Universidad de Murcia

Facultad de Informática

TRABAJO FIN DE GRADO

Grado en Ingeniería Informática

Enriquecimiento de ontologías basado en reutilización y modularización

Tutor: Jesualdo Tomás Fernández Breis
Estudiante: Álvaro López Fernández

Julio 2022

Resumen

Las ontologías son esquemas del conocimiento de un dominio determinado. La reutilización de contenido existente es una buena práctica en la construcción de ontologías. Existen varias formas de reutilización de ontologías. La primera se basa en reutilizar únicamente el identificador del término, de forma que se sepa a qué concepto de otra ontología se hace referencia. La segunda se basa en reutilizar, además del término, los axiomas lógicos asociados a dicho concepto. Esta segunda forma es más adecuada pero computacionalmente más compleja, ya que requiere identificar el conjunto mínimo de axiomas a importar. Una posible solución a este problema es la adopción de métodos basados en modularización de ontologías.

Para este propósito se encuentran diseñadas una serie de estrategias que extraen los módulos solicitados de una ontología dada, así como todos los axiomas relacionados con el de manera que puedan añadirse de forma íntegra en otra. En este proyecto se usarán 3 de ellas: “Star”, “Bot” y “Top”, se comparará el resultado, sus diferencias y se evaluará su rendimiento en repositorios de ontologías biomédicas como BioPortal y OBO Foundry.

Los objetivos de este trabajo son:

Realizar una aplicación en Java que, dada una ontología como parámetro bien como fichero o bien vía URI sea capaz utilizando un extractor modular y cualquiera de las estrategias comentadas previamente de insertar en ella todas la entidades y axiomas relacionadas con las clases importadas en ella desde otras ontologías, generando así una ontología “completa”.

Realizar una comparación del rendimiento, de los resultados de cada una de las estrategias comparando los axiomas de más o de menos que haya extraído cada una y explicar brevemente su funcionamiento mediante dichas diferencias.

Comprobar las ontologías resultantes mediante un razonador para confirmar que no se generan discrepancias o incongruencias durante el proceso, confirmando así el correcto funcionamiento tanto de los métodos de extracción como del algoritmo desarrollado en el transcurso de este trabajo.

La realización de este trabajo ha permitido confirmar la utilidad de enriquecer las ontologías insertando modularmente los conceptos reutilizados desde sus ontologías originales mediante el uso de extractores modulares y sus distintas estrategias así como que esta práctica no inserta incongruencias ni discrepancias en las ontologías enriquecidas

Sin embargo, también ha permitido mostrar que tanto estos algoritmos como la Web Semántica aún necesitan algunas mejoras para ser 100% fiables pues se han encontrado algunas limitaciones importantes durante la realización de este trabajo. También ha podido comprobarse que una de las estrategias (TOP) apenas ofrece mejora en usar extracción modular pues introduce las ontologías casi completas, y que ninguna de las estrategias es perfecta, pues nunca garantizan la extracción del menor conjunto posible de axiomas.

Extended Abstract

Ontologies are schemas of the knowledge of a given domain. To reuse the existing content is a good practice in ontology construction. There are several ways of reusing ontologies:
The first is based on reusing only the identifier of the term, so that it is known which concept of another ontology is being referred to.
The second is based on reusing, not only the term, but also the logical axioms associated with that concept. This is the most appropriated way, on the other hand, it is more complex when it comes to computation, since it requires identifying the minimum set of axioms to be imported. A possible solution to this problem is the adoption of methods based on ontologies modularization.

For this purpose, a series of strategies have been designed that extract the requested modules from a given ontology, as well as all the axioms related to it, so that they can be added in an integral way to another ontology.
In this project 3 of them will be used: "Star", "Bot" and "Top", the result and their differences will be compared, and their performance in biomedical ontology repositories such as BioPortal and OBO Foundry will be evaluated.
The objectives of this work are:

To develop a Java application that, given an ontology as a parameter, either as a file or via the Ontology URI, is capable, using a modular extractor and any of the previously mentioned strategies, of inserting into it all the entities and axioms related to the imported classes into it from other ontologies, generating a "complete" ontology.

To make a comparison of performance and the results of each of the strategies by comparing the axioms extracted by each one, and to briefly explain their performance by means of these differences.

To check the resulting ontologies by using a reasoner to confirm that no discrepancies or inconsistencies are generated during the process, confirming the correct functioning of both the extraction methods and the algorithm developed in the course of this work.

This work has confirmed the usefulness of enriching ontologies by modularly inserting reused concepts from their original ontologies through the use of modular extractors and their different strategies. Also how this practice does not insert neither inconsistencies nor discrepancies in the enriched ontologies.

However, it has also been permitted to show that both these algorithms and the Semantic Web still need some improvements to be 100% reliable, as some important limitations have been found during the course of this work.

It has also been shown that one of the strategies (TOP) hardly offers any improvement in using modular extraction, since it introduces almost complete ontologies, and that none of the strategies is perfect, as they never guarantee the extraction of the smallest possible set of axioms.

To reuse is an important step in ontologies' development, Semantic Web information nodes depend on the reuse of their content for their interconnection. Content reuse is a good practice in the construction of ontologies as it allows the creator to concentrate on the development of domain-specific content without having to worry about sub-domain information since the content, properties and axioms needed from the existing ontology for that sub-domain can be simply included.
There are two main methods of ontology reuse:

- Explicit reuse of the ontology: in this case, the creator of an ontology directly imports from other ontologies the entities it needs, as well as their axioms and properties.
- Implicit reuse of the ontology: In this case, the ontology creator reuses through the term URI (Uniform Resource Identifier) without importing the information, creating a reference to the reused ontology.

When in this work we refer to an "incomplete" ontology or to "complete" an ontology we are actually referring to an ontology that performs implicit reuse of entities and to directly import those entities and their axioms and properties to convert it into explicit reuse.

A module-based extraction implementation will be used for the realization of this work.
Module-based extraction works as follows:

Being S a set of axioms and entities of the O ontology, which wishes to be extracted, we must generate a module M that guarantees the following:
"Everything that the O ontology knows about the terms belonging to S (Seed) must be known by the module M. The rest of the information of O must be irrelevant information for the terms of S".

OWL API extraction algorithms look at the ontology to be imported as a set of axioms and, therefore, what they do is to search in that set for all axioms that are directly related to the set to be extracted passed as a parameter (seed signature). The aim is to ensure that the extraction performed preserves all the specific information of the seed signature which is part of the ontology. However, this guarantee has a problem: although it guarantees that all relevant information is extracted, it does not guarantee that information that could be considered irrelevant is extracted, which means that it does not guarantee that the extracted module is as small as possible.

The strategies it offers are the following:

- The BOT strategy guarantees that, for any class in the seed signature, both the related axioms and classes and their superclasses will be extracted, together with the axioms that relate them. The fact that an ontology class is related to a class in the seed signature does not

necessarily mean that its superclass is also related, so that in many occasions this strategy will be introducing irrelevant information to the set.

- The TOP strategy guarantees that, for any class of the seed signature, both the related axioms and classes and their subclasses will be extracted, together with the axioms that relate them. As with superclasses, just because an ontology class is related to a class in the seed signature does not mean that its subclasses are also related, and under the same logic explained in BOT, this strategy will introduce irrelevant information in the final set. This also makes TOP a dangerous strategy, because for each entity related to the seed signature it will extract its entire "subtree" with all related axioms and entities, making the extracted modules by it very large and the checking and extraction a very computationally expensive algorithm and of an exponential order to the ramifications of the ontology.

- STAR uses both of the above strategies to work. The idea behind it is that, if in a given case TOP has extracted an axiom and BOT has not, or vice versa, and on the premise that both strategies guarantee that the extracted sets have preserved all the information related to the seed signature in the ontology, then that axiom is not needed and therefore we can not include it in the extraction. Thus, STAR is the algorithm that extracts the least amount of information, and the closest to not extracting irrelevant information, although it still does not give us the guarantee of extracting the smallest possible set.

The two main parameters against which the results of the different strategies will be compared are the running time and the number of axioms extracted by each strategy.

When it comes to the time, it is the simpler of the two, the longer a strategy takes to run relative to the others, the worse it is. This metric is particularly important for the Top strategy because as previously mentioned, according to its documentation it is very expensive computationally speaking.
It is also a very subjective value because the importance given to time is relative to the user of the application and it is as valid to seek a short execution time even if it is slightly worse, as it is to want the best possible result no matter how long it takes or how many resources are consumed in the process.

The number of axioms extracted is a more difficult indicator to interpret, as it is not enough to see which strategy has extracted more or fewer axioms. It is possible that one strategy extracts more axioms than the others but that these are not really necessary. As we assume that all 3 strategies return the complete ontology, we understand that it is better to return as few axioms as possible.

To measure this, what will be done is to show examples of how each of the strategies has worked with the test cases to see which axioms have been extracted by each of them and to be able to get a picture of which of them is

closer to the expected result.

It has also been decided to run a reasoner through the resulting ontologies from the test cases to check if in all cases the results are valid or if any of the strategies has a tendency to introduce inconsistencies or some other type of error to also take this into account for comparison.

The study case chosen to document in this TFG will be the "Human Disease Ontology" (DOID) obtained from BioPortal (https://bioportal.bioontology.org/ontologies/DOID), specifically the 07/06/2022 release will be used.

The reasons why this study case has been chosen over the others are the following:

- It has axioms imported from more than 6 different ontologies, which makes it possible to test the operation with several ontologies at the same time. In addition, among them there is a great diversity in size and number of axioms, from very small ontologies such as GENO, to NCBITaxon, one of the largest in the repository.
- It is a fairly widely used and well-known ontology and contains very useful examples on which to check how axioms are being loaded by the different strategies.
- It has been updated very recently, which eliminates the possibility of encountering errors such as those that will be mentioned in the limitations section.

From the results of this work, we can draw the following conclusions:

- The difference between using Star and BOT is minimal; computationally speaking BOT needs less time, has less complexity and extracts slightly more axioms and entities than Star, but the difference between both of them is so small that the study indicates that they can be used interchangeably. The main difference is that as we haven seen, BOT will extract more superclasses from the hierarchy, which are not necessary, than Star.

- TOP extracts much more information than the other two options and its times are much longer. Its computational cost is so high that it makes it unreliable unless you use a high-capacity PC or use it in cases where the ontologies involved are known to be very small. But the main argument against this strategy is that it always extracts the imported ontology almost in its entirety, with results seen where the percentage extracted is usually 90%. Using a modular extraction algorithm does not make sense if the ontology is going to be almost entirely extracted, that is why it is better to directly import it from Protégé as it will be mentioned many times in the comparisons section.

- The fact that extraction strategies such as Star or BOT are able to complete ontologies with minimum percentages of the content of the others, makes the modular extractor a very useful tool for working with ontologies, as it allows both the creators of the ontologies and the users who use them to add the missing information of the entities which they wish to reuse from others without filling their ontology with information that is not necessary or that may be considered irrelevant.

Ontologies enriched by using the Star and BOT strategies do not increase drastically its size and remain at acceptable weights so their use for ontologies reuse is recommended. On the other hand, TOP, due to the fact that it imports the reused ontologies almost in their entirety, increases the size of the enriched ontologies to a point that may be excessive. It should be taken into account that to reuse a few axioms of a large ontology it is capable of extracting around 90% increasing its size of it, while the other two extract much less.

Índice

1 Introducción:

Una ontología es una representación formal de entidades y conceptos en conjunto con sus propiedades, características y relaciones entre sí. En el campo de la informática las ontologías son entendidas como un modelo para describir conjuntos de conocimiento sobre distintos campos del mundo real. A través de la definición de entidades, sus propiedades y sus relaciones somos capaces almacenar, utilizar información e informatizar el trabajo de campos científicos como el de la biomedicina que va a servir de caso de estudio para este trabajo [1].

Entre 2000 y 2001 nace de la mano de Tim Berners-Lee la idea de crear la Web Semántica como “Una web tejida de tal forma que no solo enlaza documentos entre ellos, sino que también identifica el significado de esos documentos”, es decir, una web que, en lugar de consistir en numerosos nodos con una información determinada, es un único nodo gigantesco con toda la información entrelazada.

Esta red sirve de base para el almacenamiento de información mediante ontologías, donde estas suponen los nodos de información que la Web Semántica entrelaza formando “una sola isla de información” [2].

Hay distintas estrategias que pueden seguirse a la hora de desarrollar una ontología desde 0, pero el proceso siempre consta de los siguientes pasos para un correcto desarrollo:

1. Determinar el dominio y el ámbito de la ontología teniendo siempre en mente el campo de estudio que se está abarcando, el uso que se le quiere dar a la ontología, a que tipo de preguntas debería responder la información de la ontología y el futuro de la misma, su mantenimiento y evolución.
2. Considerar la reutilización de ontologías. Es en este paso donde se centra el estudio de este trabajo. Siempre es aconsejable considerar qué otras ontologías de la WEB contienen material relacionado con nuestro campo, que información tiene, si podemos partir de sus axiomas y entidades y extenderlas con la información seleccionada en el primer paso, o si otras

ontologías ofrecen información de otros ámbitos que necesitamos para el estudio del nuestro. Y lo más importante, cómo reutilizamos esas ontologías, donde la extracción modular estudiada en este trabajo podría ser una opción apropiada.

3. Enumerar los términos importantes para nuestra ontología, que propiedades tienen y que información queremos dar sobre ellos.
4. Definir las clases y su jerarquía.
5. Definir las propiedades de las clases.
6. Definir características de las clases como su cardinalidad o su tipo.
7. Crear las instancias de las distintas clases de la jerarquía de la ontología [3].

Como hemos visto la reutilización es un paso importante en el desarrollo de las ontologías, los nodos de información de la Web Semántica dependen de la reutilización de su contenido para su interconexión. La reutilización de contenido es una buena práctica en la construcción de ontologías pues permite al creador concentrarse en el desarrollo de contenido específico a su dominio sin tener que preocuparse por la información referente a un subdominio pues puede simplemente incluir los contenidos, propiedades y axiomas que necesite de la ontología existente para dicho subdominio [4].

Existen dos métodos principales de reutilización de ontologías:

- Reutilización explícita de la ontología: En este caso, el creador de una ontología importa directamente de otras las entidades que necesita, así como sus axiomas y propiedades.
- Reutilización implícita de la ontología: En este caso el creador de la ontología reutiliza a través del término URI (Uniform Resource Identifier) sin importar la información, creando una referencia a la ontología reutilizada.

Veremos en profundad esta diferencia en el apartado 5.1.

Cuando en este trabajo nos referimos a una ontología “incompleta” o a “completar” una ontología en realidad nos estamos refiriendo a una ontología que realiza reutilización implícita de entidades y a importar directamente esas entidades y sus axiomas y propiedades para convertirla en reutilización explícita.

Para la realización de este trabajo se utilizará una implementación de

extracción basada en módulos, que funciona de la siguiente manera:

Todo lo que la ontología importada sepa sobre el conjunto de entidades a completar debe formar parte del modulo a extraer y todo lo que no se incluya en dicho módulo debe ser irrelevante para dicho conjunto de entidades.

Para este trabajo se utilizará OWL API, una interfaz de programación de aplicaciones para creación, modificación y trabajo sobre ontologías escrito en Java y desarrollado por la Universidad de Manchester [5].

Los algoritmos de extracción de OWL API miran a la ontología a importar como un conjunto de axiomas y, por tanto, lo que hacen es buscar en ese conjunto todos los axiomas que estén directamente relacionados con el conjunto a extraer pasado como parámetro (conjunto semilla). El objetivo es asegurar que en la extracción realizada se preserva toda la información específica del conjunto semilla que forme parte de la ontología. Esta garantía tiene sin embargo un problema, si bien garantiza que toda la información relevante es extraída, no garantiza que se extraiga información que pueda considerarse irrelevante, es decir, no garantiza que el módulo extraído sea lo más pequeño posible.

2 Estado del Arte:

2.1 Algoritmos basados en conjuntos semilla

Para la modularización de Ontologías basadas en conjuntos semilla como el anterior también podemos encontrar las siguientes implementaciones y metodologías:

PATO

PATO es una herramienta de particionado de ontologías que sigue los siguientes pasos:

I. Crea un grafo con el que representar las dependencias entre todas las entidades de la ontología donde los nodos de los grafos tienen los valores de "rdf:label" or "rdf:ID"

II. Se genera un set con los nodos en un rango de tamaños máximo y mínimo en base a que la fuerza de la conexión entre los nodos de dentro del set sea mayor que la conexión de cualquiera de ellos con los nodos de fuera, determinando así que elementos deben estar en el módulo.

III. Se crea la ontología distribuida a partir de la partición del punto anterior [6].

OAPT

OAPT son las siglas para “Ontology análisis and partitioning tool” y su objetivo es particionar las ontologías en un conjunto de módulos usando algoritmos de clustering basados en semillas. Sigue los siguientes pasos:

I. Generar un ranking de conceptos en base a la importancia de cada uno, es decir, su capacidad para generar nodos de entidades alrededor de ellos.

II. Utilizando el ranking del paso a anterior se selecciona una serie de entidades para ser cluster heads.

III. Comienza el particionado con el algoritmo basado en sedes. A cada entidad denominada cluster head se asignan aquellos conceptos que sean sus hijos directos y para los demás se aplica una función “membership” cuya función es asignarlos al cluster más conveniente para ellos.

IV. Se genera un módulo para cada cluster preservando las relaciones entre conceptos requeridas tanto dentro de cada cluster como de las entidades pertenecientes a distintas particiones [6].

AD

AD son las siglas para “Atomic descomposition”. La AD de una ontología O es un par consistente en (1) el conjunto de átomos y (2) relaciones de dependencia directa entre esos átomos donde un átomo es el conjunto máximo de axiomas que están fuertemente ligados entre ellos.

La implementación actual de AD soporta la extracción de 3 tipos de módulos, bottom modules, top modules y star modules [6].

2.2 Estrategias implementadas en OWL API

Las estrategias que ofrece son las siguientes:

- La estrategia BOT garantiza que, para cualquier clase del conjunto semilla, serán extraídos tanto las clases y axiomas relacionados como **sus superclases**, en conjunto con los axiomas que los relacionen. Que una clase de la ontología esté relacionada con una clase del conjunto semilla no quiere decir que necesariamente su superclase también lo esté, por lo que en numerosas ocasiones esta estrategia estará introduciendo información irrelevante al conjunto.

- La estrategia TOP garantiza que, para cualquier clase del conjunto semilla, serán extraídos tanto las clases y axiomas relacionados como **sus subclases**, en conjunto con los axiomas que los relacionen. Al igual que pasaba con las superclases, que una clase de la ontología esté relacionada con una clase del conjunto semilla no quiere decir que sus subclases también lo estén y bajo la misma lógica explicada en BOT, esta estrategia introducirá información irrelevante en el conjunto final. Esto hace además de TOP una estrategia peligrosa, pues para cada entidad relacionada con el conjunto semilla extraerá todo su “subárbol” con todas las entidades y axiomas relacionados, haciendo de los módulos extraídos por el muy grandes y de la comprobación y extracción un algoritmo muy costoso computacionalmente y de un orden exponencial a las ramificaciones de la ontología.

- STAR utiliza las dos estrategias previas para funcionar. La idea detrás es que, si en un caso dado TOP ha extraído un axioma y BOT no lo ha hecho, o viceversa, y partiendo de la premisa de que ambas estrategias garantizan que los conjuntos extraídos han preservado toda la información relacionada con el conjunto semilla en la ontología, entonces ese axioma no es necesario y por tanto podemos no incluirlo en la extracción. De esta forma, STAR es el algoritmo que extrae la menor cantidad de información, y la más cercana a no extraer información irrelevante, aunque sigue sin darnos la garantía de extraer el conjunto más pequeño posible.

Vamos a mostrar su funcionamiento con un ejemplo:

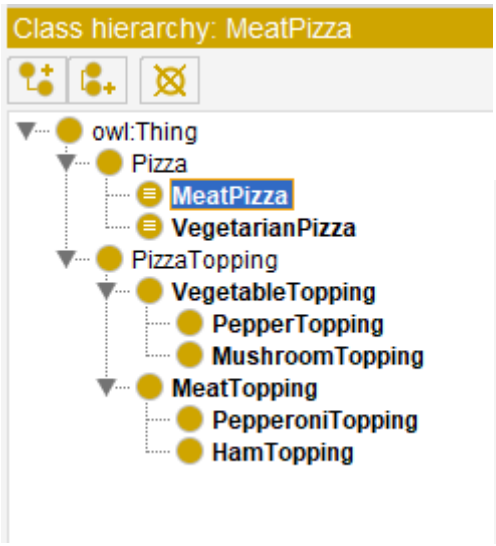

Ilustración 1- Ontología Ejemplo Pizza

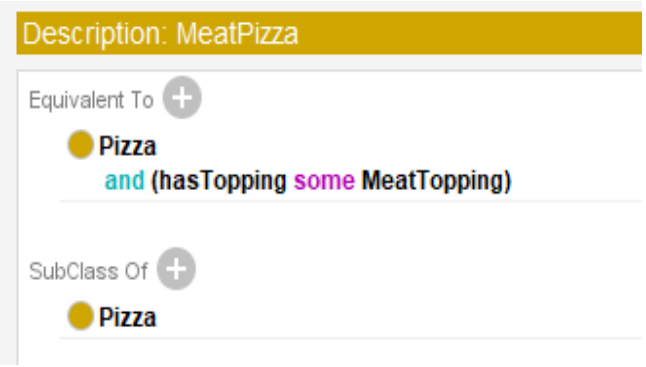

Ilustración 2 - Descripción MeatPizza

Imaginemos que estamos haciendo una ontología de comidas (como FOODON) y queremos reutilizar el concepto MeatPizza de la ontología mostrada en la Ilustración 1 en ella. Para ello utilizamos cada una de las 3 estrategias explicadas anteriormente usando como conjunto semilla únicamente la entidad “MeatPizza”.

Como queremos reutilizar el concepto MeatPizza los conceptos que queremos son Pizza, MeatPizza, MeatTopping y VegetableTopping así como los axiomas que aparecen en la Ilustración 2.

Si extraemos usando la ontología BOT obtendremos esto:

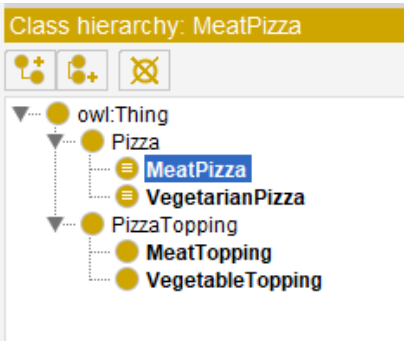

Ilustración 3 - Extracción con Bot

Como vemos en la Ilustración 3, Bot ha extraído toda la ontología menos las subclases de MeatTopping y VegetalTopping. Vamos a analizar punto a punto por qué se ha comportado de esta manera.

En primer lugar, siempre se cogen las ontologías del conjunto semilla y sus superclases, por lo que MeatPizza y Pizza son extraídas directamente. Luego el algoritmo mira sus axiomas y detecta que necesita MeatTopping y VegetableTopping; y como hemos explicado previamente Bot garantiza que también son extraídas las superclases de todo lo que se ha obtenido en el paso anterior, así como las entidades y axiomas relacionados con ellas. Por lo tanto, obtenemos PizzaTopping por ser superclase de MeatTopping y VegetableTopping y VegetablePizza por estar relacionada con VegetableTopping.

Veamos ahora si extraemos usando la ontología TOP:

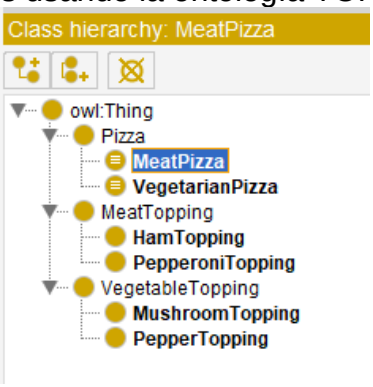

Ilustración 4 - Extracción con Top

En la ilustración 4 vemos que Top se ha limitado a extraer la ontología entera, pero eliminando PizzaTopping. La razón por la que ha actuado de esta manera es la siguiente:

Al igual que en Bot, se extraen las entidades del conjunto semilla, sus superclases, sus axiomas y las entidades directamente relacionadas con ella. Esto quiere decir que primero extrae MeatPizza, Pizza, MeatTopping y

VegetableTopping. Como se ha explicado antes Top garantiza que serán extraídas las clases del conjunto semilla, las clases relacionadas con ellas y todos los subárboles de subclases generados a partir de ellas, así como sus entidades y axiomas relacionados.

Como se ha extraído MeatTopping se han añadido también sus subclases, y se habrían añadido también las subclases de éstas si las hubiera.
Como se ha extraído Pizza se ha añadido también VegetarianPizza.
Como se ha extraído VegetableTopping se han añadido también sus subclases.
Por último, veamos el resultado al utilizar la estrategia Star.

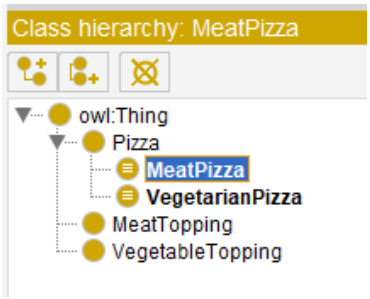

Ilustración 5 - Extracción con Star

Vemos en la ilustración 5 que Star es la que menos axiomas y entidades ha extraído. Como se ha explicado antes Star combina las estrategias Bot y Top y devuelve la unión de estás, es decir, devuelve solo las entidades y axiomas que habían sido devueltos por las dos estrategias anteriores.

Hemos podido comprobar el funcionamiento de cada una de las estrategias, y podemos ver su capacidad, pero también sus limitaciones, pues ninguna de las tres nos ha devuelto exactamente el conjunto que esperábamos. La más cercana ha sido Star, a la que solo le ha sobrado VegetarianPizza.

Lo que también hemos podido confirmar es que, aunque las 3 estrategias incluyen información irrelevante, ninguna de ellas nos ha devuelto menos de la información que necesitamos, pues todas han devuelto al menos las 4 entidades que establecimos como necesarias al principio, así como los axiomas asociados a ellas.

3 Objetivos y metodología:

3.1 Introducción del desarrollo

El objetivo final de este trabajo es ser capaces de comparar el funcionamiento de las 3 estrategias de extracción modular de ontologías (Star, Bot y Top) así como sus resultados al ser aplicadas a ontologías biomédicas.

Para realizar este estudio se va a desarrollar una aplicación que nos permita enriquecer ontologías mediante reutilización de otras a través de la extracción modular. Esta aplicación será una herramienta que nos devolverá las ontologías resultantes de la aplicación de cada una de las distintas estrategias. De esta forma podremos comprobar como estos algoritmos y esta metodología de reutilización funcionan a la hora de desarrollar ontologías, podremos ver como de fiables son, la cantidad de axiomas y entidades que introducen y las diferencias entre las estrategias, como aumenta el tamaño de las ontologías y, en definitiva, cualquier característica que pueda ser relevante en el proceso de desarrollo que se explicó anteriormente.

La fiabilidad de esta estrategia no se medirá solo por el correcto funcionamiento de sus algoritmos, sino también por comprobar que las ontologías enriquecidas mantienen su consistencia y su congruencia a través de un razonador.

La aplicación también devolverá el tiempo de ejecución acumulado por cada estrategia para comparar también ese dato.

3.2 Herramientas y tecnologías de desarrollo

Como se mencionó en la introducción para la implementación en Java se utilizará la librería OWL API de la Web Semántica que contiene todo lo necesario para implementar todos los puntos anteriores con la ayuda de las librerías básicas de Java. En concreto se usará la distribución jdk.1.8.0_181 y la OWL API 5.0.0. Se construirá sobre un proyecto Maven v.4.0.0 y se programará usando el entorno Eclipse Java 2019.

Para la visualización de ontologías, así como su revisión, análisis y comparación se está utilizando Protégé 5.5.0. Herramienta open-source disponible en https://protege.stanford.edu.

El razonador utilizado para comprobar la consistencia de las ontologías enriquecidas ha sido el HermiT OWL Reasoner disponible en http://www.hermit-reasoner.com. Concretamente se ha utilizado la versión 1.3.8.500 puesto que es la única compatible con OWL API 5.0.0. [7].

3.3 Diseño de la aplicación en Java y funcionamiento

El proyecto Java constará de las siguientes clases para su funcionamiento:

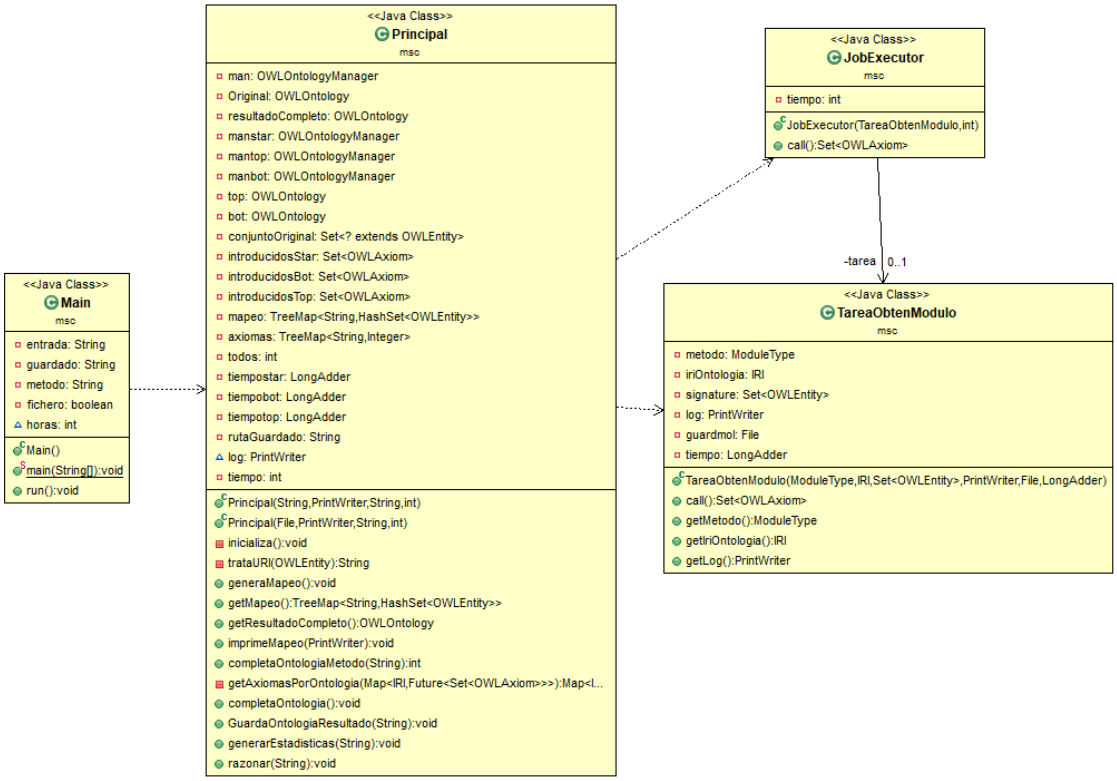

Ilustración - Diagrama de clases de la aplicación

- Clase Principal: Implementará todos los métodos que compongan la funcionalidad indicada en el punto anterior, guardará todas las ontologías y métricas necesarias para el funcionamiento y manejará la mayoría de las excepciones que puedan ir apareciendo en tiempo de ejecución.

- Clase Main: Creará y gestionará las instancias de la clase Principal y servirá como intermediario entre el usuario y la aplicación para saber que parámetros hay que utilizar y que funciones de la clase Principal hay que invocar para llevar a cabo los objetivos especificados. También será la encargada de definir las variables de entrada salida, como la ruta de la

ontología a cargar o donde guardar los ficheros generados como el mapeo o las ontologías cargadas.

- Clase TareaObtenModulo: Implementa la funcionalidad de extracción llamando al método de extracción de SyntacticLocalityModuleExtractor. Sus entidades son creadas desde Principal pero son ejecutadas desde JobExecutor.

- Clase JobExecutor: Clase que crea el hilo y lanza la ejecución correspondiente a una extracción de una ontología importada con una estrategia concreta y que gestiona su TimeOut. Las entidades de esta clase son creadas y lanzadas desde Principal, que crea una instancia de esta clase para cada ontología importada y estrategia.

3.4 Metodología a utilizar en el análisis de resultados y comparación

Los dos parámetros principales con los que se compararán los resultados de las distintas estrategias son el tiempo de ejecución y la cantidad de axiomas extraídos por cada una de ellas.

El caso del tiempo es el más sencillo de los dos, cuanto más tiempo requiera una estrategia para funcionar con respecto a las demás peor. Esta métrica es particularmente importante para la estrategia Top pues como se ha mencionado previamente, según su documentación es muy costoso computacionalmente hablando.
Se trata también de un valor muy subjetivo pues la importancia que se le dé al tiempo es relativa a quien utiliza la aplicación y es tan válido buscar un tiempo de ejecución corto aunque el resultado sea ligeramente peor como querer el mejor resultado posible sin importar lo que tarde o los recursos que consuma en el proceso.

La cantidad de axiomas extraídos es un indicador más difícil de interpretar, pues no basta con ver cuál de las estrategias ha extraído más o menos. Es

posible que una estrategia extraiga más axiomas que los demás pero que estos no fueran realmente necesarios tal y como hemos explicado en la introducción. Como partimos de la base de que las 3 estrategias devuelven la ontología completa entendemos como mejor que nos devuelva los menos axiomas posibles.

Para medir esto, lo que se hará es mostrar ejemplos de cómo han funcionado cada una de las estrategias con los casos de prueba para ver qué axiomas han extraído cada uno de ellos y poder hacernos así una imagen de cuál de ellos se aproxima más al resultado esperado.

También se ha decidido pasar un razonador por las ontologías resultantes de los casos de prueba para comprobar si en todos los casos los resultados son válidos o si alguna de las estrategias tiene tendencia a introducir inconsistencias o algún otro tipo de error para tenerlo también en cuenta de cara a la comparación.

4 Implementación:

4.1 Análisis de Requisitos

En vista de todo lo necesario para llevar a cabo el estudio definido la aplicación deberá ser capaz de todo lo siguiente:

- Leer una ontología que bien le sea proporcionada mediate fichero o mediante su URI, y por tanto ser capaz de resolver y cargar la ontología de internet en ese último caso.
- Generar un mapeo de todas la entidades y axiomas que estén cargados dentro de la ontología usada en el que sean clasificados según la URI de su ontología original. De esta forma sabremos de qué ontología debemos extraer cada uno de los objetos que necesitamos insertar para enriquecer la ontología inicial.
- Cargar desde Internet las ontologías originales a las que hacen referencia los objetos según el mapeo anterior para poder aplicar sobre ellas los distintos métodos de extracción.
- Llevar un registro de todos los objetos cargados para cada una de las 3 estrategias.
- Integrar los objetos extraídos en la ontología pasada como parámetro para así enriquecerla y guardarla en una ruta para que el usuario sea capaz de usarla y visualizarla posteriormente usando un editor de ontologías como Protégé.
- Calcular para cada estrategia la cantidad de objetos extraídos y cargados, así como métricas referentes al tiempo consumido por cada una de ellas.
- Evaluar mediante un razonador si las ontologías resultantes son correctas o si por lo contrario se ha generado algún tipo de consistencia o incongruencia.

4.2 Implementación de la lógica de programa en la aplicación

La lógica detrás del funcionamiento de la aplicación final constará de los siguientes pasos:

1. Main creará un objeto de la entidad Principal usando los parámetros recibidos por la ejecución, siendo estos la ontología, la ruta de guardado y la estrategia que desea aplicarse (Star, Bot, Top o Todas). En su inicialización, Principal cargará la ontología y abortará la ejecución si esta no es accesible.
2. Principal generará el mapeo asociado a la ontología como un TreeMap que asocia la URI de cada ontología importada con el conjunto semilla que hay que extraer de ella y lo guardará en la ruta especificada por el usuario para poder consultarse.
3. Cada una de las ontologías agregadas en el mapeo a excepción de la inicial será cargada desde Internet usando su URI y una vez descargada se aplicarán los métodos de extracción especificados por parámetros, y el resultado se insertará en la inicial. Para ello Principal creará una instancia de JobExecutor y otra de TareaObtenModulo que insertará en la primera. JobExecutor lanzará un hilo que ejecutará el código de extracción implementado en su instancia de TareaObtenModulo. Después de la ejecución de estos Principal obtendrá el módulo extraído y lo incruirá en la ontología original. Si alguna de las ontologías del mapeo no puede ser descargada se encapsulará el error generado, se tratará de descargar con otro formato y, en caso de volver a fallar, se informará de que dicha ontología no ha podido ser descargada y que por tanto se pasará a la siguiente, indicando que los axiomas de ese caso no serán cargados.
4. Para cada extracción TareaObtenModulo escribe en pantalla y en el log las entidades y los axiomas cargados, así como el tiempo de ejecución.
5. Principal guardará la ontología original enriquecida con cada estrategia para poder utilizarse y comprobarse posteriormente.

4.3 Configuración de recursos y entorno de Ejecución

Las pruebas se han llevado a cabo en el servidor de la Universidad de Murcia semantics.inf.um.es. que dispone de Procesador Intel Xeon Intel Xeon E5-2697a con 2.6Ghz, 32 cores y 512 GB de RAM

Las ejecuciones de prueba se han realizado utilizando la opción “Todas” de la aplicación para que sacar métricas de todas las estrategias. El lanzamiento

se realizó asignado 40GB de memoria Ram pues como se ha explicado previamente el algoritmo TOP puede ser muy costoso computacionalmente cuando hay que importar una ontología muy pesada.

En el caso concreto de DOID se ha pasado la ontología como fichero .owl usando la opción -f. La ontología ha sido descargada previamente desde la siguiente dirección https://bioportal.bioontology.org/ontologies/DOID.

4.4 Accesibilidad a la aplicación

La aplicación se encuentra disponible en el siguiente repositorio de GitHub para ser vista y utilizada por cualquiera: https://github.com/AlbarroAkeno/TFG
Para utilizar la aplicación hay que descargar msc-0.0.1-SNAPSHOT.jar de https://github.com/AlbarroAkeno/TFG/tree/master/msc/target y el ejecutable recibe los siguientes parámetros por orden, siendo todos ellos necesarios.

1. Ontología: Por defecto la aplicación espera recibir la URI correspondiente a la ontología que va a enriquecerse. Si se desea pasar un fichero en formato .owl se debe utilizar la opción -f.
2. Ruta de salida: Es el directorio donde la aplicación va a guardar las ontologías enriquecidas, el log de la ejecución, los módulos extraídos y el mapeo de la ontología original.
3. Método: Las opciones son Star, Bot, Top y Todas. Indica a la aplicación que estrategia de extracción modular debe utilizar o, en el último caso, que debe utilizarlas todas.

Existe otra opción de lanzamiento, -t x donde x es el número de horas que la aplicación va a esperar a cada extracción antes de abortarla. Si no se utiliza, el límite por defecto son 2 horas.
La línea de ejecución para el caso de estudio de este TFG sería.
“Java -jar msc-0.0.1-SNAPSHOT.jar -f -t 5 doid.owl salidas/ Todas”
Esta línea completaría la ontología DOID pasada como fichero usando las 3 estrategias disponibles, dando 5 horas a cada extracción para terminar y guardaría todos los resultados en la carpeta salidas.
Como apunte, si se va a usar TOP o Todas es recomendable establecer un límite de memoria alto a la ejecución con la opción de Java -Xmx.

5 Análisis de resultados

5.1 Análisis del funcionamiento con ontologías reales

El caso de estudio elegido para documentar en este TFG será la "Human Disease Ontology" [8] (DOID) obtenida desde BioPortal (https://bioportal.bioontology.org/ontologies/DOID), concretamente se utilizará la release del 07/06/2022.

Las razones por las que se ha elegido este caso de estudio sobre las demás son las siguientes:

- Tiene axiomas importados desde más de 6 ontologías distintas, lo que permite comprobar el funcionamiento con varias ontologías a la vez. Además, entre ellas hay mucha diversidad de tamaños y cantidad de axiomas, desde ontologías muy pequeñas como GENO a la NCBITaxon, una de las más grandes del repositorio.
- Es una ontología bastante utilizada y conocida y que contiene ejemplos muy útiles sobre los que comprobar como los axiomas están siendo cargados por las distintas estrategias.
- Ha sido actualizada muy recientemente, lo que elimina la probabilidad de encontrar errores como los que se mencionarán en el apartado de limitaciones.

Para más información sobre la importancia de la ontología DOID puede consultarse https://academic.oup.com/nar/article/40/D1/D940/2903651 [9]

También se completará con la aplicación la ontología "Ontology for Biomedical Investigations" [10] (OBI) obtenida desde BioPortal (https://bioportal.bioontology.org/ontologies/OBI), concretamente se utilizará la release del 03/01/2022. Pero a diferencia de DOID, esta ontología no se estudiará en detalle, sino que se mostrarán sus resultados en la tabla final de comparaciones.

Para comenzar, se mostrará un ejemplo del problema que estamos tratando:

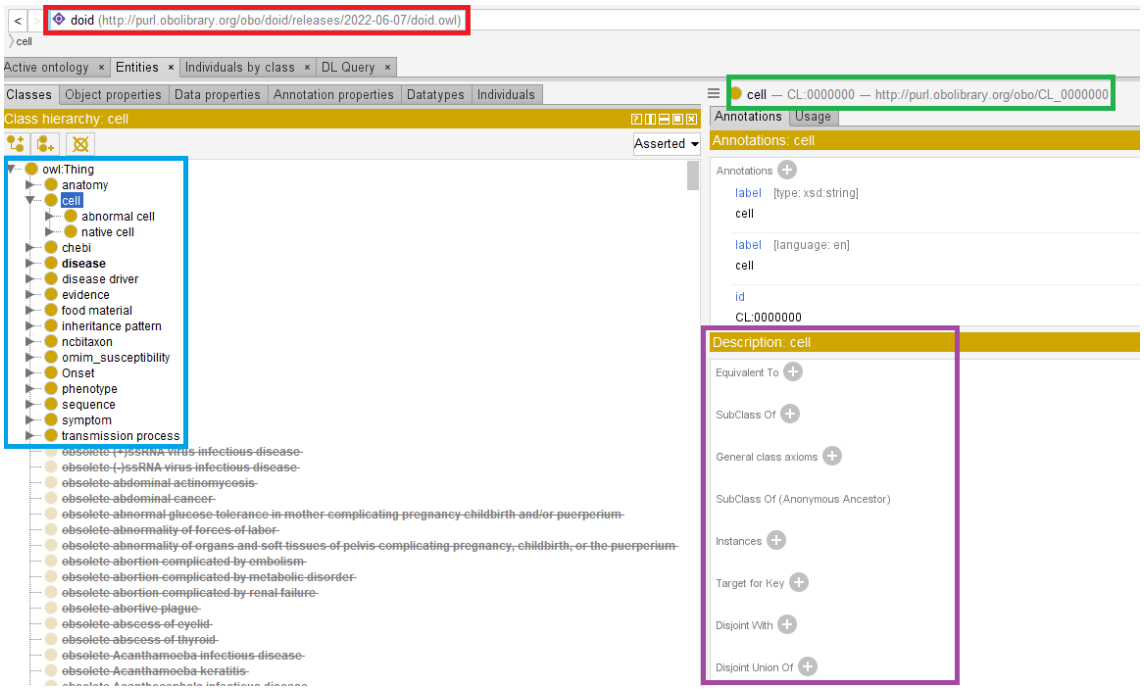

Ilustración 6 – Cell en Ontología DOID

La ilustración 6 muestra la ontología DOID cargada en Protégé y concretamente a la entidad Cell.

Lo primero que nos interesa saber es que esta entidad es importada, esto lo podemos comprobar al comparar la URI del recuadro rojo, que es la URI de la ontología cargada (DOID en este caso) y la URI del recuadro verde, que es la URI de la entidad seleccionada. Como podemos ver, el concepto Cell ha sido importado desde la Cell Ontology (CL) [11].

En el recuadro azul podemos ver la jerarquía de entidades de nuestra ontología, podemos ver que para Cell tenemos dos subtipos, “abnormal cell” y “native cell”.

Sin embargo, en el recuadro morado es donde deberíamos ver todos los axiomas relacionados con esta entidad, pero como se puede observar, está vacío. Esto no nos indica directamente que la ontología se haya exportado implícitamente pues es posible que se trate de una entidad sin axiomas, pero si es la ontología original CL los tenía, aquí también deberíamos tenerlos si queremos que se trate de una importación explícita.

Para comprobar esto se mostrará esta misma entidad en Protégé pero su versión de CL:

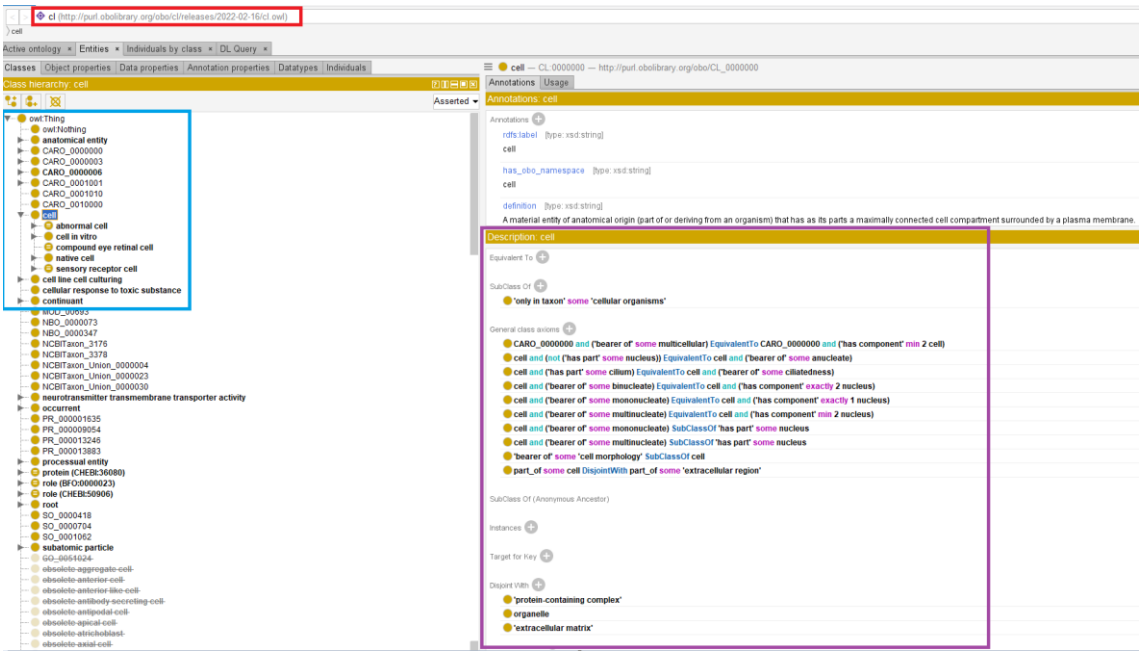

Ilustración 7 - Cell en Ontología CL original

En la versión de CL que vemos en la Ilustración 7 observamos dos diferencias significativas con respecto a la Cell de la DOID:

Que en el recuadro azul en la jerarquía de CL hay más subtipos de célula que en la de DOID, teniendo esta vez además de “abnormal cell” y “native cell” también a “cell in vitro” “compound eye retinal cell” y “sensory receptor cell”.

Que en el recuadro morado hay gran cantidad de axiomas que no estaban cargados en la versión de DOID.

La primera diferencia, el creador de la Disease Ontology consideró que solo necesitaba esos dos subtipos de Cell y por ello solo importó de CL esos. Esto por tanto no tiene nada que ver con el tipo de importación realizada.

La segunda diferencia sí tiene que ver, pues significa que una entidad que el creador importó solo la URI de las entidades, y esto afecta a la información que se almacena en esta ontología.

Como se ha explicado previamente esto no se trata de un error, significa que el creador de DOID ha utilizado reutilización implícita y ha añadido únicamente las URIs de los conceptos que deseaba reusar. Vamos a utilizar la metodología explicada para convertir la reutilización en implícita importando todas esas entidades y sus conceptos y axiomas asociados.

En primer lugar, como se mencionó en la sección anterior de este trabajo debemos generar el mapeo en el sacaremos el conjuntos entidades (OWLEntity)

que necesitamos completar.

Esta imagen sería el ejemplo de dicho mapeo para este caso de estudio concreto:

Ilustración 8 - Ejemplo Mapeo generado

Lo que vemos en la Ilustración 8 es una representación en texto del mapa generado.

En verde tenemos la URI de la Ontología original a la que pertenecía cada entidad y debajo vienen juntos todos esos conceptos. En este caso en verde tenemos la URI de la Ontología CL y en rojo tenemos la URI de la entidad Cell que estamos estudiando junto al resto de entidades de CL.

Esto se hace para todas las que puedan aparecer en DOID, por ejemplo, en azul tenemos las entidades importadas desde la Ontología de bioquímica CHEBI.

Ahora la aplicación descargará cada ontología del mapa y extraerá de ella el conjunto correspondiente de ontologías como conjunto semilla usando el método correspondiente del SyntacticLocalityModuleExtractor proporcionado a la OWL API por la Universidad de Manchester usando las tres estrategias que se han comentado en este trabajo.

Con el objetivo de llevar a cabo la comparación se ha implementado una funcionalidad concreta para poder ver que extrae cada estrategia para cada una de las 6 ontologías importadas en DOID de manera que puedan visualizarse con

Protégé. Los módulos extraídos se guardan como ficheros .owl en la carpeta de destino pasada a la aplicación como parámetro.

Antes de empezar con la comparación vamos a anotar las métricas de la ontología CL original para contrastar luego con la cantidad de axiomas que extrae cada estrategia:

Tabla 1 - Axiomas y entidades extraídos de CL original usando los conjuntos semilla de DOID para cada estrategia

Caso	Axiomas	Porcentaje axiomas	Clases	Porcentaje clases	Peso en MB
Original	238898	100%	16568	100%	57.43
Star	122097	51%	8344	50.36%	23.56
Bot	122357	51.22%	8358	50.45%	23.60
Top	237117	99.25%	16465	99.38%	46.35

Como podemos ver en la Tabla 1 la ontología CL original tiene 238989 axiomas y 165358 clases en total.

También vamos a tener en cuenta que según el mapeo en DOID hay 63 clases importadas de CL que debemos extraer modularmente, es decir, esas 63 entidades son nuestro conjunto semilla.

A continuación, vamos a analizar los resultados para cada estrategia:

Star:

Según los datos de la Tabla 1 la estrategia Star ha interpretado que necesita 122097 axiomas de la ontología original de CL para completar todas las entidades del conjunto semilla. Esto supone un 51% del total de los axiomas y un 50.36% de las entidades. Vamos a ver qué ha extraído en detalle:

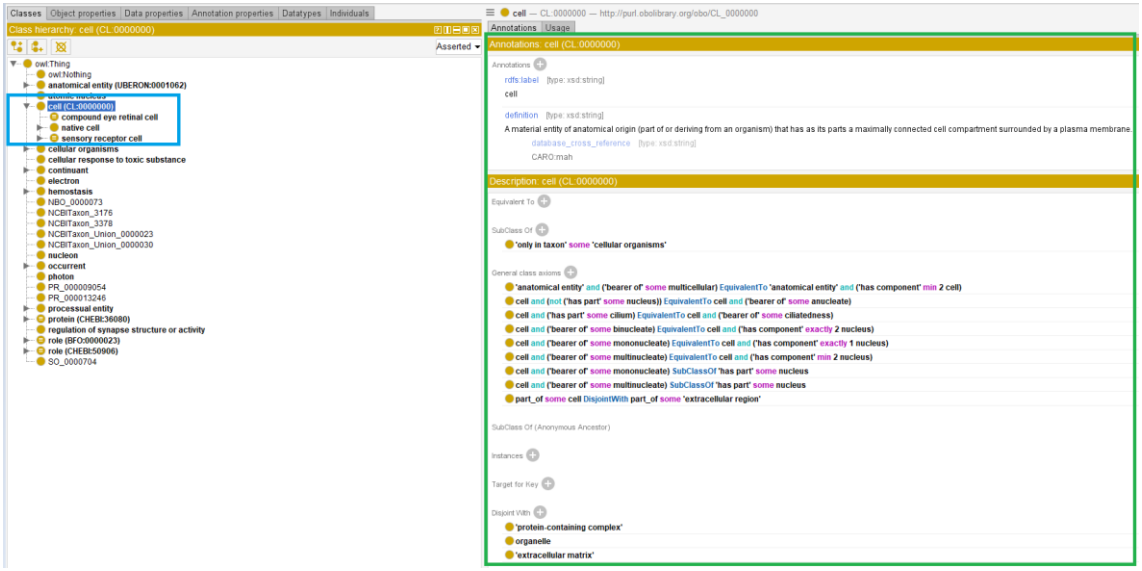

Ilustración 9 - Extracción de Star en CL

Lo primero que salta a la vista en la ilustración 9 es el recuadro verde, donde podemos ver que tanto las anotaciones como los axiomas han sido cargados, así que al introducir el resultado de SyntacticLocalityModuleExtractor en DOID esta tendrá toda esta información disponible, cosa que no ocurría en la Ilustración 6.

Hay que fijarse también en la nueva jerarquía (recuadro azul), donde se visualiza que ha cambiado mucho, pues ha desaparecido “abnormal cell” y se han introducido otros dos subtipos de Cell de los implementados en la CL, “compound eye retinal cell” y “sensory receptor cell”. Tengamos en cuenta que la aplicación está pensada para completar en este caso la DOID, lo que significa que aquí está lo que Star considera que hace falta introducir, eso no quiere decir que no puedan salir entidades u axiomas que ya estaban, pero sí que en ningún caso se va a perder información. En cuanto a las dos que han entrado nuevas están aquí porque Star ha encontrado axiomas que relacionaban a las entidades de Cell importadas en la DOID con ellas y por tanto los ha añadido.

Vamos a ver la jerarquía resultante en más detalle pues el hecho de que haya introducido la “native cell” que sí aparecía en DOID podría ser indicativo de que ha introducido más subtipos de “native cell” o porque ha introducido axiomas nuevos en los subtipos que ya había, como por ejemplo “animal cell”:

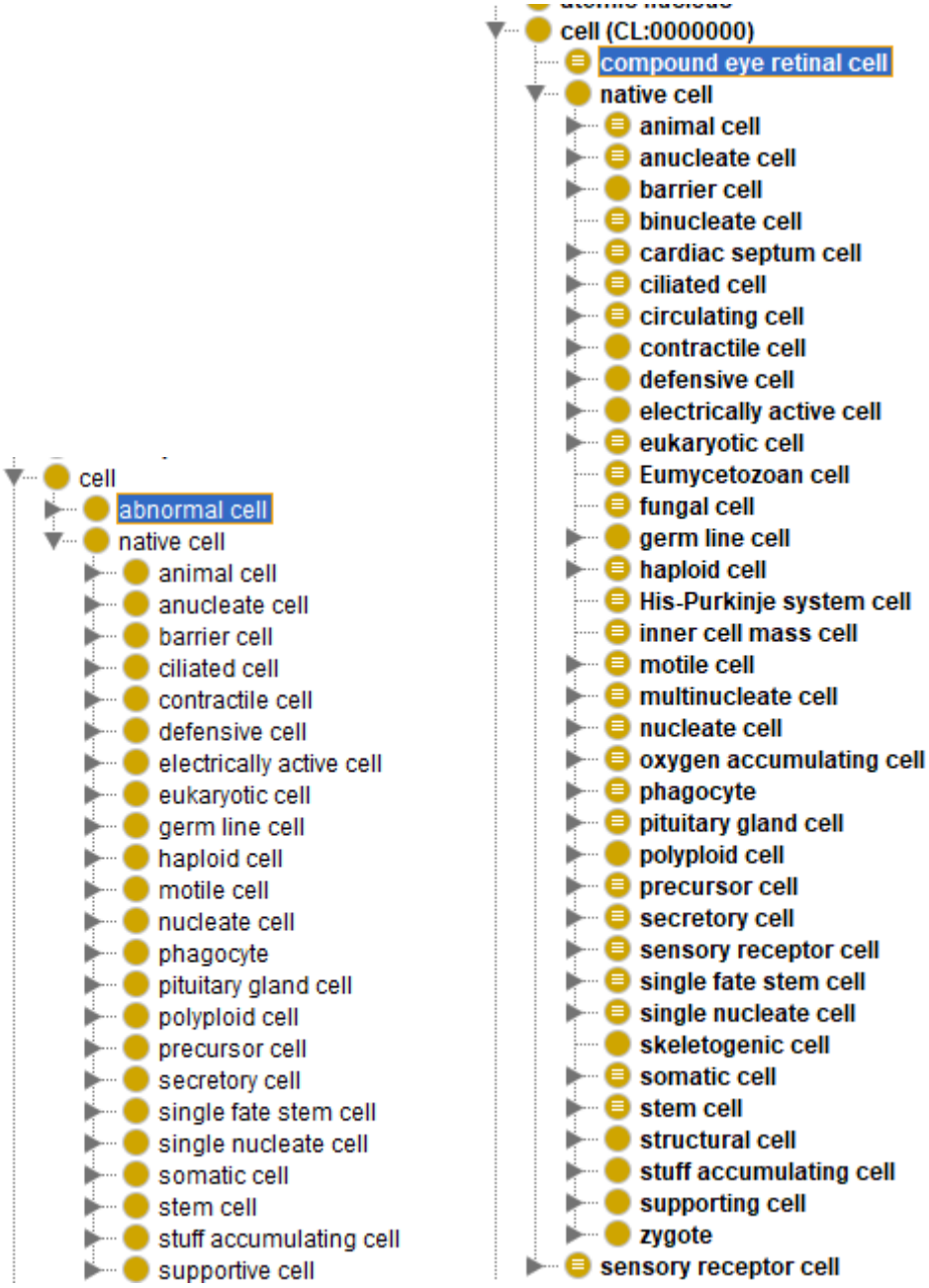

Ilustración 10 - Comparación de Jerarquía Original de DOID y Jerarquia resultante de Star y CL

Efectivamente, la razón por la que ha vuelto a introducir “native cell” es porque la necesitaba para indicar a la DOID que las nuevas entidades introducidas como por ejemplo “fungal cell” son en efecto subclases de la “native cell”. También ha introducido información nueva en las demás, siguiendo el

ejemplo dado previamente de “animal cell”:

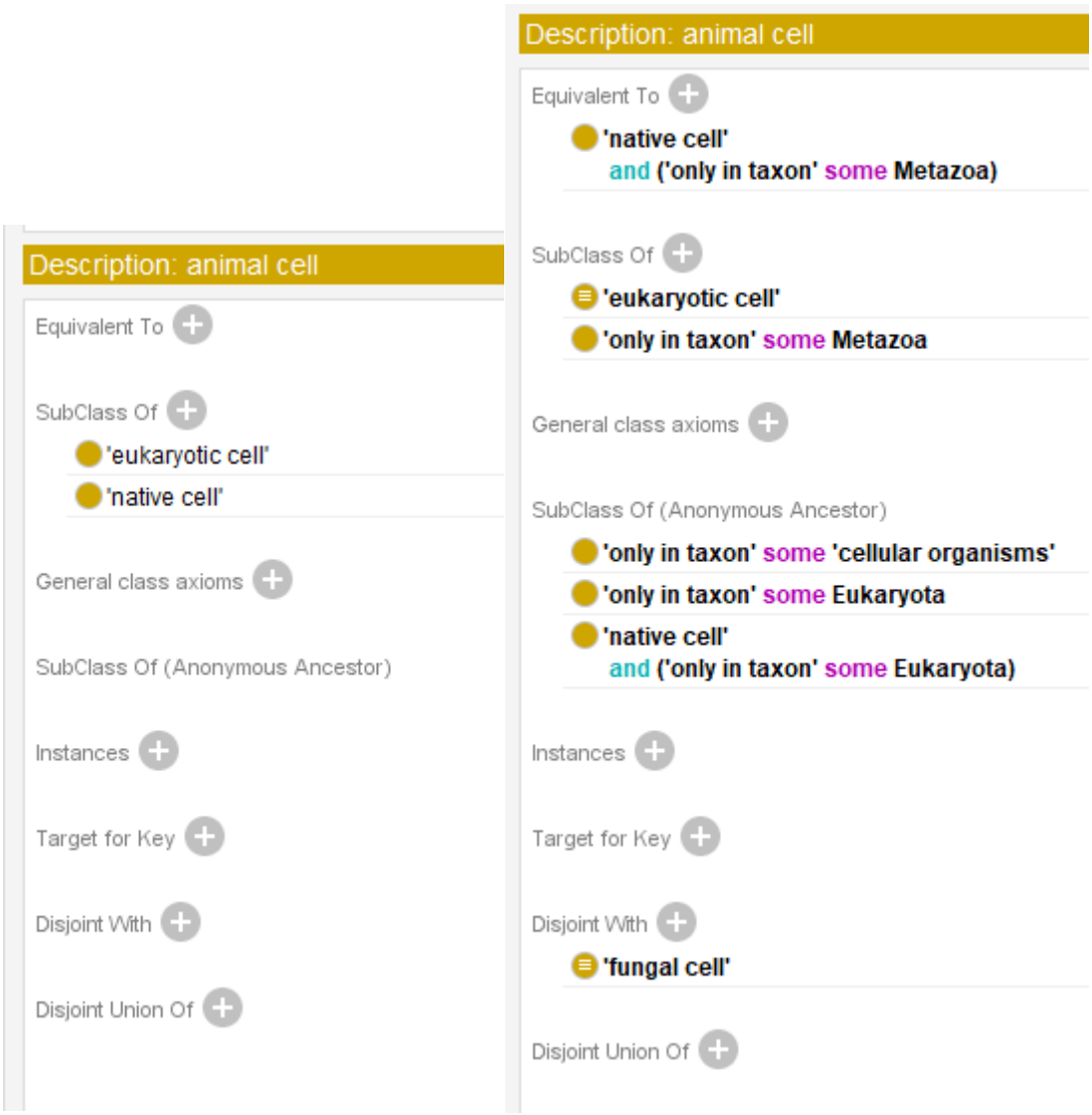

Ilustración 11 - Comparación subclase Animal Cell Doid - Star CL

Como vemos ha completado también las subclases de Cell añadiendo sus axiomas. Y también podemos ver que ha decidido introducir “fungal cell” porque está relacionada con “animal cell” que sí forma parte de DOID.

En general, se observa que Star completa adecuadamente la ontología pasada como parámetro sin necesidad de importar la CL entera, pues le ha bastado con el 50% de esta.

BOT:

Como vemos en la Tabla 1 la estrategia Bot ha extraído 122357 axiomas y 8358 entidades de CL para completar DOID, lo que supone respectivamente un 51.2% y un 50.45% de la CL original. En este sentido, la diferencia con la estrategia Star ha sido mínima por lo que no se espera una diferencia significativa.

Vamos a observar la extracción en detalle:

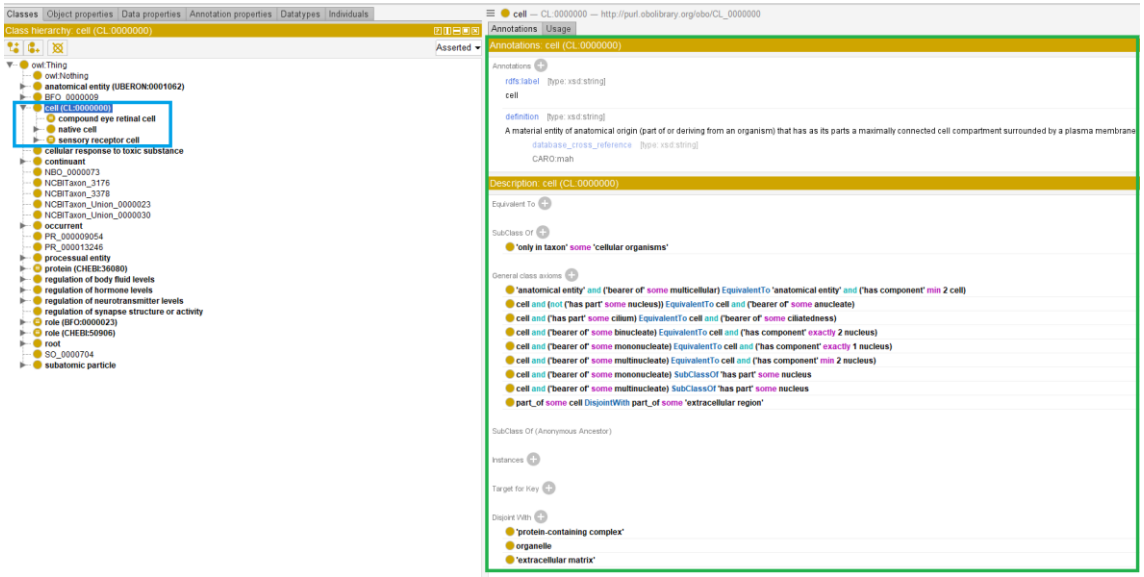

Ilustración 12 - Extracción de CL usando Bot

Podemos ver en la Ilustración 12 que la estrategia BOT también completa correctamente las anotaciones y los axiomas de Cell para DOID.

Al igual que la estrategia Star, ha evitado volver a extraer la entidad “abnormal cell” y ha optado por añadir “compound eye retinal cell” y “sensory receptor cell”.

Las causas por las que ha tomado estas decisiones son las mismas explicadas en el apartado anterior para Star, hay axiomas que enlazan estas entidades directamente con otras importadas en la DOID y que por tanto ha considerado necesarias.

Mostremos de nuevo la comparación de las subclases de “native cell”

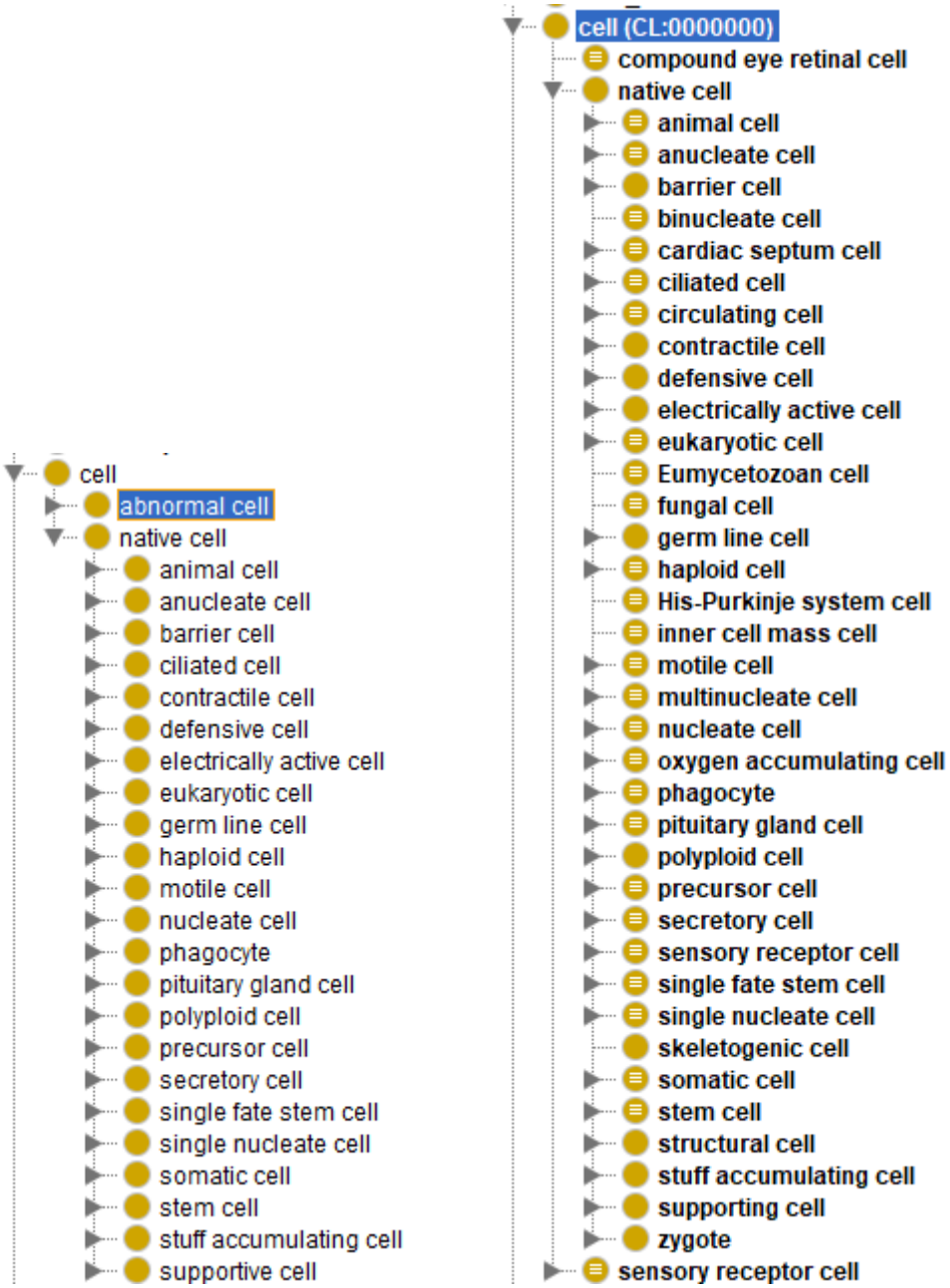

Ilustración 13 - Comparación Jerarquia Original DOID con Jerarquia de Bot en CL

Vemos en Ilustración 13 que ha extraído las mismas entidades que Star por las mismas razones, no hay ninguna diferencia aparente entre ninguna de las dos estrategias para este caso concreto. Era esperable pues solo ha cargado 260 axiomas más, una diferencia de 0.15%.

TOP:

En la Tabla 1 vemos que la estrategia TOP ha extraído para este caso de estudio 237117 axiomas y 16465 entidades, lo que supone respectivamente un 99,21% y un 99.38% de la CL original. Esto supone que casi no habría habido diferencia en este caso entre usar un extractor modular e importar la totalidad de la ontología de CL desde la DOID original.

Vamos a ver en detalle la extracción:

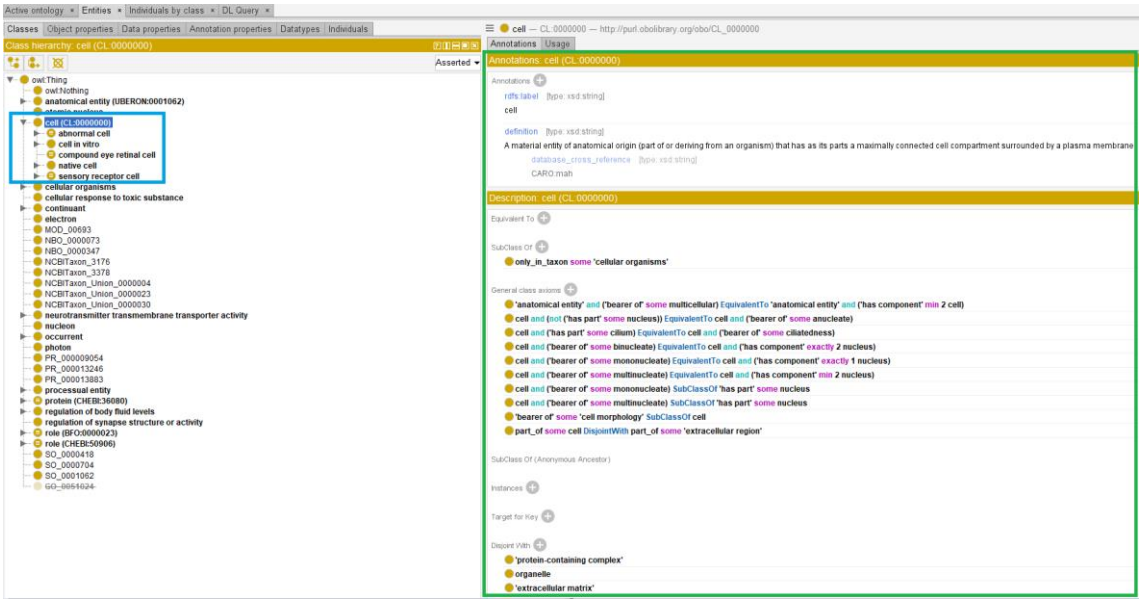

Ilustración 14 - Extracción de CL usando TOP

En la Ilustración 14 vemos que al igual que las otras dos estrategias ha completado correctamente las anotaciones y los axiomas directos de Cell, lo cual nos indica que para este objetivo cualquiera de las tres opciones es utilizable. Ahora debemos fijarnos en la jerarquía final de entidades para comprobar cuantas ha considerado necesarias incluir en comparación. Como podemos ver en el recuadro azul TOP sí ha extraído de nuevo “abnormal cell”, cosa que no han hecho las otras dos, pero es que ha extraído también “cell in vitro”, lo cual, si nos fijamos en la Ilustración 1 CL original implica que ha extraído todas las subclases de Cell. Teniendo en cuenta que TOP ha extraído casi la totalidad de CL es esperable que la jerarquía extraída por ella sea casi idéntica si no exactamente la de CL original.

Ilustración 15 - Comparación Jerarquías Cell Original con DOID - Extracción TOP CL

En la Ilustración 15 podemos confirmar que efectivamente TOP ha extraído la totalidad de la rama de Cell de CL para introducirla.

Obviamente, si importamos totalmente las ontologías externas a DOID la tendremos completa sin absolutamente ningún tipo de pérdida de información; sin embargo, esto la llenará axiomas y entidades que el creador original no consideraba necesarias, supone no obtener ninguna ventaja al elegir usar un extractor modular en lugar de importarla entera directamente.

Aun así, es pronto para descartar el uso de la estrategia TOP pues es posible que este resultado se deba a un caso particular para la ontología CL, todo lo que hemos podido demostrar es que Star y BOT encuentran una forma de completar la ontología sin necesidad de importarla entera, sino con aproximadamente la mitad de los axiomas.

En el apartado siguiente vamos a indagar más en esa última posibilidad de que TOP funcione mejor en otros casos, comparando sin entrar en detalle cómo ha funcionado para el resto de las ontologías implicadas en nuestro caso de estudio de DOID.

5.2 Comparación de resultados y estudio

En el apartado anterior hemos visto detalladamente como han funcionado las distintas estrategias en el caso de importar la ontología CL a DOID para poder hacernos una idea de que está pasando a bajo nivel.
Sin embargo, en este apartado queremos comparar como han funcionado solo desde el punto de vista de los números, siguiendo los criterios que se explicaron en la sección de análisis 3.5

Seguiremos usando el caso de estudio de DOID que se ha mencionado previamente. Vamos a tener en cuenta el porcentaje de axiomas extraídos de cada una con respecto a su ontología original y también se compararán los tiempos de cada uno de ellos para cada caso:

CHEBI:

CHEBI [12] son las siglas para “Chemical Entities of Biological Interest Ontology” y la referencia es la siguiente URI http://purl.obolibrary.org/obo/CHEBI. Con 600MB en su versión .owl, 180297 entidades y 2987597 axiomas es la segunda más grande de las 6 y una de las más extensas del repositorio.

En DOID hay 89 clases importadas de CHEBI que completar. Su extracción ha tardado los siguientes tiempos para cada estrategia:

```
Leyendo: http://purl.obolibrary.org/obo/CHEBI
Leido http://purl.obolibrary.org/obo/CHEBI
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CHEBI 27 segundos.
Procesado Star http://purl.obolibrary.org/obo/CHEBI
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CHEBI 27 segundos.
Procesado BOT http://purl.obolibrary.org/obo/CHEBI
La estrategia TOP ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CHEBI 6528 segundos.
Procesado http://purl.obolibrary.org/obo/CHEBI
```

Ilustración 16 - Resultado de la Terminal para Chebi

Como vemos en los resultados de la Ilustración 16 tanto Star como BOT han tardado lo mismo, necesitando unos 27 segundos, suponiendo un tiempo bastante pequeño, lo que contrasta bastante con TOP, que ha necesitado 1 hora, 48 minutos y 48 segundos solo para esta ontología.

Veamos ahora entidades y axiomas:

Star: Para completar Doid Star ha extraído de CHEBI 17721 axiomas y 981 entidades. Esto supone respectivamente un 0,6% de los axiomas y un 0.54% de los conceptos de la ontología original. El módulo ocupa 3.28 MB.

BOT: Para completar DOID Bot ha extraído de CHEBI 18275 axiomas y 1030 entidades. Esto supone respectivamente un 0.61% de los axiomas y un 0.57% de los conceptos de la ontología original. El módulo ocupa 3.44 MB

TOP: Para completar DOID Top ha extraído de CHEBI 2912640 axiomas y 161670 entidades. Esto supone respectivamente un 97.49% de los axiomas y un 89,67% de los conceptos de la ontología original. El módulo ocupa 541.82 MB.

Tabla 2 - Resultados Extracción Chebi para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	17721	0.6%	981	0.54%	27	3.28
Bot	18275	0.61%	1030	0.57%	27	3.44
Top	2912640	97.49%	161670	89.67%	6528	541.82

Con CHEBI nos encontramos un caso muy parecido al que hemos estudiado con CL. Para completar unas 89 entidades importadas en DOID a Star y BOT les ha bastado con alrededor de un 0.6% de los axiomas y entidades de CHEBI, mientras que TOP ha necesitado de nuevo casi la totalidad de la ontología y además para hacerlo ha tardado casi 2 horas.
Incluso si el resultado de TOP fuera mejor, que no lo es, y el tiempo nos diera igual, seguiría siendo mejor opción cargar CHEBI en DOID directamente con Protégé en lugar de usar un extractor modular.

FOODON:

FOODON son las siglas para “The FoofOn Food Ontology” [13] y la referencia es la siguiente URI: http://purl.obolibrary.org/obo/FOODON.
Con 7 MB en su versión .owl, 32516 entidades y 314196 axiomas se trata de una ontología mediana y del tamaño más habitual del repositorio de OBOFoundry.

DOID importa 24 clases de esta ontología que debemos completar. Su extracción ha tardado los siguientes tiempos para cada estrategia:

```
Leyendo: http://purl.obolibrary.org/obo/FOODON
Leido http://purl.obolibrary.org/obo/FOODON
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/FOODON 3 segundos.
Procesado Star http://purl.obolibrary.org/obo/FOODON
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/FOODON 2 segundos.
Procesado BOT http://purl.obolibrary.org/obo/FOODON
La estrategia TOP ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/FOODON 195 segundos.
Procesado http://purl.obolibrary.org/obo/FOODON
```

Ilustración 17 – Resultado de la terminal para FOODON

De nuevo volvemos a encontrarnos con que el algoritmo TOP tarda significativamente más que Star y BOT, aunque esta vez al tratarse de una ontología más pequeña es un tiempo más razonable.

Star ha necesitado 3 segundos y BOT ha necesitado 2, haciendo las extracciones casi al instante. TOP ha necesitado 3 minutos y 15 segundos.
Veamos ahora entidades y axiomas:

Star: Para completar Doid Star ha extraído de FOODON 2749 axiomas y 282 entidades. Esto supone respectivamente un 0,87% de los axiomas y un 0.87% de los conceptos de la ontología original. El módulo ocupa 0.45 MB.

BOT: Para completar DOID Bot ha extraído de FOODON 2790 axiomas y

286 entidades. Esto supone respectivamente un 0.89% de los axiomas y un 0.88% de los conceptos de la ontología original. El módulo ocupa 0.46 MB.

TOP: Para completar DOID Top ha extraído de FOODON 280634 axiomas y 30476 entidades. Esto supone respectivamente un 89.32% de los axiomas y un 93,72% de los conceptos de la ontología original. El módulo ocupa 47.66 MB, lo cual es bastante extraño pues la FOODON original pesa 7 MB. Nótese que, en todos los casos, el peso de los módulos extraídos es mucho mayor en proporción. Esto se debe al formato de guardado de OWL API.

Tabla 3 - Resultados extracción Foodon DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	2749	0.87%	282	0.87%	3	0.45
Bot	2790	0.89%	286	0.88%	2	0.46
Top	280634	89.32%	30476	93.72%	195	47.66

De la carga de FOODON sacamos las mismas conclusiones que de la carga de CHEBI pues hemos visto el mismo comportamiento para las 3 estrategias.

Tanto Star como BOT necesitan descargar menos de 1% de la ontología original para completar sus entidades en la DOID, pero de nuevo TOP necesita extraer prácticamente la totalidad de la ontología.

La conclusión es la misma que en CHEBI, si el resultado de aplicar TOP fuese mejor que el de las otras dos estrategias seguiría siendo preferible importar FOODON directamente a DOID que utilizar un extractor modular pues la diferencia es de solo el 10% de las entidades y axiomas.

CL:

Los datos de CL [11] ya los sabemos pues se han explicado en el apartado anterior, así que simplemente se van a mostrar los tiempos de ejecución para

cada estrategia, pues es el único dato que no se ha comentado aún.

```
Leyendo: http://purl.obolibrary.org/obo/CL
Leido http://purl.obolibrary.org/obo/CL
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CL 42 segundos.
Procesado Star http://purl.obolibrary.org/obo/CL
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CL 17 segundos.
Procesado BOT http://purl.obolibrary.org/obo/CL
La estrategia TOP ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/CL 44 segundos.
Procesado http://purl.obolibrary.org/obo/CL
```

Ilustración 18 - Resultados de la terminal para CL

Los tiempos de CL se salen un poco de la norma que hemos visto en las dos ontologías anteriores, pues esta vez la estrategia Star y TOP han tardado casi lo mismo (42 y 44 segundos) dejando a BOT con el mejor tiempo de las tres con sus 17 segundos.

Recordemos que tanto Star como BOT extraían aproximadamente el 50% de los axiomas y entidades de CL mientras que TOP sacaba prácticamente el 100%, por lo que en cuanto a cantidad de axiomas extraídos el comportamiento sí ha sido el mismo, siendo lo único reseñable que esta vez Star y BOT han necesitado un porcentaje mucho mayor del que han necesitado previamente, pasando menos del 1% a la mitad de la ontología.

GENO:

GENO son las siglas para “Genotype Ontology” [14] y su referencia en DOID es la siguiente URI: http://purl.obolibrary.org/obo/GENO.
Con 0,62 MB en su versión .owl, 424 entidades y 3545 axiomas es la segunda ontología más pequeña de nuestro caso de estudio.

DOID importa 9 clases de esta ontología que debemos completar. Su extracción ha tardado los siguientes tiempos para cada estrategia:

```
Leyendo: http://purl.obolibrary.org/obo/GENO
Leido http://purl.obolibrary.org/obo/GENO
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/GENO 0 segundos.
Procesado Star http://purl.obolibrary.org/obo/GENO
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/GENO 0 segundos.
Procesado BOT http://purl.obolibrary.org/obo/GENO
La estrategia TOP ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/GENO 0 segundos.
Procesado http://purl.obolibrary.org/obo/GENO
```

Ilustración 19 - Resultados de la terminal para GENO

Este caso concreto es tan pequeño que no hemos podido captar ninguna diferencia de tiempo entre las estrategias, las tres han sido instantáneas.
Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de GENO 173 axiomas y 12

entidades. Esto supone respectivamente un 4,88% de los axiomas y un 2.83% de los conceptos de la ontología original. Su módulo pesa 0.03 MB.

BOT: Para completar DOID Bot ha extraído de GENO 311 axiomas y 25 entidades. Esto supone respectivamente un 8.77% de los axiomas y un 5.9% de los conceptos de la ontología original. Su módulo pesa 0.05 MB.

TOP: Para completar DOID Top ha extraído de GENO 2723 axiomas y 375 entidades. Esto supone respectivamente un 76.81% de los axiomas y un 88,44% de los conceptos de la ontología original. Su módulo pesa 0.52MB

Tabla 4 - Resultados de Extracción Geno para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	173	4.88%	12	2.83%	0	0.03
Bot	311	8.77%	25	5.9%	0	0.05
Top	2723	76.81%	375	88.44%	0	0.52

El comportamiento de GENO es, de nuevo, muy parecido al del resto de ontologías vistas.

Star y BOT necesitan un porcentaje pequeño de la ontología original para completar los conceptos reutilizados en DOID, pero TOP ha necesitado de nuevo un porcentaje muy grande, un 76% de axiomas y un 88% de las entidades. Esta es la vez que TOP ha necesitado menos y aun así el porcentaje es muy grande. Tratándose de una ontología pequeña, es de nuevo mejor integrar GENO entera en DOID que utilizar el extractor modular con la estrategia TOP.

HP:

HP son las siglas para “Human Phenotype Ontology” [15] y su referencia en DOID es la siguiente URI: http://purl.obolibrary.org/obo/HP.
Con 93 MB en su versión .owl, 31277 entidades y 386306 axiomas se considera de tamaño medio.

DOID importa 102 conceptos de HP que debemos completar. Su extracción ha devuelto los siguientes tiempos para cada estrategia:

```
Leyendo: http://purl.obolibrary.org/obo/HP
Leido http://purl.obolibrary.org/obo/HP
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/HP 77 segundos.
Procesado Star http://purl.obolibrary.org/obo/HP
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/HP 31 segundos.
Procesado BOT http://purl.obolibrary.org/obo/HP
La estrategia TOP ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/HP 185 segundos.
Procesado http://purl.obolibrary.org/obo/HP
```

Ilustración 10 - Resultados Terminal para HP

Esta vez tenemos tiempos distintos para las 3 estrategias. De nuevo BOT es el mejor algoritmo con 31 segundos, seguido de Star que ha necesitado 1 minuto 17 segundos y TOP vuelve a ser el que más tiempo requiere con 3 minutos y 5 segundos.

De nuevo, una ontología de tamaño medio devuelve unos tiempos aceptables para las 3 estrategias, aunque la diferencia entre TOP y BOT sea significativa pues TOP tarda 6 veces más.

Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de HP 173661 axiomas y 11344 entidades. Esto supone respectivamente un 44,95% de los axiomas y un 36.27% de los conceptos de la ontología original. Su módulo pesa 34.32 MB.

BOT: Para completar DOID Bot ha extraído de HP 173940 axiomas y 11360 entidades. Esto supone respectivamente un 45.03% de los axiomas y un 36,32% de los conceptos de la ontología original. Su módulo pesa 34.36 MB.

TOP: Para completar DOID Top ha extraído de HP 376784 axiomas y 30039 entidades. Esto supone respectivamente un 97.54% de los axiomas y un 96,04% de los conceptos de la ontología original. Su módulo pesa 72.7 MB.

Tabla 5 - Resultados Extracción HP para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	173661	44.95%	11344	36.27%	77	34.32
Bot	173940	45.03%	11360	36.32%	31	34.36
Top	376784	97.54%	30039	96.04%	185	72.7

La extracción sobre HP ha tenido unos resultados muy parecidos a los que tuvimos en CL, y es que, si bien de nuevo TOP extrae la ontología casi entera mientras que BOT y Star requieren mucho menos, esta vez las dos últimas han extraído un porcentaje mayor de la ontología de entre el 40 y 50%.

Una vez más utilizar el extractor modular para usar TOP sigue sin ser mejor opción que importar directamente HP entera a DOID con Protégé, mientras que usar Star y Boti sí es una opción.

NCBITaxon

NCBITaxon son las siglas para “National Center for Biotechnology Information (NCBI) Organismal Classification” [16] cuya referencia en DOID se corresponde con la siguiente URI: http://purl.obolibrary.org/obo/NCBITaxon.
Con 1,837 GB en su versión .owl, 14614235 axiomas y 2401826 entidades es la ontología más grande utilizada por DOID y una de las más grandes del repositorio. Es tan grande que solo abrirla en Protégé requiere de 12,5 GB de Ram:

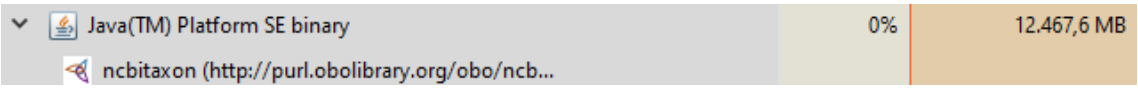

Ilustración 11 - Consumo de Ram al cargar en memoria Ontologías grandes

Y esto ha causado los siguientes problemas la hora de ejecutar la aplicación para extracción:

```
Leyendo: http://purl.obolibrary.org/obo/NCBITaxon
Leido http://purl.obolibrary.org/obo/NCBITaxon
La estrategia star ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/NCBITaxon 196 segundos.
Procesado Star http://purl.obolibrary.org/obo/NCBITaxon
La estrategia BOT ha tardado en extraer los axiomas de http://purl.obolibrary.org/obo/NCBITaxon 172 segundos.
Procesado BOT http://purl.obolibrary.org/obo/NCBITaxon
```

Ilustración 12 - Resultados Terminan NCBITaxom

DOID importa 318 conceptos de NCBITaxon.

El proceso de extracción ha funcionado para Star y para BOT, pero no se ha completado para TOP en más de 8 días de ejecución, se hablará de este problema en la sección de limitaciones.

Star ha necesitado 3 minutos y 16 segundos mientras que BOT ha necesitado 2 minutos y 52 segundos.

Star: Para completar DOID Star ha extraído de NCBITaxon 8203 axiomas y 973 entidades. Esto supone respectivamente un 0.056% de los axiomas y un 0.04% de los conceptos de la ontología original. Su módulo ocupa 1.35 MB.

BOT: Para completar DOID Bot ha extraído de NCBITaxon 8217 axiomas

y 975 entidades. Esto supone respectivamente un 0.056% de los axiomas y un 0,041% de los conceptos de la ontología original. Su módulo ocupa 1.35 MB

TOP: No ha terminado tras 1 semana de ejecución. El algoritmo de extracción parece que tiene algún tipo de error o de circunstancia que da lugar a bucles infinitos dado que implementan funciones recurrentes que podrían no converger nunca.

Tabla 6 - Resultados extracción NCBITaxiom para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	8203	0.056%	973	0,04%	196	1.35
Bot	8217	0.056%	975	0.041%	172	1.5
Top	/	/	/	/	+8 Días	/

En esta última ontología vista de DOID los resultados siguen constantes, con Star y Bot requiriendo un porcentaje muy semejante de la ontología original en todos los casos vistos, pero con la diferencia de que es la primera vez que vemos al método de extracción incapaz de terminar una, en este caso, TOP.

SO

SO es la abreviatura para Sequence Types and Features Ontology [17] cuya referencia en DOID se corresponde con la siguiente URI http://purl.obolibrary.org/obo/SO
Con 5 MB en su versión .owl, 25584 axiomas y 2728 entra en la categoría de ontologías pequeñas.

DOID importa 16 conceptos de SO. Su extracción ha devuelto los siguientes tiempos para cada estrategia:

```
Leyendo: <http://purl.obolibrary.org/obo/so.owl>
Leido: <http://purl.obolibrary.org/obo/so.owl>
La estrategia star ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/so.owl> 0 segundos.
Se han extraido 588 axiomas y 58 entidades con nested de <http://purl.obolibrary.org/obo/so.owl>
La estrategia bottom ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/so.owl> 0 segundos.
Se han extraido 622 axiomas y 61 entidades con bottom de <http://purl.obolibrary.org/obo/so.owl>
La estrategia top ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/so.owl> 2 segundos.
Se han extraido 22626 axiomas y 2337 entidades con top de <http://purl.obolibrary.org/obo/so.owl>
```

Ilustración 13 - Resultados Terminal SO

De nuevo tenemos una ontología tan pequeña que la diferencia entre Star y Bot no es apreciable. Si bien Top sí ha necesitado mas tiempo que las otras

dos, un tiempo de 2 segundos no es relevante.

Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de SO 588 axiomas y 58 entidades. Esto supone respectivamente un 2,3% de los axiomas y un 2.13% de los conceptos de la ontología original. El módulo ocupa 0.1 MB.

BOT: Para completar DOID Bot ha extraído de SO 622 axiomas y 61 entidades. Esto supone respectivamente un 2.43% de los axiomas y un 2,24% de los conceptos de la ontología original. El módulo ocupa 0.11 MB.

TOP: Para completar DOID Top ha extraído de SO 22626 axiomas y 2337 entidades. Esto supone respectivamente un 88.43% de los axiomas y un 85,66% de los conceptos de la ontología original. El módulo ocupa 3.75 MB.

Tabla 7 - Resultado extracción SO para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	588	2.3%	58	2.13%	0	0.1
Bot	622	2.43%	61	2.24%	0	0.11
Top	22626	88.43%	2337	85.66%	2	3.75

Las estrategias de extracción para SO de han comportado de la misma manera que con la mayoría de las ontologías vistas hasta ahora, Star y Bot apenas extraen información para completar DOID mientras que TOP ha extraído casi todos los axiomas y entidades de SO.

SYMP

SYMP es la abreviatura de Symptom Ontology [18] cuya referencia en DOID se corresponde con la siguiente URI http://purl.obolibrary.org/obo/SYMP Con 0,89 MB en su versión .owl, 6025 axiomas y 968 entidades es una de las ontologías más pequeñas importadas en nuestro caso de estudio.

DOID importa 266 conceptos de SYMP, lo cual supone un porcentaje considerable de la ontología (25% aproximadamente). Su extracción ha devuelto los siguientes tiempos para cada estrategia:

```
Leyendo: <http://purl.obolibrary.org/obo/symp.owl>
Leido: <http://purl.obolibrary.org/obo/symp.owl>
La estrategia star ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/symp.owl> 0 segundos.
Se han extraido 2130 axiomas y 315 entidades con star de <http://purl.obolibrary.org/obo/symp.owl>
La estrategia bottom ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/symp.owl> 0 segundos.
Se han extraido 2130 axiomas y 315 entidades con bottom de <http://purl.obolibrary.org/obo/symp.owl>
La estrategia top ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/symp.owl> 0 segundos.
Se han extraido 5470 axiomas y 867 entidades con top de <http://purl.obolibrary.org/obo/symp.owl>
```

Ilustración 14 - Resultados terminal Symp

Al igual que con GENO y SO esta ontología es tan pequeña que no hay diferencia de tiempo entre las distintas estrategias pues ninguna de las tres ha pasado del segundo.

Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de SYMP 2130 axiomas y 315 entidades. Esto supone respectivamente un 35.35% de los axiomas y un 32.54% de los conceptos de la ontología original. Su módulo ocupa 0.32 MB

BOT: Para completar DOID Bot ha extraído de SYMP 2130 axiomas y 315 entidades. Esto supone respectivamente un 35.35% de los axiomas y un 32.54% de los conceptos de la ontología original. Su módulo ocupa 0.32 MB

TOP: Para completar DOID Top ha extraído de SYMP 5470 axiomas y 869 entidades. Esto supone respectivamente un 90.79% de los axiomas y un 89,77% de los conceptos de la ontología original. Su módulo ocupa 0.8 MB.

Tabla 8 - Resultado Extracción SYMP para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	2130	35.35%	315	32.54%	0	0.32
Bot	2130	35.35%	315	32.54%	0	0.32
Top	5470	90.79%	869	89.77%	0	0.8

Por primera vez en nuestro caso de estudio Star y Bot han tenido exactamente el mismo resultado, lo que significa que en este caso no hay nada que Bot extraiga y Top no. Tenemos una subida para ambos en el porcentaje extraído que lo eleva a un 35% lo cual aun siendo moderado es más de a lo que nos tienen acostumbrados estas estrategias, pero esto se debe a que DOID importa un porcentaje bastante elevado de SYMP, así que es normal que necesiten un porcentaje parecido. Una vez más, Top extrae la ontología original prácticamente completa.

TRANS

TRANS es la abreviatura de Nurse Transitional [19] cuya referencia en DOID se corresponde con la siguiente URI http://purl.obolibrary.org/obo/TRANS Con 0.057 MB en su versión .owl, 232 axiomas y 35 entidades es la ontología más pequeña de nuestro caso de estudio.

DOID importa 13 conceptos de TRANS, lo cual supone un porcentaje considerable de la ontología (35% aproximadamente).

Su extracción ha devuelto los siguientes tiempos para cada estrategia

```
Leyendo: <http://purl.obolibrary.org/obo/trans.owl>
Leido: <http://purl.obolibrary.org/obo/trans.owl>
La estrategia star ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/trans.owl> 0 segundos.
Se han extraido 83 axiomas y 15 entidades con star de <http://purl.obolibrary.org/obo/trans.owl>
La estrategia bottom ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/trans.owl> 0 segundos.
Se han extraido 114 axiomas y 19 entidades con bottom de <http://purl.obolibrary.org/obo/trans.owl>
La estrategia top ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/trans.owl> 0 segundos.
Se han extraido 163 axiomas y 28 entidades con top de <http://purl.obolibrary.org/obo/trans.owl>
```

Ilustración 15 - Resultado Terminal Trans

Como con todas las ontologías muy pequeñas, y especialmente esta que es la más pequeña el tiempo en todos los casos ha sido menor al segundo, por lo que no hay ninguna diferencia en tiempo para este caso.
Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de TRANS 83 axiomas y 15 entidades. Esto supone respectivamente un 35.78% de los axiomas y un 42.86% de los conceptos de la ontología original. Su módulo ocupa 0.02 MB.

BOT: Para completar DOID Bot ha extraído de TRANS 114 axiomas y 19 entidades. Esto supone respectivamente un 49.14% de los axiomas y un 54.29% de los conceptos de la ontología original. Su módulo ocupa 0.02 MB.

TOP: Para completar DOID Top ha extraído de TRANS 163 axiomas y 28 entidades. Esto supone respectivamente un 70.26% de los axiomas y un 80% de los conceptos de la ontología original. Su módulo ocupa 0.03 MB

Tabla 9 - Resultados Extracción TRANS para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	83	35.78%	15	42.86%	0	0.02
Bot	114	49.14%	19	54.29%	0	0.02
Top	163	70.26%	28	80%	0	0.03

Este es el caso hasta ahora más equilibrado entre las tres estrategias debido a varias causas. La primera es que el porcentaje de entidades importadas inicialmente por DOID desde TRANS es bastante alto lo que ha hecho los resultados de Star y Top hayan sido más altos y el hecho de TOP ha extraído un porcentaje más bajo de lo que suele obtener, el cual, aunque alto, es lo suficiente lejano al 100% como para justificar su uso.

Aun siendo equilibrado, Star y Bot siguen consiguiendo un resultado significativamente mejor, y en este caso Star particularmente mejor que Bot.

UBERON

UBERON es la abreviatura para Uber Anatomy Ontology [20] cuya referencia en DOID es la siguiente URI http://purl.obolibrary.org/obo/UBERON Con 65 MB en su versión .owl, 284591 axiomas y 21083 entidades en una ontología mediana y es la última importada en nuestro caso de estudio.

DOID importa 391 conceptos de UBERON. Su extracción ha devuelto los siguientes tiempos para cada estrategia:

```
Leyendo: <http://purl.obolibrary.org/obo/uberon.owl>
Leido: <http://purl.obolibrary.org/obo/uberon.owl>
La estrategia star ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/uberon.owl> 42 segundos.
Se han extraido 105652 axiomas y 6160 entidades con star de <http://purl.obolibrary.org/obo/uberon.owl>
La estrategia bottom ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/uberon.owl> 18 segundos.
Se han extraido 106005 axiomas y 6172 entidades con bottom de <http://purl.obolibrary.org/obo/uberon.owl>
La estrategia top ha tardado en extraer los axiomas de <http://purl.obolibrary.org/obo/uberon.owl> 138 segundos.
Se han extraido 271315 axiomas y 19460 entidades con top de <http://purl.obolibrary.org/obo/uberon.owl>
```

Ilustración 16 - Resultados Terminal UBERON

Esta última ontología mantiene la temática vista en otras ontologías previas de tamaño parecido, Star y Bot tienen un tiempo muy significativamente mejor que Top, y de nuevo Bot obtiene mejor tiempo que star.
Veamos ahora entidades y axiomas:

Star: Para completar DOID Star ha extraído de UBERON 105652 axiomas y 6160 entidades. Esto supone respectivamente un 37.12% de los axiomas y un

29.22% de los conceptos de la ontología original. Su módulo ocupa 20.63 MB.

BOT: Para completar DOID Bot ha extraído de UBERON 106005 axiomas y 6172 entidades. Esto supone respectivamente un 37.25% de los axiomas y un 29.27% de los conceptos de la ontología original. Su módulo ocupa 20.68 MB.

TOP: Para completar DOID Top ha extraído de UBERON 271315 axiomas y 19460 entidades. Esto supone respectivamente un 95.34% de los axiomas y un 92.3% de los conceptos de la ontología original. Su módulo ocupa 52.36 MB.

Tabla 10 - Resultados Extracción UBERON para DOID

Estrategia	Axiomas extraídos	Porcentaje axiomas	Clases extraídas	Porcentaje clases	Tiempo en segundos	Tamaño Módulo MB
Star	105652	37.12%	6160	29.22%	42	20.63
Bot	106005	37.25%	6172	37.25%	18	20.68
Top	271315	95.34%	19460	92.3%	138	52.36

En esta última extracción de nuestro caso de estudio encontramos datos muy parecidos a los que hemos observado hasta ahora, por un lado, Star vuelve a ser la estrategia que completa la ontología con el menor número de axiomas, pero de nuevo con una diferencia mínima con respecto a Bot. En cuanto a Top, en este caso nos devuelve otro resultado que prácticamente supone una extracción de la totalidad de la ontología, con lo que una vez más, no nos estaría suponiendo ninguna diferencia utilizar el extractor modular.

Veamos ahora cuanto ha aumentado el tamaño de DOID con cada estrategia. Hay que tener en cuenta que TOP no ha importado NCBITaxiom.
DOID originalmente pesaba 28,51 MB.

Al enriquecerla con STAR el resultado ha ocupado 45,29 MB.

Al enriquecerla con BOT el resultado ha ocupado 48,73 MB.

Al enriquecerla con TOP el resultado ha ocupado 662,42 MB.

Podemos ver que la diferencia entre BOT y Star es pequeña, y que la variación de tamaño para estas dos estrategias es moderada. Sin embargo, TOP como hemos visto extrae las ontologías casi enteras, disparando el tamaño de

la ontología enriquecida hasta los 662 MB, y eso cuando además no ha podido cargar la ontología más pesada; de haberlo hecho podría haberse disparado muchísimo más en tamaño.

Una vez han finalizado las extracciones de todas las ontologías a importar el último paso restante a nuestra aplicación es comprobar mediante un razonador que las ontologías generadas son consistentes y no han introducido ninguna incongruencia por el camino.

Este es el resultado devuelto:

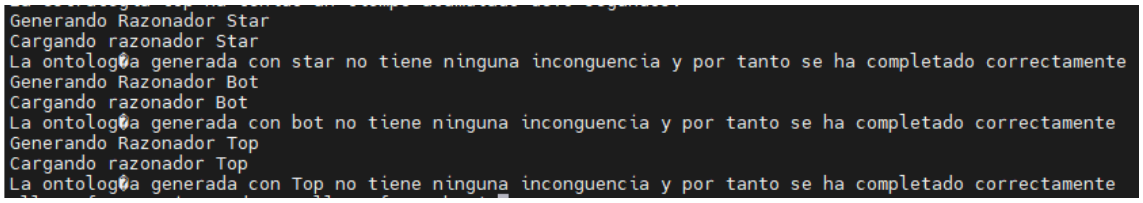

Ilustración 17 - Resultado en terminal del razonador

Ninguna de las 3 estrategias ha dado lugar a una ontología inconsistente.

Con todos estos datos, ya podemos sacar conclusiones en cuanto a que estrategia usar en función de nuestros objetivos con la ontología inicial.

5.3 Recopilación de los datos del caso de estudio

DOID

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades Star	Porcentaje entidades Star	Axiomas Star	Porcentaje Axiomas Star	Peso Star MB	Tiempo Star (s)
CHEBI - DOID	180297	2987597	675,36	89	0,05%	981	0,54%	17721	0,59%	3,28	27
Foodon - DOID	32516	314196	6,938	24	0,07%	282	0,87%	2749	0,87%	0,45	3
CL - DOID	16568	238898	57,43	63	0,38%	8344	50,36%	122097	51,11%	23,56	78
GENO - DOID	424	3545	0,62	9	2,12%	12	2,83%	173	4,88%	0,03	0
HP - DOID	31277	386306	93,109	102	0,33%	11344	36,27%	173661	44,95%	34,32	77
NCBITaxom - DOID	2401826	1,5E+07	1837,89	318	0,01%	973	0,04%	8203	0,06%	1,35	196
SO - DOID	2728	25584	5,148	16	0,59%	58	2,13%	588	2,30%	0,1	0
SYMP - DOID	968	6025	0,89	266	27,48%	315	32,54%	2130	35,35%	0,32	0
TRANS - DOID	35	232	0,057	13	37,14%	15	42,86%	83	35,78%	0,02	0
UBERON - DOID	21083	284591	65,916	391	1,85%	6160	29,22%	105652	37,12%	20,63	42
DOID Enriquecida	17898	179536	28,51	No Aplica	No aplica	30911	172,71%	423198	235,72%	45,303	423

Ilustración 18 - Resultado Star DOID completo

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades BOT	Porcentaje entidades BOT	Axiomas BOT	Porcentaje Axiomas BOT	Peso BOT MB	Tiempo BOT (s)
CHEBI - DOID	180297	2987597	675,36	89	0,05%	1030	0,57%	18275	0,61%	3,44	27
Foodon - DOID	32516	314196	6,938	24	0,07%	286	0,88%	2790	0,89%	0,46	2
CL - DOID	16568	238898	57,43	63	0,38%	8358	50,45%	122357	51,22%	23,6	31
GENO - DOID	424	3545	0,62	9	2,12%	25	5,90%	311	8,77%	0,05	0
HP - DOID	31277	386306	93,109	102	0,33%	11360	36,32%	173940	45,03%	34,36	31
NCBITaxom - DOID	2401826	1,5E+07	1837,89	318	0,01%	975	0,04%	8217	0,06%	1,5	172
SO - DOID	2728	25584	5,148	16	0,59%	61	2,24%	622	2,43%	0,11	0
SYMP - DOID	968	6025	0,89	266	27,48%	315	32,54%	2130	35,35%	0,32	0
TRANS - DOID	35	232	0,057	13	37,14%	19	54,29%	114	49,14%	0,02	0
UBERON - DOID	21083	284591	65,916	391	1,85%	6172	29,27%	106005	37,25%	20,68	18
DOID Enriquecida	17898	179536	28,51	No Aplica	No aplica	30944	172,89%	423881	236,10%	45,374	281

Ilustración 19 - Resultado BOT DOID completo

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades TOP	Porcentaje Entidades TOP	Axiomas TOP	Porcentaje axiomas TOP	Peso TOP MB	Tiempo TOP (s)
CHEBI - DOID	180297	2987597	675,36	89	0,05%	161670	89,67%	2912640	97,49%	541,82	6528
Foodon - DOID	32516	314196	6,938	24	0,07%	30476	93,73%	280634	89,32%	47,66	195
CL - DOID	16568	238898	57,43	63	0,38%	16465	99,38%	237117	99,25%	46,35	90
GENO - DOID	424	3545	0,62	9	2,12%	375	88,44%	2723	76,81%	0,52	0
HP - DOID	31277	386306	93,109	102	0,33%	30039	96,04%	376784	97,54%	72,7	185
NCBITaxom - DOID	2401826	1,5E+07	1837,89	318	0,01%	NO	NO	NO	NO	NO	NO
SO - DOID	2728	25584	5,148	16	0,59%	2337	85,67%	22626	88,44%	3,75	2
SYMP - DOID	968	6025	0,89	266	27,48%	869	89,77%	5470	90,79%	0,8	0
TRANS - DOID	35	232	0,057	13	37,14%	28	80,00%	163	70,26%	0,03	0
UBERON - DOID	21083	284591	65,916	391	1,85%	19460	92,30%	271315	95,34%	52,36	138
DOID Enriquecida	17898	179536	28,51	No Aplica	No aplica	252494	1410,74%	3949260	2199,70%	492,21	7138

Ilustración 20 - Resultado TOP DOID completo

OBI

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades Star	Porcentaje entidades Star	Axiomas Star	Porcentaje Axiomas Star	Peso Star MB	Tiempo Star (s)
APOLLO_SV - OBI	1679	17265	1,861	6	0,36%	75	4,47%	1535	8,89%	0,283	0
BFO - OBI	36	576	0,155	36	100,00%	35	97,22%	529	91,84%	0,12	0
CHEBI - OBI	180297	2987597	675,36	112	0,06%	953	0,53%	16064	0,54%	2,907	40
CHMO - OBI	3093	27798	6,107	4	0,13%	22	0,71%	201	0,72%	0,035	0
CL - OBI	16568	238898	57,43	40	0,24%	8317	50,20%	122639	51,34%	23,735	84
CLO - OBI	43325	351749	50,573	6	0,01%	78	0,18%	1411	0,40%	0,236	3
COB - OBI	72	398	0,055	1	1,39%	4	5,56%	32	8,04%	0,006	0
ENVO - OBI	6566	49216	9,893	3	0,05%	466	7,10%	4150	8,43%	0,708	1
GO - OBI	50896	564630	123,176	160	0,31%	828	1,63%	10485	1,86%	1,915	9
HP - OBI	31277	386306	93,109	1	0,00%	9043	28,91%	138860	35,95%	27,432	129
IAO - OBI	259	3565	0,548	192	74,13%	227	87,64%	2643	74,14%	0,447	0
IDO - OBI	519	4660	0,75	5	0,96%	61	11,75%	866	18,58%	0,164	0
NCBITaxom - OBI	2401826	14614235	1837,89	48	0,00%	230	0,01%	1844	0,01%	0,305	267
OGMS - OBI	187	1812	0,064	6	3,21%	23	12,30%	343	18,93%	0,064	0
OMIABIS - OBI	427	5092	0,967	2	0,47%	41	9,60%	872	17,12%	0,176	0
OMRSE - OBI	623	7367	1,208	1	0,16%	40	6,42%	1522	20,66%	0,276	0
OPL - OBI	446	5686	0,944	2	0,45%	107	23,99%	1452	25,54%	0,248	0
PATO - OBI	9841	126452	27,766	77	0,78%	100	1,02%	1929	1,53%	0,319	1
PR - OBI	334040	3926681	1211,942	36	0,01%	239	0,07%	2894	0,07%	0,563	43
SO - OBI	2728	25584	5,148	9	0,33%	35	1,28%	385	1,50%	0,062	0
UBERON - OBI	21083	284591	65,916	106	0,50%	4686	22,23%	79126	27,80%	15,293	30
UO - OBI	634	3582	0,601	11	1,74%	402	63,41%	2434	67,95%	0,305	0
VO - OBI	6906	59450	7,988	2	0,03%	48	0,70%	1667	2,80%	0,258	0
OBI Enriquecida	4629	49496	8,438	No Aplica	No aplica	30689	662,97%	443379	895,79%	84,295	607

Ilustración 21 - Resultado STAR OBI completo

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades BOT	Porcentaje entidades BOT	Axiomas BOT	Porcentaje Axiomas BOT	Peso BOT MB	Tiempo BOT (s)
APOLLO_SV - OBI	1679	17265	1,861	6	0,36%	75	4,47%	1615	9,35%	0,299	0
BFO - OBI	36	576	0,155	36	100,00%	35	97,22%	529	91,84%	0,12	0
CHEBI - OBI	180297	2987597	675,36	112	0,06%	1181	0,66%	19078	0,64%	3,456	41
CHMO - OBI	3093	27798	6,107	4	0,13%	37	1,20%	326	1,17%	0,054	0
CL - OBI	16568	238898	57,43	40	0,24%	8331	50,28%	122900	51,44%	23,779	37
CLO - OBI	43325	351749	50,573	6	0,01%	93	0,21%	1825	0,52%	0,311	3
COB - OBI	72	398	0,055	1	1,39%	6	8,33%	42	10,55%	0,007	0
ENVO - OBI	6566	49216	9,893	3	0,05%	473	7,20%	4295	8,73%	0,732	1
GO - OBI	50896	564630	123,176	160	0,31%	828	1,63%	10485	1,86%	1,915	8
HP - OBI	31277	386306	93,109	1	0,00%	9056	28,95%	139080	36,00%	27468	65
IAO - OBI	259	3565	0,548	192	74,13%	228	88,03%	2661	74,64%	0,449	0
IDO - OBI	519	4660	0,75	5	0,96%	62	11,95%	882	18,93%	0,164	0
NCBITaxom - OBI	2401826	14614235	1837,89	48	0,00%	232	0,01%	1859	0,01%	0,307	257
OGMS - OBI	187	1812	0,064	6	3,21%	23	12,30%	343	18,93%	0,068	0
OMIABIS - OBI	427	5092	0,967	2	0,47%	41	9,60%	890	17,48%	0,179	0
OMRSE - OBI	623	7367	1,208	1	0,16%	40	6,42%	1619	21,98%	0,292	0
OPL - OBI	446	5686	0,944	2	0,45%	108	24,22%	1483	26,08%	0,254	0
PATO - OBI	9841	126452	27,766	77	0,78%	100	1,02%	2027	1,60%	0,335	1
PR - OBI	334040	3926681	1211,942	36	0,01%	255	0,08%	3010	0,08%	0,581	44
SO - OBI	2728	25584	5,148	9	0,33%	35	1,28%	385	1,50%	0,062	0
UBERON - OBI	21083	284591	65,916	106	0,50%	4698	22,28%	79421	27,91%	15,336	15
UO - OBI	634	3582	0,601	11	1,74%	526	82,97%	3132	87,44%	0,386	0
VO - OBI	6906	59450	7,988	2	0,03%	62	0,90%	1970	3,31%	0,318	0
OBI Enriquecida	4629	49496	8,438	No Aplica	No aplica	31154	673,02%	449353	907,86%	27525,8	472

Ilustración 22 - Resultado BOT OBI completo

Ontología	Entidades Original	Axiomas Original	Peso original MB	Entidades a extrer	Porcentaje a extraer	Entidades TOP	Porcentaje Entidades TOP	Axiomas TOP	Porcentaje axiomas TOP	Peso TOP MB	Tiempo TOP (s)
APOLLO_SV - OBI	1679	17265	1,861	6	0,36%	1647	98,09%	14317	82,92%	2,417	1
BFO - OBI	36	576	0,155	36	100,00%	35	97,22%	529	91,84%	0,12	1
CHEBI - OBI	180297	2987597	675,36	112	0,06%	161628	89,65%	2911627	97,46%	541,68	14982
CHMO - OBI	3093	27798	6,107	4	0,13%	2929	94,70%	26987	97,08%	4,924	3
CL - OBI	16568	238898	57,43	40	0,24%	16465	99,38%	237117	99,25%	46,353	108
CLO - OBI	43325	351749	50,573	6	0,01%	42898	99,01%	341167	96,99%	46,85	994
COB - OBI	72	398	0,055	1	1,39%	64	88,89%	338	84,92%	0,045	0
ENVO - OBI	6566	49216	9,893	3	0,05%	5614	85,50%	43044	87,46%	7,357	13
GO - OBI	50896	564630	123,176	160	0,31%	43617	85,70%	507907	89,95%	98,258	1071
HP - OBI	31277	386306	93,109	1	0,00%	30039	96,04%	376784	97,54%	72,696	503
IAO - OBI	259	3565	0,548	192	74,13%	257	99,23%	3223	90,41%	0,56	0
IDO - OBI	519	4660	0,75	5	0,96%	344	66,28%	3568	76,57%	0,597	0
NCBITaxom - OBI	2401826	14614235	1837,89	48	0,00%	NO	NO	NO	NO	NO	NO
OGMS - OBI	187	1812	0,064	6	3,21%	180	96,26%	1511	83,39%	0,268	0
OMIABIS - OBI	427	5092	0,967	2	0,47%	424	99,30%	4895	96,13%	0,891	0
OMRSE - OBI	623	7367	1,208	1	0,16%	568	91,17%	6393	86,78%	1,1	0
OPL - OBI	446	5686	0,944	2	0,45%	445	99,78%	5290	93,04%	0,868	0
PATO - OBI	9841	126452	27,766	77	0,78%	8846	89,89%	120936	95,64%	22,811	38
PR - OBI	334040	3926681	1211,942	36	0,01%	NO	NO	NO	NO	NO	NO
SO - OBI	2728	25584	5,148	9	0,33%	2175	79,73%	20644	80,69%	3,63	2
UBERON - OBI	21083	284591	65,916	106	0,50%	19469	92,34%	271646	95,45%	23,4	202
UO - OBI	634	3582	0,601	11	1,74%	429	67,67%	2586	72,19%	0,71	0
VO - OBI	6906	59450	7,988	2	0,03%	6904	99,97%	57286	96,36%	19,92	23
OBI Enriquecida	4629	49496	8,438	No Aplica	No aplica	349606	7552,52%	4962424	10025,91%	903,9	17941

Ilustración 23 - Resultado TOP OBI completo

6 Conclusión, validación de resultados y limitaciones

6.1 Limitaciones y compatibilidad con extensiones o URIS

La mayor parte de las limitaciones de la aplicación vienen de la descarga de las ontologías a importar desde internet, pues no se puede garantizar que se puedan descargar todas las ontologías de las rutas extraídas de nuestra Ontología inicial.

Imaginemos el caso de una ontología pasada como parámetro A que tiene entidades de otra ontología B. Al generar el mapeo obtendremos la URI de B, que es la URL en la que se hospedaba en el momento en el que A la importó, sin embargo, podrían aparecer todos estos problemas.

- La ruta de B no está accesible porque su propietario la ha cerrado temporalmente.
- La ontología B se ha movido a otro servidor o a otra ruta.
- B ha sido integrada en otra ontología y ya no aparece disponible por separado.
- Su propietario ha decidido cambiarla de nombre y no sabemos el nuevo nombre del recurso que debemos descargar.
- Simplemente ha sido eliminada.
- El repositorio web está caído en el momento de la ejecución.

Y podrían ocurrir más problemas de este tipo. La resolución de estos problemas no es posible de implementarse en tiempo de ejecución pues la única solución es buscar B en otro directorio web y descargarlo ahí.

Además de esto tenemos otro problema, la URI nos da información sobre la dirección del recurso a descargar para obtener la ontología, pero no nos indica su formato y lo necesitamos para hacer la petición HTTP al servidor, y no es posible implementar todos formatos existentes.

El formato más extendido es owl, y la gran mayoría de las consultas funcionan asumiendo que es owl, pero no siempre es así. En los repositorios de ontologías biomédicas indicados en la introducción (BioPortal y OBO Foundry) en los que esta aplicación está pensada para funcionar es el formato mayoritario y es resoluble en casi todos los casos.

6.2 Limitaciones técnicas y asociadas a OWL API

Tenemos la limitación de memoria requerida. En los repositorios anteriores muchas de las ontologías más utilizadas son muy pesadas, llegando algunas a pesar más de 2 gigas. Para poder funcionar correctamente necesitamos mantener en memoria RAM en todo momento la ontología inicial con los axiomas añadidos y durante la fase de extracción hay que mantener la última ontología que se haya descargado (que en caso de ser una grande, puede ocupar fácilmente 13 GB o más), la cual no puede ser eliminada hasta que el método extractor haya terminado con ella.

En algunos casos, puede llegar a necesitarse una cantidad considerable de memoria RAM. Además de eso, una de las 3 estrategias de extracción, la estrategia Top, es extremadamente costosa computacionalmente según su documentación y para ontologías grandes puede dispararse debido al funcionamiento explicado en la introducción por el cual deben comprobarse todos los subárboles del conjunto semilla.

Por último, durante las pruebas se ha comprobado que el método de extracción en algunos casos da lugar a ejecuciones excesivamente costosas en tiempo, llegando a no haber terminado en días de ejecución. Para lidiar con este problema se ha implementado un limitador de tiempo parametrizable tras el cual la ejecución pasará a la siguiente ontología e informará de qué ontología no ha sido capaz de extraer. Por defecto se ha establecido en 2 horas por ontología y el usuario puede modificarlo añadiendo la opción -t x donde x es el numero de horas a las que se limita cada extracción.

6.3 Conclusiones generales

Después de haber analizado los resultados del apartado anterior podemos extraer las siguientes conclusiones:

- La diferencia entre usar Star y BOT es mínima; computacionalmente hablando BOT necesita menos tiempo, tiene menos complejidad y extrae ligeramente más axiomas y entidades que Star, pero la diferencia entre ambos es tan pequeña que el estudio indica que pueden utilizarse indistintamente. La principal diferencia es que como vimos, BOT extraerá más superclases de la jerarquía que Star las cuales no son necesarias.

- TOP extrae mucha más información que las otras dos opciones y sus tiempos son mucho mayores. Su coste computacional es tan alto que lo vuelve no fiable si no se utiliza un pc de gran capacidad o se usa en casos donde se sabe que las ontologías implicadas son muy pequeñas. Pero el principal argumento en contra de esta estrategia es que siempre extrae la ontología importada casi en su totalidad, con unos resultados visto donde el porcentaje extraído suele ser del 90%. Utilizar un algoritmo de extracción modular no tiene sentido si van a extraer la ontología casi entera, para eso es mejor importarla completa directamente desde Protégé como se ha mencionado en numerosas ocasiones en el apartado de comparaciones.

- Que estrategias de extracción como Star o BOT sean capaces de completar ontologías con porcentajes mínimos del contenido de las demás hacen del extractor modular una herramienta muy útil para el trabajo con ontologías pues permiten tanto a los creadores de las mismas como a los usuarios que las utilizan añadir la información faltante de las entidades que desean reutilizar desde otras sin llenar su ontología con información que no les es necesaria o que pueda ser considerada irrelevante.

- Las ontologías enriquecidas usando las estrategias Star y BOT no aumentan drásticamente de tamaño y se mantienen con pesos aceptables por lo que su uso para reutilización de ontologías es recomendable. TOP en cambio, debido a que importa las ontologías reutilizadas casi en su totalidad dispara los tamaños de las ontologías enriquecidas hasta puntos que pueden resultar excesivos. Hay que tener en cuenta que para reutilizar unos pocos axiomas de una ontología grande es capaz de extraer en torno al 90% de ella disparando los tamaños mientras que las otras dos extraen muchísimo menos.

6.4 Vías futuras

Conforme la Web Semántica vaya avanzando en el futuro, con mayor numero de ontologías, cada vez más especializadas y con más variedad, más útil será la reutilización de contenidos en el desarrollo las mismas. Es por esto que es importante trabajar ahora con mecanismos con los que aplicar esta práctica, como la extracción modular que se ha estudiado en este trabajo.

Sin embargo, con el tiempo puede que aparezcan nuevos mecanismos que sean más eficientes o interesantes, o quizás formatos nuevos para trabajo con ontologías distinto al .owl mayoritario hoy en día y que nos obliguen a estudiar nuevas formas de trabajar y aplicar estos conceptos.

En lo que refiere a este estudio, puede que las estrategias mejoren y sean capaces de garantizar la extracción de módulos mínimos, o de trabajar mejor con ontologías grandes y puedan garantizar el correcto funcionamiento de todas ellas para todas las ontologías, cambiando drásticamente los resultados de este trabajo.

En lo que refiere a la aplicación desarrollada podrían realizarse mejoras futuras, como por ejemplo adaptarla a más formatos de ontología que cojan peso o importancia en el futuro o que con las mejoras de la Web Semántica sea capaz de implementarse un mecanismo para encontrar siempre las ontologías referenciadas a través de URI, cuya posibilidad de fallo es una de las limitaciones estudiadas previamente.

En cualquier caso, el concepto de reutilización es el futuro en el desarrollo de las ontologías y por tanto, el estudio de los métodos para lograr la accesibilidad y la importación de la información de la Web Semántica.

7 Bibliografía

[1] Thomas R. Gruber at Stanford University , «https://tomgruber.org,» 1993. [En línea]. Available: https://tomgruber.org/writing/ontolingua-kaj-1993.pdf. [Último acceso: 2022].

[2] J. H. a. O. L. TIM BERNERS-LEE, «THE SEMANTIC WEB,» *Scientific American,* vol. 284, nº 5, pp. 35-43, 2001.

[3] Natalya F. Noy and Deborah L. McGuinness Stanford University, «https://corais.org/,» 26 04 2012. [En línea]. Available: https://corais.org/sites/default/files/ontology_development_101_aguide_to_creating_your_first_ontology.pdf. [Último acceso: 2022].

[4] J. T. F.-B. Manuel Quesada-Martínez, «Studying the reuse of content in biomedical,» Universidad de Murcia, Murcia, 2020.

[5] M. |. B. S. Horridge, «The OWL API: A Java API for OWL ontologies,» *Semantic Web,* vol. 2, nº 1, pp. 11-21, 2011.

[6] A. A. a. B. König-Ries, «http://ceur-ws.org,» 12 10 2019. [En línea]. Available: http://ceur-ws.org/Vol-2849/paper-10.pdf. [Último acceso: 2022].

[7] Boris Motik, Rob Shearer, Birte Glimm, Giorgos Stoilos, and Ian Horrocks at the Department of Computer Science in the University of Oxford, «http://www.hermit-reasoner.com,» [En línea]. Available: http://www.hermit-reasoner.com. [Último acceso: 2022].

[8] L. Schriml, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/DOID. [Último acceso: 2022].

[9] C. A. S. N. Y.-W. W. C. M. M. V. F. G. F. W. A. K. Lynn Marie Schriml, «Disease Ontology: a backbone for disease semantic integration,» *Nucleic Acids Research,* vol. 40, nº D1, p. D940–D946, 2012.

[10] B. Peters, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/OBI. [Último acceso: 2022].

[11] A. Diehl, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/CL. [Último acceso: 2022].

[12] G. Owen, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/CHEBI. [Último acceso: 2022].

[13] D. Dooley, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/FOODON. [Último acceso: 2022].

[14] M. Brush, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/GENO. [Último acceso: 2022].

[15] P. R. Sebastian Köhler, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/HP. [Último acceso: 2022].

[16] NCBI information, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/NCBITAXON. [Último acceso: 2022].

[17] SO Administrators, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/SO. [Último acceso: 2022].

[18] L. Schriml, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/SYMP. [Último acceso: 2022].

[19] P. Shields, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/TRANS. [Último acceso: 2022].

[20] C. Mungall, «https://bioportal.bioontology.org,» [En línea]. Available: https://bioportal.bioontology.org/ontologies/UBERON. [Último acceso: 2022].

[2 R. G. B. P. U. S. Chiara Del Vescovo, «http://owl.cs.manchester.ac.uk,» The

1] University of Manchester, [En línea]. Available: http://owl.cs.manchester.ac.uk/research/modularity/. [Último acceso: 2022].