uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería de Tecnologías de Telecomunicaciones

2021-2022

*Trabajo Fin de Grado*

# "Estructuración Automática de Información Biomédica a partir de Grafos de Conocimiento"

---

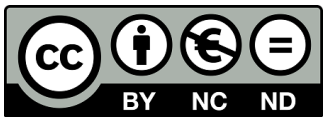
Claudia Llorente Pascual

Tutores

Pedro Manuel Moreno Marcos

Carlos Óscar Sánchez Sorzano

Leganés, 14 de julio de 2022



# RESUMEN

Con el avance de la globalización y especialmente desde la pandemia de la COVID-19, el aumento de la producción de la información biomédica y sobre todo el interés por el público general en acceder a la misma ha aumentado considerablemente.

Para las herramientas de gestión de información, ordenarla, categorizarla y presentarla de manera organizada ha supuesto un reto y ha popularizado herramientas de generación automática de resúmenes como el clústering o los grafos de conocimiento.

Con el objetivo de accesibilizar información biomédica desde el caso concreto de la Asociación COpenMed, a lo largo de este Trabajo de Fin de Grado se desarrolla un sistema de estructuración automática de información a partir de la creación y exploración de un grafo de conocimiento.

Además, se analizan las relaciones no explícitas entre los nodos del grafo para categorizar la información en grupos que faciliten su organización. De esta forma, se combina un análisis teórico de las similitudes entre las entidades tratadas, como causas de enfermedades o condiciones, con el procedimiento automático de estructuración.

Por último, se realiza la exploración completa de todas las entidades con las que se trabaja para generar un documento final. Este documento es generado programáticamente utilizando LaTeX y contiene la información estructurada de todas las entidades. Esta información aparece vinculada entre sí con referencias cruzadas por medio de identificadores que mantienen las relaciones exploradas en el grafo, de tal manera que es posible acceder a la información de una entidad desde cualquiera de sus relaciones.

Este Trabajo de Fin de Grado busca aplicar los elementos de la teoría de grafos al problema de la información biomédica, mejorando su estructuración y consecuentemente su accesibilidad.

**Palabras clave:** Grafos de conocimiento, información biomédica, estructuración automática.

# ABSTRACT

Due to the progression of globalization and specially since the COVID-19 pandemic, we have seen a rise in the production of documents and studies in the field of biomedicine. This has been followed by an increment in the general public's interest in the field, that demands a change in the way information is presented.

As a consequence of the substantial growth in documentation, developing tools to organize, categorize and present the information has been a challenge in the last years. In this field, solutions based on knowledge graphs as well as text summarization tools using clustering methods, have been popularised.

To make biomedical information accessible from the standpoint of the particular problem faced by Asociación COpenMed, this Final Degree Project generates a system able to automatically structure biomedical information from the creation and exploration of a knowledge graph.

It also considers the non-explicit relationships between the nodes of the graph to categorize them into groups, to ease the organization. In this way, the theoretical analysis applied to the similarities between entity types, like the causes of diseases being similar to those of conditions, is combined with the automatic structuring process of the information.

Finally, the information extracted from the knowledge graph is organized into a document, that contains the characteristics of each entity categorized into chapters and sections. The document also contains labels and references to each entity whenever it appears, maintaining the relations considered and explored in the knowledge graph.

This Final Degree Project applies concepts from graph theory to the problem of biomedical information, improving it's structure and consequently making it more accessible.

**Keywords:** Knowledge graphs, automatic information processing, biomedical information.

# DEDICATORIA

Quisiera empezar esta dedicatoria agradeciéndoles su trabajo a mis tutores Pedro y Carlos Óscar, por resolver todas mis dudas y ayudarme en el camino a desarrollar este trabajo.

Me gustaría también agradecerle a mi familia, especialmente a mis padres y a mi hermano, su apoyo incondicional así como su preocupación compartida.

Por otro lado, quisiera agradecer a mis compañeros Juan y Víctor su compañía estos cuatro años y las infinitas dudas y quejas resultas especialmente en el último año. Y a Miriam, por haber pasado los tres últimos años cuidándonos y por haberme ofrecido su ayuda siempre que lo he necesitado.

Finalmente quiero agradecer a las Asocis, especialmente a las chicas de la Asociación Lovelace por ser un refugio durante estos cuatro años de carrera.

# ÍNDICE GENERAL

# ÍNDICE DE FIGURAS

# ÍNDICE DE TABLAS

# 1. INTRODUCCIÓN

En este primer capítulo, se incluye una introducción al objeto de estudio donde se plantea el problema y se expone la motivación para llevar a cabo el trabajo. Además, se exponen los objetivos y por último se detalla la estructura que seguirá este documento, así como el contenido de cada uno de los capítulos.

## 1.1. Planteamiento del problema y motivación

El acceso a la información biomédica se ha entendido en la sociedad como un área de conocimiento accesible únicamente a doctores o personas con alto grado de formación en ese ámbito. Además, el gran volumen de información que se produce en esta era, genera un reto para el procesamiento de texto y la organización y filtrado de la misma.

Una de las formas de estructuración de conocimiento que se utiliza en ámbitos de gran interrelación son los grafos de conocimiento. A partir de una base de datos relacional, se puede construir un grafo a través del cual explorar la pertinencia de las relaciones entre distintos bloques de información.

Por otro lado, aunque la representación gráfica puede servir de ayuda para formar una idea del contenido que se está tratando, la visualización de un grafo de conocimiento como forma de acceder a él, no es cómoda para un público no especializado. Por esto, además del problema de explorar las relaciones, se plantea la manera de extraer el contenido relevante y organizarlo de manera automática.

Poniendo el foco en la información biomédica, se trata de un campo muy amplio con grandes cantidades de información. Por este motivo, surge la necesidad de establecer procesos automáticos capaces de extraer información de calidad, relevante y pertinente. Bases de datos como MEDLINE recogen más de 20 millones de referencias bibliográficas así como acceso a más de 700 revistas activas [1].

Este Trabajo de Fin de Grado se basa en una de las soluciones propuestas para organizar información, los grafos de conocimiento. Generalmente aplicado a la organización de documentos para la generación automática de resúmenes así como a las páginas web, se decide aplicar a toda una base de datos biomédicos contenida en un archivo Excel. Además de la motivación de aplicar el conocimiento del área de los grafos de conocimiento a otras tareas, se pretende accesibilizar información biomédica sin perder rigurosidad.

El proyecto nace de la colaboración con la asociación sin ánimo de lucro COpenMed, que surge en 2019 a partir de la idea del Dr Sorzano de crear un razonador automático que ayude en la identificación de posibles enfermedades compatibles con una serie de síntomas. Aunque no pretende generar una herramienta de diagnóstico, busca generar un documento de consulta como primer paso para una actualización de su página web.

A partir del aumento de complejidad de los datos manejados por la asociación, así como la inclusión de más tipos de entidades, surge la necesidad de crear una nueva línea de trabajo al margen del razonador. Este proyecto pretende automatizar la organización de la información manejada por la asociación para facilitar su futuro uso.

De esta forma, en este Trabajo de Fin de Grado se persigue proponer una solución a uno de los problemas del área de la biomedicina, la estructuración de información, aplicada a la información manejada por la Asociación COpenMed.

Este proyecto consiste en realizar el análisis de ambos problemas y presentar una solución. El contenido de partida es un archivo tipo Excel descargado de una base de datos relacional. Desde aquí, se generará el grafo de conocimiento, del que se exploran las relaciones y se extraerá la información pertinente. El resultado final será un documento con la información organizada de todas las entidades del archivo y la relación entre sí.

## 1.2. Objetivos

En el presente Trabajo de Fin de Grado se han definido una serie de objetivos a cumplir con la finalización del mismo, el objetivo principal es implementar un código de estructuración automática de información, capaz de recorrer un grafo de conocimiento formado con información biomédica, para producir un documento final con fichas informativas de cada entidad contenida en el grafo. Además, se definen los siguientes objetivos específicos:

1. Generar grafos de conocimiento a partir de archivos Excel con información no estructurada.
2. Analizar las distintas opciones para estructurar la información y considerar su utilidad para recorrer grafos de conocimiento.
3. Implementar un proceso de estructuración automática de información.
4. Analizar las claves de la información biomédica manejada para establecer relaciones entre entidades.
5. Producir un documento final con toda la información contenida en el grafo de manera organizada y debidamente vinculada entre sí.

## 1.3. Estructura del documento

En este apartado se indican los diferentes capítulos que se incluyen en este documento así como los contenidos de cada uno:

1. Introducción: incluye el planteamiento del problema, la motivación del trabajo y los objetivos del mismo.

2. Estado del arte: este apartado incluye cuatro subsecciones referidas al conocimiento utilizado en el trabajo como información basada en grafos, optimización de código, sistemas de estructuración automática y aplicaciones similares existentes.

3. Metodología: este apartado recoge los métodos utilizados para el desarrollo de este trabajo así como las justificaciones para su elección, incluyendo el lenguaje de programación y el uso de grafos de conocimiento.

4. Desarrollo de las funciones: este apartado detalla la implementación de cada función de propagación del grafo así como las funciones de separación en categorías.

5. Agrupaciones por tipo de entidad: en el capítulo se detallan los criterios escogidos para realizar agrupaciones de entidades para su estudio y estructuración así como las funciones aplicables a cada grupo.

6. Verificación de los resultados: se definen las maneras de verificar los resultados del procesado respecto a la base de datos.

7. Visualización: se discuten las posibles salidas y se define la implementación del tipo de visualización elegida.

8. Conclusiones: se incluyen las conclusiones obtenidas del trabajo así como posibles líneas de trabajo futuro.

9. Anexo 1 - Gestión del proyecto: se incluye la planificación temporal y el presupuesto asociados a la realización del trabajo.

10. Anexo 2: Marco regulador: este anexo recoge y analiza las regulaciones aplicables a este trabajo.

11. Anexo 3 - Marco socio-económico: en el apartado se recoge un análisis del impacto socioeconómico del trabajo realizado.

# 2. ESTADO DEL ARTE

En este capítulo se exponen las bases teóricas de los grafos de conocimiento, así como los métodos actuales de optimización y las aplicaciones biomédicas actuales.

## 2.1. Grafos de Conocimiento

Los grafos de conocimiento son sistemas de estructuración de información. En esta sección se expone la Teoría de Grafos, las definiciones actuales existentes y algunas aplicaciones de los mismos.

### 2.1.1. Teoría de Grafos

La Teoría de Grafos, también llamada teoría de las gráficas, es una rama de la Matemática Discreta que estudia problemas que relacionan un número de puntos con ciertos trazos que los unen [2]. Se basa en el problema de los puentes de Könishberg, este problema surge en la ciudad de Könishberg, que tenía una isla en medio del río que atravesaba la ciudad [3]. Cada una de las cuatro regiones en las que se divide la ciudad se conectan entre sí mediante siete puentes, se pueden ver en la figura 2.1:

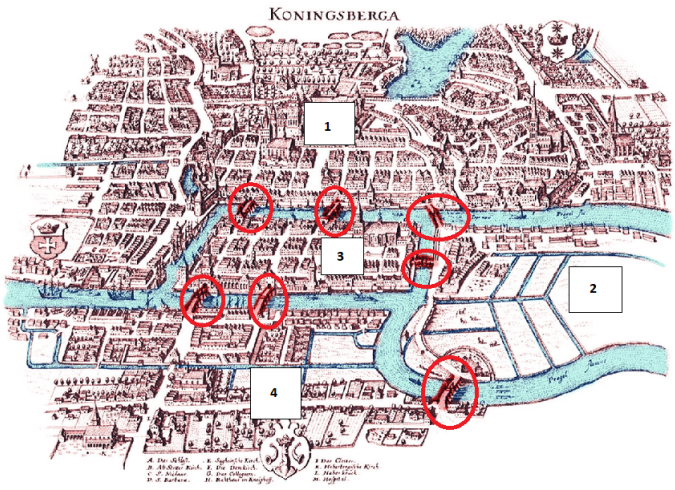



Fig. 2.1. Los puentes de Koenigsberg. Imagen adaptada [4]

La preguntá de Euler era: ¿Es posible encontrar una ruta en la ciudad que recorra los siete puentes, cruzando cada uno de ellos una sola vez y regresando al punto de partida? [5]

En su artículo publicado en 1736, el matemático demostró que no era posible, sentando la base del concepto de grafo euleriano. Este término se refiere a *"aquel grafo que*

*puede ser dibujado sin levantar el lápiz del papel sin pasar dos veces por la misma línea y acabando en el punto de partida"* [2]. A partir la definición del grafo euleriano y del inicio de la teoría de grafos, otros matemáticos como Vandermonde o Kirchoff también presentaron sus contribuciones al campo y la teoría de grafos se utilizó para resolver juegos matemáticos, estudiar circuitos eléctricos y el desarrollo de aplicaciones en áreas como la economía o la física nuclear.

### 2.1.2. Definiciones

Los grafos de conocimiento han sido objeto de estudio desde 2012 y desde ese momento se han generado un gran número de definiciones y aplicaciones para los mismos. Partiendo de las numerosas definiciones, Paulheim [6] define en 2016 unas bases comunes para diferenciar los grafos de conocimiento de otras organizaciones de conocimiento, que los restringe a representaciones gráficas de conocimiento.

Desde la matemática, distinguimos la definición geométrica y la algebraica. Geométricamente, un grafo simple es un conjunto de datos en el espacio, algunos de los cuales están unidos entre sí mediante líneas. Se considera que un grafo contiene únicamente información sobre las conectividades entre puntos y carece de información geométrica. Para formalizar esta definición de manera algebraica, siendo V el conjunto de los vértices y E el conjunto de los lados, el grafo G se define como un par ordenado

$$G = (V, E) = (V(G), E(G)) \tag{2.1}$$

En relación con la definición geométrica, V aporta la información de puntos en el espacio topológico mientras que E es un conjunto de pares no ordenados de los elementos de V [2]. A nivel visual, si consideramos el siguiente grafo G de la figura 2.2:

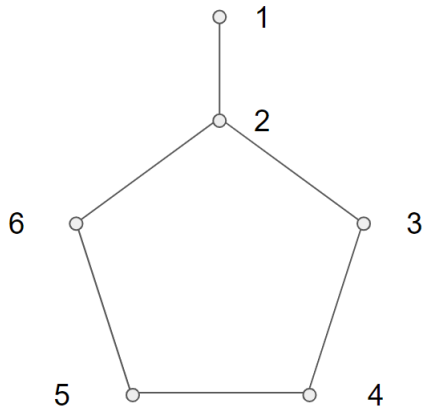



Fig. 2.2. Grafo simple G

El grafo se compone de los nodos, en este caso representados como puntos, así como de las relaciones entre los mismos. De tal forma que V, definido como el conjunto de los vértices, sería en este grafo:

$$V = V(G) = [v_1, v_2, v_3, v_4, v_5, v_6] = [1, 2, 3, 4, 5, 6]$$

Mientras que E, definido como el conjunto de los lados, representa la relaciones entre los nodos, siendo:

$$E = E(G) = [(v_1, v_2), (v_2, v_3), (v_3, v_4), (v_4, v_5), (v_5, v_6), (v_6, v_2)]$$

$$E = [(1, 2), (2, 3), (3, 4), (4, 5), (5, 6), (6, 2)]$$

De tal forma que el conjunto de V y E constituye el grafo.

Sin embargo, cabe resaltar la importancia de la existencia de otros tipos de grafos como el grafo general o el grafo dirigido. En este último, a las aristas se les asigna un sentido y la diferencia formal entre ambos consiste en que los pares de vértices que definen los lados de un grafo general son no ordenados, mientras que en uno dirigido sí son pares ordenados [2].

En el ámbito de la ingeniería, los grafos de conocimiento son sistemas de organización de la información, comúnmente asociados con las bases de datos. Según la definición de Lisa Ehrlinger, el grafo es un sistema que *"adquiere e integra conocimiento de una ontología y aplica un razonador para derivar un nuevo conocimiento"*[7]. Estos sistemas describen entidades del mundo real, en este caso del campo de la biomedicina, y sus interrelaciones de manera organizada. Además, las clases y relaciones se pueden organizar en un esquema de tipos y funcionalidades.

Aunque algunas de las definiciones se contradicen entre sí, la mayoría resaltan la capacidad de los grafos para organizar grandes (aunque esta capacidad no se concreta) cantidades de información así como una función para extraer información de los mismos.

### 2.1.3. Aplicaciones actuales

En cuanto a los grafos en sus aplicaciones tecnológicas, el término fue introducido en los años 80 por la Universidad de Groningen para describir su modelo de descripción de lenguaje natural. En 2012, Google utilizó el concepto para definir una de sus funciones de búsqueda semántica que permitía buscar objetos del mundo real, aunque no se discute su implementación [8]. A partir de ese momento, el término se menciona en la implementación de DBPedia, Freebase, Wikidata y otras aplicaciones. En todas ellas la implementación y propósito es diferente, con el único denominador común siendo el uso de datos enlazados.

En el caso de BDpedia, lo que comenzó en 2007 como una base de conocimiento, extrae desde junio de 2011 información de Wikipedia con un proceso de generación de información [9]. En 2012, se presenta otro de los proyectos mencionados, Wikidata, que también extrae información de Wikipedia [10], aunque toman acercamientos distintos. En el caso de BDpedia, se pretende formalizar la información de las llamadas infoboxes,

las plantillas wiki en las que se define una estructura de datos común [11]. Su ontología, sistema de datos que define las relaciones existentes entre los conceptos de un dominio o área del conocimiento [12], se organiza en una taxonomía de clases que se ve en la figura 2.3:

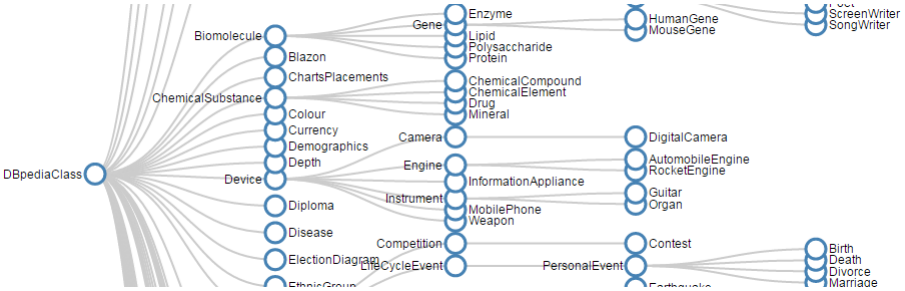



Fig. 2.3. Taxonomía de clases DBpedia; Fuente: DBpedia [13]

Para estructurar su información, extrae grafos de tipo Resource Description Framework (RDF) de Wikipedia. RDF es un formato de datos dirigidos y etiquetados para representar información en la Web [14]. Integra en los datos de DBpedia, aunque limita las posibilidades de conexión con otros datasets [11].

Por otro lado, en el caso de Wikidata se plantea producir los datos en vez de extraerlos. Wikidata se constituye como datos estructurados en los que cada entidad se representa mediante una IRI [11]. Una IRI es una secuencia de referencia, utilizado como identificador [15]. La herramienta desarrolla su propio modelo de datos, que permite generar diferentes serializaciones RDF [11]. Aunque mediante procesos distintos, ambos proyectos sirven para estructurar información a partir de Wikipedia y dependen de su capacidad de análisis de esta para producir buenos resultados.

La otra aplicación que se ha mencionado es Freebase. Esta herramienta fue lanzada en 2007 como base de conocimiento colaborativa, y adquirida en 2010 por Google para servir como base de su grafo de conocimiento [16]. Los datos de esta base, migraron progresivamente a Wikidata y Freebase cerró en 2016. Hasta ese momento, fue sido utilizada como ejemplo de base de conocimiento para estructurar varios trabajos [17].

Los conceptos manejados para la generación de estos grafos derivan de la Web Semántica, que propone introducir descripciones explícitas de los conceptos para que las propias máquinas puedan hacerse cargo de su aprendizaje. Utiliza los principios de los grafos, ya que cada recurso representa un nodo con un tipo concreto y los enlaces representan relaciones diferenciadas. Por ejemplo, un nodo puede tener como tipo profesor o tienda, y su relación será del tipo profesor-departamento. Un ejemplo de su uso es el establecimiento de ontologías sobre cuadros, pintores y pintura en la que un museo virtual puede organizar sus contenidos y publicarlos en la red semántica [18].

Como ejemplo práctico de su uso, encontramos la exploración de la web del Museo del Prado. Desde 2012, el museo ha ido cambiando su forma de estructurar información

pasando del modelo convencional que encontramos por ejemplo en Ceres [1], la Red Digital de Colecciones de Museos de España, a un modelo semántico basado en grafos. Define su grafo de la siguiente manera:

*"El Grafo de Conocimiento del Museo del Prado es un sistema de representación del conjunto de sus contenidos y recursos digitales que entiende hechos relacionados con los autores, las obras de arte, sus contenidos, temas, épocas y estilos, así como cualquier objeto potencialmente enlazado con ellos y, especialmente, entiende el modo en que este conjunto de entidades está todo él conectado"* [19]

El éxito de este proyecto, radica en el cambio de sistema de búsqueda de información, que permite el acceso a una colección organizada, así como a contenidos multimedia y una bibliografía básica asociada a cada obra, lo que les concedió dos premios Webby en 2016 [20].

Actualmente, una de las aplicaciones principales de los grafos de conocimiento son la investigación y seguimiento del SARS - CoV2. Uno de estos ejemplos, es la creación de un grafo de conocimiento argumentativo a partir de la base de datos abierta para el COVID-19 (CORD-19). En un ensayo de la Revista Cubana de Información en Ciencias de la Salud [21], se estudia su utilidad para el seguimiento de brotes. Sin embargo, actualmente se utilizan para generar información para investigadores, principalmente con fines de diagnóstico [21].

## 2.2. Optimización

En informática, el proceso de optimización se refiere al conjunto de métodos utilizados para determinar los mejores valores de las variables y procesos de un sistema para conseguir el mejor resultado posible. En términos de información, el mejor resultado se refiere al más rápido y pertinente. A lo largo de esta sección se exponen las metodologías para la optimización de estructuración de información, recorrido de grafos y procesos automáticos en este ámbito.

### 2.2.1. Search Engine Optimization

En el ámbito de información, estas técnicas de análisis y posicionamiento se pueden aplicar a motores de búsqueda y nacen métodos como SEO (Search Engine Optimization). Esta técnica nació en los 90 bajo la premisa de que las páginas de empresas tuvieran mayor visibilidad y por lo tanto más tráfico. Al tratar de mejorar de forma natural la visibilidad de una página y su posición en los buscadores, se trata de organizar la información y utilizar palabras clave para que su identificación sea mejor [22].

[1] http://ceres.mcu.es/

El mecanismo interno de SEO se basa en el contenido de la información. Utiliza los datos estadísticos sobre la repetición de conceptos y palabras, por lo que es relevante hacer buen uso de las palabras clave. Además, un concepto relevante es la usabilidad, que determina hasta qué punto es accesible la interfaz y el contenido de una web, lo que repercute en la propia arquitectura de la red, y la manera en que estructura su conocimiento [23].

Las prácticas de SEO se basan en primer lugar en comprender cómo se visualiza la información, lo cual depende del buscador a través del cuál se accede a la información [24]. Por otro lado, la parte más relevante a este trabajo es que las estrategias SEO deben tomar en cuenta el punto de vista de los motores de búsqueda y usuarios [24]. Aplicar estos principios al trabajo es relevante en cuanto a la visualización de los datos, de cara a generar un documento efectivo, es necesario asumir el punto de vista de usuarios con diferentes niveles de conocimiento en el área de la biomedicina.

En el área de la medicina, muchos de estos resultados de buscadores están enlazados a bases de datos. Este es el ejemplo de Pubmed, que extrae información de Medline [25]. Pubmed es un portal que recoge literatura biomédica, que sirve como motor de búsqueda para acceder a este tipo de literatura y aunque no incluye artículos completos sí los enlaza [25]. En este ámbito, también son relevantes las palabras clave, y en buscadores como Pubmed, se organizan en un Tesauro. Este concepto se refiere a una lista de palabras o términos utilizados para representar ciertos conceptos. Permite convertir el lenguaje natural de los documentos en un lenguaje controlado y común a la base de datos o buscador especializado.

Es importante destacar, que PubMed permite hacer búsquedas complejas a través de campos o términos MeSh [26]. MeSh (Medical Subjec Headings) representa el vocabulario controlado de PubMed y representa los términos y conocmiento de la NLM (National Library of Medicine) [27]. MeSH se utiliza para realizar consultas al buscador, y en ese ámbito, se encuentra su utilización para el mapeo automático de términos (ATM - Automatic Term Mapping). En evaluaciones realizadas [28], se concluye que MeSH mejora el número de documentos relevantes devueltos ante una consulta, aunque puede no ser útil si se buscan solo los más relevantes, ya que devolverá más. Se ve que la forma en la que se organiza el conocimiento antes de estructurarlo influye en el resultado de diferentes formas: en relación al número de resultados obtenidos y en la relevancia de los mismos.

Por lo tanto, la optimización que se realiza en los buscadores, especialmente aquellos especializados en el área de la biomedicina, depende de la estructuración de sus conceptos. En primer lugar eligiendo correctamente las palabras clave que definen la información y en segundo lugar de acuerdo a su accesibilidad.

### 2.2.2. Optimización de grafos

Recordando el problema inicial de la teoría de grafos, los puentes de Könishberg, vemos que sería posible utilizar la fuerza bruta para recorrer todos los caminos posibles. Actualmente existen procedimientos de búsqueda por fuerza bruta, que recorren el grafo hasta encontrar un nodo que cumpla las condiciones, algunos ejemplos son:

- Procedimiento del Museo Británico: Este método, definido en los 90, recorre el grafo en busca de todas las soluciones posibles a la búsqueda realizada y elige la que muestra mejor función de evaluación. Encuentra la solución óptima así como otras soluciones factibles [29].
- Búsqueda en profundidad: este algoritmo, también conocido como estrategia de búsqueda LIFO, explora siempre un vértice hijo del vértice de partida. Si dicho vértice no tiene hijos, se realiza un proceso de backtracking para volver al nodo anterior y se reanuda la búsqueda hasta encontrar un nodo que sirva como solución. En este caso no se encuentra la solución óptima necesariamente [29].
- Búsqueda best first search: esta estrategia termina cuando se encuentra la primera solución factible. La exploración del grafo con esta estrategia se basa en encontrar el vértice más prometedor del estado de partida. Se selecciona el valor mínimo como el más prometedor en base a la función de evaluación aplicada. No es relevante la distancia ni posición del vértice, únicamente la función de evaluación, por lo que debe adecuarse a su utilización concreta. [29].

Actualmente, la perspectiva de explorar grafos completos para encontrar soluciones con el volumen de conocimiento que se maneja en el ámbito de la biomedicina, no representa una opción realista para muchos proyectos. Por esto, se han desarrollado otros algoritmos.

Encontramos por un lado procedimientos de búsqueda basados en programación dinámica. En estos algoritmos, una decisión provoca un cambio de estado, que a su vez lleva a una secuencia de decisiones que provocan la trayectoria a seguir. Cada una de estas trayectorias lleva de un estado inicial previamente definido a uno final, también definido por ser una solución factible. La optimización de estos algoritmos, se basa en optimizar las políticas de decisión que provocan los cambios de estado. En cada una de las etapas de estos algoritmos interviene una variable de optimización, por lo que el entrenamiento de este algoritmo es más complejo en cuanto a que posee más variables a optimizar [29].

La optimización de grafo se aplicado a diferentes ámbitos aplicando algoritmos adecuados a cada propósito. Uno de los algoritmos más conocidos actualmente para la optimización de grafos es el algoritmo de Dijkstra. Este algoritmo sirve para encontrar la distancia más corta de un nodo de origen a otro cualquiera, por lo que se aplica en grafos donde el factor relevante es la distancia entre los nodos, por ejemplo para calcular rutas de transporte [30]. En telecomunicaciones se aplica también a algoritmos de routing en redes y de manera más aplicable a este trabajo, en la búsqueda de ontologías [31].

La exploración del grafo en este trabajo se realiza sin conocer el nodo de llegada, solo el de partida, por lo que no se aplica la lógica de Dijkstra. En este caso, se conoce la distancia máxima que se quiere recorrer para cada respuesta y el umbral para su función de evaluación. De manera similar al procedimiento del Museo Británico, busca todas las soluciones factibles y las ordena en cuanto a su relevancia. Dado que el objetivo no es devolver la mejor solución a cada característica sino todas las posibles, no se aplican técnicas de optimización que devuelven una sola respuesta.

## 2.3. Aplicaciones del sector

El aumento de la investigación y la información biomédica, ha hecho que a lo largo de la última década hayan aumentado las bases de datos y webs léxicas que tratan de aunar este conocimiento. A continuación se citan trabajos y aplicaciones en el área de la generación automática de resúmenes así como en el tratamiento de bases de datos de información biomédica.

### 2.3.1. Generación automática de resúmenes

En el área de la medicina y biología, contar con resúmenes de los documentos a investigar puede ayudar a acelerar el proceso, no necesitando una lectura exhaustiva para conocer el contenido de los documentos.Actualmente, bases de datos como MEDLINE añaden semanalmente alrededor de 10 000 artículos científicos. En este aspecto, las herramientas del procesamiento del lenguaje natural han avanzado para generar dichos resúmenes a partir de los conceptos del documento [32].

Una de las métricas de resúmen que se utiliza más a menudo en este área es la frecuencia de aparición de los conceptos, el *Inverse Document Frequency* (IDF). Este proceso consiste en extraer la frecuencia inversa del documento, después de su tokenización (normalmente el primer paso del preprocesado de texto, consiste en dividir el texto en partes más pequeñas como palabra o signos de puntuación) y del procesamiento adecuado al texto. Este tipo de representación del documento, permite su aplicación a algoritmos de clustering, ya que conserva la centralidad (la relevancia de los conceptos) del documento, el concepto principal discutido, a partir de la combinación lineas de sus subelementos [33]. A partir de esta representación, se pueden aplicar funciones de similitud como la distancia coseno o funciones de prioridad para comparar los resultados de estas funciones con la representación del documento, lo que permite extraer los elementos centrales del documento para elaborar un resumen. También aporta un ranking descendente sobre los conceptos relevantes del documento sobre el que elaborar un resumen más detallado si se requiere [33].

Por otro lado, se llevan también a cabo resúmenes a partir de métodos de clustering. Para ello, oraciones que tratan conceptos similares se organizan en un cluster del cual se

extrae la oración central que se considera más representativa y se establece como resumen del concepto [34]. Este tipo de algoritmos utilizan la representación *Term Frequency - Inverse Document Frequency* (TF-IDF) que extrae la frecuencia de un término concreto respecto al documento en el que se encuentra [35]. De nuevo, esta herramienta sirve para encontrar los conceptos más relevantes en un documento que, aplicados a algoritmos de clustering, sirven como centroides [35].

Relacionando este procedimiento con los grafos de conocimiento que se tratan en este trabajo encontramos LexRank. Se trata de un método no supervisado para la elaboración de resúmenes a partir de bases de datos basadas en grafos [36]. La idea detrás de este proyecto es que las frases recomienden frases similares al lector de tal manera que los conceptos más repetidos a lo largo de un documento tomen más importancia. En este algoritmo se establece un ranking con dichos conceptos y los más relevantes se colocan en el resumen. El algoritmo no necesita conocer realmente el contenido del documento, por lo que se puede aplicar a cualquier texto [36].

También como algoritmo basado en grafos, existe PageRank. Este algoritmo desarrollado en los 90, es una herramienta de búsqueda de información derivada de los procesos de *Information Retrieval* (IR) [33]. El modelo de este algoritmo, se basa en los índices de citas aplicados a trabajos científicos [37] y es el algoritmo utilizado por Google para las búsquedas de archivos. Se suele relacionar con el algoritmo *Hyperlink Induced Topic Search* (HITS), un algoritmo de búsqueda que realiza un ranking de páginas web para una búsqueda concreta a partir de los hipervínculos presentes en las páginas web [38].

Al igual que con LexRank, en PageRank se evalua la pertinencia de los conceptos sin importar el contenido, es decir, se puede aplicar a cualquier área de conocimiento. En este caso, se basa en las citas que aparecen en un documento o página web para evaluar el contenido [37]. Trabaja bajo la premisa de que no existen páginas sin citas o hipervínculos entre su contenido y su grafo se organiza con relaciones direccionales, es decir, se consideran referencias de entrada y de salida a cada documento (que actúa como nodo). La limitación principal de este algoritmo es que depende de la regularidad de las referencias, implicando que una distribución irregular de citas y vínculos puede incrementar la relevancia de unos documentos y disminuir la de otros sin tener en cuenta la pertinencia de los conceptos [37].

Al igual que para PageRank, el algoritmo desarrollado en este trabajo tiene en cuenta la direccionalidad de las relaciones, que pueden ser unidireccionales o bidireccionales. Sin embargo, al no considerar documentos y no tener la intención de realizar un resumen de los mismos el funcionamiento es distinto. De la misma forma que en LexRank y TexRank, se realiza una evaluación de la pertinencia de las respuestas y se ordenan en base a la misma y se devuelven todas las respuestas factibles a la cuestión. Por otro lado, al contar con una entrada distinta a los algoritmos mencionados, no se basa en las citas o vínculos, sino que explora todas las relaciones entrantes y salientes de un nodo y devuelve aquellas que se adaptan a las condiciones impuestas en la realización de la consulta.

### 2.3.2. Bases de datos orientadas a grafos

Las aplicaciones descritas anteriormente sobre la Generación Automática de Resúmenes (GAR) se superponen con el desarrollo de aplicaciones de extracción de información biomédica. A nivel universitario, encontramos trabajos que proponen el uso de grafos de conceptos para este propósito. En uno de los trabajos publicados por la Sociedad Española de Procesamiento del Lenguaje Natural [32], se propone la extracción de estos conceptos a partir de un grafo. Para ello, cada uno de los documentos seleccionados pasa por un proceso de tokenización, a partir del cual se identifican los conceptos de *Unified Medical Language System* (UMLS) asociados con cada término para constituir los nodos del grafo, mientras que las relaciones entre dichos conceptos se corresponden con las aristas. A nivel sintáctico se considera también la posición de los conceptos en la oración, de tal forma que el resultado final es un grafo que engloba todo el documento, pero del que se pueden extraer pequeños grafos a nivel de oración en los que se vinculan los conceptos. Este tipo de trabajos aporta una nueva visión sobre la propia estructuración de los documentos y su interacción con la web, desde la que se accede a los documentos.

Por otro lado, estas aplicaciones se suelen centrar en organizar bases de datos concretas. Se ha mencionado antes que MEDLINE es una de las bases de datos de mayor tamaño en el área de la medicina y que el número de artículos que se publican en ella crece considerablemente. Por ello trabajos como el de Illhoi Yoo et al. [39] se centran en proponer un grafo semántico y evaluarlo en esta base de datos. De manera similar a la aplicación anterior, se sirve de los conceptos recogidos en ontologías reconocidas por la base de datos en la que evalúan su algoritmo y se sirven de métodos de clustering para obtener resultados. El trabajo busca mejorar estos algoritmos mediante la representación de la información en un grafo, de manera que el criterio para establecer un cluster no se base únicamente en la similitud entre frases sino también en su relación en el grafo [39].

### 2.3.3. Grafos en biomedicina

Durante este capítulo, se han explicado las bases de la teoría de grafos, sus métodos de funcionamiento y aplicaciones en el área de la biomedicina. En esta sección se exponen métodos específicos utilizados en biomedicina para representar información de forma gráfica.

Una de las formas de representación más comunes en este área y a la vez más similares al trabajo de la Asociación COpenMed es el uso del grafo acíclico dirigido (DAG - Directed Acyclic Graph). Este tipo de representación se utiliza para asociar las entidades de tipo causa a efecto [40]. También se utiliza para el control de sesgos. Uno de los problemas que se considera en la biomedicina, es predecir el resultado de aplicar, por ejemplo, un tratamiento a un grupo sin partir de los sesgos de las personas realizando la predicción [41]. Teniendo en cuenta que los grafos en este ámbito tienen a ser complicados y requieren tener en cuenta muchas relaciones para realizar una predicción adecuada, este tipo de

grafos considera la introducción de variables que afecten tanto a la entidad inicial como a la entidad definida en la predicción, de tal manera que sirvan para controlar los resultados [41]. Sin embargo, la introducción de más variables aplicadas a grandes cantidades de información genera un proceso muy costoso.

En comparación a otras aplicaciones descritas, la forma de procesamiento de información mediante el grafo acícliclo dirigido se asemeja más al trabajo realizado. Al tratar las entidades como nodos en vez de definir los nodos como documentos completos, se adapta mejor a la información adoptada en este trabajo. Además, los vértices del grafo son las relaciones entre entidades, de la misma forma que en el grafo generado a lo largo del trabajo. Sin embargo, esta aplicación está orientada a predecir resultados o servir como herramienta de diagnóstico, esto es un proceso adecuado a un razonador, pero no se aplica al código generado en este trabajo.

Otras aplicaciones de los grafos en medicina se basan en la representación de procesos de esta forma. Aunque se aleja de la implementación de este trabajo, a continuación se presentan algunas ideas de su aplicación. Una de estas aplicaciones es la representación gráfica del potencial eléctrico del corazón [42]. Esta forma de visualización permite crear modelos que ayuden a entender el funcionamiento de las estructuras implicadas. Si bien en el trabajo citado se aplica al potencial eléctrico, esta forma de representación también se ha utilizado para estudiar el flujo sanguíneo [43].

### 2.3.4. Conclusiones

A lo largo de esa sección, se han analizado diferentes aplicaciones orientadas al sector de la biomedicina así como a la información basada en grafos. En este aspecto, se observa que algunas de las técnicas utilizadas en este trabajo son similares a aquellas descritas en el capítulo. Por ejemplo, los rankings de respuestas establecidos en *PageRank* o *LexRank*. Por otro lado, se ha observado que la estructuración en forma de grafo de la información biomédica se ha aplicado a bases de datos mayoritarias así como a herramientas de búsqueda, como PubMed.

Partiendo de las bases sentadas por aplicaciones previas en el sector, este trabajo pretende generar una herramienta orientada a un público más amplio. Si bien bases de datos como MEDLINE poseen más información que además aumenta a mayor velocidad, estas están orientada a la búsqueda de personal de investigación. La herramienta generada en este trabajo busca manejar información de cara a un público no especializado para el cual otras bases de datos son inaccesibles.

Otra diferencia introducida por este trabajo es el formato de la información manejada. A diferencia de *TextRank*, la construcción de las conexiones del grafo no se realiza a partir de los hipervículos asociados a cada entidad, ya que no se manejan documentos sino entidades. Por esto mismo, el tratamiento de la información es distinto y hace que las relaciones en el grafo representen las relaciones entre entidades. Al tratarse de infor-

mación de caracter biomédico, esto permite mayor especificidad en el tratamiento de las relaciones, ya que los tipos de relaciones a tratar están definidos en base a los tipos de entidades posibles y por tanto se adaptan mejor a las necesidades de la información.

# 3. METODOLOGÍA

En esta sección se describe el procedimiento seguido para la elaboración de este trabajo. Se incluyen las decisiones previas a la realización del trabajo así como la estructura general del trabajo y los diagramas del flujo del código.

La realización de este trabajo, se ha realizado de manera iterativa, terminando en cada etapa un prototipo de estructuración de información al que se le han añadido funcionalidades hasta llegar al producto final. Como se verá en el cronograma, se realizaron primero las funciones de información no dependiente del grafo, después aquellas que dependían del mismo, hasta refinar el trabajo y realizar cambios en la visualización. Este tipo de metodología se enmarca en un entorno Agile, en el que se realiza el proyecto de manera iterativa de forma organizada en sprints. Al comienzo de cada sprint se definen los objetivos del mismo, que deben generar una herramienta funcional. En este caso cada sprint ha consistido en desarrollar ciertas funcionalidades, primero funciones no dependientes del grafo, después funciones dependientes del grafo hasta la visualización y verificación de los resultados.

En la metodología Agile, se incluye además una sesión de planificación al inicio y el resultado es verificado por el cliente, de tal manera que tareas que no hayan sido completadas de manera satisfactoria se pueden incluir en el siguiente sprint, así como nuevos requisitos que puedan aparecer en el proceso. En este caso, esas tareas se han realizado junto al tutor Carlos Óscar Sánchez Sorzano ya que, como presidente de la Asociación COpenMed, representa a quien va a recibir el resultado de este trabajo.

## 3.1. Lenguaje y entorno de desarrollo

El código se ha realizado en Python, se trata de un lenguaje de alto nivel diseñado por Guido Van Rossum que apareció por primera vez en 1991. Es un lenguaje dinámico que permite orientación a objetos así como programación imperativa. Su flexibilidad permite aplicarlo a diversas tareas, aunque actualmente se usa especialmente en aplicaciones de aprendizaje máquina [44]. La decisión para adoptar este lenguaje para el proyecto es, a parte de su flexibilidad, para mantener la continuidad con trabajos desarrollados anteriormente por la Asociación COpenMed, para permitir así integrar funciones desarrolladas en este trabajo en futuros trabajos y poder hacer uso de herramientas ya desarrolladas. Además, se encuentra entre los lenguajes de programación más utilizados actualmente, lo que coadyuva a la integración de este trabajo con otras aplicaciones [45].

Por otro lado, se ha utilizado el entorno Spyder[2] para el desarrollo del código. Se trata de un entorno gratuito de código abierto diseñado para la programación científica [46]. Se

[2]https://www.spyder-ide.org/

ha elegido este entorno porque está diseñado específicamente para el tratamiento de datos científicos. Además, provee herramientas de profiling que permiten conocer el tiempo gastado en cada función para poder optimizar el código. En comparación a un entorno como Jupyter, se considera mejor el entorno elegido para una aplicación con múltiples scripts que se referencian entre sí. Además al no tratarse de una aplicación que requiera herramientas de control de versiones o refactorización de código, se ha elegido Spyder frente a entornos más avanzados como PyCharm [47].

Como se explicará más adelante, a lo largo del código se han utilizado archivos tipo pickle para guardar resultados y optimizar el código, este tipo de archivos guardan como un flujo de bytes objetos de python [48]. Para su uso en Spyder, requiere una librería concreta, "pickle". Además, se han utilizado otros módulos como pandas o numpy que vienen incluídos con la instalación del programa y se utilizan para el manejo de estructuras de datos asi como expresiones matemáticas.

## 3.2. Estructura y diagrama de flujo

Para la realización de este trabajo, el código se ha organizado en dos scripts. El primero de ellos, llamado copenmed tools contiene funciones desarrolladas por la Asociación COpenMed, que son comunes a varios proyectos de la asociación. Contiene las funciones de creación del grafo, así como las funciones de propagación, aunque como se explica más adelante, estas se han editado a lo largo de este trabajo para servir al propósito concreto de la estructuración de información. La entrada de este código es el archivo Excel que contiene la información sobre entidades y relaciones, mientras que su salida es un archivo tipo pickle que contiene los diccionarios creados para manejar la información de los nodos.

Los archivos tipo pickle, permiten almacenar objetos de Python como un flujo de bytes sin realizar ninguna conversión [48]. Pickle es generalmente un módulo preinstalado con la versión de python que se utilice. Para generar uno de estos archivos, se genera un camino y nombre para el mismo y se utiliza la función "dump"para introducir contenido. En este trabajo se ha realizado como se ve en la figura 3.1:

```
obj = (_, _, self.entity_type_dict, _, self.entity_dict, _, \
  self.edge_type_dict, _, _, self.edge_dict, _, self.details_dict, \
   _, self.resources_dict, _ ,self.descriptions_dict, _, self.graph)

self.read_weights()

fh = open("summarizer.pkl","wb")
pickle.dump(obj,fh)
fh.close()
```

Fig. 3.1. Utilización de pickle

En el desarrollo de este trabajo se ha utilizado para acceder a los diccionarios asociados con las entidades así como para guardar la información del grafo. La existencia de

estos archivos permite no volver a ejecutar código que genera objetos ya guardados en archivos pickle, favoreciendo la optimización.

Por otro lado, el código principal sobre el que se ha trabajado se llama summarizer. La entrada de este código es el archivo tipo pickle que se obtiene como salida del código anterior, a partir del cual se llama a la función de construcción del grafo contenida en copenmed tools, este archivo es un código que contiene herramientas comunes a todos los proyectos de copenmed (como las funciones de construcción del grafo). La salida de este código es un documento de LaTeX con las fichas de todas las entidades del archivo Excel. Además, cuenta también con un archivo intermedio llamado summarizer.pkl que almacena el grafo. Este archivo se genera una vez por cada cambio de archivo inicial (la entrada de copenmed tools) para evitar prolongar la ejecución más de lo necesario.

El resultado es la distribución de códigos y archivos que se ve en la tabla 3.1:

TABLA 3.1. DISTRIBUCIÓN DE ARCHIVOS

| **COpenMed Tools** | | **Summarizer** | |
|---|---|---|---|
| Entrada | Salida | Entrada | Salida |
| Excel con entidades | copenmed.pkl | copenmed.pkl | Documentación de entidades |
| | | Excel con pesos | |

Se puede ver en la tabla 3.1 que el trabajo se ha realizado utilizando dos archivos de código distintos y dos archivos Excel. El segundo de estos archivos Excel, referido como *Excel con pesos* en la tabla 3.1, contiene todas las características a explorar (causas, consecuencias, observables, etc) en hojas. Dentro de cada hoja se encuentran todos los tipos de relaciones y la seguridad (el peso) asociada a cada una de ellas según la característica. Su construcción se explica en la sección 4.3 llamada *Archivos de entrada*.

Como se ha explicado, el trabajo se centra en el desarrollo del script llamado summarizer, aunque se modifican funciones de las herramientas que se describirán más adelante. A continuación se muestra el diagrama de flujo del código principal:

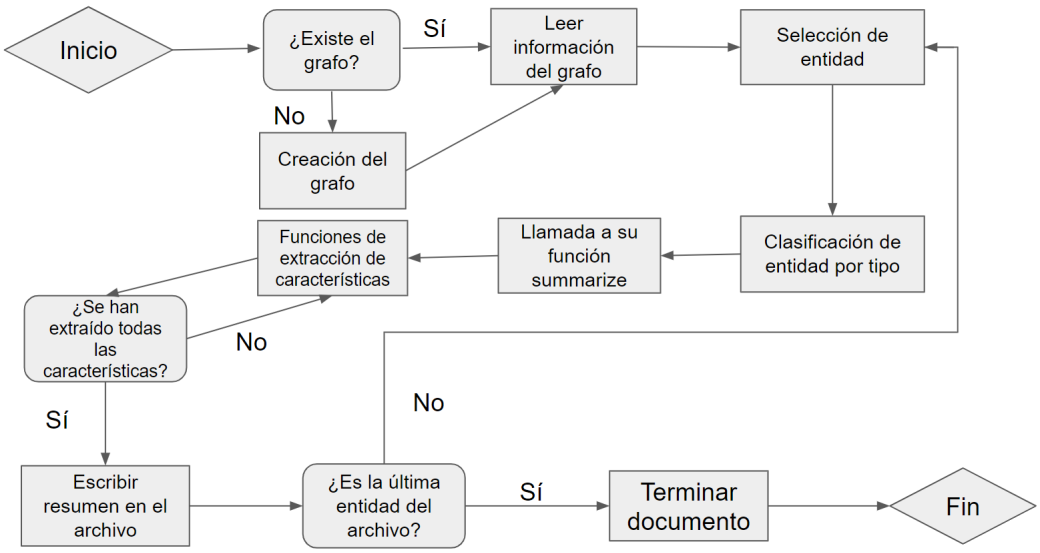



Fig. 3.2. Diagrama del flujo Summarizer

Como se ve en la figura 3.2, después de crear o leer el grafo, el programa repite para cada entidad el proceso de extracción de características. Se explicará en el capítulo de visualización que el documento se construye progresivamente con cada entidad. Después de la última entidad se incluyen los comandos para la finalización del archivo y termina la ejecución.

# 4. DESARROLLO DE LAS FUNCIONES

En esta sección se describe el punto de partida de las funciones del trabajo, así como los objetivos de funcionamiento y el desarrollo de las funciones para obtener dichos resultados.

## 4.1. Punto de partida

Como se ha comentado anteriormente, este trabajo se ha realizado en colaboración con la Asociación COpenMed, por lo tanto, la investigación parte de los códigos y estructuración se ha llevado a cabo por dicha asociación.

Se ha explicado en la definición de los grafos de conocimiento, que suelen llevar asociados un razonador para extraer información del grafo; este caso no es distinto. La entrada de este razonador, es un archivo Excel que contiene los pasos asociados a cada tipo de relación, que a su vez están enmarcadas en un tipo concreto. Es decir, este archivo se divide en diez hojas en base a características asociadas al grafo. Cada hoja contiene todos los diferentes tipos de relaciones, las aristas del grafo, y tienen como título causas, consecuencias, observables, tratamientos, tests, prevención, similar, atención, especialidad y equipo. Se puede ver un ejemplo de la visualización de este archivo en la figura 4.1:

| IdTipoAsociacion | IdTipoEntidad1 | IdTipoEntidad2 | TipoAsociacion | TipoEntidad1 | TipoEntidad2 | A1+, Down | A1-, Down | A2+, Up | A2-, Up |
|---|---|---|---|---|---|---|---|---|---|
| 440 | 15 | 20 | Function is similar to Activity | Function | Activity | | | | |
| 441 | 7 | 15 | Symptom is related to Function | Symptom | Function | | | | |
| 442 | 13 | 22 | GroupOfTreartments should be avoided with GroupOfDiseases | GroupOfTreatments | GroupOfDiseases | | | | |
| 443 | 6 | 22 | Disease prevents GroupOfDiseases | Disease | GroupOfDiseases | -1 | 0,25 | -1 | 0,25 |
| 444 | 22 | 22 | Group1 prevents Group2 | GroupOfDiseases | GroupOfDiseases | -1 | 0,25 | -1 | 0,25 |
| 445 | 7 | 22 | Symptom is never observed in Group | Symptom | GroupOfDiseases | | | | |

Fig. 4.1. Ejemplo de archivo de entrada al razonador

Dentro de cada una de estas hojas se encuentran todos los tipos de relaciones. Si estos son pertinentes al tipo de hoja, por ejemplo *entidad 1 provoca entidad 2* en la hoja de consecuencias, se les asigna un peso en función de la seguridad de la relación. Estas relaciones tienen, en el razonador, cuatro pesos asociados: si están presentes las dos entidades, si está presente la primera, si está presente la segunda y si no están presentes ninguna de las dos entidades.

Por ejemplo, la presencia de una entidad puede indicar el desarrollo de la otra, como la presencia de un síntoma puede indicar determinada enfermedad, por lo que la relación tendrá un peso positivo y más cercano a 1 cuanta más seguridad se tenga en esa relación, pero si la entidad 1 no está presente lo más probable es que no queramos explorar esa relación, por lo que el peso será negativo.

Se puede observar un ejemplo en la figura 4.2:

| | | | | | |
|---|---|---|---|---|---|
| Treatment should be avoided if this Group is given | Treatment | GroupOfSubstances | | | -10 |
| Substance should be avoided if Group is present | Substance | GroupOfDiseases | | | -10 |
| Group enhances the effect of Treatment | GroupOfTreatments | Treatment | 1 | | 1 |

Fig. 4.2. Ejemplo de tratamientos en el razonador

En la imagen 4.2, aparecen los pesos asignados a tres relaciones de la hoja de tratamientos, donde se observa la utilización en el razonador de los pesos negativos. En las dos primeras relaciones *Treatment should be avoided if this Group is given* y *Substance should be avoided if Group is present* se ve que es una relación a evitar y, desde el punto de vista del razonador, se establece un peso muy negativo (-10) para que la relación no se explore nunca.

A la hora de realizar este trabajo no se ha usado el mismo archivo, sino que se ha modificado para servir a la extracción de información, pero es relevante conocer el archivo del razonador para entender la base de este trabajo.

## 4.2. Construcción del grafo

El objetivo principal de este trabajo es generar un documento de información estructurada automáticamente a partir de la exploración y extracción de conocimiento a partir de un grafo. Para esto, el grafo a explorar se ha construido a partir de un archivo Excel que contiene las entidades que servirán como nodos, así como las relaciones que serán las aristas del grafo, los tipos en los que se engloban cada una de estas características e información que servirá para estructurar la información. Estas últimas características son el idioma, una descripción de la entidad y algunos detalles sobre las entidades, como sinónimos.

Para el caso concreto de análisis de información biomédica, las aristas del grafo incluyen direccionalidad. Pueden ser bidireccionales para el caso de relaciones de coexistencia, o unidireccionales para indicar, por ejemplo, consecuencias. A la hora de generar el grafo, se crea un nodo por cada entidad del diccionario, después se crean todas las aristas y se generan los caminos. Estos caminos, tienen por defecto una distancia máxima de dos nodos, ya que a la hora de explorar el grafo, entidades a mayor distancia dejan de ser relevantes, y un umbral de 0.25. Este límite sirve para medir la pertinencia de la relación y se calcula multiplicando el peso, es decir la seguridad, de las relaciones por las que pasa, si este número baja de 0.25 se dejan de explorar los caminos del nodo.

En cada nodo, podemos obtener las relaciones de entrada así como las de salida. Además, se asigna un diccionario de características a cada entidad que se utilizan para el documento final. Estas características son: idiomas, tipo de entidad, tipo de relación (que contiene las relaciones a través de las aristas de entrada y salida al nodo), detalles (que contiene los sinónimos y otras referencias a la entidad), recursos y descripciones. Todo

esto se guarda en un archivo tipo pickle que se utiliza como entrada para el resto del código.

## 4.3. Archivos de entrada

Como se ha explicado en el apartado anterior, la entrada del razonador se compone de un archivo en el que se dividen las relaciones en tipo y se asigna un peso a cada relación según su seguridad y su tipo. Una de las primeras premisas que se siguió en este trabajo fue la de generar un código que trabajara de forma independiente del razonador, ya que su desarrollo se realiza en otro departamento de la asociación y la forma de tratar las entidades es diferente. Sin embargo, se tomó como base este archivo para la entrada del código.

Al igual que en el razonador, el archivo está dividido en hojas según la categoría que se quiere explorar, encontramos: causas, observables, consecuencias, tratamientos, tests, prevención, similar, atención, especialidad y equipo. Cada una de las hojas incluye todos los tipos de relación, aunque solo tienen un peso asignado si son pertinentes. Por ejemplo, en la hoja de equipo que se ve en la figura 4.3, las relaciones que tienen un peso asociado empiezan o terminan con una entidad de tipo “Instrument/Device”.

| | | | | | |
|---|---|---|---|---|---|
| Treatment uses Instrument | Treatment | Instrument/Device | 1 | | 1 |
| Group uses Instrument | GroupOfTreatments | Instrument/Device | 1 | | 1 |
| Symptom can be measured with Device | Symptom | Instrument/Device | 1 | | 1 |

Fig. 4.3. Ejemplo de relaciones de equipo

Cada una de las hojas del archivo de entrada del *Summarizer* contiene ocho columnas, que indican los siguientes aspectos:

- Identificador del tipo de asociación
- Identificador del tipo de la primera entidad
- Identificador del tipo de la segunda entidad
- Nombre del tipo de asociación
- Nombre del tipo de la primera entidad
- Nombre del tipo de la segunda entidad
- Peso de la relación si se comienza en la primera entidad
- Peso de la relación si se comienza en la segunda entidad

Los pesos del archivo del razonador se organizaban en cuatro columnas, sin embargo en este trabajo la entrada se organiza en dos tipos de pesos. Dado que la exploración

del grafo siempre se realiza desde una entidad de partida, los pesos se organizan en una columna si es la entidad uno la que está presente y en otra columna si es la segunda entidad la que está presente.

Por otro lado, los pesos del razonador indican la seguridad de la relación, y se incluyen pesos negativos cuando no se quiere explorar la relación para la entidad. Por ejemplo, en el caso del razonador si se quieren explorar los tratamientos para una determinada enfermedad, se quiere evitar que aparezcan relacionados aquellos que empeoran dicha enfermedad.

Sin embargo, para la realización de fichas informativas, es relevante que aparezcan estas relaciones negativas. Por lo tanto, en este archivo se ha organizado la seguridad de las relaciones en un rango x E [-1, 1] de forma simétrica en torno a cero. Esto es, las relaciones positivas con total seguridad tendrán un peso uno, como *esta entidad está en el dominio de está especialidad* y relaciones positivas con menos seguridad tendrán un peso menor manteniéndose por encima de cero. Mientras que relaciones negativas con total seguridad como *Este tratamiento se debe evitar para este grupo* tendrán peso menos uno. La diferenciación de estas relaciones y la exploración del grafo se realiza después a nivel de función.

De esta manera, relaciones que en el archivo tenían un peso muy negativo como las mostradas en la figura 4.2, en el Excel utilizado en este trabajo, se enmarcan entre -1 y 1 como se ve en la figura 4.4:

| | | | | | |
|---|---|---|---|---|---|
| Treatment should be avoided if this Group is given | Treatment | GroupOfSubstances | | | -1 |
| Substance should be avoided if Group is present | Substance | GroupOfDiseases | | | -1 |
| Group enhances the effect of Treatment | GroupOfTreatments | Treatment | 1 | | 1 |

Fig. 4.4. Ejemplo de relaciones de tratamiento

Por otro lado, este trabajo también parte del archivo de salida descrito en el apartado anterior, que incluye los diccionarios asociados a cada entidad. La creación de este archivo ha sido mayoritariamente llevada a cabo por la Asociación COpenMed. La información que se utiliza del archivo listado como copenmed.pkl en la Tabla 3.1 es la siguiente:

- Diccionario con los tipos de entidades: Contiene el identificador así como el nombre asociado a cada tipo de entidad. En este trabajo se consideran un total de 22 tipos de entidades, se muestra un ejemplo en la figura 4.5.

| IdTipoEntida | TipoEntidad | IdIdioma | Idioma |
|---|---|---|---|
| 20 | Activity | 0 | null |
| 9 | Anatomy | 0 | null |
| 21 | Cause | 0 | null |
| 10 | Condition | 0 | null |

Fig. 4.5. Tipos de entidades

- Diccionario con las entidades: Contiene todas las entidades presentes en el grafo. Se consideran 9695 entidades, se ven algunas de ellas en la figura 4.6.

| IdEntidad | Entidad | IdTipoEntida | TipoEntidad | Idioma | Comentario |
|---|---|---|---|---|---|
| 9 | Reacción alérgica grave | 6 | Disease | Español | null |
| 11 | Alergia alimentaria | 6 | Disease | Español | null |
| 12 | Reacción alérgica por mordeduras y pi | 6 | Disease | Español | null |

Fig. 4.6. Entidades

- Diccionario con los tipos de relaciones: Contiene los identificadores de los tipos de relación así como los nombres, así como el tipo y nombre de las entidades de partida y de llegada. Se consideran 437 tipos de relaciones, algunos de estos se ven en la figura 4.7.

| IdTipoAsocia | IdTipoEntida | IdTipoEntida | IdIdioma | TipoAsociaci | Idioma | TipoEntidad | TipoEntidad2 |
|---|---|---|---|---|---|---|---|
| 3 | 9 | 9 | 4 | Anatomy1 is | English | Anatomy | Anatomy |
| 4 | 9 | 9 | 0 | Anatomy1 is | null | Anatomy | Anatomy |
| 5 | 12 | 13 | 0 | Treatment b | null | Treatment | GroupOfTreatments |

Fig. 4.7. Tipos de relaciones

- Diccionario con las relaciones: Contiene las 94 364 relaciones consideradas entre entidades. Se identifican a partir del tipo de asociación, los identificadores de las entidades implicadas y el identificador de la relación. Se puede ver la estructura así como ejemplos en la figura 4.8

| IdAsociacion | IdEntidad1 | IdEntidad2 | IdTipoAsocia | TipoAsociacion | Descripcion |
|---|---|---|---|---|---|
| 21 | 18 | 9 | 53 | Disease1 seldom evolves to Disease2 | |
| 29 | 16 | 25 | 274 | Treatment may cause Symptom | |
| 33 | 28 | 27 | 19 | Symptom1 implies Symptom2 | |

Fig. 4.8. Relaciones

- Diccionario con los detalles: Incluye el identificador y nombre de la entidad asociada, el identificador del idioma en el que se escribe (3 = Español, 4 = Inglés etc) así como un idenficador del recurso asociado y el tipo de entidad que se trata. Esta estructura se puede ver en la figura 4.9:

| IdRecurso | IdIdioma | Idioma | IdEntidad | Entidad | IdTipoEntida | TipoEntidad | Nivel |
|---|---|---|---|---|---|---|---|
| 11 | 3 | Español | 11 | Alergia alime | 6 | Disease | 0 |
| 12 | 3 | Español | 12 | Reacción alé | 6 | Disease | 0 |
| 14 | 3 | Español | 14 | Morfina | 12 | Treatment | 0 |

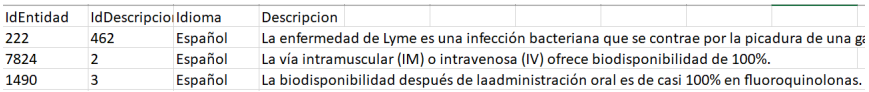
Fig. 4.9. Detalles

- Diccionario con los recursos: Como en la hoja anterior, contiene el identificador del recurso, el nivel del recurso que indica la fiabilidad del resurso y el URL del mismo, como se ve en la figura 4.10.

| IdRecurso | IdEntidad | Nivel | URL |
|---|---|---|---|
| 7 | 11 | 0 | https://medlineplus.gov/spanish/ency/article/000817.htm |
| 8 | 12 | 0 | https://medlineplus.gov/spanish/ency/article/000033.htm |
| 10 | 14 | 0 | https://medlineplus.gov/spanish/druginfo/meds/a682133-es.html |

Fig. 4.10. Recursos

- Diccionario con descripciones: esta hoja contiene el identificador de la identidad a la que pertenece la descripción, el identificador de la descripción, el idioma y la propia descripción. Se muestra un ejemplo en la figura 4.11.



| IdEntidad | IdDescripcio | Idioma | Descripcion |
|---|---|---|---|
| 222 | 462 | Español | La enfermedad de Lyme es una infección bacteriana que se contrae por la picadura de una g |
| 7824 | 2 | Español | La vía intramuscular (IM) o intravenosa (IV) ofrece biodisponibilidad de 100%. |
| 1490 | 3 | Español | La biodisponibilidad después de laadministración oral es de casi 100% en fluoroquinolonas. |

Fig. 4.11. Descripciones

Estos diccionarios utilizados se almacenan como un flujo de bytes en el archivo copenmed.pkl. A lo largo de este trabajo se incluyó el diccionario de descripciones, mientras que el resto habían sido ya incluidos por la Asociación previamente a la realización del proyecto.

## 4.4. Funciones de exploración del grafo

Como se ha explicado en la estructura del código, el primer paso es generar el grafo. Para optimizar el código, se decidió guardar el grafo en un archivo tipo pickle, de tal forma que la construcción del mismo solo se debe realizar una vez entre cada cambio de archivo de entrada. Una vez generado el grafo, se separan las entidades por tipos y se define la función a través de la cuál se leen los pesos de las relaciones. Esta función utiliza los pesos de cada hoja para propagar las relaciones desde la entidad de partida.

Las funciones de exploración del grafo son las siguientes:

- Propagate Up: esta función se utiliza para extraer las relaciones de entrada al nodo. Utiliza el atributo reachable, que almacena el camino, en este caso hasta el nodo desde otros y realiza el camino inverso desde el nodo de entrada hasta los nodos finales del camino. Cada vez que se añade un nodo al camino se multiplica el threshold por la fuerza de la relación tratada, almacenada en el parámetro edgeStrength y antes de añadir el siguiente nodo se comprueba que la fuerza del camino se mantiene dentro del límite de relevancia.

- Propagate Down: esta función funciona de manera muy similar a la anterior y sirve para extraer las relaciones de salida del nodo, los outgoing edges. En este caso, el camino es del nodo de entrada hacia los nodos de salida, por lo que se realiza de manera directa. De igual manera, se comprueba que la seguridad de la relación esté dentro del límite.

- Propagate: esta última función llama a las dos anteriores para extraer los caminos tanto de los nodos de entrada como de salida.

Como se ha explicado en el apartado sobre los archivos de entrada, los pesos de las relaciones están divididos en negativos y positivos y esta diferencia se trata a nivel de función

en la propagación. Para evitar cambiar los mecanismos de comprobación del límite de seguridad así como el recorrido de los caminos del grafo, se decidió introducir un nuevo parámetro llamado tipo de relación para separar entre positivos y negativos. Para esto, si el parámetro está marcado a negativo, se toma el valor absoluto del peso y se explora el grafo de igual manera que se haría con una relación positiva.

Por otro lado, es importante justificar la elección del límite a través del que se exploran los caminos. Como ya se ha explicado, marca la seguridad de la relación. Para ver una diferencia, relaciones del tipo "este tratamiento puede prevenir esta entidad" tienen menos seguridad que relaciones categóricas como "esta causa promueve este patógeno". Para establecer la relevancia se establece la doble comprobación de longitud del camino así como del límite de seguridad. Aunque nos puede interesar explorar dos relaciones com seguridad media, del tipo "puede provocar" o "a veces causa", relaciones de tipo "en ocasiones provoca" que tienen poca seguridad provocan que al explorar un nodo más del grafo se obtengan entidades poco relevantes a la entidad tratada. Además, se ha observado que establecer un límite más bajo provoca que sea siempre la distancia, dos, la que corta la exploración del grafo.

## 4.5. Funciones de extracción de características

Se ha visto anteriormente en el diagrama de flujo general, que las funciones de resumen de cada entidad llaman a funciones de extracción de características. Es a partir de estas funciones que se generan las subsecciones dentro de cada entidad, que contienen la información pertinente.

Estas funciones se dividen en dos grupos, el primer grupo de funciones extrae información de la propia entidad, considerando los diccionarios asociados a cada nodo. Estas funciones son las más sencillas ya que no requieren explorar el grafo y se desarrollaron primero. Dependen de los diccionarios explicados anteriormente y acceden concretamente al diccionario de detalles, de descripciones y de recursos.

El segundo grupo son funciones que exploran el grafo para obtener las características; estas se basan en los pesos provistos por el archivo Excel así como en las funciones de propagación.

### 4.5.1. Características de la entidad

Como ya se ha explicado, estas funciones se basan en los parámetros de cada entidad para obtener la información, a continuación se muestra en la figura 4.12 un diagrama de flujo genérico para estas funciones:

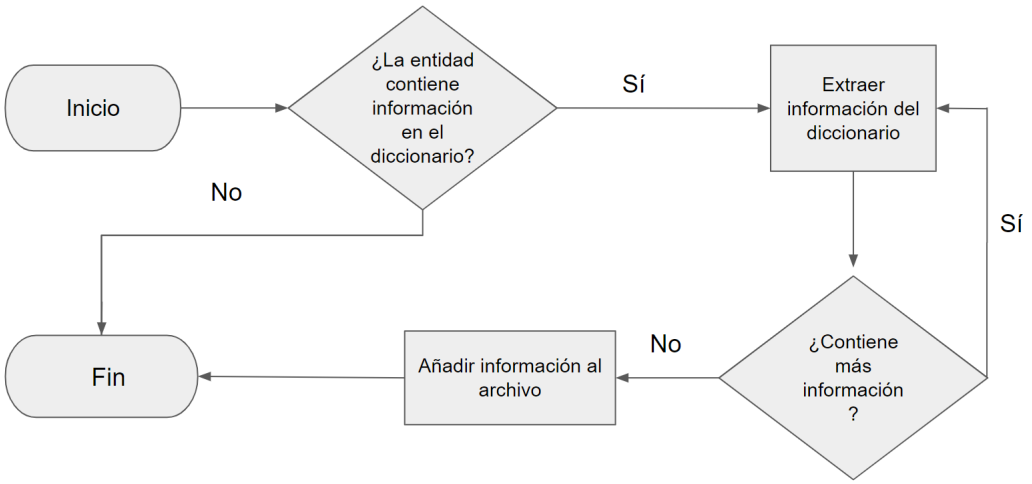



Fig. 4.12. Diagrama de flujo de las funciones de entidad

Las características que se extraen con este método son:

- Sinónimos: en esta función se extraen los sinónimos, nombres científicos y comunes de la entidad. Se obtienen a través del diccionario de detalles y se dividen en varias categorías. En primer lugar, los sinónimos se ordenan por nivel de tecnicismo en tres niveles. Para este trabajo se han dividido en dos niveles: nombres comunes y nombres técnicos. Esta separación genera dos subsecciones distintas, una por cada tipo. Por otro lado, se considera también, si lo tiene, el nombre en inglés de la entidad. A nivel de función se ve de la siguiente forma:

```
def addSynonyms(self, idEntity, mainName):
    retval_tec=""
    retval_com=""
    retval_en =""
    for _,row in self.details_dict[idEntity].iterrows():
        if row.Entidad!=mainName:
            if row.Nivel== 2 :
                if retval_tec=="":
                    continue
                else:
                    retval_tec+=", "

                retval_tec+=row.Entidad
            else:
                if row.IdIdioma ==4:
                    if retval_en =="":
                        continue
                    else:
                        retval_en+= ","
                    retval_en += row.Entidad
                else:
                    if retval_com == "":
                        continue
                    else:
                        retval_com+=", "
                    retval_com+=row.Entidad
```

Fig. 4.13. Función de extracción de sinónimos

En este caso *retval com* representa los sinónimos comunes, *retval tec* la denominación técnica y *retval en* el nombre en inglés. En la visualización de las entidades se ve de la siguiente forma:

```
ENFERMEDAD. 18. Alergia al polen.
Conocido de forma técnica como: Astenia, Polinosis
```

Fig. 4.14. Ejemplo de visualización de nombre técnico

- Recursos: a través de esta función se obtienen recursos web e imágenes que aportan más información sobre la entidad. Esta característica está guardada en el diccionario de recursos y se divide en dos partes. Se tratan por un lado las URL y por otro las imágenes, que se encuentran presentes sobre todo en entidades anatómicas. De nuevo cada tipo se separa en una subsección distinta. En este caso la función se comporta de la siguiente manera:

```
#URLs de referencia
def addResources(self, idEntity):
    retval_im = ""
    retval_rec = ""
    image_strings = [".jpg", ".png", ".jpeg", ".bmp", ".gif", ".tif" \
        ,".tiff" , ".psd" , ".eps" , ".svg", ".raw"]
    if idEntity in self.resources_dict:
        for i,row in self.resources_dict[idEntity].iterrows():
            if "data:image" in row.URL:
                continue
            for image_string in image_strings :
                if image_string in row.URL:
                    retval_im+="    "+row.URL+"\n"
            else:
                retval_rec+="    "+row.URL+"\n"

    if retval_rec!= "":
        title1 = "Para saber más:"
        latex.writeURL(title1, retval_rec)
    if retval_im != "":
        entity = self.entity_dict[idEntity][0]
        title2= "Puede ver la siguiente imagen:"
        latex.includeImage(title2,entity, retval_im)
```

Fig. 4.15. Función de recursos

En este caso *retval rec* recoge los URL de los recursos mientras que en *retval im* se recogen las imágenes. En la versión final de la visualización los URL se pueden abrir directamente:

**2.1.5 Para saber más:**

https://es.wikipedia.org/wiki/Faringe

Fig. 4.16. Ejemplo de recursos

- Descripción: las descripciones de las entidades se encuentran en su propio diccionario, en este caso el proceso es directo ya que no se divide en subcategorías. En este caso la función es más sencilla, ya que solo requiere acceder a la entidad correspondiente:

```
#añadir descripción
def addDescription(self,idEntity):

    aux = " "
    if idEntity in self.descriptions_dict:
        for i,row in self.descriptions_dict[idEntity].iterrows():
            aux+="\n"+row.Descripcion+"\n"

    if aux!="":
        title = "Descripción: "
        latex.writeOtherSubsection(title, aux)
```

Fig. 4.17. Función de descripción

La visualización también es bastante directa, además esta función es la primera que se añade después del título de la sección:

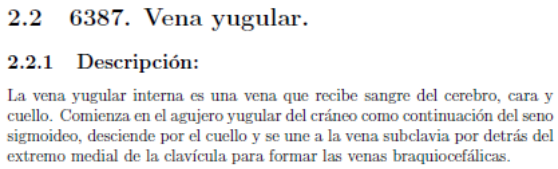



Fig. 4.18. Visualización de la descripción

### 4.5.2. Características del grafo

A continuación se explicarán las funciones que extraen características a través del grafo, estas funciones utilizan la propagación para conseguir la información relevante y en general siguen el diagrama de flujo mostrado en la figura 4.19:

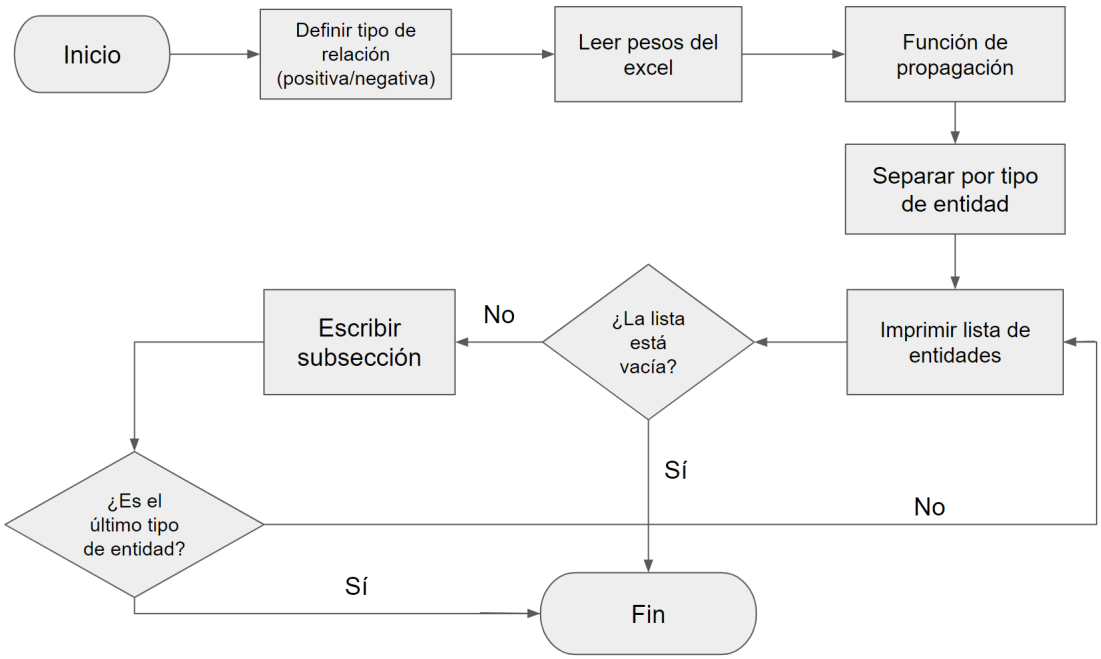



Fig. 4.19. Diagrama de flujo de las funciones grafo

En este caso las funciones que encontramos son:

- Similar: esta función explora el grafo en función de las relaciones que tienen un peso asignado en la hoja "Similar", en este caso se exploran únicamente las relaciones positivas, y no se establece un tipo de entidad concreto a buscar. Un ejemplo del tipo de relaciones que se consideran en esta función es *Substance is related to Anatomy*

- Causas: esta función explora las causas de la entidad seleccionada, y solo se aplica a ciertos tipos de entidad. Por ejemplo, se aplican a las enfermedades y a los síntomas, pero no a los tratamientos. En este caso, la propagación es distinta para distintos tipos de entidades, en el grupo uno que contiene enfermedades, causas, condiciones, etc La propagación se realiza descendentemente en la hoja de consecuencias y de forma ascendente en la de causas, ya que ser consecuencia de una entidad hace que esta sea tu causa. Esta función considera relaciones como *Pathogen can cause Group* o *Function may cause Group*. Aplicado a una enfermedad, se ve de la siguiente forma:

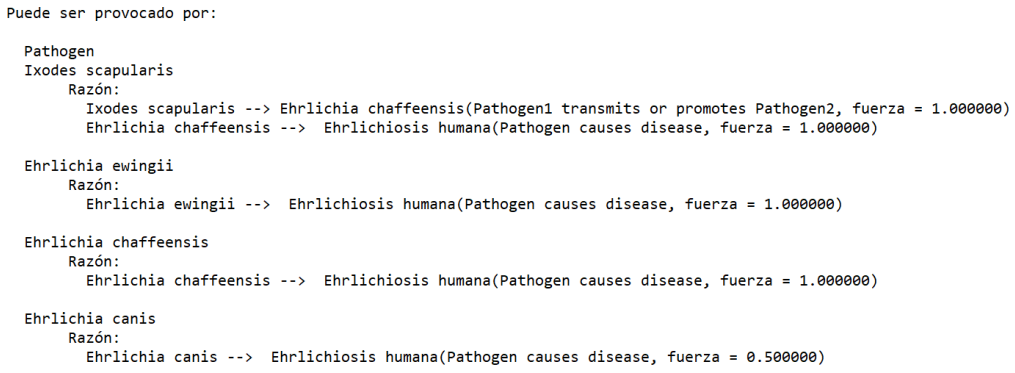



Fig. 4.20. Ejemplo de causas

Por otro lado, también se considera una función separada que explora las relaciones negativas, es decir, entidades que no pueden ser causas de la entidad seleccionada. Esta función se aplica a los mismos grupos de entidades de los que se extraen las causas. Alguna de estas relaciones negativas son: *Disease prevents GroupOfDiseases* y *Disease seldom evolves to Condition*

- Consecuencias: al igual que en las causas, esta función extrae consecuencias de la entidad seleccionada explorando las hojas de causas y consecuencias y sus asociaciones con el grafo. Se aplica a tratamientos, enfermedades, pruebas y otras entidades y explora de manera ascendente las causas y de manera descendente las consecuencias. El tipo de relaciones que se consideran en esta función son: *Pathogen is more observed when this Disease is present*, *Cause may cause condition* o *Group increases Risk*.

  De manera similar a las causas, también se exploran las relaciones negativas en otra función para obtener entidades que no pueden o suelen derivar de la entidad inicial. Algunos ejemplos de estas relaciones son: *Treatment prevents Activity* o *Treatment inhibits Substance*.

- Tratamientos: esta función extrae los tratamientos respecto a cierta entidad. En este caso, la función de propagación incluye un parámetro llamado tipos de entidad, a través del cual se limitan los tipos de entidad que puede devolver, en este caso solo aquellos marcados como tratamientos o equipo. Algunos ejemplos de las relaciones que se exploran en esta función son: *Treatment can be used against Pathogen*, *Group enhances the effect of Treatment* o *Group can be treated with Substance*.

  Igual que en las anteriores, aquí también se genera una función separada que explora las relaciones negativas, siendo en este caso los tratamientos contraindicados. Las relaciones exploradas tienen connotaciones negativas para el tratamiento, algunos ejemplos son: *Treatment should be avoided if this Group is given* o *Condition cannot be treated with Group*.

- Prevención: esta función indica medidas de prevención para evitar ciertas entidades, se aplica a enfermedades, grupos de enfermedades y entidades similares. Al igual que en los tratamientos, solo acepta entidades pertenecientes al grupo de tratamientos. En este caso, la propagación se realiza en ambas direcciones, para obtener todas las entidades que tengan un peso asignado en prevención y que sean tratamientos. Algunas de las relaciones consideradas en esta función son: *This Substance is beneficial for this Anatomy* o *Treatment may prevent Symptom*.

- Test: esta función recoge los test y pruebas que se deben hacer para diagnosticar entidades o para indicar lo que diagnostican. En este caso se divide cuatro funciones. La primera de ellas se aplica a enfermedades y entidades de su grupo e indica con qué pruebas se diagnostica la entidad.

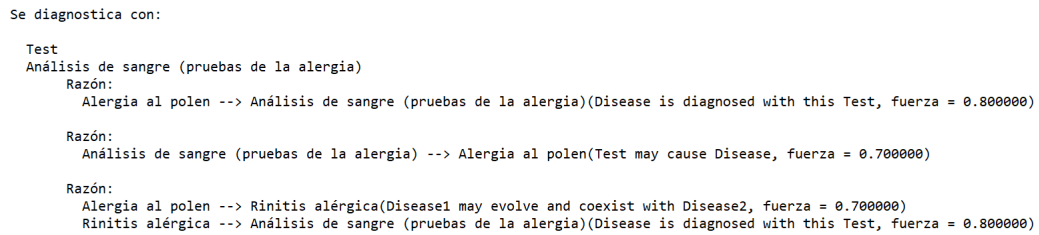



Fig. 4.21. Ejemplo de prueba en enfermedades

La segunda función se asocia a resultados o grupos de resultados y los relaciona con las pruebas de las que se suelen obtener dichos resultados. La tercera función se aplica a las propias pruebas, y obtiene la funcionalidad de las mismas, es decir, qué diagnostican. Cuando se aplica a una prueba se ve de la siguiente forma:

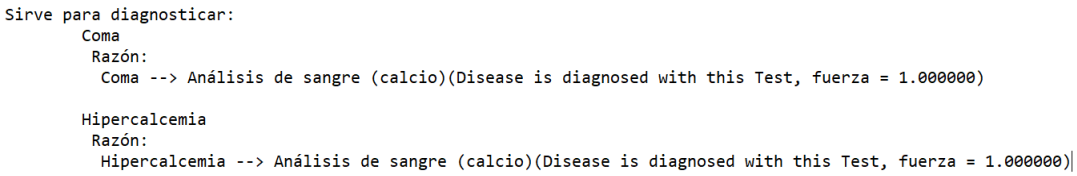



Fig. 4.22. Ejemplo de Tests aplicado al análisis de calcio en la sangre

Algunas relaciones positivas consideradas son: *eaturFe is studied with Test* o *GroupOfDiseases requires GroupOfTests*

Por último, la cuarta función explora las relaciones negativas, esta función se aplica junto con la primera a las enfermedades, ya que indica los tests que se deben evitar para un diagnóstico. Podemos encontrar el siguiente ejemplo de relación en esta función: *Test is contraindicated if Symptom is present*.

- Atención: la función de atención indica entidades, sean síntomas, actividades u otras entidades a las que se debe prestar especial atención si están presentes en la entidad

de partida. Se aplica a todos los grupos de entidades y se realiza una búsqueda en ambas direcciones. Cuenta también con una función para explorar las relaciones negativas, que indican entidades a evitar cuando están en contacto con la entidad inicial. Indica por ejemplo actividades que no se pueden realizar durante un tratamiento. Algunas de las relaciones que se tienen en cuenta en estas funciones son: *Pathogen is more observed when this Disease is present*, *Cause cannot be treated by Treatment*, *Test is contraindicated if this Group is present*.

En este caso, aunque relaciones positivas y negativas se tratan en funciones distintas el resultado es el mismo: entidades y relaciones a las que se debe prestar atención ya tengan un efecto positivo o negativo en la entidad inicial.

- Equipo: la función de equipo sirve para extraer los instrumentos y máquinas usadas en ciertas entidades. Se aplica principalmente a los tratamientos y tests. Algunas de las relaciones consideradas en esta función son: *Device is used in Anatomy*, *Treatment uses Instrument* o *Symptom can be treated with Device*.

- Observables: esta función indica en qué entidades se observa la entidad de partida. Por ejemplo, en qué anatomía se presentan los síntomas de una enfermedad y relaciones similares. Se aplica a los síntomas, enfermedades y entidades de estas categorías. Se consideran relaciones como: *Feature occurs in Anatomy*, *Disease implies TestResult* o *Group promotes Activity*. Se puede ver un ejemplo de su visualización en la figura 4.23.

```
Se puede observar:

  Anatomy
  Cabeza
       Razón:
          Ehrlichiosis humana --> Dolor de cabeza(Disease causes Symptom, fuerza = 1.000000)
         Dolor de cabeza --> Cabeza(Symptom can only be observed in Anatomy, fuerza = 1.000000)

  Estómago
       Razón:
          Ehrlichiosis humana --> Vómito(Disease causes Symptom, fuerza = 1.000000)
         Vómito --> Estómago(Symptom is related to Anatomy, fuerza = 1.000000)

       Razón:
          Ehrlichiosis humana --> Náuseas(Disease causes Symptom, fuerza = 1.000000)
         Náuseas --> Estómago(Symptom is related to Anatomy, fuerza = 1.000000)
```

Fig. 4.23. Ejemplo de Observables aplicado a la Ehrlichiosis humana

- Especialidad: por último la función de especialidad se utiliza para indicar a qué área pertenece la entidad. Se aplica a todos los tipos de entidad y se realiza la búsqueda en todas las direcciones. El tipo de relaciones tratadas en esta función son: *Condition is in the domain of Specialty* o *Device is in the domain of Specialty*. Aparece de la siguiente forma en las fichas de las entidades:

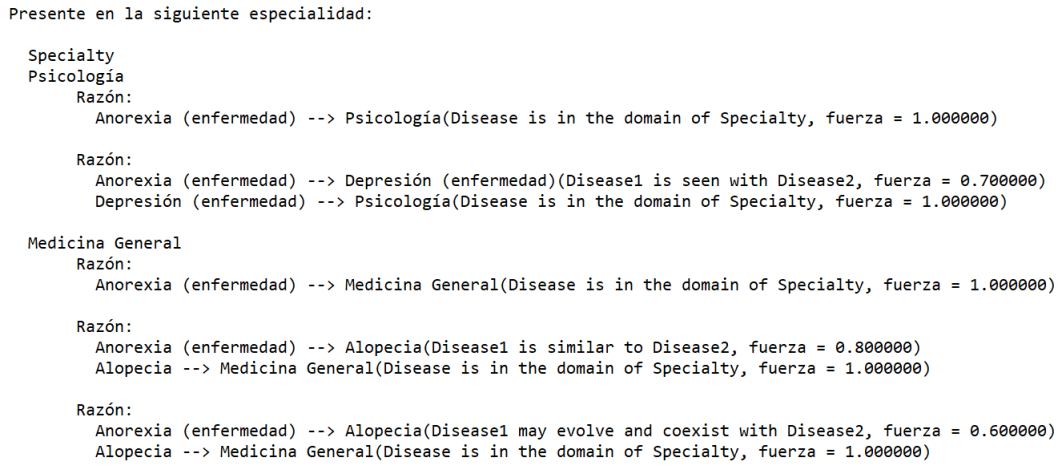



Fig. 4.24. Ejemplo de Especialidad aplicado a la anorexia

### 4.5.3. Funciones comunes

Se ha mencionado anteriormente la existencia de una función común en el diagrama de flujo de las funciones del grafo, se trata de una función llamada print list. Esta función llama a su vez a una función print path. En esta sección se explica el desarrollo y funcionamiento de las mismas.

- Print List: como su nombre indica, esta función se encarga de generar una lista, concretamente genera la lista de las entidades relacionadas junto con la justificación de su relación. Se ha indicado anteriormente que se ejecuta después de la propagación, por lo que la entrada de esta función es la lista de índices que supone la salida de la función de propagación. Además, recibe también la entidad de inicio y un parámetro con el tipo de entidad y los tipos válidos a devolver. En primer lugar ordena los índices de las entidades y para cada uno comprueba que cumplan las condiciones, que pertenezcan al tipo de entidad a devolver, que no sean la entidad de partida, que existan dentro del diccionario de entidades, etc. Si se cumplen las condiciones se decide mostrar la entidad, para ello se accede al nodo desde el grafo, ya que contiene los atributos reachableUp y reachableDown que nos permiten obtener el camino desde la entidad de partida. Con este camino, se llama a la función print path para obtener la razón de la relación. La salida de esta función es una lista donde cada elemento en un primer nivel es el identificador de la entidad de llegada con su razón y la fuerza de la relación.

  Se puede observar el diagrama de flujo de la función en la figura 4.25:

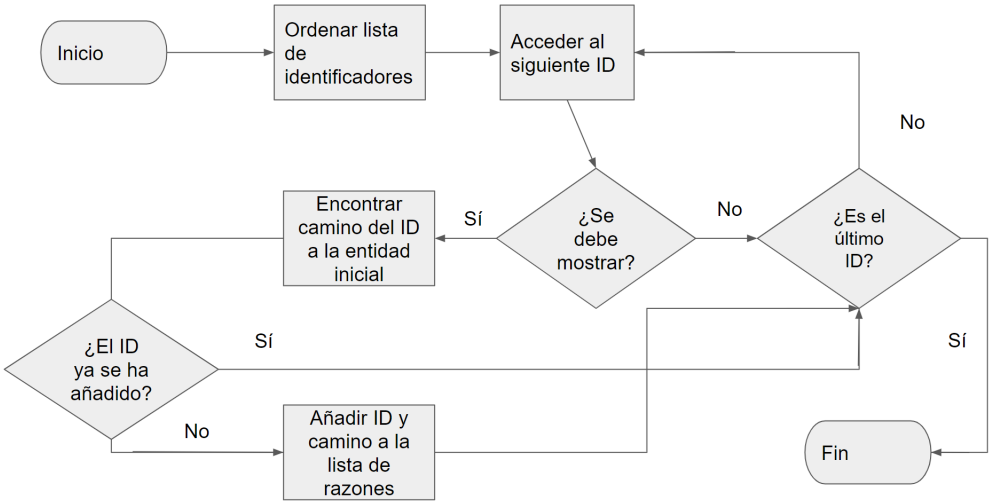



Fig. 4.25. Diagrama de flujo de la función printList

- Print Path: Esta función se usa para justificar la conexión entre la entidad de partida y la entidad final. Recibe estos dos parámetros así como el recorrido de la función print List para devolver el tipo de relación que une las dos entidades, incluyendo también entidades intermedias si las hubiera. Además, devuelve la fuerza total de la conexión, es decir, la multiplicación de los pesos de todas las relaciones por las que pasa el camino, lo que permite ordenar las relaciones para mostrarlas después por orden de relevancia.

  A continuación se muestra el diagrama de flujo asociado a la función:

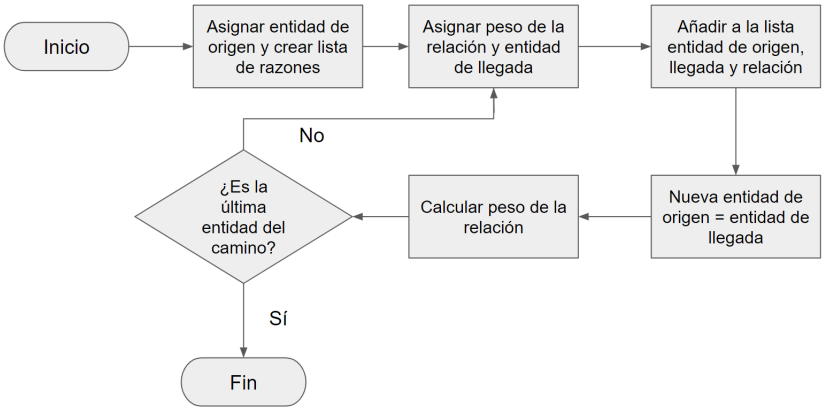



Fig. 4.26. Diagrama de flujo de la función print Path

La asignación de los parámetros como el peso de la relación y la entidad de llegada se obtienen del camino a la entidad, que es un parámetro de entrada a la función.

Además de estas funciones comunes, existen también las funciones destinadas a la visualización de la información, en las que se ahondará en el apartado correspondiente.

# 5. AGRUPACIONES POR TIPO DE ENTIDAD

A lo largo de la explicación sobre las funciones, se ha mencionado que las entidades se dividen en grupos con características comunes para llamar las funciones. A lo largo de este capítulo se describen los escenarios en los que se ha utilizado este agrupamiento así como el procedimiento para realizar los mismos.

En primer lugar, es importante conocer los tipos de entidades sobre los que se trabaja, desde el archivo Excel de entrada, vemos que los tipos de entidades manejados y ejemplos de cada tipo son: actividad (por ejemplo ejercicio físico o estrés), anatomía (por ejemplo hígado o faringe) , causa (por ejemplo ácaros del polvo o moho) , condición (por ejemplo embarazo múltiple o muerte), enfermedad (por ejemplo pie de atleta o alergia alpolen), función (por ejemplo retención renal de sodio), grupo de enfermedades (por ejemplo alergias), grupo de sustancias (por ejemplo hormonas), grupo de tests (como análisis de sangre o test de embarazo), grupo de tratamientos (antihistamínicos o quimioterapia), instrumento (como fonendoscopio o respirómetro), patógeno (como coxiella brunetii), función fisiológica (como presión arterial o pulso), población (como ancianos o mujeres entre 30 y 45 años), riesgo (por ejemplo episodios previos de la enfermedad o vejez), especialidad (como psicología o medicina general), sustancia (como humo del tabaco), síntoma (como tos o urticaria), test (por ejemplo análisis de electrolitos en la sangre), resultado de test (como nivel bajo de sodio o temperatura >41.5º) y tratamiento (por ejemplo adrenalina o morfina). Se puede observar que algunos tipos son muy parecidos entre sí, mientras que otros, aunque dentro del campo de la biomedicina, tienen características muy diferentes.

Para acceder al tipo de entidad asociado, cada una de las categorías tiene un número asociado, almacenado dentro del diccionario asociado con cada identificador de entidad. De manera que desde que se asigna una entidad inicial, se tiene la siguiente estructura de información:

TABLA 5.1. ACCESO A ENTIDADES Y SUS TIPOS

| ID Entidad | Nombre entidad | ID tipo de entidad | Nombre tipo entidad |
|---|---|---|---|
| 18 | Alergia al polen | 6 | Enfermedad |
| 42 | Estrés | 20 | Actividad |

En la tabla 5.1 se puede ver que cada tipo de entidad tiene un identificador fijo, enfermedad siempre es 6, actividad siempre 20, ect. Dentro de cada tipo de entidad se recogen las entidades, cada una con su identificador particular.

A continuación se explica con algunos ejemplos el contenido de todos los tipos de entidad, con el identificador asignado a cada tipo de entidad así como el nombre con el que se les refiere en el archivo de entrada:

- Actividad: con identificador número 20, esta categoría se define como “Activity” en el archivo de entrada. Una actividad se define como la facultad de obrar, o el conjunto de tareas y operaciones propias de una entidad. En este trabajo, la categoría engloba actividades físicas realizadas por personas como abuso de alcohol, ejercicio físico o fumar. Además, contiene también actividades generales que afectan a entidades como cambios hormonales u osificación. Se puede observar que este tipo de entidades pueden servir como causas para entidades de otro grupo, como aumento del consumo de sodio puede llevar a hipertensión, y a entidades del mismo grupo, como las emociones fuertes pueden derivar en estrés.

- Anatomía: este grupo de entidades se identifica a través del número 9 bajo el nombre “Anatomy”. En este grupo, se recogen las características de los humanos como la ubicación y disposición de órganos, huesos y músculos, así como la relación existente entre ellos. En este tipo de entidad, es donde se reflejan síntomas y enfermedades, por lo que se relaciona con los observables. Además, representan lugares en lo que se aplican tratamientos y que interaccionan con el equipo médico. Para su propia ficha, son entidades descriptivas, sin causas o consecuencias.

- Causa: esta categoría se ordena bajo el nombre “Cause” siendo 21 su número de identificación. Engloba los motivos que se dan para que ocurra otra cosa determinada. En esta categoría se incluyen causas que no están recogidas en otros grupos de entidades, es decir, no aparecen enfermedades o tests. Algunos ejemplos de las entidades recogidas son: cambios del clima o agua contaminada.

- Condición: bajo el nombre “Condition” con identificador número 10, esta categoría recoge las características propias de una persona, engloba trastornos preexistentes y también se utiliza para referirse al estado actual de un paciente. Incluye entidades como barbilla prominente, muerte cerebral, embarazo o prognatismo.

- Enfermedad: con un identificador número 6 y bajo el nombre “Disease”, este grupo es el más numeroso de los listados. Una enfermedad se refiere a una alteración estructural o funcional del organismo que origina la pérdida de salud, y así es el tipo de entidades que contiene. Algunos ejemplos almacenados en esta categoría son: carcinoma espinocelular, cólera o diabetes.

- Función: bajo el nombre Function y con un identificador número 15, una función se describe como la actividad propia de un ser vivo o de sus aparatos, órganos, tejidos o células. Incluye funciones desarrolladas por el cuerpo humano y otros organismos como aumento de la producción de insulina, retención renal del potasio o quimiotaxis. Estas entidades pueden servir como causas o consecuencias de otras, así como manifestarse como causas observables de enfermedades o grupos.

- Grupo de enfermedades: este grupo tiene como identificador el número 22 y se utiliza bajo el nombre “Groups of diseases”. Está compuesto por enfermedades que

se asocian bajo el mismo grupo, incluye por ejemplo artritis, que incluye todas las enfermedades relacionadas o alergias.

- Grupo de sustancias: bajo el nombre "group of substances" y con un identificador número 24, esta agrupación engloba grupos de materiales químicos o material de estructuras biológicas con un nexo común. Algunos ejemplos de esta categoría son hormonas, polen o sustancias aditivas.

- Grupo de tests: esta categoría tiene el identificador número 23 y se llama "Group of Tests", al igual que los anteriores grupos, engloba entidades pruebas generales que luego se especifican en pruebas más concretas. Algunos ejemplos de esta categoría son análisis de sangre o exámenes de detección de embarazo. Este tipo de entidades está relacionada con tests más concretos con un tipo de relación "pertenece a" y también se considera como herramienta de diagnóstico.

- Grupo de tratamientos: "group of treatments" es la categoría con el identificador número 13. Contiene grupos de entidades que engloban tratamientos, es decir, algunos de los términos más generales de los mismos. Algunos ejemplos de esta categoría son: antihistamínicos, glucocorticoides o sulfamidas.

- Instrumento: esta categoría se identifica con el número 26 y tiene como nombre "Instrument/Device". Contiene los objetos y aparatos que sirven para registrar y observar así como para realizar curas o tratamientos. Recoge entidades como el fonendoscopio, respirómetro o termómetro y se suele relacionar con pruebas de diagnóstico y tratamientos.

- Patógeno: con el número 19 y bajo el nombre "Pathogen", esta categoría recoge microbios patógenos así como entidades que pueden causar una entidad o trastorno. Son entidades que actúan como causa, algunos ejemplos con Bartonella henselae, abiotrophia o norovirus.

- Función fisiológica: esta categoría se identifica con el número 25 y el nombre "physiological function". La categoría se define como todo lo funcional, relativo a las funciones o al funcionamiento de una célula, de un tejido, de un órgano o de un ser vivo. Algunos ejemplos de esta categoría son la frecuencia respiratoria o la presión arterial.

- Población: bajo el nombre de "population" y el identificador número 8, esta categoría contiene grupos de población determinados. Se define como el conjunto de individuos de los cuales se quiere conocer algo. Se realizan divisiones por edad, sexo o hábitos. Algunos ejemplos son: personas de más de 55 años, mineros o pacientes diabéticos.

- Riesgo: con el identificador número 16 y bajo el nombre "risk", un riesgo se define como una Situación determinada o condicionada por la presencia de eventos o

fenómenos de cualquier naturaleza a los cuales se expone el individuo en su ambiente, que están relacionados con la aparición de una enfermedad o de un efecto indeseable y que pueden ser la causa de los mismos. Se considera como un factor etiológico o causal cuando una modificación en su frecuencia implica una modificación en la frecuencia de la enfermedad o el efecto. En este caso indica factores de riesgo como antecedentes familiares de una enfermedad, sustitución hormonal o VIH positivo.

- Especialidad: "specialty" con un identificador número 14, engloba las ramas de la biomedicina cuyo estudio se centra en una parte limitada de la misma. Algunos ejemplos de esta categoría son la ginecología, las urgencias o la medicina general.

- Sustancia: o "substance", con identificador número 18, se define como Material de composición constante caracterizado por las entidades químicas (moléculas, átomos, iones) que lo componen y por las propiedades resultantes. Contiene entidades como humo de tabaco o químicos en los alimentos, que pueden servir como causas o factores de riesgo para grupos o enfermedades.

- Síntoma: "symptom", 7, se define como la manifestación de una enfermedad o de un síndrome que solo es percibida por el individuo que lo padece. En este caso, incluye también aquellos signos que pueden ser medidos y probados por el personal sanitario. Algunos ejemplos son: agitamiento, caída del pelo o dolor de cabeza.

- Test: con número de identificación 11 esta categoría contiene pruebas, los experimentos o ensayos clínicos llevados a cabo para comprobar las características biológicas de una sustancia así como la existencia de patógenos o presencia de enfermedades. Algunos ejemplos de esta categoría son: análisis de ADN, biopsia del hígado o histerosonografía.

- Resultado de test: "test result", con identificador número 16, es una categoría que recoge los resultados de las pruebas descritas en el apartado anterior. Estos resultados se enlazan con posibles tratamientos así como con los tests desde los que parten. Nivel bajo de hierro o alta temperatura corporal son ejemplos de esta categoría.

- Tratamiento: o "treatment", con identificador número 12, es el conjunto de medidas médicas, farmacológicas, quirúrgicas, físicas o de otro tipo encaminadas a curar o a aliviar las enfermedades. Es también una de las categorías mayoritarias e incluye entidades como la dopamina, morfina o salbutamol.

Las definiciones de estas categorías se han obtenido a partir del análisis de las mismas así como de las definiciones que aporta de los términos la Real Academia Nacional de Medicina de España, partiendo de su diccionario de términos médicos [49]. Estas categorías se utilizan para definir los capítulos del archivo final, que se explicarán en el capítulo de visualización. Además, sirven para agrupar las entidades relacionadas por tipo, de tal manera que aparecen en el orden de las categorías en el que se han descrito, mientras que

dentro de cada categoría las entidades están ordenadas de mayor a menor por la fuerza, la seguridad, de la relación.

La realización de esta agrupación por tipos de entidades fue realizada por la Asociación COpenMed de forma previa a la realización del trabajo y está ligada al archivo Excel de manera que el número de entidades así como los tipos de entidades recogidos dependen de este archivo. Es relevante tener en cuenta que los tipos y entidades considerados aumentan progresivamente en la Asociación, por lo que en este trabajo se han considerado aquellos presentes cuando se inció la implementación.

## 5.1. Agrupación por tipo

Durante la definición de las categorías, se ha explicado la función general de algunas de ellas, que actúan como causas, consecuencias, observable, etc. A lo largo de esta sección se profundiza en las funciones de estas categorías, se las engloba en grupos más amplios y se explica su funcionalidad en el código.

A lo largo de este trabajo, se realizan agrupaciones por tipo de entidad en varias ocasiones. En función de las categorías explicadas anteriormente se definen los capítulos del documento final. Así mismo, se utilizan para dividir las razones vinculadas con una entidad, de tal forma que se organizan en un primer nivel por categoría y en un segundo nivel por la seguridad de la relación.

Por otro lado, se agrupan en conjuntos más grandes por varios motivos: en primer lugar, se establecen funciones comunes para las categorías que requieren la extracción de las mismas características, el otro motivo es que la función printList permite devolver solo un tipo concreto de entidad, por ejemplo, la función que extrae los tests, sólo acepta como tipo de entidad aquellos que pertenecen a la categoría de tests.

Pasando a explicar el segundo motivo descrito, las categorías de entidades se agrupan a su vez en cinco clases superiores: causas, tratamientos, enfermedades, pruebas y entidades descriptivas. Como se puede observar, algunas de estas clases coinciden con las categorías explicadas anteriormente aunque en este caso incluyen más tipos de entidades. También se verá, que algunas categorías de entidades están contenidas en varias de estas clases.

Dentro de la clase de causas, encontramos enfermedades, tratamientos, grupos de enfermedades, pruebas, grupos de tratamientos, riesgos, sustancias, patógenos, actividades, síntomas, causas, grupos de pruebas y grupos de sustancias. Todas estas categorías pueden servir como causa de otras, incluso del mismo tipo. Esta clase se usa como tipos válidos a la entrada de la extracción de causas.

Para la clase de tratamientos, encontramos las categorías de tratamientos, grupos de tratamientos, sustancias, grupo de sustancias, actividades e instrumentos. Esta categoría se utiliza como válida para las funciones de tratamientos aplicadas a otras entidades, es decir, del tipo “se trata con”. Algunas de estas entidades también se pueden usar como

medidas de prevención, pero en este caso no se restringen los tipos de entidad devuelta.

Para las enfermedades, se incluyen las propias enfermedades, los grupos de enfermedades, los patógenos, actividades, causas, síntomas y condiciones. Esta categoría se aplica a la función de tratamientos cuando se aplica para el tipo "sirve para tratar", así mismo junto con las causas, se aplica a la función de pruebas del tipo "sirve para diagnosticar". La clase de pruebas recoge las pruebas, los grupos de pruebas, los resultados de tests y los instrumentos. Se usa para la función de extracción de tests del tipo "se diagnostica con".

Por último, la clase de categorías descriptivas contiene anatomía, población, función fisiológica, especialidad, función e instrumento. Esta clase se utiliza para identificar estas categorías que requieren información pero no tanto relaciones de causalidad. Se utilizan para la extracción de características observables de una entidad.

Pasando a explicar la primera razón para dividir las categorías en grupos, se utiliza para organizar las funciones de extracción de características en función de las necesidades de cada categoría. Se organizan en los siguientes grupos:

- Causas no descriptivas: contiene enfermedades, grupos de enfermedades, causas, riesgos, condiciones y patógenos.
- Tratamientos: contiene tratamientos, grupos de tratamientos, sustancias y grupos de sustancias.
- Pruebas: recoge tests, grupos de tests e instrumento/equipo.
- Resultados de pruebas.
- Categorías descriptivas: contiene anatomía, población, características fisiológicas y funciones.
- Síntomas.
- Especialidades.

El primer grupo contiene tipos de causas no descriptivos, que a su vez tienen causas y consecuencias. En este grupo se engloban: enfermedades, grupos de enfermedades, causas, riesgos, condiciones y patógenos. De este tipo de entidades se extraen los sinónimos, la descripción, entidades similares, formas de prevención, entidades observables, causas y entidades que no pueden ser la causa, consecuencias y entidades que no pueden ser consecuencias, tratamientos indicados y contraindicados, pruebas de diagnóstico así como pruebas evitables, entidades a las que prestar especial atención, especialidad desde la que se estudian y otros recursos para conocer más.

A nivel de función, se obtiene de la siguiente forma:

```
#summarize 1 para los tipos de enfermedades/síntomas
def summarize1(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d.%s.\n"%(idEntity,mainName)
    latex.writeSection(idEntity,title)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addPrevention(idEntity)
    self.addObservables(idEntity)
    self.addCauses(idEntity)
    self.addNotCauses(idEntity)
    self.addConsequences(idEntity, "Puede provocar")
    self.addNotConsequences(idEntity, "No provoca")
    self.addTreatments(idEntity)
    self.addCounterTreatments(idEntity)
    self.addTests3(idEntity)
    self.addAvoidableTests(idEntity)
    self.addAttention(idEntity)
    self.addNegativeAttention(idEntity)
    self.addSpecialty(idEntity)
    self.addResources(idEntity)
    #return retval
```

Fig. 5.1. Extracción de características causas no descriptivas

Este grupo fue definido por COpenMed de forma previa a la realización del proyecto, aunque se modificaron algunas entidades. Por ejemplo, al comienzo del trabajo los síntomas se agrupaban con las enfermedades. Las modificaciones se realizaron durante la verificación de resultados, al ver que algunas características no eran apropiadas.

El segundo grupo contiene entidades de tipo tratamientos, que tienen consecuencias pero no causas. Las entidades incluidas son: tratamientos, grupos de tratamientos, sustancias y grupos de sustancias. Las características que se extraen de este tipo de entidades son: sinónimos y descripción, que se aplican a todos los grupos, entidades similares, que permite encontrar tratamientos similares. Incluye también una función de tratamientos del tipo “sirve para tratar” que devuelve las entidades a las que se aplican estos tratamientos. Extrae además las consecuencias de los mismos y el equipo utilizado. Se devuelven también las relaciones positivas y negativas a las que prestar atención y por último recursos que consultar para conocer más información. A nivel de función, se obtiene de la siguiente forma:

```
#summarize 2 para tratamientos
def summarize2(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s."%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addTreatments2(idEntity, "Sirve para tratar o puede causar")
    self.addConsequences(idEntity, "Sirve para tratar o puede causar")
    self.addEquipment(idEntity)
    self.addAttention(idEntity)
    self.addNegativeAttention(idEntity)
    self.addSpecialty(idEntity)
    self.addResources(idEntity)
```

Fig. 5.2. Extracción de características de tratamientos

Al igual que el grupo anterior, esta segunda categoría también estaba definida antes del comienzo del proyecto. En este caso no se modificaron los tipos de entidades recogidos, puesto que las respuestas se adecuaban a las entradas. Los siguientes grupos que se describirán fueron desarrollados durante el proyecto.

El tercer grupo se utiliza para devolver las características de pruebas, agrupa los siguientes tipos de entidades: test, grupo de test e instrumento/equipo. De nuevo incluye los sinónimos y la descripción así como entidades similares. Se extraen también las relaciones del tipo “sirve para diagnosticar” que explora las relaciones ascendentes desde los tests. Además se incluyen las relaciones negativas de tipo “no sirve para diagnosticar”. Por otro lado, se extrae también el equipo utilizado, las entidades a las que prestar atención y otros recursos para obtener más información. A nivel de función, se obtiene de la siguiente forma:

```
#summarize 3 para las pruebas
def summarize3(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s."%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addTests(idEntity)
    self.addAvoidableTests(idEntity)
    self.addConsequences(idEntity, "Puede causar")
    self.addEquipment(idEntity)
    self.addAttention(idEntity)
    self.addNegativeAttention(idEntity)
    self.addSpecialty(idEntity)
    self.addResources(idEntity)
```

Fig. 5.3. Extracción de características de pruebas

El cuarto grupo se utiliza exclusivamente para los resultados de las pruebas, esto es porque es el único tipo de entidad con estas características. En esta función se extraen los sinónimos, descripción y entidades similares. Así mismo, se extraen las causas y las pruebas que llevan a este resultado. A diferencia de las otras funciones de extracción de pruebas, en esta se exploran las relaciones en ambas direcciones (arriba y abajo) del grafo

Se incluyen además los tests desde los que no parte el resultado, características a las que prestar atención y recursos extra, así como la especialidad a la que pertenecen. A nivel de función, se puede ver en la figura 5.4:

```
#summarize 4 para los resultados
def summarize4(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s.\n"%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addCauses(idEntity)
    self.addTests2(idEntity)
    self.addAvoidableTests(idEntity)
    self.addAttention(idEntity)
    self.addNegativeAttention(idEntity)
    self.addSpecialty(idEntity)
    self.addResources(idEntity)
```

Fig. 5.4. Extracción de características de resultados

El quinto grupo se aplica a categorías descriptivas, es decir, que contienen información, pero la mayoría de sus relaciones no son causales. Incluye los siguientes tipos de entidades: anatomía, población, característica fisiológica y función. De estos tipos de entidades se extrae su descripción, sinónimos, entidades similares, equipo utilizado en ellas, entidades que se pueden observar en estas otras, tratamientos indicados y contraindicados para aplicar a las mismas, la especialidad a la que pertenecen y recursos extra. A nivel de función, se obtiene de la siguiente forma:

```
#summarize 5 para las categorías descriptivas
def summarize5(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s."%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addEquipment(idEntity)
    self.addObservables(idEntity)
    self.addTreatments(idEntity)
    self.addCounterTreatments(idEntity)
    self.addSpecialty(idEntity)
    self.addResources(idEntity)
```

Fig. 5.5. Extracción de características de características descriptivas

El sexto grupo se utiliza para extraer las características de los síntomas. Inicialmente este tipo de entidad se había integrado con el primer grupo, sin embargo a lo largo del desarrollo del trabajo se vio que es una entidad que no es en sí mismo una causa, sino que, aunque puede indicar la causa de otra entidad, sirve realmente como indicador. Para este tipo de entidad se extraen los sinónimos, descripción y entidades similares. Así mismo se devuelven las causas y entidades que no pueden ser causas, tratamientos indicados

y contraindicados para tratar el síntoma, entidades donde se puede observar el síntoma, equipo utilizado y pruebas, si las hay, para comprobarlo. Se devuelven también entidades a las que prestar atención y recursos para obtener más información. A nivel de función, se puede ver su extracción en la figura 5.6:

```
#summarize 6 para los síntomas
def summarize6(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s."%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addCausesUp(idEntity)
    self.addNotCauses(idEntity)
    self.addTreatments(idEntity)
    self.addCounterTreatments(idEntity)
    self.addObservables(idEntity)
    self.addEquipment(idEntity)
    self.addAvoidableTests(idEntity)
    self.addAttention(idEntity)
    self.addNegativeAttention(idEntity)
    self.addResources(idEntity)
```

Fig. 5.6. Extracción de características de síntomas

El último grupo se utiliza para las especialidades. Inicialmente se había incluido con las categorías descriptivas puesto que es así como se comporta. Sin embargo en esa categoría se decidió incluir la especialidad a la que pertenecen mientras que, al explorar estas relaciones en una especialidad, podía devolver todas las entidades que pertenecen a dicha entidad, lo que, por el tamaño de esa lista, sería un volumen de datos muy grande para leer y fallaría al objetivo de facilitar la lectura. Para este tipo de entidad se incluyen los sinónimos, la descripción, entidades similares, equipo, entidades observables y recursos. Esto nos permite tener una muestra de las entidades que pertenecen a una especialidad sin necesitar la lista completa. A nivel de función, se obtiene de la siguiente forma:

```
#summarize 7 para la especialidad
def summarize7(self, idEntity, typeName):
    mainName = self.entity_dict[idEntity][0]
    title = "%d. %s."%(idEntity,mainName)
    latex.writeSection(title,idEntity)
    self.addSynonyms(idEntity, mainName)
    self.addDescription(idEntity)
    self.addSimilar(idEntity)
    self.addEquipment(idEntity)
    self.addObservables(idEntity)
    self.addResources(idEntity)
```

Fig. 5.7. Extracción de características de especialidades

Todos estos grupos son llamados por una función común que, después de comprobar que la entidad está presente en el diccionario y que el tipo de entidad es el que se pretende de volver, obtiene el grupo al que pertenece dicha entidad y llama a la función correspondiente. Esta función se refleja de la siguiente forma:

```
def summarize(self, idEntity,entity_type):
    if idEntity not in self.entity_dict:
        print( "La entidad no está presente" , idEntity)
        return ""

    entityType = self.entity_dict[idEntity][1]
    typeName = self.entity_dict[idEntity][2]
    if typeName != entity_type:
        return

    if entityType == self.id_dict['Disease']:
        return self.summarize1(idEntity, "ENFERMEDAD")
    elif entityType == self.id_dict['GroupOfDiseases']:
        return self.summarize1(idEntity, "GRUPO DE ENFERMEDADES")
    elif entityType == self.id_dict['Cause']:
        return self.summarize1(idEntity, "CAUSA")
    elif entityType == self.id_dict['Risk']:
        return self.summarize1(idEntity, "RISK")
    if entityType == self.id_dict['Condition']:
        return self.summarize1(idEntity, "CONDICIÓN")
    elif entityType == self.id_dict['Pathogen']:
        return self.summarize1(idEntity, "PATOGENO")
```

Fig. 5.8. Función común de resumen

Como se observa en la figura 5.8, las primeras líneas comprueban que la entidad está presente en el diccionario de entidades, si no es así se imprime en la consola el identificador de la entidad que se ha querido extraer con el mensaje indicado. Al trabajar con un archivo cambiante, este tipo de mensajes sirven para comprobar que se está trabajando con la versión deseada del archivo y, si el archivo es correcto pero la entidad no está presente, que se ha realizado la ejecución pertinente en el código *copenmed tools*.

Antes de comenzar con estre Trabajo de Fin de Grado, la categoría de causas no descriptivas ya estaba definida, así como la categoría de tratamientos, aunque incluían algunos tipos de entidades distintos, como por ejemplo síntomas, que se encontraba en la categoría de causas. A lo largo de este trabajo se han creado el resto de categorías y se han modificado en función de los nuevos tipos de entidad presentes en el archivo Excel de entrada del código.

# 6. VERIFICACIÓN DE RESULTADOS

Como se ha mencionado en la metodología, este trabajo se ha desarrollado de forma iterativa, generando un modelo funcional, aunque incompleto, en cada etapa y añadiendo funcionalidades a medida que avanzaba el proyecto. En este capítulo se describe la verificación de resultados llevada a cabo en cada etapa del proceso y se ordena en las iteraciones llevadas a cabo.

Para verificar los resultados de los diferentes pasos, se ha organizado de la siguiente manera:

- Verificación de las funciones
  - Funciones de diccionarios
    - Función de recursos
    - Función de descripciones
    - Función de recursos
  - Funciones de exploración del grafo
    - Funciones de propagación
    - Función *Print List*
    - Función *Print Path*
- Verificación de resultados
  - Relaciones a distancia uno
  - Relaciones a más distancia
  - Pertinencia de las respuestas a las agrupaciones

## 6.1. Verificación de funciones

La primera fase del desarrollo de este proyecto fue tras la implementación de las funciones. Como se ha explicado, estas se desarrollaron en dos etapas: en primer lugar aquellas cuya información está almacenada en diccionarios asociados a la entidad y en segundo lugar aquellas que dependían de la exploración del grafo.

Antes de empezar a adaptar las funciones a los tipos de entidad, se aseguró que estaban devolviendo la información correcta, así como que la función realizaba todas las comprobaciones necesarias. Para las funciones que acceden a los diccionarios, estas comprobaciones se realizaron accediendo individualmente a cada uno de los parámetros utilizados por la función y comprobando, al aplicar la función a entidades concretas, que se recorre

de manera adecuada el diccionario. Estas son las funciones explicadas en la sección 4.5.1: las funciones de extracción de características.

Para la función de recursos, era necesario comprobar que se accedía al diccionario correctamente. Además, al considerar por separado las imágenes también se debía comprobar de qué tipo de recurso se trataba. En este caso la función solo accede a un diccionario, y recorre todas las filas asociadas a la entidad de entrada para devolver los recursos. Este diccionario no cuenta con un parámetro que indique el tipo de recurso por lo que, para comprobar que se trataba de una imagen se verifican las terminaciones de los enlaces y si coinciden con las de una imagen (.jpg, .png, .tiff, etc) se clasifican como tal. Las entidades que más imágenes contienen son las de tipo anatomía, por lo que la comprobación de esta división se realizó extrayendo las características de este tipo de entidades.

De esta manera, si se trata de un URL se devuelve de la siguiente manera:

```
Para saber más:
  https://www.msdmanuals.com/es/hogar/trastornos-otorrinolaringol%C3%B3gicos/biolog%C3%ADa-de-los-o%C3%ADdos-la-nariz-y-la-garganta/garganta
  https://es.wikipedia.org/wiki/Garganta
```

Fig. 6.1. Ejemplo de URL

Mientras que si se trata de una imagen, también queda indicado, como se ve en la figura 6.2:

```
Para saber más:
  https://es.wikipedia.org/wiki/Pie

Puede ver la siguiente imagen:
  https://i.pinimg.com/originals/6e/3a/09/6e3a09586efb2a90710f8279a6487e10.jpg
```

Fig. 6.2. Ejemplo de enlace a una imagen

En el caso de la función de descripción, también se poseía un diccionario único. Además, en este caso no era necesario hacer comprobaciones de parámetros puesto que no era necesario dividir en subcategorías. Las pruebas de esta función se hicieron aplicándola a diferentes entidades. Al estar el archivo escrito por personas distintas, esta función sirvió para detectar errores en el formato de las descripciones asociadas a diversas entidades.

Para acabar con las funciones de diccionarios, se realizaron también verificaciones en la función de extracción de sinónimos, en cuyo caso era necesario comprobar el nivel, es decir, el grado de tecnicismo del sinónimo, y el idioma. Para comprobar el tecnicismo del sinónimo se comprueba un parámetro llamado nivel, con números 0, 1 y 2. En este caso la verificación se hizo extrayendo sinónimos de varias entidades y comprobando a través del diccionario que la categoría asignada a los mismos correspondía con su nivel asignado. Para el idioma se trabajó de manera similar, pero comprobando el identificador de idioma. En este caso solo se separa en inglés porque actualmente, aparte del castellano, es el único otro idioma que se maneja. En el futuro, si aumenta el número de idiomas manejado por

la asociación, sería recomendable añadir comprobaciones de los identificadores de los demás idiomas para clasificarlos correctamente.

Por ejemplo, en presencia de un sinónimo de alto nivel, técnico, aparece de la siguiente forma:

```
ENFERMEDAD. 2848. Anorexia (enfermedad).
Conocido de forma técnica como: Anorexia nerviosa
```

Fig. 6.3. Ejemplo de sinónimo técnico

Mientras que si aparecen varias categorías se ve de la siguiente forma:

```
ENFERMEDAD. 18. Alergia al polen.
  Conocido de forma técnica como: Astenia, Polinosis
  También conocido como: Fiebre del heno, Alergia primaveral
```

Fig. 6.4. Ejemplo de varios sinónimos

En la tabla 6.1 se muestra un resumen de las comprobaciones realizadas por este tipo de funciones:

TABLA 6.1. VERIFICACIONES DE LAS FUNCIONES DE DICCIONARIO

| **Función** | **Comprobaciones** | | |
|---|---|---|---|
| Recursos | Acceso al diccionario | Recorrido completo | Imagen VS URL |
| Descripción | Aplicación correcta a entidades | | |
| Sinónimos | Acceso al diccionario | Nivel (técnico/común) | Idioma (español/inglés) |

Pasando ahora a las funciones de exploración del grafo, es importante tener en cuenta que utilizan las funciones de propagación, *Print List* y *Print path*. En este caso la verificación de resultados se ha realizado individualmente en cada una de las funciones internas, ya que la estructura de todas las funciones de extracción de características es la misma.

La primera función a evaluar fue la propagación, en este caso, dentro de la propia función se evalúan la distancia de camino y el límite de seguridad de las relaciones. Se ha discutido anteriormente que estos valores se establecieron en dos para la distancia y 0,25 para el límite. El factor a considerar al aplicar estas funciones a las características es la dirección de la propagación, hacia arriba, hacia abajo o en ambas direcciones. En este caso, la evaluación se realizó para cada una de las funciones, en ocasiones desarrollando todas las opciones para aplicarlas a diferentes grupos de entidades. Por ejemplo, para las funciones de pruebas se desarrollaron tres opciones: la propagación hacia arriba devuelve relaciones del tipo “sirve para diagnosticar“ y se relaciona con las pruebas, la propagación hacia abajo devuelve relaciones del tipo “se diagnostica con” y se aplica a las

enfermedades y entidades de su grupo. Además se decidió desarrollar una tercera función con propagación en las dos direcciones para aplicar a los resultados.

La verificación de este tipo de funciones, se realizó a base de comparar resultados de la propagación en distintas direcciones. A nivel teórico es posible saber en qué dirección del grafo se debe propagar una función para obtener los resultados requeridos. Por ejemplo para encontrar las causas de una entidad se propaga hacia arriba desde la hoja de consecuencias, "si yo soy la consecuencia de una entidad, esa entidad es mi causa". Sin embargo, en funciones como los observables o los tratamientos donde no es tan intuitivo se realizaron pruebas en varias entidades del mismo tipo para determinar qué tipo de propagación aporta los resultados más relevantes.

Para la función *Print list* así como para *Print path*, las comprobaciones se realizaron de manera conjunta, ya que devuelven un resultado común a la extracción de características. Al llamar a la función de impresión de la lista, lo primera comprobación que realiza es que la entidad esté presente en el diccionario y que cumpla las condiciones. A lo largo del trabajo, se ampliaron estas condiciones para que la entidad devuelta fuera además del tipo requerido, lo que permitiría ordenar las respuestas por tipo de entidad. Para verificar la corrección de estos resultados se comprobó que la función validaba correctamente la condición y accedía al parámetro adecuado del diccionario. Estas comprobaciones determinan si la entidad se debe enseñar o no, en el caso de que así sea se pasa a la segunda parte de la función.

En la función, estas comprobaciones se ven de la siguiente forma:

```
for idx in idxSorted:
     if not idx in self.entity_dict:
        continue
     if y[idx]>threshold and idx!=idEntity:
        show = False
        if validTypes is not None:
            if self.entity_dict[idx][1] in validTypes \
                and self.entity_dict[idx][2]==entity_type:
                show = True
        elif sameType==0 and \
            self.entity_dict[idx][2] == entity_type:
            show = True
        elif sameType==1 and \
             self.entity_dict[idEntity][1]==self.entity_dict[idx][1]\
                 and self.entity_dict[idx][2]==entity_type:
            show = True
        elif sameType==-1 and \
            self.entity_dict[idEntity][1]!=self.entity_dict[idx][1]\
                and self.entity_dict[idx][2]==entity_type:
            show = True
```

Fig. 6.5. Comprobaciones en *Print Path*

Tras determinar que la entidad debe enseñarse, la función obtiene el camino desde la entidad inicial hasta la que se está tratando, propagando hacia arriba y hacia abajo. En este caso, se determinó que los caminos de longitud mayor que uno cuyos nodos intermedios tuvieran más de veinte relaciones de salida no eran relevantes para la entidad y no debían incluirse. Esto es así porque entidades muy genéricas como fiebre o infección bacteriana

se relacionan con mucha seguridad con otras entidades que pueden no ser relevantes para la entidad inicial, ya que cubren aspectos muy amplios. Por ello se utilizó el atributo *outgoing edges* que existe para todos los nodos del grafo:

```
for weight, idEnt, idEdgeType, edgeStrength, idEdge in path.path:
    edges = 0
    if idEnt is not None:
            current_node = self.graph.getNode(idEnt)
            #necesito contar los edges del nodo intermedio
            edges += len(current_node.outgoingEdges)
            if edges > 20:
                delete = True

if delete == True:
    continue
else:
    if idx not in shownEntities:
        print("Path Down",path.pretty(self.entity_dict, self.edge_type_dict))
        string = "down"
        reasonList.append(self.printPath(path, string))
        shownEntities.append(idx)
```

Fig. 6.6. Relaciones de salida de la entidad

En este punto, fue relevante comprobar que la condición se estaba evaluando para evitar respuestas genéricas, una vez comprobada, la función llama a *Print path*. Tras los cambios realizados en la estructura del código durante la implementación final de la visualización, esta función devuelve una lista de listas con cada razón, donde la primera lista contiene el identificador de la entidad final, la segunda lista contiene el camino recorrido desde la entidad de partida hasta la de llegada y la tercera el peso final de la relación.

Para la función de *Print path*, el paso más importante es comprobar que el cálculo de la seguridad final de la relación estaba bien calculado. Este valor se calcula multiplicando el peso de cada relación por la que pasa el camino y se comprobó realizando los cálculos por separado y comparando con los resultados del programa. Además se comprobó que los identificadores de inicio y final del camino se dieran correctamente y la lista no se sobreescribiera en cada paso del camino. Se puede ver el resultado en la figura 6.7:

```
#print path
def printPath(self,path, string):
    idOrigin = path.originEntity
    retval = []
    ret_edgeStrength = 1
    for weight, idEntity, idEdgeType, edgeStrength, idEdge in path.path:
        if idEntity is not None:
                idFinal = idEntity
                retval.append((idOrigin,idFinal,\
                                self.edge_type_dict[idEdgeType][3],
                                                edgeStrength))
                ret_edgeStrength *= edgeStrength
                idOrigin = idFinal
    return string, retval, ret_edgeStrength
```

Fig. 6.7. Función *Print Path* resultante

Al volver a la función de la lista, esta se ordena en función de los pesos devueltos como el tercer parámetro de esta función y se devuelven a la función de características.

Al volver a esta función, fue necesario comprobar el contenido de la lista de forma manual, para asegurar que no estaba vacía y que todas las entidades habían sido añadidas antes de incluir un título para la subsección. Una vez realizado esto, se finalizaron las verificaciones de las funciones.

A continuación se muestra el resumen de las verificaciones de estas funciones en la tabla 6.2:

TABLA 6.2. VERIFICACIONES DE LAS FUNCIONES DE PROPAGACIÓN, LISTA Y CAMINO

| **Función** | **Comprobaciones** | | |
|---|---|---|---|
| Propagación | Distancia propagada | Límite de seguridad | Dirección de propagación |
| *Print List* | Entidad presente | Número de conexiones | Lista final |
| *Print Path* | Cálculo del peso del camino | Camino devuelto | |

## 6.2. Verificación de resultados

Una vez verificadas individualmente las funciones, se pasó a evaluar el resultado común de las mismas y su aplicación a los distintos tipos de entidades. Es a lo largo de este proceso cuando se decidieron los grupos para la extracción de características de las entidades, así como el tipo de funciones a aplicar en cada uno.

Los pasos llevados a cabo en esta verificación han sido:

1. Comprobar agrupaciones realizadas
2. Para cada tipo de entidad comprobar:
    *a*) Relaciones a distancia uno
    *b*) Relaciones a más distancia
    *c*) Repetir comprobaciones para relaciones negativas

Los grupos quedaron definidos como se ha explicado en el apartado anterior después de realizar pruebas con todos los tipos de entidades. Como se ha explicado anteriormente, en un primer momento los síntomas estaban en el mismo grupo que las enfermedades. Sin embargo, tras la realización de pruebas se vio que esta agrupación no era adecuada, ya que se trataban los síntomas como causas de enfermedades, no como indicadores. Por otro lado, los patógenos también se encontraban inicialmente en este grupo, dada la

relevancia de su descripción así como sus relaciones con otros patógenos o portadores. Se decidió comprobar la diferencia de pertinencia entre las respuestas situándolos también en el grupo de las entidades descriptivas. Al final se decidió mantenerlos con las entidades de tipo causas, porque con las entidades descriptivas no se devolvían posibles consecuencias de los mismos.

Por otro lado, es en este paso cuando se decidió separar la especialidad de otras entidades. La verificación de resultados de estas agrupaciones se realizó eligiendo al azar un mínimo de tres ejemplos de cada entidad, extrayendo sus resultados y analizando la pertinencia de los resultados. A lo largo de este proceso se modificaron algunas funciones de extracción de características que estaban teóricamente mal planteadas. Por ejemplo, en el caso de la extracción de causas, la propagación en la hoja de causas se estaba realizando en todas las direcciones solo en esa hoja. Sin embargo tras realizar pruebas se vio que era mejor cambiar la propagación de manera descendente en la hoja de causas e incluir la propagación ascendente en las consecuencias.

En este caso, para comprobar las entidades devueltas se consultó la página de la Asociación COpenMed [50], que contiene todas las relaciones de longitud uno de una entidad. Aunque estas relaciones no aparecen ordenadas, es posible comparar las entidades devueltas por el código con estas para encontrar diferencias. Un ejemplo de la visualización de las entidades en la página web es el siguiente:

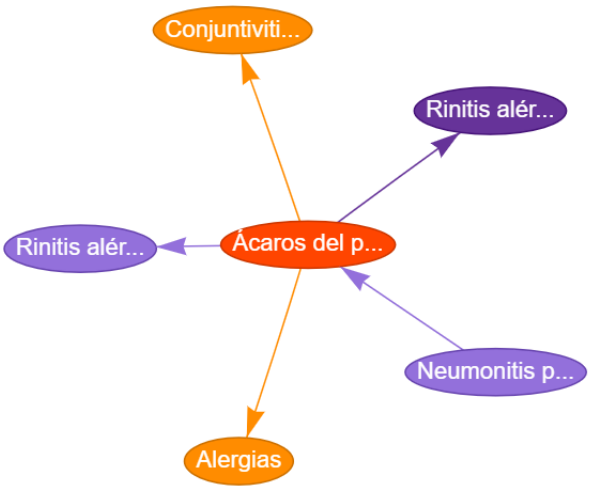



Fig. 6.8. Representación gráfica ácaros del polvo; Fuente: Asociación COpenMed

Además de la representación gráfica mostrada en la figura 6.8, se puede ver la tabla mostrada en la figura 6.9 con las asociaciones:

| Entidad | Tipo de Asociación ▼ | Fuerza | Descripción |
|---|---|---|---|
| *Alergias* | Cause can cause this Group | 0.3 | |
| *Conjuntivitis* | Cause can cause this Group | 0.5 | |
| *Rinitis alérgica* | Cause may cause Disease | 0.7 | |
| *Rinitis alérgica* | Cause may cause Symptom | 0.7 | |

Fig. 6.9. Tabla de relaciones de los ácaros del polvo; Fuente: Asociación COpenMed

A partir de está información, se comprobó si las funciones estaban devolviendo todas las entidades a distancia uno de la entidad inicial. Para analizar las entidades a mayor distancia, se prestó atención a la fuerza de las relaciones, si esta se mantenía por encima del límite se devolvía en el archivo. Para el caso mostrado de la página web, ácaros del polvo, el código devolvió las siguientes entidades:

- Observables: Alérgeno, goteo postnasal, rinitis alérgica, estornudo, líquido claro por la nariz, mocos, congestión nasal, coriza, picor nasal, ojos llorosos, tos, trasudado de líquidos, sinusitis, vasodilatación periférica, edema, picor de ojos, pérdida de olfato, liberación de mediadores de la inflamación, estrechamiento de los bronquios, ojos enrojecidos o inflamados, dolor de cabeza, sibilancias, hiperemia.
- Consecuencias: Presencia de objeto extraño en el ojo, rinitis alérgica, conjuntivitis, queratitis y alergias.

Como se puede ver, devuelve todas las entidades a distancia uno así como muchas entidades derivadas. En este caso, una de las entidades con las que tiene relación es la **Rinitis Alérgica**, esta entidad tiene muchas relaciones externas, se pueden ver estas relaciones en la figura 6.10:

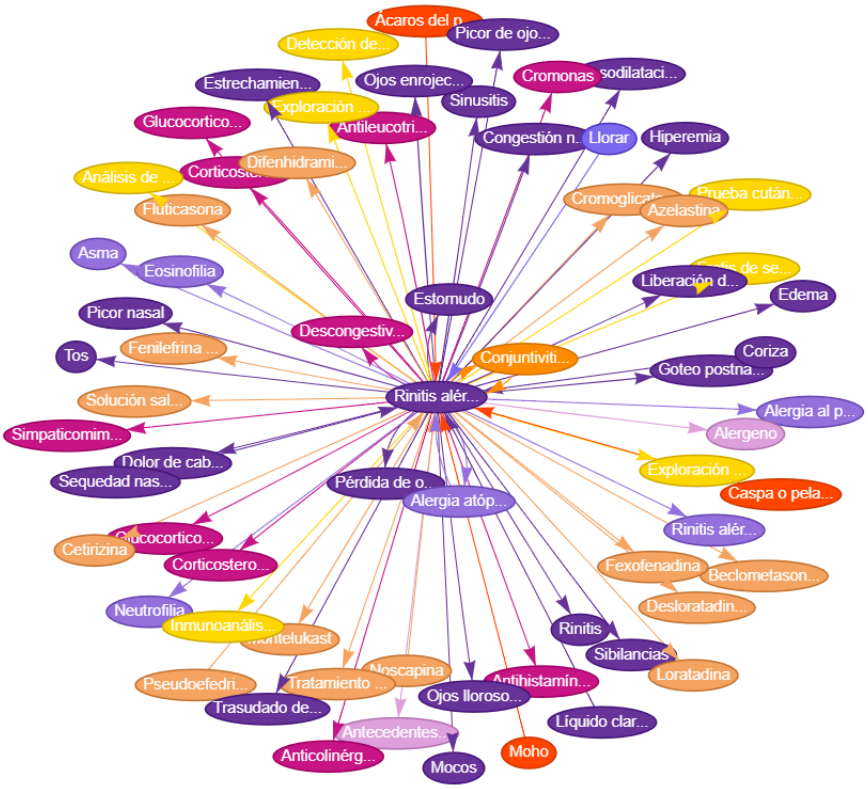



Fig. 6.10. Rinitis alérgica

Se ha explicado anteriormente, que para evitar que aparezcan relaciones demasiado generales, se establece un límite de relaciones de salida que pueden tener las entidades. En el caso de la Rinitis, se comprobó que poseía más enlaces de los indicados:

```
IPdb [4]: self.entity_dict[idEnt][0]
'Rinitis alérgica'

IPdb [5]: !next

IPdb [5]: edges
62
```

Fig. 6.11. Enlaces rinitis alérgica

Se puede ver en la figura 6.11 que la rinitis alérgica tiene 62 enlaces de salida asociados, por lo que relaciones derivadas de esta entidad no deben aparecer.

Como se ha explicado anteriormente, en el archivo Excel de entrada se consideran relaciones negativas y positivas, que se evaluan en las funciones de propagación. Esta diferenciación se realizó a lo largo del proyecto, no estando definida en un primer momento.

La verificación de las relaciones negativas, se realizó una vez terminada la validación de los grupos con relaciones positivas. Después de realizar los cambios en el archivo y adecuar las funciones de propagación a este tipo de entidades, se comprobó que efectivamente se devuelven relaciones negativas en el código. En este caso, la información

expuesta en la página web de la asociación sólo considera relaciones positivas, por lo que la verificación de estos resultados se realizó a mano, analizando la pertinencia de estas relaciones.

Una vez realizados experimentos para todos los tipos de entidad con relaciones tanto positivas como negativas y devuelto satisfactoriamente entidades pertinentes, se pasó a mejorar la visualización de los documentos.

# 7. VISUALIZACIÓN

## 7.1. Planteamiento inical

En los primeros estadios de este proyecto, cuando solo se devolvía información de los diccionarios de las entidades, la visualización de dicha información se realizaba desde la consola de Spyder. Sin embargo, a medida que se implementaron las funciones de extracción de características este método era insuficiente para visualizar la información.

Pasado este método, se decidió incluir la información en un archivo de texto. De esta manera, se diseñaron las funciones para devolver una cadena con la información obtenida. Este método se mantuvo durante la mayoría de la implementación y verificación de los resultados, por lo que se editaron las cadenas utilizando espacios y retornos de carro para facilitar la lectura lo máximo posible.

En la figura 7.1 se puede ver el esquema del documento generado inicialmente:

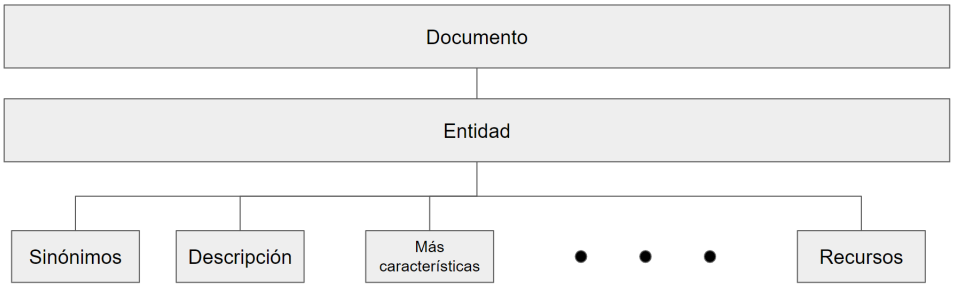



Fig. 7.1. Esquema inicial de visualización

De esta manera, se establecía un título con el nombre, tipo e identificador de la entidad y para cada sección se escribía el título, que normalmente es la frase que engloba las relaciones, por ejemplo el título de una función de tratamientos puede ser “Sirve para tratar:". Además los nombres de las entidades relacionadas en cada sección se colocaron con mayor indentación para indicar la importancia. Se puede ver en el ejemplo mostrado en la figura 7.2:

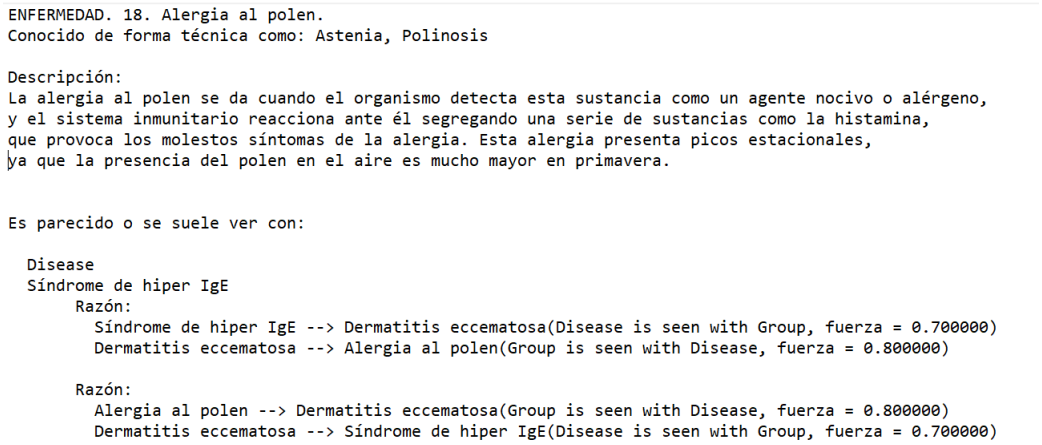



Fig. 7.2. Ejemplo de visualización: Alergia al Pólen

Con este planteamiento, se genera un documento para cada entidad, por lo que de cara a generar documentación de una gran cantidad de entidades, se vuelve un método poco lógico. Además, está formado a partir de cadenas simples, por lo que no se puede maquetar para por ejemplo realzar entidades relacionadas o minimizar los caminos.

Esta forma de visualización no es la más cómoda para manejar grandes volúmenes de información. La imposibilidad de maquetar el archivo con títulos o listas dificulta la lectura. Además, en este punto se devolvía información de una entidad cada vez, pero el objetivo del trabajo era elaborar un documento completo con datos de todas las entidades, que se haría muy difícil de tratar en formato de texto.

Por ello, se decidió cambiar el formato de visualización a uno que permitiera editar el texto así como manejar y leer de manera más cómoda la información.

## 7.2. Desarrollo del modelo final

Como se ha explicado, el objetivo final era generar un archivo conteniendo la información estructurada de todas las entidades del archivo de entrada. Para ello, se eligió realizar la visualización y maquetación del documento en LaTex. LaTeX es un sistema de composición de textos que permite obtener resultados de calidad profesional fácilmente. Se trata de un lenguaje de programación orientado a la generación de texto. La versión actual data de 1994 y ha sido precedida por varias versiones de TeX y LaTeX [51].

La aplicación de este lenguaje al trabajo se ha realizado a través de Spyder generando un documento tipo tex, para después compilarlo con un editor de LaTex, en este caso se ha utilizado TeXworks, una aplicación de uso libre diseñada para generar PDFs.

### 7.2.1. Implementación de las funciones

Para la estructura general del documento, se decidió usar un formato de libro. Como este archivo contiene entidades de todos los tipos, la primera función que se diseñó servía para dividir las entidades en tipos y generar así los capítulos del libro. Además, esta función se encarga de empezar y terminar el documento. La estructura general de la visualizaciónaparece mostrada en la figura 7.3:

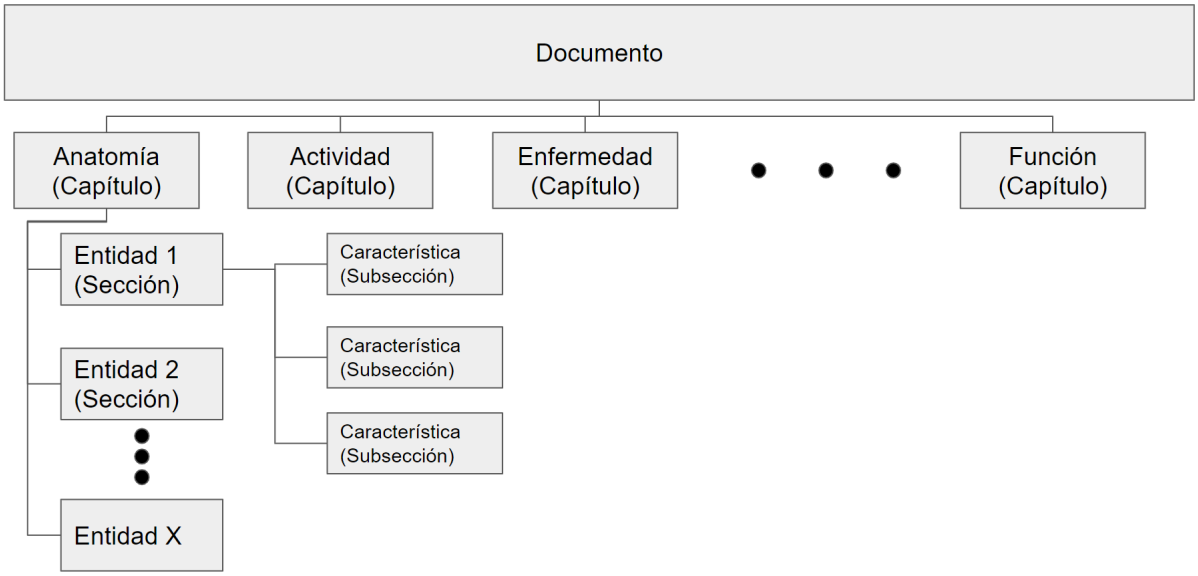



Fig. 7.3. Estructura general del documento

Para generar esta estructura, a parte de la función descrita, se implementaron funciones para generar las secciones, las subsecciones, incluir figuras y listar las razones vinculadas a cada entidad. Para hacer esto desde Spyder, el documento se genera en formato TeX, y las respuestas de cada función se incluyen en una cadena con el formato adecuado que se va escribiendo en el documento. Para guardar el formato de LaTeX, se llama primero a las funciones de inicio de sección o subsección y después a las relaciones con otras entidades. Un ejemplo de este tipo de funciones es la de inclusión de las razones, que se puede ver en la figura 7.4:

```
def writeReasons(self, idEntity, entity, reasonList):
    toWrite=\
        """
        \\underline{\\ref{id:%s} %s}
        \\begin{small}
        """%(idEntity,entity)
    toWrite += self.latexItemize(reasonList)
    toWrite +=\
        """
        \\end{small}
        """
    self.fh.write(toWrite)
    return
```

Fig. 7.4. Ejemplo de Función

Esta función recibe un identificador de entidad, el nombre de la misma y una lista de razones ordenadas. Se puede ver que estos datos se almacenan en una cadena, que tiene el formato asociado a LaTex, el comando *underline* sirve para subrayar el contenido entre corchetes, mientras que el comando *ref* establece una referencia entre el identificador y la ficha completa de la entidad. Esto permite acceder a las secciones de las entidades relacionadas con una identidad que se esté buscando si se quiere tener más información y perpetúa la conectividad entre entidades que aporta el uso de grafos de conocimiento.

Las funciones de subsección y razones son la misma para todas las funciones que devuelven una lista de razones, es decir, aquellas que exploran el grafo. Sin embargo, para las funciones de acceso al diccionario se han generado: una función que incluye el título de la subsección y la respuesta en la misma cadena para aplicar a los sinónimos, recursos y descripción. Así como una función para tratar las imágenes devueltas de los recursos.

Todo esto ha generado un cambio en la estructura global del código, que cambió su forma de funcionamiento para adaptarse a este archivo y se explicará a continuación.

### 7.2.2. Cambios en la estructura del código

En cuanto a los cambios en la estructura, inicialmente el archivo funcionaba de tal forma que toda la información se recopila en la primera función llamada, de esta forma, cada función devuelve una cadena con las relaciones, entidades o datos a devolver a la función anterior, y así hasta llegar a la primera función. De tal forma, la función print path devuelve la información a print list, que devuelve, pasó a las funciones de características, estas a la función del grupo de la entidad y finalmente a la función inicial.

Sin embargo, para escribir el documento,se cambió la estructura del código de tal forma que cada función que antes devolvía una cadena, pasa a enviar una lista o título a una función del documento. Uno de los cambios fundamentales se realizó en la función print List, que en primera instancia devolvía una cadena con el nombre de la entidad relacionada y sus razones, cada una de ellas con el camino asociado, pasa a devolver una lista de listas que para cada entidad consta de tres elementos: un indicador Up o Down en función del tipo de propagación aplicada para llegar a la entidad inicial, una lista de razones y la fuerza de cada asociación. Se puede ver el siguiente ejemplo del formato:

[('down', [(287, 491, 'Anatomy1 is related to Anatomy2', 1.0), (491, 6387, 'Anatomy1 is related to Anatomy2', 1.0)], 1.0), ('down', [(287, 1050, 'Anatomy1 is related to Anatomy2', 1.0), (1050, 5382, 'Anatomy1 is related to Anatomy2', 1.0)], 1.0) . . . ]

Finalmente, el resultado es una lista de listas con todas las entidades relacionadas y los elementos descritos anteriormente. Además tras la comprobación de esta lista, se descubrió que no se estaba comprobando si la entidad ya estaba incluída en la lista antes de añadirla, por lo que aparecían entidades repetidas, tras incluirse esta comprobación la función daba el resultado esperado. A continuación se puede ver la comprobación en el

código en la figura 7.5:

```
for path in node.reachableDown[idx]:
    if len(path.path)==1:
        if idx not in shownEntities:
            print("Path Down 1",path.pretty(self.entity_dict, self.edge_type_dict))
            string = "down"
            reasonList.append(self.printPath(path,string))
            shownEntities.append(idx)
```

Fig. 7.5. Comprobación de la presencia de entidades en la lista de razones

Para procesar este tipo de datos, se generó una función añadida que con la lista descrita anteriormente. Para cada entidad relacionada se añade su identificador de referencia, su nombre y la razón de la relación con nombres de identidades en vez de números. De tal manera que la estructura queda como se muestra en la figura 7.6:

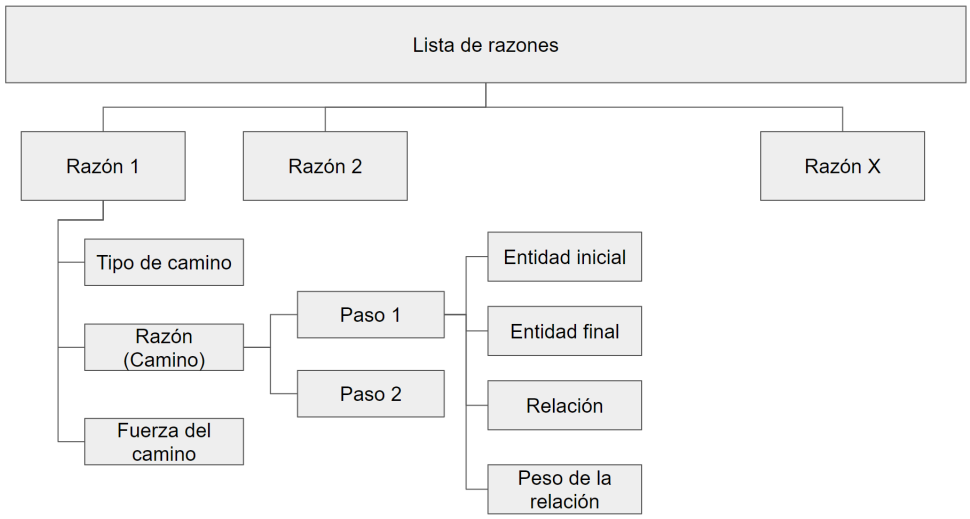



Fig. 7.6. Estructura de la lista de razones

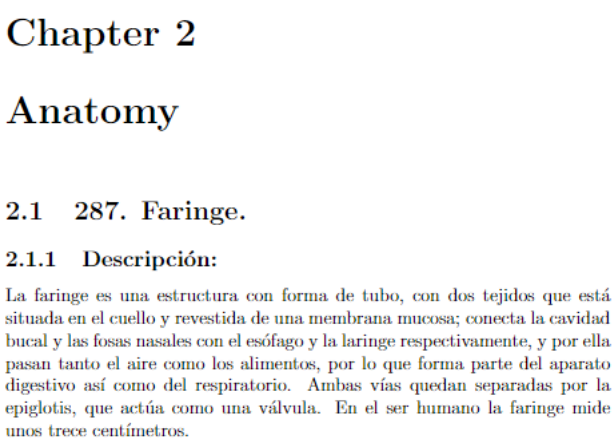
Para esto la función comprueba para cada elemento de la lista, el identificador de la primera entidad, si la propagación es descendente, o el último del camino si es ascendente. Si este identificador es el mismo que en el elemento anterior, se añaden las razones asociadas a la misma lista, puesto que se tratan de varias razones de la misma entidad. Si no es así, la función llama a la función write Reasons de LaTeX para escribir la lista en el documento y la vacía para añadir las entidades nuevas.

Con esta función se ordenan y escriben todas las razones asociadas a una entidad, además sirve para diferenciar a qué entidades está asociada cada razón y no mezclar las listas. Esta función envía el identificador de la entidad relacionada para servir de referencia, su nombre y la lista de razones que se utilizan en la función ejemplificada anteriormente.

## 7.3. Resultado

El resultado final es un archivo con un capítulo por cada tipo de entidad, contiene la información organizada de las casi diez mil entidades consideradas.

En la siguiente imagen, se puede observar el inicio del capítulo segundo, en este caso, todas las entidades se corresponden con la categoría de anatomía:



Chapter 2

Anatomy

2.1 287. Faringe.

2.1.1 Descripción:

La faringe es una estructura con forma de tubo, con dos tejidos que está situada en el cuello y revestida de una membrana mucosa; conecta la cavidad bucal y las fosas nasales con el esófago y la laringe respectivamente, y por ella pasan tanto el aire como los alimentos, por lo que forma parte del aparato digestivo así como del respiratorio. Ambas vías quedan separadas por la epiglotis, que actúa como una válvula. En el ser humano la faringe mide unos trece centímetros.

Fig. 7.7. Ejemplo de inicio de capítulo: Anatomía

Se puede ver que la primera entidad de este capítulo es la Faringe, con identificador número 287. Este identificador sirve como referencia (se utiliza el comando label) a esta ficha si se accede a la entidad desde una relación con otra. Atendiendo al código en LaTex, se puede ver la definición de la referencia de la siguiente manera:

```
\chapter{Anatomy}

    \section{\label{id:287}287. Faringe.}
        \subsection{Descripción: }
            La faringe es una estructura con forma de tubo, con dos tejidos que está situada en el cuello y revestida de una membrana mucosa;
            conecta la cavidad bucal y las fosas nasales con el esófago y la laringe respectivamente, y por ella pasan tanto el aire como los alimentos,
            por lo que forma parte del aparato digestivo así como del respiratorio. Ambas vías quedan separadas por la epiglotis, que actúa como una válvula.
            En el ser humano la faringe mide unos trece centímetros.
```

Fig. 7.8. Ejemplo de inicio de capítulo en LaTeX

Finalmente, para tener ejemplos de las referencias en fichas de otras entidades, se incluyen la siguiente figura:

### 2.1.2 Es parecido o se suele ver con:

2.2 Vena yugular

- Faringe – Cuello(Anatomy1 is related to Anatomy2, fuerza = 1.000000)
- Cuello – Vena yugular(Anatomy1 is related to Anatomy2, fuerza = 1.000000)

2.3 Histiocito

- Faringe – Amígdalas palatinas(Anatomy1 is related to Anatomy2, fuerza = 1.000000)
- Amígdalas palatinas – Histiocito(Anatomy1 is related to Anatomy2, fuerza = 1.000000)

Fig. 7.9. Entidades relacionadas: Faringe

En la figura se pueden ver entidades relacionadas con la Faringe, que poníamos de ejemplo al inicio del capítulo. En este caso es posible ver incluídos en un cuadrado rojo el número en cada entidad relacionada, esto hace significa que la entidad tiene una referencia cruzada a su propia ficha de información. En el código de LaTeX se ve de la siguiente forma:

```
\subsection{Es parecido o se suele ver con:}


  \underline{\ref{id:6387} Vena yugular}
  \begin{small}
  \begin{itemize}
      \item Faringe -- Cuello(Anatomy1 is related to Anatomy2, fuerza = 1.000000)
      \item Cuello -- Vena yugular(Anatomy1 is related to Anatomy2, fuerza = 1.000000)
  \end{itemize}
  \end{small}
```

Fig. 7.10. Entidades relacionadas en LaTeX: Faringe

Al pulsar en el número de la entidad, este lleva a la página en la que comienza la ficha de la entidad seleccionada:

## 2.2 6387. Vena yugular.

### 2.2.1 Descripción:

La vena yugular interna es una vena que recibe sangre del cerebro, cara y cuello. Comienza en el agujero yugular del cráneo como continuación del seno sigmoideo, desciende por el cuello y se une a la vena subclavia por detrás del extremo medial de la clavícula para formar las venas braquiocefálicas.

Fig. 7.11. Entidades relacionadas: Vena Yugular

La construcción de estas referencias se puede ver en la figura 7.3, en el ejemplo de función presentado, ya que es en esa función donde se establece la referencia a partir del número identificador de cada entidad. Ya que se han generado este tipo de referencias para todas las entidades del documento, en el resultado final todas las relaciones del grafo están reflejadas en estos cruces.

# 8. CONCLUSIONES

En este capítulo se exponen las conclusiones del trabajo, el grado de consecución de los objetivos propuestos y posibles vías de trabajo futuro

## 8.1. Conclusiones del trabajo realizado

A lo largo del trabajo, se han alcanzado los objetivos que se marcaron al comienzo del trabajo.

En primer lugar, se ha generado un grafo de conocimiento común a todas las entidades a partir de un archivo Excel. Además se han implementado funciones para recorrerlo de manera eficiente teniendo en cuenta la direccionalidad de las relaciones. Cumpliendo el primero de los objetivos particulares propuestos en el trabajo.

Cumpliendo el segundo objetivo, se han analizado distintas formas de recorrer grafos, como el algoritmo de Dijkstra, y se han expuesto distintas formas de estructurar información, como los algoritmos de clustering.

Por otro lado, se ha aplicado la lógica de estas relaciones para extraer características de entidades de información biomédica, considerando la relación entre la propagación en el grafo y el tipo de relación a extraer.

Además, se han analizado las características comunes de los diferentes tipos de entidades tratados, generando grupos con características similares y aplicándolos a las funciones de resumen, lo que concluye el cuarto objetivo.

Para finalizar, se ha generado utilizando un lenguaje de generación de textos (LaTeX) un documento coherente que mantiene la relación entre las entidades no sólo mostrando la información sino incluyendo también referencias cruzadas que permiten acceder a la información de todas las entidades relacionadas desde una inicial, cumpliendo así el objetivo quinto del trabajo, de generar un documento estructurado a partir de un código de estructuración automática de información.

Siguiendo el tercer objetivo propuesto, se ha conseguido así un proceso de estructuración automática de información que permite recorrer un grafo de conocimiento y producir un documento con los nodos y relaciones de dicho grafo de manera debidamente ordenada.

En cumplimiento del objetivo principal del trabajo, se ha conseguido generar un código capaz de recorrer un grafo desde una entidad de entrada para estructurar toda la información pertinente a esa entidad.

Finalmente, el desarrollo de una estructuración automática de la información de la Asociación COpenMed permite a los integrantes de la misma acceder a la información

organizada. A nivel interno, permite ver las relaciones entre entidades y comprobar, de cara a mejorar el grafo, si algunas de ellas tienen pocas relaciones o qué tipo de entidades es necesario añadir. A nivel externo, el trabajo permite a los usuarios, especializados o no, acceder a la información biomédica de forma cómoda.

Este trabajo ha permitido también aplicar conocimientos de programación y gestión de proyectos adquiridos en la carrera, así como la posibilidad de ampliar de manera autónoma conocimientos teóricos sobre grafos y las bases matemáticas de los mismos.

Por otro lado, la posibilidad de utilizar un lenguaje de edición de texto muy utilizado en el entorno profesional así como uno de los lenguajes de programación más utilizados ha sido de gran interés para un desarrollo futuro en un entorno laboral.

## 8.2. Limitaciones del Trabajo

Cómo se ha explicado a lo largo del trabajo, el código desarrollado para la estructuración automática de información se adecua a la información de entrada. Esto es, se ha desarrollado en base al archivo Excel provisto por la asociación COpenMed. Esta es una de las limitaciones principales del proyecto. Aunque teóricamente la forma de estructurar la información sería aplicable a varios escenarios, la realidad es que las estructuras de información están definidas para adaptarse al archivo concreto con el que se ha trabajado. El código es fácilmente escalable a una versión nueva del archivo producida por la asociación COpenMed, pero un cambio de formato provocaría fallos en el código.

Por otro lado, la forma de visualización elegida es, aunque adecuada al propósito, también una limitación. Dada la especificidad de los comandos de edición de texto en LaTeX, la maquetación del texto realizada solo sirve para visualizarse en este formato. Como se expondrá en el trabajo futuro, una de las vías deseables del desarrollo del proyecto sería su integración en una página web. En este sentido, el proceso de visualización debería realizarse de nuevo para adaptarse a este nuevo sistema.

Finalmente, se podría considerar que el tiempo y tamaño de los archivos manejados también es una limitación. Si bien se ha tratado de optimizar el código para tardar el menor tiempo posible en realizar la ejecución, realizar el documento completo de todas las entidades es una tarea costosa. A medida que crezca el número de entidades a resumir, esto podría limitar el equipo en el que se realiza esta tarea o alargar el tiempo requerido para la misma. Sin embargo, dado que solo requiere una ejecución para cada nueva versión del archivo de entidades, no se considera una gran limitación.

## 8.3. Trabajo futuro

En cuanto a posibles vías de trabajo futuro, es importante considerar que la asociación COpenMed ha seguido desarrollando de forma paralela su archivo de entidades, aumen-

tando tanto el número de entidades como los tipos. Por lo tanto, una de las primeras vías de desarrollo sería adaptar el código a nuevos tipos de entidades, así como incluir funciones para procesar los tipos de entidad que no estén considerados, extrayendo solo las características del diccionario o explicando que el tipo de entidad no ha sido considerado en el código.

Por otro lado, el documento generado sirve actualmente para uso interno de la asociación y así como para consulta en su página web. Una vía de trabajo para hacerlo más accesible sería vincular la información generada por el código, no con un documento, sino directamente a la página web de tal manera que se pudiera acceder a la información organizada sin acceder al pdf generado. En este caso, la visualización del código estaría orientada a HTML en vez de a LaTeX y los cambios en la página web se realizarían de manera más dinámica.

Otra posible vía de trabajo, sería ampliar los tipos de relaciones considerados por el grafo así como optimizar su búsqueda para obtener resultados más rápidos. También podría ser positivo generar un resultado visual de las relaciones obtenidas, en forma de grafo, para facilitar la comprensión de las relaciones.

Otras líneas de trabajo futuro serían considerar la eficacia de este método en comparación a métodos de clustering y generación automática de resúmenes expuestos en el estado del arte, así como aplicar este procedimiento de estructuración de información a otros campos más allá de la biomedicina.

# BIBLIOGRAFÍA


[1] EBSCO, *Medline complete: EBSCO*, 2021. [En línea]. Disponible en: https://www.ebsco.com/es/products/research-databases/medline-complete.

[2] A. Menéndez Velazquez, "Una breve introducción a la teoría de grafos," *SUMA*, vol. 28, pp. 11-26, 1998. DOI: http://hdl.handle.net/11162/13526.

[3] V. Meavilla Seguí, "El problema de los siete puentes de Königsberg," *Nueva revista de enseñanzas medias*, vol. 6, pp. 83-88, 1984. [En línea]. Disponible en: http://hdl.handle.net/11162/74591.

[4] E. Romero y J. Urbina Alonso, "El uso de redes complejas en economía: alcances y perspectivas," *INTERdisciplina*, vol. 5, p. 9, 2017. DOI: 10.22201/ceiich.24485705e.2017.12.61462.

[5] Luisyep, *El problema de los puentes de königsberg*, 2019. [En línea]. Disponible en: https://ingenieriabasica.es/el-problema-de-los-puentes-de-konigsberg/.

[6] H. Paulheim, "Knowledge graph refinement: A survey of approaches and evaluation methods," *Semantic Web*, vol. 8, n.º 3, pp. 489-508, 2016. [En línea]. Disponible en: http://semantic-web-journal.net/system/files/swj1167.pdf.

[7] L. Ehrlinger y W. Wöß, "Towards a Definition of Knowledge Graphs," *Institute for Application Oriented Knowledge Processing*, vol. 1695, 2016. [En línea]. Disponible en: http://ceur-ws.org/Vol-1695/paper4.pdf.

[8] A. Singhal, *Introducing the knowledge graph: Things, not strings*, 2012. [En línea]. Disponible en: https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[9] M. Á. García, *DBPedia del Español*, es.dbpedia.org, 2020. [En línea]. Disponible en: https://es.dbpedia.org/.

[10] Wikidata, *Main page*, 2019. [En línea]. Disponible en: https://www.wikidata.org/wiki/Wikidata:Main_Page.

[11] T. Saorín y J.-A. Pastor-Sánchez, "Wikidata y DBpedia: viaje al centro de la web de datos," *Anuario ThinkEPI*, vol. 12, pp. 207-214, 2018. DOI: https://doi.org/10.3145/thinkepi.2018.31.

[12] R. Asale, *Ontología: Diccionario de la Lengua Española*, 2022. [En línea]. Disponible en: https://dle.rae.es/ontolog%C3%ADa.

[13] M. Rico, *Mostrando las clases de DBPedia*, 2018. [En línea]. Disponible en: https://es.dbpedia.org/wiki/Wiki.jsp?page=Mostrando+las+clases+de+DBpedia.

[14] E. Prud'hommenaux y A. Seaborne, *SPARQL Lenguaje de consulta para RDF*, 2008. [En línea]. Disponible en: https://skos.um.es/TR/rdf-sparql-query/.

[15] R. Working Group, *Resource Description Framework (RDF): Concepts and Abstract Syntax*, 2014. [En línea]. Disponible en: https://www.w3.org/TR/2004/REC-rdf-concepts-20040210/#dfn-URI-reference.

[16] N. Chah, *OK Google, What Is Your Ontology? Or: Exploring Freebase Classification to Understand Google's Knowledge Graph*, 2018. DOI: 10.48550/ARXIV.1805.03885.

[17] X. Li, G. Tur, D. Hakkani-Tür y Q. Li, "Personal knowledge graph population from user utterances in conversational understanding," en *2014 IEEE Spoken Language Technology Workshop (SLT)*, 2014, pp. 224-229. DOI: 10.1109/SLT.2014.7078578.

[18] P. Castells, "La web semántica," *Ciencia y técnica: Sistemas interactivos y colaborativos en la web*, vol. 47, pp. 195-212, 2003.

[19] M. del Prado, *El Grafo de Conocimiento del Museo del Prado*, 2015. [En línea]. Disponible en: https://www.museodelprado.es/grafo-de-conocimiento/el-grafo-de-conocimiento-del-museo-del-prado.

[20] T. Saorín, "Grafos de conocimiento y bases de datos en grafo: conceptos fundamentales a partir de una "obra maestra" del Museo del Prado," *Anuario ThinkEPI*, vol. 13, 2019. DOI: 10.3145/thinkepi.2019.e13f05.

[21] T. Delgado Fernández, M. L. Stuart Cárdenas y M. Delgado Fernández, "Grafos de conocimiento para gestionar información epidemiológica sobre COVID-19," es, *Revista Cubana de Información en Ciencias de la Salud*, vol. 32, 2021. [En línea]. Disponible en: http://scielo.sld.cu/scielo.php?script=sci_arttext&pid=S2307-21132021000400002&nrm=iso.

[22] Y. R. Mesa, A. M. M. Gómez y L. A. Valladares, "Optimización para los motores de búsqueda (SEO) y la garantıa de posicionamiento en los buscadores," *MediSur*, vol. 19, n.º 1, pp. 188-192, 2021.

[23] L. Iglesias-García Mar Codina, "Los cibermedios y la importancia estratégica del posicionamientoen buscadores (SEO)," Español, *Opción*, 2016. [En línea]. Disponible en: https://www.redalyc.org/articulo.oa?id=31048482052.

[24] H. Davis, *Search engine optimization*. O'Reilly Media, Inc., 2006.

[25] NCBI, *About - PubMed*. [En línea]. Disponible en: https://pubmed.ncbi.nlm.nih.gov/about/.

[26] R. Trueba-Gómez y J.-M. Estrada-Lorenzo, "La base de datos PubMed y la búsqueda de información científica," *Seminarios de la Fundación Española de Reumatología*, vol. 11, n.º 2, pp. 49-63, 2010. DOI: https://doi.org/10.1016/j.semreu.2010.02.005.

[27] NLH, *Home - Mesh - NCBI*, 2016. [En línea]. Disponible en: https://www.ncbi.nlm.nih.gov/mesh/.

[28] Z. Lu, W. Kim y W. J. Wilbur, "Evaluation of query expansion using MeSH in PubMed," *Information retrieval*, vol. 12, n.º 1, pp. 69-80, 2009.

[29] R. Pastor, "Metalgoritmo de Optimización combinatoria Mediante La Exploración de grafos.," Tesis doct., 2009. [En línea]. Disponible en: http://hdl.handle.net/10803/6781.

[30] J. H. Restrepo y J. J. Sánchez, "Aplicación de la teoría de grafos y el algoritmo de Dijkstra para determinar las distancias y las rutas más cortas en una ciudad," *Scientia et technica*, vol. 10, n.º 26, pp. 121-126, 2004.

[31] M. Montón, D. Castells, A. Portero y J. Carrabina, "Implementación SystemC sintetizable de un procesador asociativo para el algoritmo de Dijkstra," *Dept. Microelectrónica i Sistemas Electrónics, Universitat Autónoma de Barcelona*, 2005.

[32] L. P. Morales, A. D. Esteban y P. Gervás, "Uso de Grafos de Conceptos para la Generacion Automática de Resúmenes en Biomedicina," *Procesamiento del lenguaje Natural*, vol. 41, pp. 191-198, 2008.

[33] L. O. Camarena, "Estudio del estado del arte de Resumen Automático de Texto," 2019. [En línea]. Disponible en: https://www.researchgate.net/publication/331641998_Estudio_del_estado_del_arte_de_Resumen_Automatico_de_Texto.

[34] G. Erkan y D. R. Radev, "LexRank: Graph-based Lexical Centrality as Salience in Text Summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457-479, 2004. DOI: 10.1613/jair.1523. [En línea]. Disponible en: https://doi.org/10.1613%5C%2Fjair.1523.

[35] A. Simbha, *Understanding TF-IDF for Machine Learning*, 2021. [En línea]. Disponible en: https://www.capitalone.com/tech/machine-learning/understanding-tf-idf/.

[36] Crabcamp, *CraLexRank algorithm for text summarization*, 2019. [En línea]. Disponible en: https://github.com/crabcamp/lexrank.

[37] M. Bianchini, M. Gori y F. Scarselli, "Inside PageRank," *ACM Trans. Internet Technol.*, vol. 5, n.º 1, pp. 92-128, 2005. DOI: 10.1145/1052934.1052938.

[38] Chonyy, *Hits algorithm: Link analysis explanation and python implementation from scratch*, 2021. [En línea]. Disponible en: https://towardsdatascience.com/hits-algorithm-link-analysis-explanation-and-python-implementation-61f0762fd7cf.

[39] I. Yoo, X. Hu e I.-Y. Song, "A coherent graph-based semantic clustering and summarization approach for biomedical literature and a new summarization evaluation method," en *BMC bioinformatics*, Springer, vol. 8, 2007, pp. 1-15.

[40] N. S.-d.-C. y Larisa Zamora-Matamoros, "Métodos gráficos en la investigación biomédica de causalidad," *Revista Electrónica Dr. Zoilo E. Marinello Vidaurreta*, vol. 44, n.º 4, 2019. [En línea]. Disponible en: http://revzoilomarinello.sld.cu/index.php/zmv/article/view/1846.

[41] I. Shrier y R. W. Platt, "Reducing bias through directed acyclic graphs," *BMC medical research methodology*, vol. 8, n.º 1, pp. 1-15, 2008. DOI: https://doi.org/10.1186/1471-2288-8-70.

[42] N. González-Cervantes, A. Espinoza-Valdez y R. Salido-Ruiz, "Potencial Eléctrico en el Corazón: Representación Mediante un Grafo," *ReCIBE. Revista electrónica de Computación, Informática, Biomédica y Electrónica*, vol. 5, n.º 3, 2016.

[43] B. Benchamardimath y R. Hegadi, "A GRAPH THEORETICAL NETWORK MODEL ON HUMAN HEART," *International Journal of Applied Engineering Research*, vol. 9, n.º 20, 2014. [En línea]. Disponible en: https://www.researchgate.net/publication/340827007_A_GRAPH_THEORETICAL_NETWORK_MODEL_ON_HUMAN_HEART.

[44] U. Santander, *Python: Qué es y por qué deberías aprender a utilizarlo*, 2021. [En línea]. Disponible en: https://www.becas-santander.com/es/blog/python-que-es.html.

[45] C. Garcia, *Los lenguajes de Programación Más Populares (2021)*, 2021. [En línea]. Disponible en: https://codelearn.es/blog/los-lenguajes-de-programacion-mas-populares-2021/.

[46] S. Team, *Home - spyder ide*, 2021. [En línea]. Disponible en: https://www.spyder-ide.org/.

[47] *Pycharm vs Spyder vs Jupyter vs visual studio vs Anaconda vs intellij*, 2021. [En línea]. Disponible en: https://ritza.co/articles/pycharm-vs-spyder-vs-jupyter-vs-visual-studio-vs-anaconda-vs-intellij/#pycharm-vs-spyder.

[48] A.-R. Ali, *Python's pickles*, 2017. [En línea]. Disponible en: https://code.tutsplus.com/tutorials/pythons-pickles--cms-28839.

[49] *Real Academia Nacional de Medicina*, 2018. [En línea]. Disponible en: http://dtme.ranm.es/buscador.aspx.

[50] A. COpenMed, *Copenmed*, 2022. [En línea]. Disponible en: https://copenmed.org/#/paginaInicio.

[51] C. Ivorra, "Introducción al LATEX," 2019. [En línea]. Disponible en: https://www.uv.es/~ivorra/Latex/LaTeX.pdf.

[52] C. Ó. Sánchez Sorzano, *Copenmed*, 2019. [En línea]. Disponible en: http://www.copenmed.org/.

[53] C. Commons, *Creative Commons License Deed*. [En línea]. Disponible en: https://creativecommons.org/licenses/by-sa/4.0/.

[54] D. A, *Spyder, Un potente entorno de desarrollo interactivo para python*, 2017. [En línea]. Disponible en: https://ubunlog.com/spyder-entorno-desarrollo-python/.

[55] *Various licenses and comments about them - GNU Project - Free Software Foundation*. [En línea]. Disponible en: https://www.gnu.org/licenses/license-list.en.html#GPLCompatibleLicenses.

[56] F. S. Foundation, *The GNU general public license v3.0 - GNU Project - Free Software Foundation*, 2007. [En línea]. Disponible en: https://www.gnu.org/licenses/gpl-3.0.html.

[57] N. Unidas, *La Falta de Agua O Saneamiento en los Centros de Salud Pone en Riesgo a miles de millones de personas | noticias onu*, 2019. [En línea]. Disponible en: https://news.un.org/es/story/2019/04/1453811.

# A. PLANIFICACIÓN TEMPORAL

En este anexo se detalla la planificación temporal del proyecto. Comenzando con la identificación de las actividades, estimación temporal de cada una, secuenciación de las actividades y por último elaboración del cronograma del proyecto.

### Identifiación de actividades y estimación temporal

A continuación se detallan las actividades realizadas organizadas en las diferentes fases del proyecto con la estimación temporal inicial realizada.

- Formación teórica
  - Búsqueda de información acerca de grafos de conocimiento - 2 semanas
  - Manejo de LaTeX a través de Python -1 semana
- Planteamiento del trabajo
  - Definición de objetivos - 1 semana
- Búsqueda de información del estado del arte
  - Conocimientos previos sobre teoría de grafos - 1 semana
  - Búsqueda de información sobre la aplicación de grafos de conocimiento a sistemas de información - 1 semana
  - Documentación sobre sistemas de información biomédica - 1 semana
- Análisis teórico
  - Definición del grafo - 1 semana
  - Estructuración de los documentos - 1 semana
- Implementación
  - Creación del grafo - 2 semanas
  - Diseño de las funciones - 3 semanas
  - Agrupación de las entidades - 3 semanas
- Visualización
  - Generación del documento - 2 semanas
- Pruebas

    - Verificación de la implementación - 4 semanas

- Generación de la documentación del Trabajo de Fin de Grado
    - Memoria del proyecto - 4 semanas
    - Presentación del proyecto - 1 semana

**Secuenciación de las actividades**

En este apartado se realiza la secuenciación de las actividades así como la identificación del camino crítico.

El trabajo se ha realizado a lo largo de veintiocho semanas. Para facilitar la inclusión de estas actividades en un diagrama, se ha elaborado la siguiente tabla, asociando cada actividad con una letra, incluyendo además una duración estimada de cada tarea:

| Nombre de la Actividad | Letra Asociada | Duración en semanas |
|---|---|---|
| Información grafos | A | 2 |
| Manejo de LaTeX | B | 1 |
| Definición de objetivos | C | 1 |
| Teoría de grafos | D | 1 |
| Sistemas basados en grafos | E | 1 |
| Información biomédica | F | 1 |
| Definición del grafo | G | 1 |
| Estructuración documentos | H | 1 |
| Creación del grafo | I | 2 |
| Diseño de las funciones | J | 3 |
| Agrupación entidades | K | 3 |
| Generación documento | L | 2 |
| Verificación de implementación | M | 4 |
| Memoria del proyecto | N | 4 |
| Presentación del proyecto | O | 1 |

Fig. A.1. Duración de actividades

Después, se ha desarrollado una matriz de dependencia entre las actividades, que se muestra en la siguiente imagen, con el fin de establecer secuencias:

| | | Precedentes | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Sucesoras | | A | B | C | D | E | F | G | H | I | J | K | L | M | N | O |
| | A | | | 1 | | | | | | | | | | | | |
| | B | | | 1 | | | | | | | | | | | | |
| | C | | | | | | | | | | | | | | | |
| | D | 1 | | | | | | | | | | | | | | |
| | E | | | | | | | | | | | | | | | |
| | F | | | | | | | | | | | | | | | |
| | G | 1 | | 1 | 1 | 1 | | | | | | | | | | |
| | H | | | 1 | | | | | | | | | | | | |
| | I | | | | | | | 1 | | | | | | | | |
| | J | | | | | | | | 1 | 1 | | | | | | |
| | K | | | | | | | | | 1 | 1 | | | | | |
| | L | | 1 | | | | | | | | 1 | 1 | | | | |
| | M | | | | | | | | | | | | 1 | | | |
| | N | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | | |
| | O | | | | | | | | | | | | | | 1 | |

Fig. A.2. Matriz de dependencias

Se puede observar que actividades como la realización de la memoria dependen de todos los apartados anteriores; en este caso, se ha decidido realizar de manera simultánea, incluyendo cada apartado en la memoria durante su realización. En cuanto a las actividades de implementación, se realizan de manera secuencial y dependen de las anteriores, por lo que el camino crítico se construye de esta forma:

Definición de objetivos – teoría de grafos – definición del grafo – creación del grafo – diseño de las funciones – agrupación de las entidades – generación del documento – verificación de la implementación – memoria del proyecto – presentación del proyecto

## Cronograma

Por último, en base a la secuenciación mostrada anteriormente, se ha elaborado el siguiente cronograma:

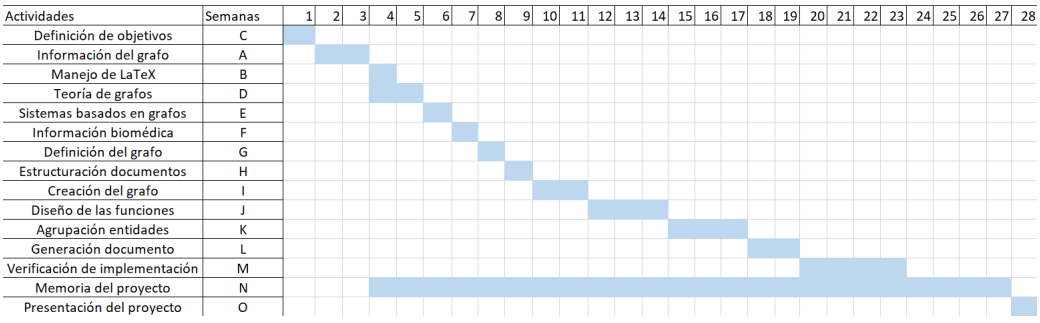



Fig. A.3. Cronograma del proyecto

Se puede ver que el proyecto se desarrolla a lo largo de veintiocho semanas y, aunque algunas actividades no tienen dependencia con otras anteriores, al ser un proyecto realizado por una sola persona, los recursos humanos dedicados son limitados e impiden la realización en paralelo de varias actividades sin aumentar el tiempo total del trabajo. Se puede ver que la duración estimada del desarrollo de la memoria era de cuatro semanas,

mientras que en el cronograma se extiende a 16. sto es así, porque se decidió comenzar en paralelo a otras tareas para evitar la pérdida de información, aunque luego se dedicaron cuatro semanas exclusivamente a su redacción.

Finalmente, se listan las fechas del comienzo de las actividades:

- Formación teórica
  - Búsqueda de información acerca de grafos de conocimiento - 15 de noviembre
  - Manejo de LaTeX a través de Python - 22 de noviembre
- Planteamiento del trabajo
  - Definición de objetivos - 8 de noviembre
- Búsqueda de información del estado del arte
  - Conocimientos previos sobre teoría de grafos - 6 de diciembre
  - Búsqueda de información sobre la aplicación de grafos de conocimiento a sistemas de información - 13 de diciembre
  - Documentación sobre sistemas de información biomédica - 20 de diciembre
- Análisis teórico
  - Definición del grafo - 27 de diciembre
  - Estructuración de los documentos - 3 de enero
- Implementación
  - Creación del grafo - 24 de enero
  - Diseño de las funciones - 14 de febrero
  - Agrupación de las entidades - 7 de marzo
- Visualización
  - Generación del documento - 21 de marzo
- Pruebas
  - Verificación de la implementación - 18 de abril
- Generación de la documentación del Trabajo de Fin de Grado
  - Memoria del proyecto - 24 de enero
  - Presentación del proyecto - 27 de junio

# B. PRESUPUESTO

En este apartado se elabora el presupuesto del Trabajo de Fin de Grado. Contiene los gastos diferenciados en los materiales utilizados, los costes de recursos humanos y los costes indirectos.

La unidad monetaria utilizada es el euro y se tendrá en cuenta un redondeo con dos decimales. Así mismo se aplican los impuestos adecuados a cada cálculo.

### Costes de recursos humanos

Para el cálculo de los costes de recursos humanos, se han contabilizado las horas dedicadas por Claudia Llorente Pascual, autora del trabajo, Carlos Óscar Sánchez Sorzano, tutor de la Asociación COpenMed y Pedro Manuel Moreno Marcos, tutor de la Universidad Carlos III de Madrid.

Las horas dedicadas por la alumna se indican en una tabla que presenta los meses transcurridos en la realización del trabajo, las horas semanales dedicadas y las horas totales en el mes. Se presentan en tablas separadas las horas para cada una de las personas mencionadas.

TABLA B.1. HORAS TRABAJADAS POR CLAUDIA LLORENTE PASCUAL

| Mes | Semanas Trabajadas | Horas por Semana | Horas totales |
|---|---|---|---|
| Noviembre | 3 | 12 | 36 |
| Diciembre | 3 | 12 | 36 |
| Enero | 3 | 12 | 36 |
| Febrero | 4 | 12 | 48 |
| Marzo | 4 | 12 | 48 |
| Abril | 4 | 12 | 48 |
| Mayo | 4 | 12 | 48 |
| Junio | 4 | 12 | 48 |
| Julio | 1 | 12 | 12 |
| Total | | | 360 |

En la tabla B.1 se observa que las horas totales dedicadas al proyecto han sido 360. Se muestran a continuación las horas dedicadas por cada uno de los tutores:

TABLA B.2. HORAS TRABAJADAS POR EL TUTOR CARLOS ÓSCAR SÁNCHEZ SORZANO

| Mes | Semanas Trabajadas | Horas por Semana | Horas totales |
|---|---|---|---|
| Noviembre | 3 | 1 | 3 |
| Diciembre | 3 | 1 | 3 |
| Enero | 3 | 1 | 3 |
| Febrero | 4 | 1 | 4 |
| Marzo | 4 | 1 | 4 |
| Abril | 4 | 1 | 4 |
| Mayo | 4 | 1 | 4 |
| Junio | 3 | 1 | 3 |
| Total | | | 32 |

Como se muestra en la tabla B.2, el tiempo de dedicación para el tutor Carlos Oscar Sanchez Sorzado asciende a 32 horas. En el caso de Pedro Manuel Moreno, al haber realizado el desarrollo del trabajo en la Asociación COpenMed, las horas se concentran en los dos últimos meses de trabajo, donde se ha elaborado la memoria y documentación correspondiente al Trabajo de Fin de Grado. Se pueden ver las horas dedicadas en la tabla B.3:

TABLA B.3. HORAS TRABAJADAS POR EL TUTOR PEDRO MANUEL MORENO MARCOS

| Mes | Semanas Trabajadas | Horas por Semana | Horas totales |
|---|---|---|---|
| Mayo | 4 | 2 | 8 |
| Junio | 3 | 3 | 9 |
| Total | | | 17 |

A partir de las horas totales dedicadas, se han calculado los costes asociados a recursos humanos. Se tiene en cuenta un precio por hora para cada persona indicada anteriormente. Para Claudia Llorente Pascual, como autora del trabajo, se establece un coste de 25€/ hora para un Ingeniero Junior. Para los tutores del trabajo, Carlos Oscar Sánchez Sorzano y Pedro Manuel Moreno Marcos, como Ingenieros, se establece un coste de 35€/ hora:

TABLA B.4. COSTES DE PERSONAL

| Personal | Categoría | Coste (€/hora) | Horas dedicadas | Total (€) |
|---|---|---|---|---|
| Claudia Llorente Pascual | Ingeniero Junior | 25,00 | 360 | 9 000,00 |
| Carlos Óscar Sánchez | Ingeniero | 35,00 | 32 | 1120,00 |
| Pedro Manuel Moreno | Ingeniero | 35,00 | 17 | 595,00 |

Como se ve en la tabla B.4, en total los costes asociados a personal ascienden a 10 715,00€.

### Costes materiales

Para realizar el cálculo asociado a los recursos materiales se ha calculado el coste de uso de cada elemento teniendo en cuenta su precio inicial, porcentaje de uso y amortización. El cálculo se realiza mediante la siguiente fórmula:

$$Coste = (PrecioDelMaterial - ValorResidual) * \frac{PeriodoUtilizacin}{PeriodoAmortizacin} \quad \text{(B.1)}$$

Se indica el cálculo realizado en la tabla B.5:

TABLA B.5. COSTES MATERIALES DEL TRABAJO DE FIN DE GRADO

| **Material** | **Tipo** | **Precio (€)** | **Valor residual (€)** |
|---|---|---|---|
| Lenovo Yoga 720 13” i7 | HW | 1 637,00 | 300,00 |
| **Dedicación (semanas)** | **Periodo de amortización (semanas)** | **Coste (€)** | |
| 28 | 192 (4 años) | 194,98 | |

En cuanto al software, las herramientas utilizadas han sido Spyder y TexWorks, ambas opciones de software libre, por lo que el coste total de ambas es cero. Para la elaboración de la memoria del Trabajo de Fin de Grado se ha utilizado Overleaf, que es gratuito para proyectos de un solo colaborador.

En total, los costes materiales son de 194,98€.

### Costes indirectos

Para los costes indirectos de este proyecto, se ha tenido en cuenta el precio de la conexión a Internet, así como el uso de luz que se detalla en la siguiente tabla:

TABLA B.6. COSTES INDIRECTOS

| Descripción | Precio | Dedicación (semanas) | Coste (€) |
|---|---|---|---|
| Fibra óptica 100Mbps | 20€/mes | 28 | 140 |
| Gastos de luz | 0,25599 €/kWh | 28 (consumo aproximado 3kWh/semana) | 21,50 |

Como se puede ver en la tabla B.6, los costes indirectos están formados por la conexión a internet y los gastos de luz. Por lo tanto, los costes indirectos se sitúan en un total de 161,50€.

### Coste total

Por último, se incluye en la tabla B.7 el coste final del proyecto, aplicando el IVA al importe final:

TABLA B.7. PRESUPUESTO TOTAL DEL PROYECTO

| Descripción | Coste (€) |
|---|---|
| Coste total de RRHH (€) | 10 715,00 |
| Costes totales de material | 194,98 |
| Costes indirectos totales | 161,50 |
| Importe total | 11 071,48 |
| IVA (21 %) | 2 325,01 |
| **Presupuesto total** | 13 396,49 |

# C. MARCO REGULATORIO

En este apartado, se analiza la legislación y las cuestiones relacionadas con la propiedad intelectual e industrial de las herramientas aplicadas y generadas en este trabajo.

Una parte relevante de los trabajos de estructuración de información es la legislación o las licencias de uso bajo las que se regulan. En el caso de este trabajo, la información se ha obtenido de la Asociación COpenMed. Toda la información publicada por dicha asociación está bajo una licencia Creative Commons CC-BY-SA 4.0 [52]. Este tipo de licencia, del mismo tipo utilizado por Wikipedia, permite compartir, copiar, adaptar y redistribuir el contenido publicado bajo dos condiciones: la primera de ellas es la atribución del trabajo, es decir, se debe citar apropiadamente al autor de la información. La segunda condición es que si se adapta el contenido publicado y se construye o desarrolla sobre el mismo, este debe ser compartido bajo el mismo tipo de licencia [53]. Al realizar este trabajo para la misma Asociación, se asume que su posible integración con la web así como su publicación o utilización futura se sitúa bajo el mismo tipo de licencia.

Por otro lado, la utilización de software para el desarrollo de este trabajo también supone la regulación bajo ciertas licencias. En el caso de Spyder, este entorno de desarrollo integrado y multiplataforma de código abierto está liberado bajo la licencia de MIT [54]. Esta licencia, creada en el Instituto Tecnológico de Massachusetts, permite la creación de trabajos no libres a partir de la misma. Además, concede al usuario derechos para usar, copiar, modificar, publicar, distribuir o sublicenciar el software. El texto de la licencia se divide en tres partes [55]:

1. Condiciones: la nota de copyright así como los derechos deben incluirse en todas las copias del software realizadas así como en aquellas modificaciones que utilicen partes sustanciales.

2. Derechos: no existen restricciones de uso, copia, modificación o integración. Se permite también publicar o vender copias del software y permite, a quién se entregue dicho software, hacer lo mismo.

3. Limitación de responsabilidad: se incluye una nota de limitación de responsabilidad.

Por lo tanto, esta licencia no limita las condiciones de creación de trabajos desarrollados a través de este software, de manera que se pueden asociar a la misma o a otras licencias. Para el software utilizado para la compilación del documento final, TexWorks, se trabaja bajo una licencia GNU - General Public License (GPL) versión 3. Esta licencia establece que cualquiera tiene potestad para copiar y distribuir copias del documento bajo la licencia. De manera similar a la licencia de Creative Commons, se debe establecer la autoría original del software utilizado. Además se otorga permiso para copiar, distribuir y

modificar el producto. Se puede encontrar más información en la página oficial de GNU [56]. En lo que respecta a este trabajo, no se han realizado cambios al software de TexWorks, sino que se ha usado como medio para producir el documento final, por lo que aparece citado en la metodología de este trabajo.

# D. MARCO SOCIOECONÓMICO

En este apartado se define el plan de explotación del proyecto, así como el impacto económico, social y ético.

### Plan de explotación

Actualmente, la herramienta no genera beneficios económicos de forma externa a la Asociación COpenMed. En su etapa inicial, el propósito del trabajo es su aplicación como uso interno para la asociación COpenMed para generar documentos de consulta a partir de su base de datos. Se plantea este uso durante un tiempo prolongado con el fin de encontrar fallos en el sistema así como refinar las categorías y adaptarlas a nuevos modelos de información generados por la asociación.

Una vez obtenido un modelo refinado y con alta precisión, se propone la inclusión de este sistema en la página web de la Asociación COpenMed, de manera que la estructuración de la información se realice de manera no supervisada y genere un resultado público.

A su vez, al tratarse de una herramienta orientada a la estructuración de información a partir de grafos de conocimiento, esta herramienta se podría generalizar para estructurar información de varios ámbitos. De esta manera podría aplicarse para explorar las entidades de otras empresas que tengan su información organizada en grafos, de manera que pueda estructurar la información de manera no gráfica o servir para evaluar las relaciones existentes entre entidades, identificando por ejemplo aquellas menos o más conectadas.

La ampliación del uso de esta herramienta produciría un beneficio para el personal involucrado en su desarrollo.

### Impacto económico

En cuanto al impacto económico, esta herramienta facilitará el acceso a la información de la Asociación COpenMed y servirá además como punto de partida para la elaboración de una página web. En este sentido, la elaboración de un proceso automático, permite no depender de terceras personas para la vinculación del grafo con la página web, reduciendo los gastos de la asociación.

Actualmente, el acceso a un documento ordenado con la información recogida por esta asociación, permite el acceso a información de alto nivel organizada para ser comprendida con facilidad. De esta manera permite a los usuarios acceder con agilidad a la información de interés, lo que a nivel económico reduce su necesidad de acudir al médico para todas

las consultas, ya que provee una herramienta que puede reducir aquellas que se realizan con propósitos puramente informativos. De cara al sistema sanitario, esto alivia la carga del personal y puede reducir la presión sobre los centros de atención primaria.

Finalmente, de cara a su uso como herramienta de consulta para estudiantes, puede impactar en su consumo de libros físicos. Además, si la herramienta de estructuración se aplicara a otras bases de datos o áreas de conocimiento, podría suponer un impacto ecológico positivo en la impresión de material físico.

### Impacto social

Según las Naciones Unidas más de la mitad de la población no tiene acceso a servicios de salud esenciales y carece de información de ningún tipo en ese ámbito [57]. Además, la elitización del lenguaje médico, dificulta la comprensión y entorpece el acceso a la población general. Esto provoca que parte de la población no tenga acceso a conocimiento biomédico de ningún tipo. Esta herramienta presente divulgar en la medida de lo posible este conocimiento y ponerlo al alcance de cualquier persona con una conexión a internet.

El hecho de contar con una herramienta capaz de organizar y estructurar la información facilita el acceso a esta información sin perder la rigurosidad científica de la medicina. De cara a su uso por estudiantes en el ámbito de la biomedicina, esta herramienta provee una forma de estudio y repaso, así como una opción más de consulta que puede resultar más ágil que un libro físico, La inclusión en el documento de referencias y vínculos entre las entidades permite además una lectura rápida y un movimiento intuitivo a través de la información.

Para finalizar, la existencia de información estructurada que utiliza lenguaje común que mantiene la corrección técnica puede servir para reducir la brecha digital. Al presentar una forma de acceder a la información de manera intuitiva, propone un sistema accesible que provee las facilidades de leer un libro, contando con las funcionalidades de un archivo digital.

### Impacto ético

Respecto al impacto ético de la herramienta, generar un documento de información orientado a la medicina puede crear controversia en tanto que se asocie a la sustitución de personal médico. Este trabajo no pretende en ningún momento sustituir al personal médico y no está pensado como una herramienta de diagnóstico. Teniendo en cuenta además, que la información tratada en este trabajo ha sido incluida en el documento por gran variedad de personas, la asociación COpenMed no se responsabiliza de las decisiones tomadas en base a la información publicada en su página.

En este sentido, se considera importante incluir un aviso al inicio del documento, indi-

cando que no se debe usar como herramienta de diagnóstico o para tratar una enfermedad, sino como herramienta de consulta, no sustituyendo de ninguna manera al personal sanitario.