Centric Knowledge Bases, pages 110–152. 09 2019.

[29] Y. Sun, J. Tang, and Z. Zhu. A method of english test knowledge graph construction. Journal of Computer and Communications, 9:99–107, sep 2021.