CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Ingeniería Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de
Ingenieros Informáticos

TRABAJO FIN DE GRADO

Estructuración jerárquica y semántica de la información extraída de imágenes de tomografía computerizada y resonancia magnética

Autor: Roberto Garrido García

Directora: Consuelo Gonzalo Martín

MADRID, JULIO DE 2019

En primer lugar, dar las gracias a mi tutora, Consuelo, por la confianza depositada en mí para este trabajo, por guiarme durante toda su realización y por ser paciente y comprensiva conmigo.

A César, por ayudarme en la discusión de los resultados y por hacerme comprender ciertas partes de este trabajo de manera más clara.

A mis compañeros del laboratorio Medal del CTB, por haberme ayudado siempre que lo he necesitado y por haberme acogido tan bien desde que empecé aquí.

A Diego, Pablo y al resto de la Batmafia por todo el apoyo que nos hemos dado mutuamente durante estos años de carrera. La carrera hubiera sido mucho más dura de no ser por la complicidad que nos ha unido.

A Laura, por ayudarme y apoyarme siempre de manera incondicional. Por estar a mi lado y por animarme en los momentos más duros. Gracias por quererme tanto y tan bien.

Por último, quiero dar las gracias a mis padres. Gracias por la paciencia y por darme el espacio y la libertad que necesitaba.

Índice general

Índice de figuras

Índice de tablas

Resumen

Durante los últimos años, la tecnología ha permitido un avance cualitativo en la detección y prevención de enfermedades. El proceso de inclusión de la tecnología al ámbito médico ha introducido nuevos conceptos y ha añadido complejidad. Una de las brechas más importantes es la conocida como *Gap Semántico*. El *Gap Semántico* es la discrepancia existente entre las características de bajo nivel relativas a la propia imagen y los conceptos médicos de alto nivel.

En este Trabajo de Fin de Grado, se intenta reducir esta brecha al mínimo analizando imágenes de Tomografía Axial Computerizada de tórax e imágenes de Resonancia Magnética de cráneo, anotando las características de bajo nivel extraídas de ellas y enriqueciéndolas con conceptos ontológicos. El análisis de ambos casos de estudio se divide en diferentes etapas, donde se segmentan las imágenes, se extrae su conjunto de características y se les aplican técnicas de aprendizaje automático.

Durante la realización de este trabajo, enmarcado dentro del proyecto europeo IASIS [1], se ha desarrollado un módulo de enriquecimiento semántico que asocia a los conceptos extraídos directamente de las imágenes médicas, el término equivalente del diccionario ontológico correspondiente.

Los términos resultantes podrán ser compartidos con otros centros y con otros laboratorios de investigación, ya que al estar debidamente anonimizados y enriquecidos semánticamente, se consigue universalizar el conocimiento generado.

Abstract

During the last years, technology has allowed a qualitative enhancement in the detection and prevention of diseases. The process of including technology within the medical scope has introduced new concepts and it has added more complexity. One of the most important gaps is the one known as *Semantic Gap*. The *Semantic Gap* is the difference between low-level concepts directly associated with the image itself and high-level medical concepts.

In this End-of-Degree Project, this gap is attempted to be reduced to a minimum by analyzing thorax Computerized Tomography images and skull Magnetic Resonance images. The low-level characteristics extracted from these images are annotated and then enriched with ontological concepts. The analysis of both study cases have different phases, where the images are segregated, their set of characteristics is extracted and they are studied using Machine Learning techniques.

During this work, as part of the IASIS European Project [1], a semantic enrichment module has been developed, which associates concepts extracted from the medical images with their equivalent ontological concepts.

The output concepts can be shared with other research laboratories as they are correctly anonymized and semantically enriched, so knowledge extracted is universalized.

1

Introducción

En los últimos años, la medicina ha dado un salto cualitativo en la detección de enfermedades de diverso origen. Gracias al avance tecnológico que ha tenido lugar en todos los ámbitos de la sociedad, incluido en el plano médico, y a la colaboración de equipos y hospitales de diferentes partes del mundo con centros universitarios y de investigación, ha surgido la necesidad de estandarizar los términos médicos utilizados y asociarlos a ontologías para facilitar la comunicación y la cooperación entre equipos.

Este trabajo se centra en la extracción de información de imágenes de Tomografía Axial Computerizada (TAC) y Resonancia Magnética (MRI) para su estructuración jerárquica y semántica, todo ello enmarcado en el proyecto europeo *Integration and analysis of heterogeneous big data for precision medicine and suggested treatments for different types of patients* (IASIS) [1] en colaboración con hospitales públicos de la Comunidad de Madrid.

Podemos definir una ontología como un conjunto de características que engloba a una representación, a una nomenclatura formal y a una definición de las categorías, las propiedades y las relaciones entre los conceptos, los datos y las entidades que justifican uno o varios dominios. Cada campo de conocimiento crea ontologías para limitar la complejidad y organizar la información, los datos y el conocimiento de dicho campo. La creación y la utilización de ontologías facilita enormemente la divulgación de las investigaciones y de los avances en todos los campos del conocimiento, pues permite que expertos de diferentes partes del mundo utilicen un vocabulario acotado y una terminología común. Por tanto, las ontologías podrían ser consideradas como una base de datos de conocimiento cuyo uso supone una abstracción

sobre el lenguaje natural que hace que un mismo concepto tenga una única representación posible dentro de dicha ontología [2]. Posteriormente, una ontología puede ser portada, estudiada y representada en cualquier idioma.

En este proyecto, las imágenes de Tomografía Axial Computerizada (TAC) se corresponden con pacientes enfermos de cáncer de pulmón de célula no pequeña, mientras que las imágenes de Resonancia Magnética (MRI) se corresponden con pacientes que padecen la enfermedad de Alzheimer. Ambos tipos de imágenes han sido debidamente anonimizadas con anterioridad.

En el ámbito médico, al ser las imágenes tomadas a pacientes una de las fuentes más importantes de información, se busca la mejora de los diagnósticos médicos realizados a partir de estos dos tipos de imágenes anteriormente citados. Uno de los objetivos de este proyecto es establecer una relación entre características propias de la imagen en sí (coloración de diferentes regiones, rugosidades, luminosidad, brillo, etc.) con conceptos médicos que los profesionales puedan manejar. Para conseguir esta relación de conceptos, se enriquecerán los datos extraídos de las imágenes con términos semánticos obtenidos directamente de diccionarios ontológicos médicos, estableciendo entre ellos una relación formal.

El cáncer es la segunda causa de muerte en el mundo y, en concreto, el cáncer de pulmón es el más común de todos. Según la Organización Mundial de la Salud [3], se estimaba que, en 2018, el cáncer provocaría 9.6 millones de muertes a nivel mundial, siendo el cáncer de pulmón el causante del 18 % de ellas [4]. En consecuencia, la prevención y la detección temprana del cáncer es uno de los principales retos a nivel médico y tecnológico. Como se puede observar en la figura 1.1, el cáncer de pulmón, junto con el de mama, son los dos tipos de cáncer que más incidencia tienen a nivel mundial, seguidos muy de cerca por el de colon.

La figura 1.2 muestra el índice de mortalidad de los tipos de cáncer más agresivos. Como se puede observar, el cáncer de pulmón supera por más de el doble en índice de mortalidad a cualquier otro tipo de cáncer.

Además del cáncer, existen otras muchas enfermedades que afectan a una gran cantidad de población a nivel mundial, siendo una de ellas la demencia. La demencia es un síndrome que provoca el deterioro cognitivo y la capacidad de realizar tareas cotidianas básicas de la vida diaria. Se estima que unos 50 millones de personas sufren algún tipo de demencia, donde cerca del 60 % de ellas viven en países subdesarrollados o en vías de desarrollo. En concreto, la enfermedad de Alzheimer es la forma más común de demencia, que supone entre el 60 % y el 70 % de los casos diagnosticados [5]. Otras formas frecuentes son la demencia vascular o la demencia por cuerpos de Lewy. A pesar de que la demencia, en cualquiera de sus variantes, suele afectar en un mayor grado a gente de avanzada edad, no es una consecuencia natural del envejecimiento.

En la figura 1.3 se puede observar la atrofia progresiva que sufre el cerebro de un sujeto sano (CN - *Cognitively Normal*), el de un sujeto con pérdidas leves de memoria (MCI - *Mild*

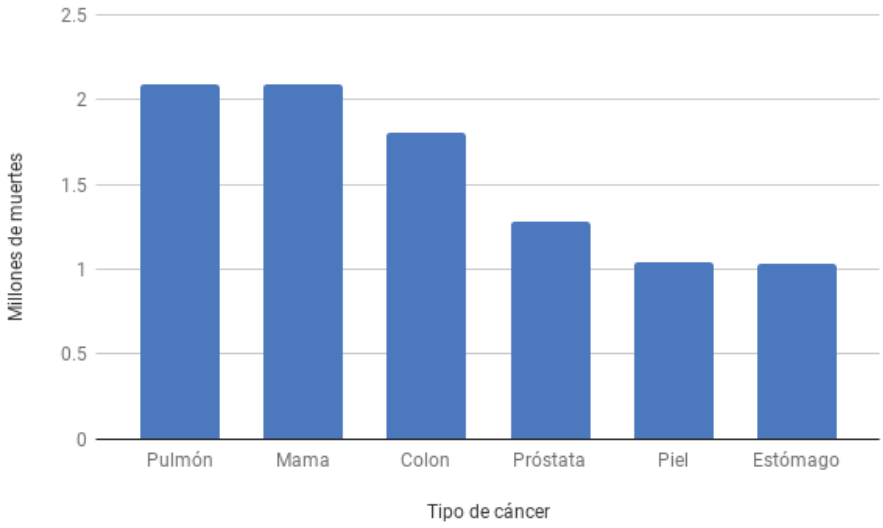

Figura 1.1: Incidencia del cáncer a nivel mundial

Cognitive Impairment) y el de un sujeto con la enfermedad de Alzheimer (AD - *Alzheimer's Disease*).

Este proyecto tiene dos objetivos principales: por un lado, se busca la estructuración jerárquica y semántica de la información extraída de las imágenes de TAC y de MRI tomadas a pacientes. Para ello, se desarrollará un módulo que clasifique y etiquete, utilizando diccionarios ontológicos, los diferentes conceptos de manera automática. Por otro lado, se analizarán las imágenes de Resonancia Magnética de los pacientes con la enfermedad de Alzheimer junto con el conjunto de imágenes que provee el proyecto *Open Access Series of Imaging Studies* (OASIS) [7] en su dataset OASIS-1, que servirán para entrenar los diferentes modelos predictivos que se generen. Para esta última tarea, utilizaremos el software Clinica [8], una plataforma desarrollada para estudios clínicos de pacientes con enfermedades neurodegenerativas y psiquiátricas, así como para la adquisición de datos multimodales.

En cuanto a las imágenes de TAC, no se realizan modelos predictivos con ellas ya que ese cometido forma parte de otro Trabajo de Fin de Grado que se estuvo realizando paralelamente a éste. Por otro lado, las imágenes de MRI no son enriquecidas semánticamente debido a que el proyecto IASIS todavía no dispone de imágenes de Resonancia Magnética de cráneo procedentes de los hospitales públicos con los que colabora. No obstante, el módulo desarrollado para el caso de cáncer de pulmón es fácilmente generalizable a otros casos de estudio, como por ejemplo, la enfermedad de Alzheimer.

1 Introducción

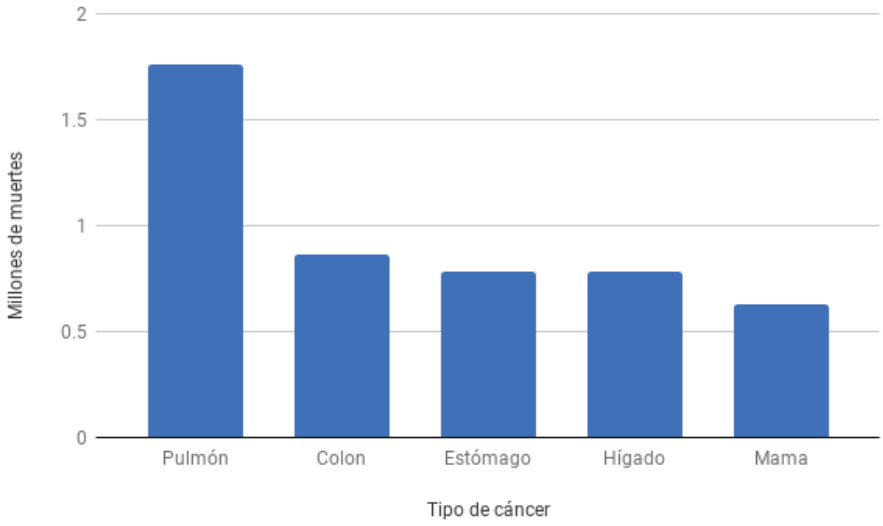

Figura 1.2: Mortalidad del cáncer a nivel mundial

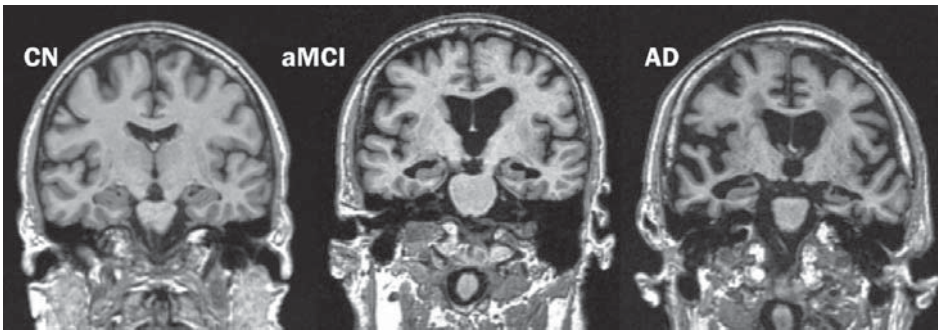

Figura 1.3: MRI de un paciente sano (CN), de un paciente con pérdidas leves de memoria (MCI) y de un paciente con la enfermedad de Alzheimer (AD). Imagen extraída de [6]

2

Estado del arte

La tecnología es utilizada a diario en múltiples y diversos campos de estudio. En concreto, en el ámbito médico, uno de los principales usos que se le da es el de la recogida de información sobre los pacientes, principalmente con la realización de pruebas médicas. Sin embargo, la tecnología prácticamente no está presente en la toma de decisiones por parte de los médicos a la hora de, por ejemplo, realizar un diagnóstico o decantarse por una línea u otra de tratamiento. Uno de los principales motivos de esta situación es la brecha existente entre las características de bajo nivel presentes en la propia imagen y los conceptos médicos a nivel semántico (conocimiento de alto nivel). Esta brecha de conocimiento se conoce como *GAP semántico* [9].

Para atajar este *GAP semántico*, se ha de utilizar una capa de conocimiento intermedio que interconecte el conocimiento de bajo nivel con el conocimiento de alto nivel. Como ya hemos adelantado en la introducción, esta capa extra es conocida como *ontología*. Dentro del mundo de la medicina, existen multitud de diccionarios ontológicos. A continuación se habla de tres de los más usados a nivel mundial:

- **SNOMED-CT** [10]
- **ICD-10** [11]
- **UMLS** [12]

2 Estado del arte

Las tres opciones proveen códigos únicos para conceptos médicos. De esta manera, se evita duplicidad de conceptos y se universalizan los términos. Si bien las tres encapsulan y estandarizan el conocimiento, UMLS [13] es la más completa de las tres ontologías, pues dispone de un *Metathesaurus* formado con los conceptos recogidos en cerca de 200 diccionarios ontológicos, entre los que se encuentran SNOMED-CT e ICD-10. UMLS está organizada por conceptos y relaciona los nombres similares con el mismo concepto. Dispone de su propio sistema de nombres y sigue una estructura jerárquica o de árbol, lo que permite que los conceptos se agrupen en familias, cada una con su propio código. Todo concepto está dentro de una familia. Los cuatro principales campos de cada concepto dentro de UMLS son:

- **CUI**: Se trata del código unívoco de un concepto médico, e.g.: *C1289436*.
- **STN**: Se trata del código de la familia a la que pertenece el concepto al que hace referencia el CUI, e.g.: *A1.2.3.1*. Cada paso que se avanza dentro de la jerarquía supone un índice maś que se añade al STN.
- **STY**: Corresponde con el nombre de la familia de conceptos, e.g.: *Region or Body Part*. Es equivalente al STN.
- **STR**: Corresponde con todos los conceptos médicos que están vinculados con el CUI correspondiente. Un solo CUI puede tener más de un STR.

Como podemos ver en los trabajos de Seifert [14] y Möller [15], utilizan el software MEDICO, que intenta salvar el *GAP semántico* vinculando las características de bajo nivel con conceptos médicos de alto nivel utilizando diccionarios ontológicos, pero solamente usan SNOMED, lo que limita el alcance de las anotaciones y de las relaciones que se pueden hacer a los conceptos. Como se puede leer en el artículo *Integrating SNOMED CT into the UMLS* [16], la integración de SNOMED dentro de UMLS se hizo de manera exhaustiva y corrigiendo los errores de discrepancia existentes en cuanto a sinónimos entre ambos diccionarios ontológicos por un comité de expertos.

En [17] se hace un estudio sobre los requisitos y los trabajos previos realizados sobre la anotación semántica en un plano muy general, es decir, se analiza la anotación semántica para el manejo de grandes volúmenes de conocimiento, no necesariamente imágenes ni necesariamente en un plano médico, por lo que nos puede servir como punto de partida, pero en ningún caso como guía sobre la estrategia a seguir.

En [18] se propone una aproximación al etiquetado semántico partiendo de características de bajo nivel y se utilizan diferentes técnicas de clasificación, como *Content-Based Image Retrieval* (CBIR). Esta técnica consiste en la búsqueda de imágenes en grandes bases de datos a través de sus características, al contrario de técnicas más tradicionales, como puede ser la técnica llamada *Content-Based Image Indexing*, que realiza la búsqueda utilizando metadatos

de la imágenes, como pueden ser palabras clave, etiquetas o descripciones asociadas a la imagen. Dentro de la técnica de CBIR, se analiza el contenido de la imagen atendiendo a colores, formas o texturas, entre otras características, todas ellas derivadas directamente de la imagen.

Si bien la técnica de CBIR es más costosa computacionalmente, es capaz de reducir fallos humanos, ya que la clasificación de imágenes a través de etiquetas y palabras clave es dependiente del factor humano, lo que hace este estudio bastante interesante. Sin embargo, para probar su modelo, utilizan imágenes TAC de hígado, lo que no nos permite asegurar la compatibilidad con el tipo de imágenes usadas durante este proyecto, debido a que las diferencias entre ellas pueden ser insalvables para el modelo generado.

2 Estado del arte

3

Metodología

3.1 Procesamiento de datos

Como ya hemos mencionado anteriormente, el trabajo que aquí se describe está enmarcado dentro de un proyecto mayor: IASIS. El proyecto IASIS está dividido en dos grandes ramas de estudio: la rama de análisis de texto y la rama de análisis de imágenes.

Por un lado, la rama del análisis de texto se encarga de analizar las notas clínicas y los informes de los pacientes para estructurar toda esa información. Las notas son procesadas mediante técnicas de Procesamiento de Lenguaje Natural (NLP por sus siglas en inglés) y, posteriormente, los datos obtenidos son enriquecidos mediante diccionarios ontológicos, un proceso en cierto modo similar al que se va a describir en este trabajo. Los resultados, una vez estructurados, son almacenados en una base de datos relacional MySQL [19]. Una vez están los resultados estructurados y almacenados de manera persistente en una base de datos a la que hacer consultas, comienzan una serie de post-procesos para calcular líneas de tratamiento, curvas de supervivencia y diferentes métricas necesarias para un posible análisis de *machine learning* o de minería de datos.

Por otro lado, en la rama de imagen, donde este proyecto se integra, la preparación, el procesamiento de los datos y su almacenamiento es diferente, aunque ambos procesos guardan una estrecha relación al estudiarse a los mismos pacientes en ambas ramas. Cabe destacar que la rama de imágenes tiene un volumen de casos menor al que tiene la rama de texto, ya que las imágenes no son tan fácilmente exportables como lo son las notas clínicas.

Las imágenes médicas pueden ser almacenadas en muchos formatos diferentes. En este proyecto, se utilizan dos de los formatos más extendidos para el almacenamiento de este tipo de imágenes: DICOM [20], con extensión *.dcm* y NIfTI [21], con extensión *.nii*. Ambos formatos permiten el almacenamiento de metadatos dentro del propio fichero. Si bien las imágenes en formato NIfTI suelen ser un único fichero de entre 20MB y 50MB que contiene todas las *slices* de la prueba realizada, las imágenes en formato DICOM están formadas por multitud de ficheros, uno por cada *slice*. De media, tienen entre 150 y 200 ficheros con un tamaño inferior a 1MB cada uno de ellos.

Estas imágenes no pueden ser visualizados en un visor de imágenes al uso, sino que necesitamos un software específico. Uno de los más completos que permite visualizar tanto las imágenes como los metadatos de las mismas es Fiji ImageJ [22], un visor de imágenes médicas que acepta ficheros tanto DICOM con NIfTI, incluso comprimidos en formato *.zip* o *.gz*. Como veremos más adelante, esto será de utilidad para visualizar los resultados del procesamiento de las imágenes de resonancia magnética.

3.2 Imágenes de Tomografía Axial Computerizada

Se decidió analizar las imágenes de TAC de pulmón mediante la extracción de características de *Radiomics* [23] utilizando la librería *Pyradiomics* [24]. Las características extraídas por *Pyradiomics* están orientadas a la propia imagen. Existía la necesidad de almacenar los datos extraídos de este análisis pero orientados a paciente. En este trabajo, se detalla el desarrollo de un módulo integrable que se encarga de realizar esta transformación y de almacenar el resultado en una base de datos MySQL [19].

Además de transformar los datos, el módulo se encarga de enriquecer semánticamente las características de *Radiomics* extraídas de las imágenes. Dado un concepto médico, se realiza la búsqueda de dicho concepto en el diccionario ontológico y se añade una tupla con el concepto médico original y la etiqueta ontológica correspondiente. Este enriquecimiento semántico nos permitirá obtener, en etapas posteriores, la relación entre las características extraídas de las imágenes y sus conceptos médicos asociados.

3.2.1 Preparación de datos

Antes de poder analizar las imágenes de TAC de pacientes de cáncer de pulmón, hay que preprocesarlas para eliminar las que sean erróneas, las que no se puedan observar correctamente o las que no dispongan de los metadatos necesarios requeridos, como pueden ser la fecha de la toma de la imagen o el género del paciente. En las imágenes procedentes de los hospitales públicos de la Comunidad de Madrid, el preprocesado ha de hacerse a mano, pues

las imágenes son enviadas directamente desde el hospital sin ningún tipo de garantía y algunas pueden estar demasiado oscuras, demasiado claras o incluso incompletas. Sin embargo, en las imágenes procedentes de bases de datos públicas, como Radiomics [25], repositorio público con 422 imágenes de cáncer de pulmón de célula grande, el preprocesado no es necesario, pues ya tenemos la garantía de que las imágenes son válidas al haber sido previamente procesadas y normalizadas.

En la figura 3.1 encontramos un TAC de tórax donde podemos observar con claridad los pulmones del paciente.

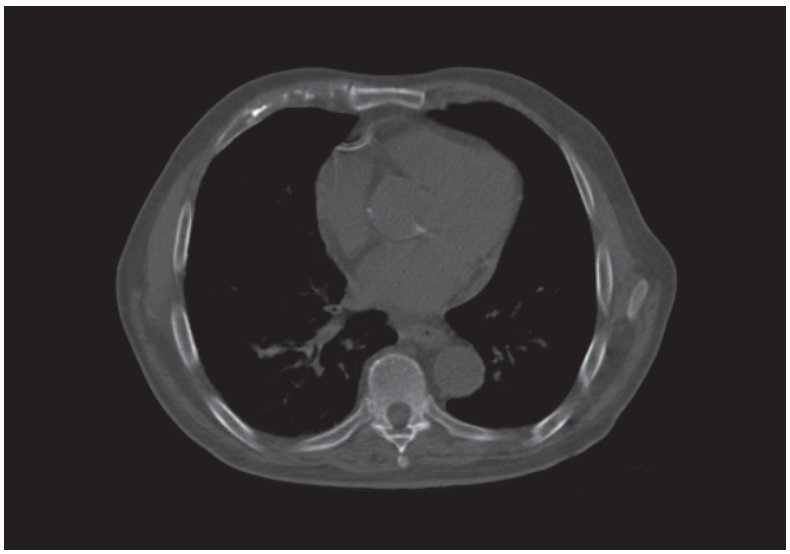

Figura 3.1: TAC de tórax

En la figura 3.2 podemos observar ciertos campos de los metadatos que se rellenan en las imágenes DICOM.

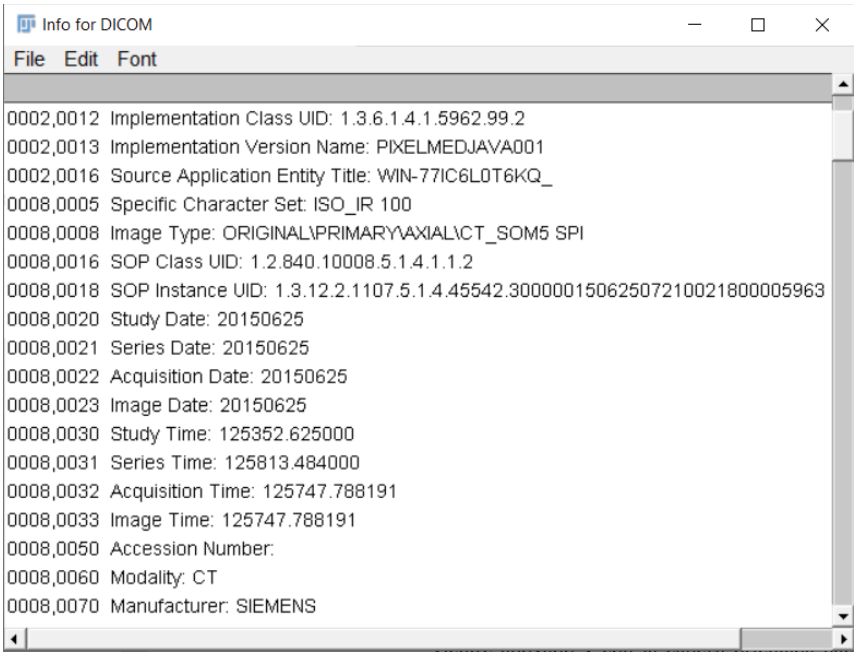

Figura 3.2: Detalle de los metadatos de una imagen DICOM

3.2.2 Extracción de características

Cada una de las imágenes es procesada mediante un *script* desarrollado en Matlab [26] que se encarga de diferenciar cada uno de los pulmones que aparecen en la imagen delimitando su contorno. Con los contornos de cada pulmón ya delimitados, se procede a generar los *bounding boxes*, unos rectángulos que circunscriben los pulmones. Una vez delimitado el contorno y los *bounding boxes*, se busca dentro de estas regiones los nódulos pulmonares que pudiera haber y se delimita, al igual que se hizo con los pulmones, su contorno. Esta división en regiones se realiza para tener acotado los puntos de estudio de cada una de las imágenes. De esta manera, podemos obviar el resto de la imagen y centrarnos solamente en las regiones delimitadas. Además, disponiendo de estas regiones, conocidas como máscaras, podremos reconstruir la imagen en cualquier momento a partir de las coordenadas espaciales de cada región dentro de la imagen.

La finalidad principal de los *bounding boxes* es maximizar la probabilidad de éxito en la detección de los nódulos. Si solamente dispusiéramos de la segmentación del pulmón y no de los *bounding boxes*, un nódulo que coincidiera con el contorno del pulmón no se detectaría, ya que parte de él quedaría fuera de los límites establecidos. Sin embargo, cualquier nódulo pulmonar queda dentro de los *bounding boxes*, por lo que no correríamos ese riesgo.

3.2 Imágenes de Tomografía Axial Computerizada

En la figura 3.3, podemos observar cómo se segmenta cada región del pulmón tras el procesamiento de una imagen de TAC con Matlab. De izquierda a derecha, tendríamos la segmentación del pulmón (el contorno), los *bounding boxes* y la segmentación del nódulo encontrado.

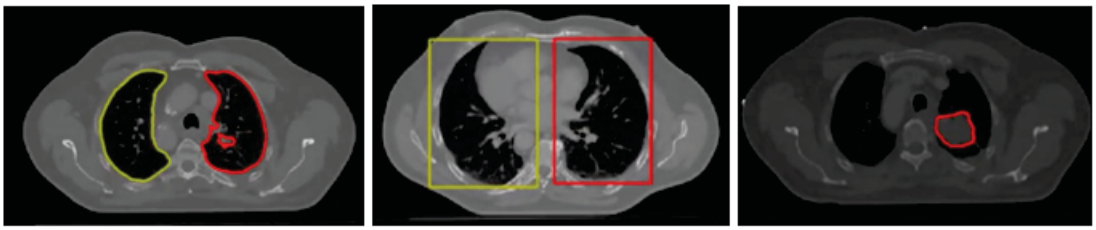

Figura 3.3: Segmentación del pulmón, *bounding boxes* y segmentación del nódulo

Además de las regiones que podemos observar en la imagen anterior, también se extraen los supervóxeles [27] de la imagen. Una imagen en dos dimensiones está formada por píxeles que se pueden agrupar en superpíxeles, una agrupación de píxeles que no se solapan con características similares de color o escala de grises y de brillo. Las imágenes en tres dimensiones, de manera análoga, están formadas por vóxeles que pueden agruparse en supervóxeles. En la figura 3.4, podemos ver la segmentación en supervóxeles de una de las imágenes de TAC de pulmón.

Figura 3.4: Segmentación en supervóxeles

Las imágenes generadas con Matlab en la etapa anterior se corresponden con los datos de entrada de un módulo escrito en Python [28] que implementa la librería *Pyradiomics* [24]. Esta librería funciona como un *framework* para la extracción de características *Radiomics* dentro de las regiones delimitadas anteriormente. Al tener las regiones de estudio previamente acotadas y los supervóxeles generados, agilizamos el proceso de extracción de características obviando las regiones que no están circunscritas por los *bounding boxes*, reduciendo así los tiempos de ejecución y la carga de trabajo del pipeline de procesamiento.

3.2.3 Almacenamiento de los resultados

Las características de tipo *Radiomics* extraídas en el apartado anterior se almacenan en un fichero de tipo *JSON* [29] generado directamente por el módulo de extracción de características. Se genera un fichero por cada sujeto analizado donde se guarda toda la información relativa al paciente (id, edad, estadío, tipo de cáncer, género) junto con todas las características extraídas. La figura 3.5 muestra una parte de los resultados generados, donde podemos ver tanto la información personal del sujeto como algunas de las características calculadas.


```
{
    "patient_id": "LUNG1-001",
    "survival_time": 2165,
    "age": 78.7515,
    "histology": "large cell",
    "gender": "male",
    "features": {
        "lungs": {
            "rigth": {
                "glcm": {
                    "original_glcm_Idn": 0.9535264695973401,
                    "original_glcm_JointEnergy": 0.013530882239059476,
                    "original_glcm_Id": 0.4659967711957344,
                    "original_glcm_ClusterShade": 3892.8962068030573,
                    "original_glcm_JointAverage": 9.173638993612853,
                    "original_glcm_DifferenceVariance": 14.855242918709745,
                    "original_glcm_Idm": 0.40511008600629383
```


Figura 3.5: Detalle de las características calculadas en formato *JSON*

3.3 Imágenes de Resonancia Magnética

Para analizar las imágenes de MRI de cráneo, utilizamos diversas herramientas. Las imágenes son procesadas utilizando el *software Clinica*, que detallaremos su funcionamiento más adelante. Los resultados calculados con *Clinica* se utilizan como valores de entrada para técnicas de aprendizaje automático.

3.3.1 Preparación de datos

Al igual que las imágenes de TAC correspondientes a pacientes de cáncer de pulmón, las imágenes de Resonancia Magnética pertenecientes a pacientes que padecen la enfermedad de Alzheimer provienen de varias fuentes diferentes: la base de datos de acceso público *Open Access Series of Imaging Studies* y el Laboratorio de Neurociencia Cognitiva y Computacional (LNCyC) [30] del Centro de Tecnología Biomédica.

Para que estas imágenes sean útiles y puedan ser estudiadas de manera conjunta, hay que realizarle una serie de preprocesos que forman parte de este proyecto y están explicados al detalle en los siguientes apartados. En la figura 3.6 podemos observar una Resonancia Magnética de cráneo.

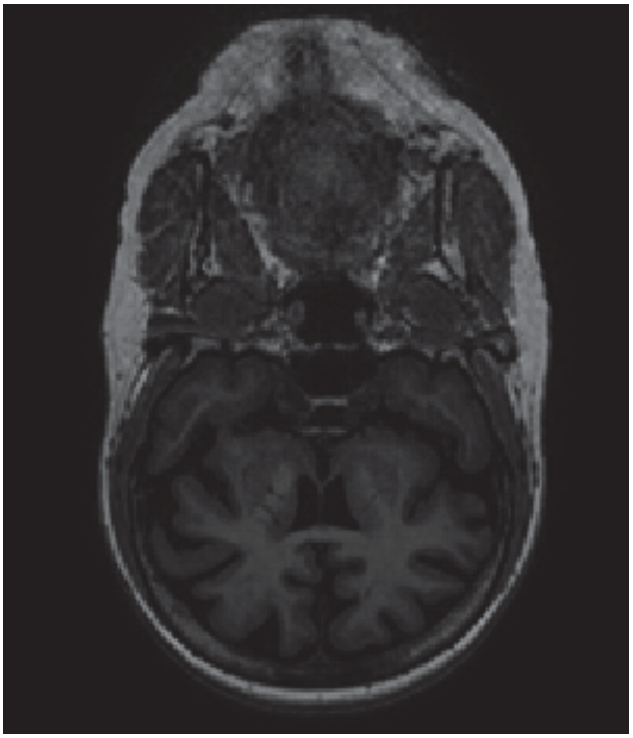

Figura 3.6: MRI de cráneo

3.3.2 Extracción de características

Software *Clinica*

La herramienta de software *Clinica* [8] ha sido desarrollada por el laboratorio francés de investigación Aramis Lab [31] y presentada en su paper *Yet Another ADNI Machine Learning Paper? Paving The Way Towards Fully-reproducible Research on Classification of Alzheimer's Disease* [32]. En este paper proponen una solución desarrollada en *Python* y basada en el *framework Nipype* [33] con diferentes utilidades a través de una interfaz de línea de comandos. En sus propias palabras:

> *Clinica* es una plataforma software para estudios de investigación clínica que impliquen a pacientes con enfermedades psiquiátricas y neurológicas, así como para la adquisición de datos multimodales (imágenes neurológicas, evaluaciones clínicas y cognitivas, genética...).

En puridad, *Clinica* funciona como un *wrapper*, esto es, como una capa de abstracción entre el usuario y las herramientas utilizadas por el software. Incluye diversos tipos de algoritmos para el análisis de imágenes médicas, así como diferentes *pipelines* de procesamiento. En este trabajo usaremos, principalmente, el *pipeline t1-volume*. Este *pipeline* es usado para analizar imágenes de resonancia magnética *T1-weighted* [34] y consta de las siguientes etapas:

- Segmentación por tejidos, corrección de sesgos y normalización espacial
- Creación de un modelo común usando DARTEL [35]
- Conversión de cada sujeto desde su espacio DARTEL a un espacio MNI [36]
- Parcelación del cerebro en atlas [37]

En la figura 3.7, podemos ver una representación gráfica de las diferentes etapas descritas en el listado anterior. Como se puede observar, la etapa de segmentación de tejidos, corrección de sesgos y normalización espacial es iterativa, es decir, las imágenes son procesadas de manera cíclica hasta que la diferencia entre dos normalizaciones consecutivas es inferior al umbral que *Clinica* tenga configurado.

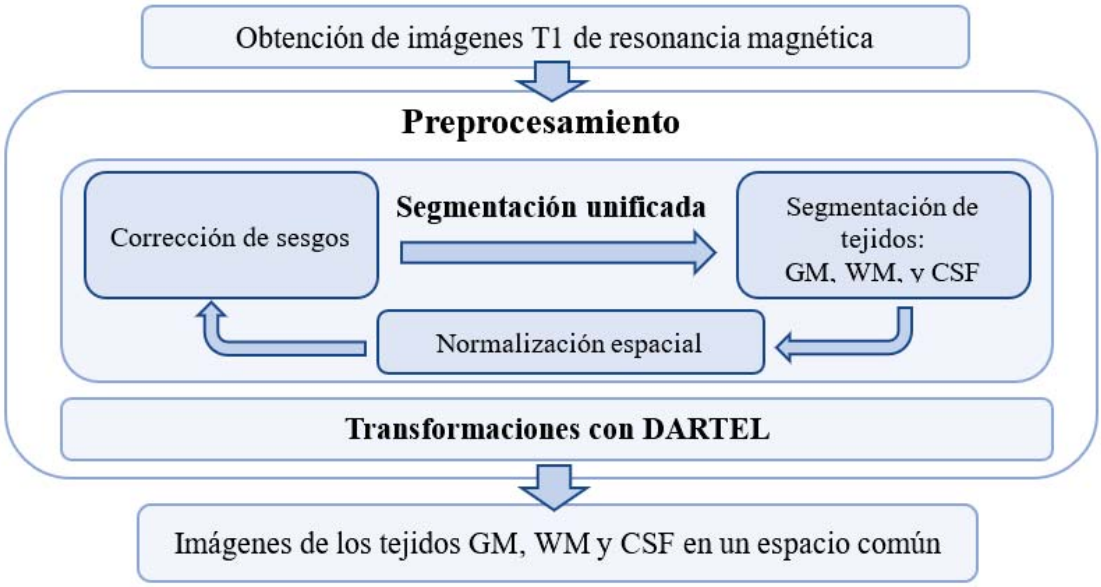

Figura 3.7: Diagrama con las diferentes etapas del *pipeline t1-volume* en *Clinica*

Conversión de imágenes MRI a formato BIDS para el software *Clinica*

Las imágenes de Resonancia Magnética, por lo general, se guardan en formato DICOM [20] o en formato NIfTI [21], dependiendo de la técnica utilizada para obtener las imágenes, incluso dependiendo del fabricante de la máquina utilizada. Sin embargo, *Clinica* solamente es capaz de trabajar con imágenes en formato BIDS [38] (*Brain Imaging Data Structure*), un estándar propuesto por Gorgolewski y su equipo en el paper *The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments* [39], cuya motivación surge de la falta de un criterio unificado o estándar a la hora de organizar datasets de imágenes de Resonancia Magnética.

A efectos prácticos, el formato BIDS no es más que una imagen en formato NIfTI, esto es, con extensión *.nii*, pero comprimida utilizando *gzip* [40] y almacenada dentro de una estructura de directorios muy concreta con una serie de ficheros TSV [41] con información acerca de la imagen y del sujeto del que fue tomada.

Las imágenes de la base de datos de OASIS [7] están en formato DICOM, es decir, con extensión *.dcm*, donde cada fichero se corresponde con una *slice* de la resonancia magnética. La conversión desde DICOM a BIDS no es trivial y no puede hacerse directamente. El primer

paso es convertir todos los ficheros *.dcm* en un solo fichero *.nii*, es decir, pasaremos de tener muchas imágenes en formato DICOM a tener una única imagen equivalente en formato NIfTI. Una vez realizada esta primera conversión, debemos generar la estructura de directorios para obtener un formato BIDS válido.

Para realizar la conversión de imágenes en formato DICOM a formato BIDS, existen diversas librerías, la mayoría de ellas de código abierto, como pueden ser *dcm2bids* [42] o *dcm2niix* [43] . Finalmente en este proyecto se optó por utilizar la herramienta *heudiconv* [44], un conversor de imágenes cerebrales que realiza los dos pasos descritos anteriormente: primero convierte los ficheros DICOM a NIfTI utilizando *dcm2niix* y después genera la jerarquía de directorios necesarios para tener la imagen en formato BIDS, lo que ya la haría compatible con *Clinica*. Toda la conversión la realiza dentro de contenedores de Docker [45], lo que nos evita tener que instalar las librerías directamente en la máquina donde se realiza la conversión y nos aporta aislamiento y desencapsulación en el proceso, así como una mayor facilidad para escalar horizontalmente la tarea en el caso de que el volumen de datos a convertir sea muy elevado.

Para agilizar el proceso anteriormente descrito, se automatizó la ejecución de las dos fases de *heudiconv* mediante un *script* [46] escrito en bash [47], así como la generación de los archivos TSV necesarios.

Por otro lado, las imágenes facilitadas por el Laboratorio de Neurociencia Cognitiva y Computacional (LNCyC) [30] del Centro de Tecnología Biomédica (CTB), están en formato NIfTI directamente. La conversión se hizo mediante una compresión con *gzip* y la generación de la estructura de directorios BIDS sin necesidad de recurrir a ningún conversor desarrollado por terceros. Al igual que en el caso anterior, la conversión se automatizó mediante un *script* escrito en *bash*. En este caso, la conversión no se hizo dentro de un contenedor Docker como sí se hizo en el caso de las imágenes de la base de datos OASIS.

3.3.3 Almacenamiento de los resultados

Cuando los datos ya han sido procesados, los resultados son almacenados en disco en un formato creado por Aramis Lab llamado CAPS [48] (*ClinicA Processed Structure*). El formato CAPS consiste en una estructuración jerárquica de los datos procesados por *Clinica*, cuyo objetivo principal es incluir en un único directorio todos los resultados de los diferentes *pipelines* de procesamiento y organizar los datos siguiendo los mismos patrones de diseño especificados en el formato DICOM.

Los resultados en CAPS están divididos en dos grupos diferenciados: los resultados relativos al sujeto de estudio, donde se encuentra la segmentación por tejidos, la corrección de sesgos y la normalización espacial, y los resultados relativos al grupo de estudio formado por todas las imágenes que han sido procesadas, donde se encuentra la plantilla DARTEL y el

espacio normalizado MNI. En la figura 3.8, podemos observar la segmentación por tejidos realizada por *Clinica*. Las zonas resaltadas se corresponden con las que más probabilidad tienen de ser el tipo de tejido especificado.

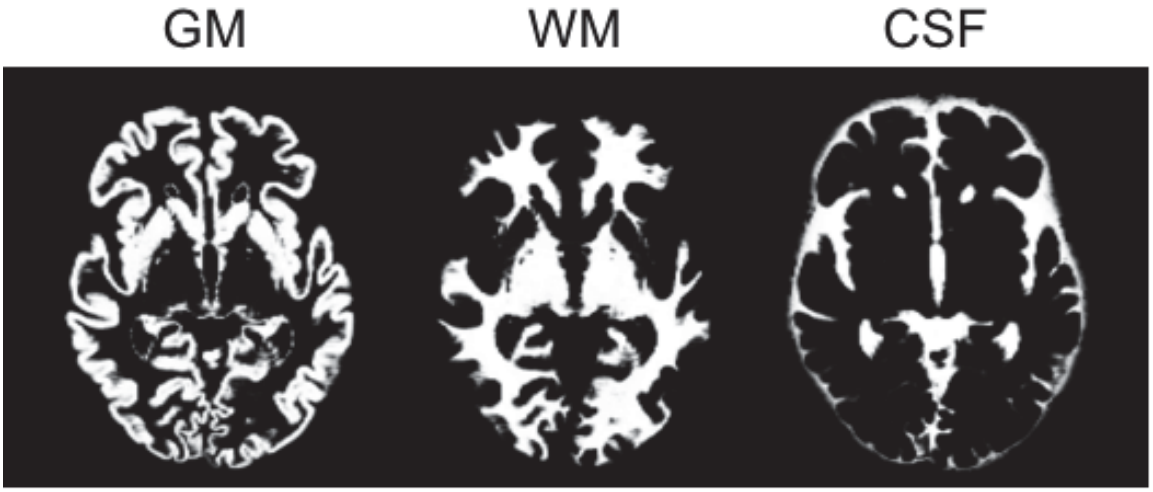

Figura 3.8: Segmentación por tejidos: materia gris, materia blanca y líquido cerebroespinal

Además de lo anteriormente descrito, dentro de la normalización espacial de cada sujeto también se han calculado los supervóxeles de la imagen, que usaremos para realizar análisis de los resultados con técnicas de *machine learning* [49] como veremos más adelante en este trabajo.

3.4 Enriquecimiento semántico mediante diccionarios ontológicos

Como ya se ha detallado con anterioridad, existe una necesidad de vincular el conocimiento de bajo nivel extraído directamente de características propias de la imagen con conceptos médicos de alto nivel. Para establecer esta relación entre conceptos, es necesario utilizar diccionarios ontológicos. En este caso, hacemos uso de UMLS (*Unified Medical Language System*), que ya integra otros diccionarios de utilidad, todo unificado en un único *metathesaurus* [50].

UMLS es un conjunto de ficheros y de utilidades software que aúnan una gran cantidad de vocabulario biomédico y estándares para facilitar la interoperabilidad entre diferentes sistemas. Así, UMLS integra terminología clave, clasificada y estandarizada. Dentro de las

principales utilidades de UMLS que propone la *United States National Library of Medicine*, se encuentran:

- Procesamiento de textos para extraer conceptos, relaciones o conocimiento.
- Facilitar la relación entre terminologías.
- Desarrollo de un sistema de recuperación de información.
- Extracción de terminología específica directamente del *Metathesaurus*.
- Creación y mantenimiento de una terminología local.
- Desarrollo de un servicio de terminología.
- Investigación de terminologías u ontologías.

En el capítulo 2, ya establecimos cómo funciona el diccionario ontológico UMLS. Los conceptos médicos y semánticos están agrupados por familias, siguiendo una jerarquía concreta, lo que nos permite, por ejemplo, buscar un concepto concreto limitando los resultados a una determinada familia de conocimiento.

Dentro del proyecto disponemos de dos instancias diferenciadas de UMLS. Una de uso general, a la que tienen acceso ambas ramas del proyecto, tanto texto como imagen, y otra instancia interna a los servicios de enriquecimiento semántico de las características extraídas de las imágenes. La instancia local está poblada con los conceptos que se sabe con certeza que se van a necesitar. De esta manera, muchos conceptos son asociados directamente con un término UMLS sin salir del ámbito del proyecto, lo que agiliza el procesamiento y reduce el tráfico de red.

Debido a los requerimientos específicos del proyecto, fue necesario diseñar e implementar una base de datos como parte de este trabajo, por lo que las relaciones establecidas por el módulo son almacenadas en una base de datos relacional MySQL. Esta base de datos está orientada a imagen, es decir, a la imagen se le asigna un id único vinculado con un paciente, además de almacenar el tipo de imagen, el diagnóstico, el género, la posición del paciente en el momento de la prueba, la parte del cuerpo examinada y todas las características extraídas. En la figura 3.9 podemos ver el esquema de esta base de datos para el caso de uso de cáncer de pulmón.

El *pipeline* de extracción de características descrito en la sección 3.2.2 no interactúa directamente con la base de datos, sino que delega toda la interacción con la base de datos en el módulo desarrollado. De esta manera, si los detalles de la conexión con la base de datos

3.4 Enriquecimiento semántico mediante diccionarios ontológicos

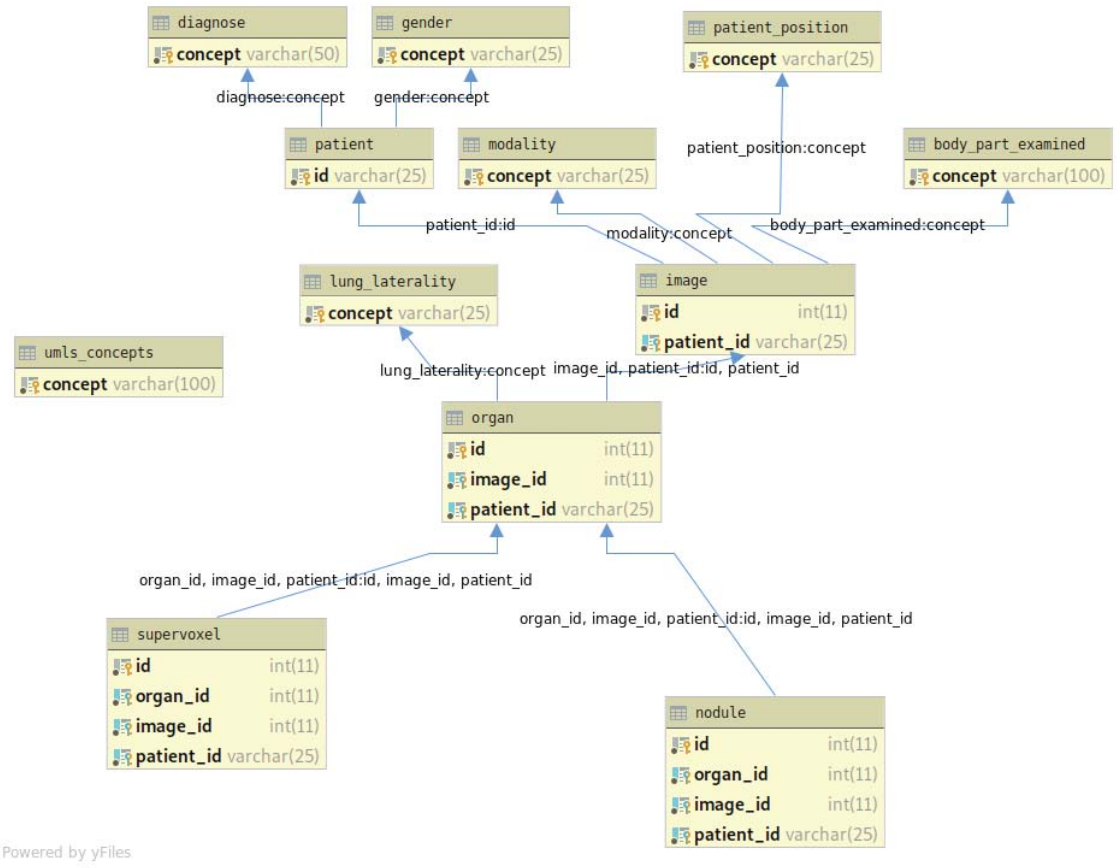

Figura 3.9: Diagrama de la base de datos interna del módulo

cambian, incluso si se decide cambiar el tipo de base de datos por cualquier otra, no sería necesario modificar el *pipeline* de extracción de características, bastaría con adaptar el módulo al nuevo paradigma.

El módulo de enriquecimiento semántico recibe los datos necesarios del paciente, de la imagen, del órgano y del nódulo y se encarga de crear los modelos necesarios con la estructura necesaria para su posterior almacenamiento en la base de datos local.

Cada atributo de cada uno de los modelos que deba tener un concepto ontológico asociado, es buscado en la instancia local de UMLS. En caso de que un concepto no se encuentre en la base de datos local de UMLS, éste se buscaría directamente en la instancia general de UMLS. Si el concepto tampoco es encontrado en la instancia general de UMLS, la inserción no se realiza y el módulo devuelve un error.

Se barajaron diferentes posibilidades sobre cómo proceder en el caso de que uno de los atributos que debe tener un concepto ontológico asociado no estuviera presente en UMLS. Una posibilidad es realizar la inserción en la base de datos sin establecer la relación entre ambos conceptos y sin generar ningún tipo de aviso o de error. Esta opción se descartó por la falta de *feedback* que provocaría, ya que habría conceptos que no estarían relacionados y no se podrían detectar en tiempo de ejecución.

También se contempló la opción de no realizar la inserción en la base de datos del modelo que no tuviera todos sus conceptos enriquecidos semánticamente, lanzar una excepción en tiempo de ejecución y continuar procesando el siguiente modelo recibido.

Finalmente, por las necesidades del proyecto, se tomó la decisión de diseño de que, si algún atributo no puede ser relacionado con un concepto de UMLS, la inserción falla y el módulo detiene su ejecución sin procesar los modelos restantes. De esta manera, nos aseguramos de que todos los datos procesados contengan las características requeridas y de que no se quede ninguna imagen procesada sin almacenar en la base de datos con sus correspondientes relaciones de conceptos ontológicos.

3.5 Tecnologías adicionales utilizadas

3.5.1 Docker

Docker es una herramienta que facilita la creación, el despliegue y la ejecución de aplicaciones utilizando contenedores. Los contenedores permiten empaquetar software de cualquier tipo con todas las dependencias necesarias. Esta encapsulación permite ejecutar la aplicación siempre en el mismo entorno, de manera que el paquete puede ser portado a cualquier máquina ejecutando cualquier sistema operativo compatible con Docker.

En cierta medida, Docker es similar a una máquina virtual, con la salvedad de que, en lugar de virtualizar una máquina al completo, Docker hace uso del *kernel* de la máquina anfitriona y solamente empaqueta las dependencias que no están instaladas.

Esta tecnología de contenedores ha sido utilizada para el despliegue de todos los servicios utilizados a lo largo de este proyecto. Además, aporta desacoplamiento, una capa extra de seguridad, compatibilidad multiplataforma y aislamiento entre procesos.

3.5.2 Jupyter Notebook

Jupyter Notebook es una aplicación web de código abierto que permite desarrollar y ejecutar código directamente desde el navegador, así como documentar todo el proceso. Jupyter ofrece compatibilidad con diversos lenguajes de programación entre los que se incluye *Python*, el usado durante este proyecto.

En el proyecto que se describe en este documento, Jupyter ha sido usado para desarrollar y ejecutar el código de la extracción de características para las imágenes de TAC de pulmón, proceso descrito en la sección 3.2.2.

3.5.3 PyCharm IDE

El IDE (*Integrated Development Environment*) utilizado para el desarrollo del módulo de enriquecimiento semántico ha sido PyCharm [51], un IDE desarrollado íntegramente por JetBrains. PyCharm, además de disponer de autocompletado y sugerencia de código, dispone de otras muchas facilidades a la hora de desarrollar código Python, entre las que se encuentran:

- Intérprete Python embebido.
- Búsqueda y descarga de librerías directamente de los repositorios oficiales de Python.
- Integración con herramientas de control de versiones.
- Sugerencias de modificación en la organización del código para seguir PEP 8 [52], guía de estilo oficial de Python.

3.6 Aprendizaje automático

Tras procesar las imágenes de Resonancia Magnética utilizando el *pipeline* de *Clinica* descrito en la sección 3.3.2, se aplicaron algoritmos de aprendizaje automático a las imágenes resultantes. La estrategia utilizada fue *hold-out* [53], que reduce la carga de trabajo en la máquina ya que solamente necesita una iteración al ser tratados los datos como independientes. Se utilizó la implementación del algoritmo desarrollada por Aramis Lab [54], el mismo grupo de desarrolladores del software *Clinica*, que está ligeramente modificada para ser más eficiente si se utiliza junto con los datos CAPS generados por los *pipelines* de procesamiento de su propia herramienta. Este algoritmo es el mismo que ellos usaron en su paper de presentación de *Clinica* [32], solo que en esa ocasión fue con la base de datos ADNI [55], un

repositorio de imágenes de cerebro de pacientes que padecen de la enfermedad de Alzheimer similar a OASIS.

Las métricas que se han utilizado para determinar la calidad de los resultados son las siguientes:

- **Precisión**: Métrica que determina el nivel de precisión del modelo predictivo. Explicado de manera sencilla, sería la tasa de aciertos sobre el total de casos estudiados. La fórmula que la representa es la siguiente:

$$Precision = \frac{VP + VN}{Total}$$

  donde VP son los verdaderos positivos y VN son los verdaderos negativos.

- **NPV**: *Negative Predictive Value*. Métrica que determina el ratio de acierto de negativos sobre el total de los sujetos clasificados como negativos por el modelo. La fórmula es:

$$NPV = \frac{VN}{VN + FN}$$

  donde FN son los falsos negativos y VN son los verdaderos negativos.

- **PPV**: *Positive Predictive Value*. Valor que representa el ratio de acierto de positivos sobre el total de los sujetos etiquetados como positivos por el modelo. La fórmula es la siguiente:

$$PPV = \frac{VP}{VP + FP}$$

  donde FP son los falsos positivos y VP son los verdaderos positivos.

- **Sensibilidad**: Métrica que indica la precisión del modelo a la hora de clasificar como enfermos de Alzheimer a los sujetos que, verdaderamente, padecen la enfermedad. La fórmula es la siguiente:

$$Sensibilidad = \frac{VP}{VP + FN}$$

  done VP son los verdaderos positivos y FN son los falsos negativos.

- **Especificidad**: Valor que determina la precisión del modelo a la hora de clasificar como sanos a los sujetos que no padecen la enfermedad de Alzheimer. La fórmula es:

$$Especificidad = \frac{VN}{VN + FP}$$

  done VN son los verdaderos negativos y FP son los falsos positivos.

Dentro de las imágenes procesadas, disponemos de dos grupos: las imágenes provenientes de la base de datos OASIS, pertenecientes a 416 sujetos, y las imágenes provistas por el Laboratorio de Neurociencia Cognitiva y Computacional (LNCyC), que ascendían a 150. Las imágenes de OASIS están balanceadas en cuanto a diagnóstico, es decir, se dispone, aproximadamente, de un 50 % de sujetos sanos y un 50 % de sujetos enfermos. En cuanto a las imágenes del LNCyC, se distribuyen de la siguiente manera:

- 50 de pacientes sanos (grupo de control)
- 50 de pacientes enfermos
- 50 de pacientes que sufrían pérdidas subjetivas de memoria pero de los que no se disponía de un diagnóstico

Estas 150 imágenes, junto con las 416 de las que disponíamos del repositorio OASIS, formaban una cantidad suficiente de imágenes como para realizar un estudio del que extraer conclusiones relevantes.

Para las pruebas, se establecieron dos etiquetas que se aplicaban a cada uno de los pacientes según sus características: sano o enfermo. Todas las pruebas fueron realizadas en un servidor dedicado, el mismo servidor donde se ejecuta el software *Clinica*, de manera que los datos no debían viajar de un servidor a otro y se aprovechaba la capacidad de cómputo de una máquina mucho más potente que un portátil personal. Debido a la gran cantidad de tiempo que consume la ejecución de un *pipeline* completo del software *Clinica*, no se pudieron realizar todas las pruebas que se hubieran deseado durante el tiempo que duró este trabajo.

3 Metodología

4

Desarrollo

4.1 Imágenes de Tomografía Axial Computerizada

4.1.1 Desarrollo del módulo de búsqueda de conceptos ontológicos

El módulo de relación de conceptos que aquí se describe está estructurado y programado de la manera más genérica posible. El programa recibe como parámetro un fichero de configuración con los detalles de la conexión con la base de datos local del módulo. Queda a elección del administrador el poblar o no la instancia local de UMLS asociada a esta base de datos con los términos a los que se tenga certeza que se va a hacer referencia según el caso de uso concreto. En la figura 4.1 podemos ver en detalle el fichero de credenciales que el módulo recibe.

```
# ip;user;pass;database;port
"192.168.1.10";"iasis-image";"RDI8lmNBBkKWPT5LoxXM";"iasis-image";"3307"
```

Figura 4.1: Detalle del fichero de credenciales que el módulo recibe como parámetro

4 Desarrollo

Aunque el módulo se realizó específicamente para el proyecto que nos ocupa, podría integrarse en otro proyecto de características similares simplemente cambiando la configuración de la conexión con la base de datos interna al ámbito del módulo y modificando los modelos para adecuarse al caso de uso en el que se integrara. El software que importe el módulo delegaría en él las tareas de inserción y consulta de la base de datos, así como la búsqueda de términos ontológicos. En la figura 4.2 se muestra cuál sería el flujo de información. El programador ejecutaría el programa en el que el módulo se integrara. Dicho software invocaría al módulo cuando fuera necesario, que hará las consultas correspondientes tanto a la base de datos local como a la instancia externa de UMLS.

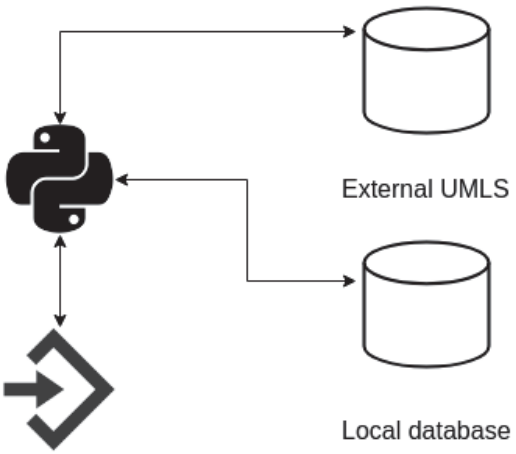

Figura 4.2: Flujo de información del módulo

Se contempló la posibilidad de utilizar algún tipo de ORM [56] para Python como *SQLAlchemy* [57] o el que viene integrado dentro de Django [58], un *framework* de desarrollo web desarrollado en Python [59], pero las modificaciones que había que realizar al comportamiento por defecto del *framework* eran demasiado numerosas y no justificaban el aumento de la carga de trabajo que eso iba a suponer. Además, la principal ventaja de utilizar un *framework* ORM es la posibilidad de usarlo como si fueran objetos de un lenguaje de alto nivel en lugar de tener que escribir directamente las consultas. Sin embargo, las consultas en nuestro caso no eran nada complejas, por lo que finalmente se decidió desarrollar manualmente la interacción con la base de datos mediante abstracciones y el uso de repositorios [60]. La organización interna del módulo está explicada con detalle en el siguiente apartado.

4.1.2 Configuración interna del módulo

El módulo, internamente, está estructurado de manera que para el programador que lo importe en su programa sea transparente la consulta que se realiza a la base de datos. Cada tabla de la base de datos tiene asociado un modelo que el programador rellena con los datos correspondientes. Los modelos se inyectan como parámetro a los repositorios, que son los que se encargan de comprobar que el modelo es correcto y que contiene los datos mínimos necesarios para realizar la inserción. Así mismo, se encarga de las consultas necesarias sobre cada tabla de la base de datos.

Para realizar la conexión a la base de datos, se dispone de una clase *Connector*, que lee del fichero de configuración recibido como parámetro las credenciales y devuelve un objeto de conexión con la base de datos, que es usado por cada repositorio para realizar las consultas pertinentes. De esta manera, la conexión está centralizada en un solo objeto siguiendo el patrón de diseño *singleton* [61], lo que lo convierte en un objeto común a todo el módulo y le aporta fiabilidad y mantenibilidad.

Todas las operaciones que se realizan contra la base de datos son gestionadas por un objeto de tipo *Manager*. La centralización de la gestión de las conexiones a la base de datos aporta robustez, fiabilidad y mantenibilidad al código, mientras que para el programador que implemente este gestor, aporta sencillez de uso ya que únicamente interactúa con un servicio.

El *manager* interactúa directamente con los repositorios, que son los encargados de realizar la lógica de negocio. De esta manera, disponemos de un repositorio por cada modelo de datos:

- Imagen
- Paciente
- Órgano
- Nódulo

La figura 4.3 es un ejemplo de una de las funciones incluidas en los repositorios. En concreto, esta función recibe un paciente como parámetro, comprueba que, efectivamente, se trata de una instancia de tipo Paciente y lo inserta en la base de datos.

Como ya hemos mencionado anteriormente, queda a elección del programador poblar previamente la instancia local de UMLS con los conceptos que conozca que se van a utilizar o no. El propio módulo, buscará primero en la base de datos local. De encontrarse el concepto, lo asociará al término de alto nivel correspondiente. Si no se encontró el concepto ontológico en la instancia local, se recurrirá a la base de datos genérica de UMLS. Una vez se

```
def save(patient):
    from Models.models import Patient
    from Utils import functions as f
    import DatabaseManager.manager as m

    if type(patient) is not Patient:
        f.eprint("Patient instance expected. Received", type(patient).__name__, "instance instead.")
        exit(1)
    else:
        controller = m.ImageDataController()
        controller.insert_into_patient(patient)
        controller.close_connection()
```

Figura 4.3: Función de inserción en la tabla de pacientes

haya obtenido el concepto, se añadirá la relación de ese concepto ontológico con el término de alto nivel a la base de datos local de UMLS. De esta manera, nos aseguramos de recurrir solamente a la base de datos externa, como máximo, una vez por concepto. El detalle de código que se muestra en la figura 4.4 muestra la función para buscar un CUI en la base de datos general. Recibe como parámetro el concepto a buscar y devolverá el CUI correspondiente si se encuentra o nulo en caso contrario.

```
def search_cui_general_umls(concept_string):
    from DatabaseManager import manager as m

    query = "SELECT DISTINCT MRCONSO.CUI FROM umls.MRSTY LEFT OUTER JOIN umls.MRCONSO ON MRSTY.CUI=MRCONSO.CUI " \
            "WHERE STR = %s"

    manager = m.UmlsController()
    cursor = manager.cursor

    cursor.execute(query, (concept_string, ))
    fetchone = cursor.fetchone()
    if fetchone is not None:
        return fetchone[0]
    else:
        return None
```

Figura 4.4: Función para la búsqueda de un concepto en la base de datos general de UMLS

Por último, el módulo dispone de un paquete con utilidades externalizadas a funciones para un mayor mantenimiento y una mejor usabilidad. La porción de código que se muestra en la figura 4.5 corresponde a la función externa que genera la consulta de inserción de cualquier elemento a cualquier tabla. Recibe el nombre de la tabla y la lista de elementos a insertar como parámetro y devuelve la consulta de inserción correspondiente ya saneada.

```
def insert_query_generator(table, params_list):
    values = ['%s' for i in range(0, len(params_list))]
    query = 'INSERT INTO ' + table + ' (%s)' % ', '.join(params_list) + ' VALUES (%s)' % ','
.join(values)

    return query
```

Figura 4.5: Función de generación de consulta genérica para inserción

4.1.3 Integración del módulo con el *pipeline* utilizado en IASIS

Tras el procesamiento de las imágenes mediante los scripts de Matlab para la generación de las máscaras de los pulmones y los nódulos y la extracción de características Radiomics con la librería *Pyradiomics*, estos datos se debían servir a través de una API [62], por lo que debían almacenarse de alguna manera persistente. En un principio, se generaba un fichero JSON [29], lo que era bastante conveniente, ya que, por norma general, se interactúa con las API enviando y recibiendo ficheros JSON. Sin embargo, las necesidades del proyecto IASIS requirieron de una consulta más minuciosa y más pormenorizada de los datos generados, por lo que era necesario incluirlos en una base de datos contra la que se pudieran ejecutar consultas.

Se planteó la posibilidad de almacenar los ficheros generados en una base de datos *MongoDB* [63], que está basada en ficheros JSON y de esta forma aprovechar cómo se estaban generando los resultados. Sin embargo, también era necesario integrar los resultados de la rama de imagen con los obtenidos en la rama de texto del proyecto, donde todos los resultados estaban siendo almacenados en una base de datos relacional MySQL [19], por lo que lo óptimo y más sencillo era modificar cómo se generaba la salida de los datos del *pipeline* de imagen para hacerlo coincidir con la estructura seguida en la rama de texto del proyecto.

El código de este gestor fue desarrollado utilizando *git* [64] para el control de versiones, una herramienta de código abierto para el manejo y la organización de los ficheros. En concreto, se utilizó la implementación desarrollada por *Gitlab* [65], una solución gratuita y con posibilidad de ser desplegada en un servidor privado sin perder funcionalidades, lo que supone un extra de privacidad.

Para integrar el gestor desarrollado en el *pipeline* de extracción de características de imágenes de TAC de pulmón, basta con clonar el repositorio e importar el paquete de la misma manera que se haría con cualquier otra librería estándar de Python.

En la figura 4.6, se muestra un diagrama con las diferentes etapas que atraviesan las imá-

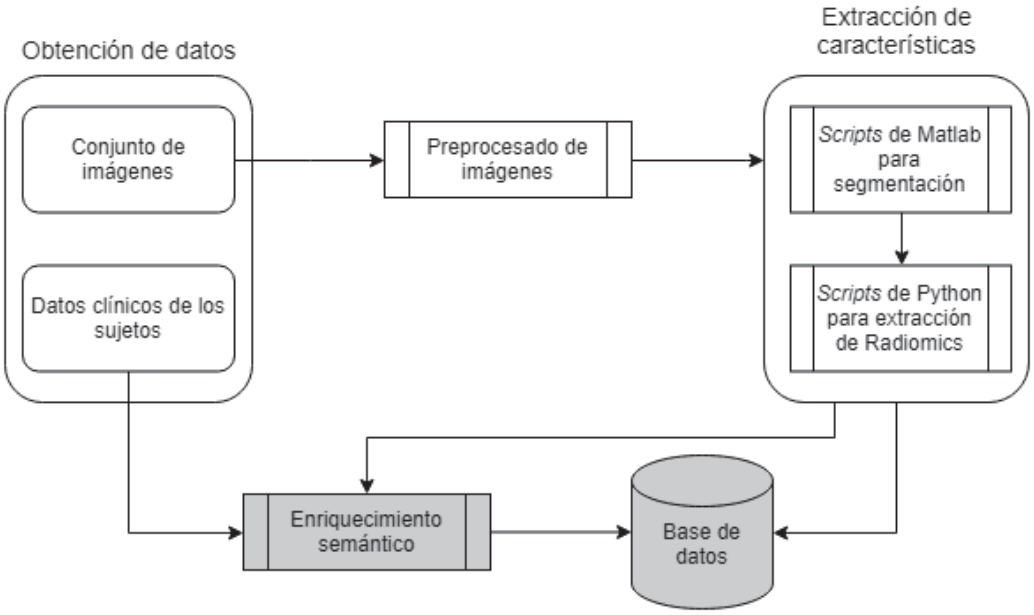

Figura 4.6: Diagrama de las etapas de trabajo con las imágenes de TAC. Las etapas sombreadas son las descritas en este trabajo.

genes de TAC de pulmón desde su obtención. Las etapas sombreadas son las correspondientes con el desarrollo descrito en este trabajo.

4.2 Imágenes de Resonancia Magnética

4.2.1 Ejecución de modelos predictivos para la enfermedad de Alzheimer

Como establecimos anteriormente, hemos utilizado el algoritmo de aprendizaje automático *Repeated Holdout*, en concreto, una implementación desarrollada por Aramis Lab. Siguiendo la documentación oficial de *Clinica* [66], a todos sus modelos predictivos se les puede especificar qué subconjunto de pacientes quiere utilizarse para entrenamiento y qué subconjunto de pacientes quiere utilizarse para la etapa de pruebas. De esta manera, se realizaron varias iteraciones de aprendizaje automático sobre los resultados generados por el *pipeline t1-volume*. En cada una de ellas, se cambiaba el conjunto de entrenamiento y el conjunto de pruebas.

4.2 Imágenes de Resonancia Magnética

La motivación principal de este procesamiento con *Machine Learning* de los resultados es comprobar la precisión de *Clinica* a la hora de generar los espacios normalizados DARTEL y MNI. Así, cuanto más similares sean las imágenes procesadas entre sí y menos disparidad haya entre sus características, menor discrepancia habrá entre el espacio normalizado y cada una de las imágenes y, por ende, mejor será el modelo generado por el algoritmo de aprendizaje automático.

La variable estudiada en este caso es binaria, ya que se intenta estimar si el sujeto está sano o enfermo de la enfermedad de Alzheimer, todo ello atendiendo únicamente a las características extraídas por *Clinica*, como son las probabilidades de los diferentes tejidos en ciertas regiones o los supervóxeles de las imágenes. Esta información, combinada con la información clínica de cada uno de los sujetos, como es la edad, la mano predominante o el sexo, nos permite generar un modelo con el que poder realizar predicciones. En el capítulo siguiente se muestran los resultados de todas estas iteraciones.

En la figura 4.7, se muestra un diagrama con las diferentes etapas que atraviesan las imágenes de MRI de cráneo desde su obtención. Las etapas sombreadas son las correspondientes con el desarrollo descrito en este trabajo.

4 Desarrollo

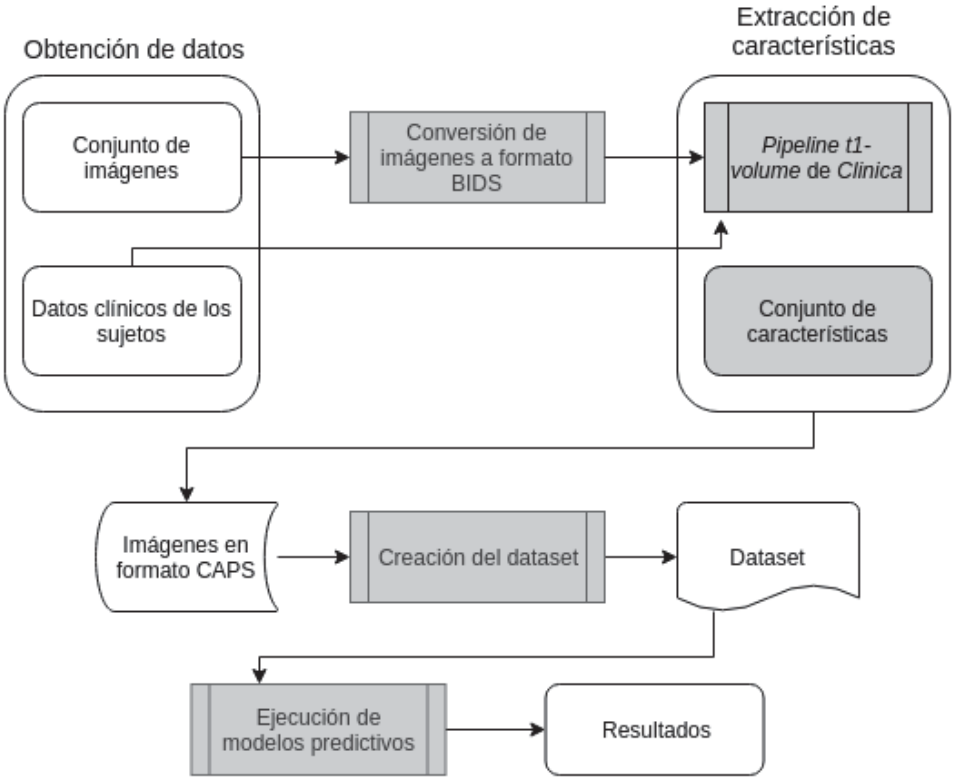

Figura 4.7: Diagrama de las etapas de trabajo con las imágenes de MRI. Las etapas sombreadas son las descritas en este trabajo.

5

Resultados y discusión

5.1 Módulo de enriquecimiento semántico

En este trabajo hemos desarrollado una librería en *Python* para el enriquecimiento semántico con conceptos ontológicos de términos médicos asociados a características de bajo nivel de imágenes médicas. A través de este módulo, hemos podido vincular diferentes conceptos médicos con una ontología universal y etiquetarlos de manera unívoca.

El módulo ha sido desarrollado siguiendo los principales patrones de diseño de la programación orientada a objetos y de la gestión de conexiones a bases de datos. Esto aporta al módulo robustez y facilidad a la hora de mantener el código actualizado o de realizar modificaciones en él. Es ligero, sencillo de usar, no almacena ningún tipo de información en disco y dispone de control de errores al realizar diversas operaciones. Además, la interacción del programador con la librería está centralizada en el *Manager*, que es el servicio con el que el usuario interactúa, delegando en la librería las diferentes consultas y lógicas de negocio que se realicen con los datos.

5.2 Base de datos estructurada

La librería desarrollada, además de realizar la búsqueda de los conceptos ontológicos correspondientes con los atributos recibidos, almacena toda la información recabada de manera persistente en una base de datos, cuyo esquema detallamos en la figura 3.9. Esta base de datos, partiendo de una imagen, nos permite obtener todas las características asociadas a ella extraídas durante las diferentes etapas de procesamiento. Además, al tratarse de una base de datos relacional, cuyo modelo entidad-relación ha sido desarrollado siguiendo las buenas prácticas de la programación SQL, aseguramos la integridad referencial de todos y cada uno de los datos insertados en ella.

Por un lado, el servidor MySQL que soporta esta base de datos ha sido desplegado utilizando contenedores de Docker, por lo que podemos portarlo a cualquier máquina con cualquier sistema operativo o externalizar el servicio y ejecutarlo en una máquina diferente a donde se esté trabajando con el módulo. Por otro lado, las credenciales y los detalles de la conexión de la librería desarrollada con esta base de datos son leídas directamente por el módulo desde un archivo en disco, lo que nos permite cambiar los detalles de la conexión, ya sea el usuario, la contraseña, la IP en la que está ejecutando el servidor, el nombre de la base de datos o el puerto expuesto por el servicio.

En definitiva, el servicio de MySQL que debe ejecutarse junto con este módulo es ligero, configurable y externalizable, lo que nos aporta desacoplamiento y posibilidad de escalabilidad horizontal.

5.3 Aprendizaje automático

Se llevaron a cabo diferentes experimentos, siempre utilizando el algoritmo *Repeated Holdout*, pero cambiando los grupos de entrenamiento y de prueba. En concreto, se llevaron a cabo cuatro agrupaciones diferentes entre todos los pacientes de los que se disponían. A continuación, se muestran los experimentos realizados junto con sus correspondientes resultados.

- El primer experimento consistió en entrenar al clasificador con las imágenes de OASIS y, como conjunto de pruebas, tomamos las imágenes del laboratorio LNCyC.

Accuracy	NPV	PPV	Sensitivity	Specificity
0.66	0.66	0.67	0.7	0.62

Tabla 5.1: Entrenamiento con OASIS y prueba con LNCyC

- El segundo experimento consistió en utilizar los sujetos de OASIS mayores de 60 años como conjunto de entrenamiento para, posteriormente, probar nuestro clasificador con los sujetos de las imágenes del LNCyC

Accuracy	NPV	PPV	Sensitivity	Specificity
0.69	0.69	0.65	0.56	0.82

Tabla 5.2: Entrenamiento con OASIS (sujetos mayores de 60 años) y prueba con LNCyC

- Para el tercer experimento, utilizamos las imágenes de OASIS y las de control y enfermos del LNCyC para el conjunto de entrenamiento, mientras que el conjunto de pruebas estaba formado por las imágenes de quejas del LNCyC.

Accuracy	NPV	PPV	Sensitivity	Specificity
0.58	0.95	0.28	0.89	0.51

Tabla 5.3: Entrenamiento con OASIS, control y enfermos de LNCyC y prueba con quejas

- Para el cuarto y último experimento, utilizamos como conjunto de entrenamiento las imágenes de OASIS y como conjunto de pruebas las imágenes del LNCyC, en este caso, sin las 50 imágenes del conjunto de quejas.

Accuracy	NPV	PPV	Sensitivity	Specificity
0.66	0.66	0.61	0.48	0.84

Tabla 5.4: Entrenamiento con OASIS y prueba con LNCyC sin quejas

Como se puede observar, los resultados no permiten sacar ninguna conclusión satisfactoria de los modelos predictivos generados. La distribución de los sujetos en los diferentes grupos de entrenamiento y pruebas no son el problema, pues están hechos de la manera correcta, ya que OASIS es un conjunto de imágenes balanceadas en cuanto al sexo del sujeto, su lateralidad (diestro o zurdo) y su edad, y siempre ha sido usado como conjunto de entrenamiento.

Las dos primeras pruebas, el conjunto de prueba son todas las imágenes del LNCyC, es decir, grupo de control (50), grupo de sujetos diagnosticados con la enfermedad de Alzheimer (50) y grupo de quejas subjetivas de pérdidas de memoria (50). El tercer grupo, el de quejas, tiene el inconveniente de que ninguna de las dos clasificaciones posibles dentro de las dos clases utilizadas para el etiquetado (enfermo o sano), es correcta, ya que el objetivo que está pendiente de validar es ver si los pacientes clasificados como enfermos habían evolucionado desde el grupo de quejas subjetivas a enfermo. Por un lado, no hay un diagnóstico médico que

determine que padecen la enfermedad de Alzheimer, por lo que etiquetarlos como enfermos no sería correcto; por otro lado, tampoco sería preciso al 100 % etiquetarlos como sanos, puesto que el paciente se queja de pérdidas de memoria. Finalmente, se decidió catalogarlos aleatoriamente en alguna de las dos categorías para intentar minimizar un posible sesgo.

Por los motivos anteriormente expuestos, los valores que aparecen en la tabla 5.1 y en la tabla 5.2 muy probablemente estén desvirtuados por la presencia del conjunto de quejas en el grupo de sujetos de pruebas.

En el caso de la ejecución cuyos resultados aparecen en la tabla 5.3, se intentó entrenar con todas las imágenes de las que se tuviera un diagnóstico fiable y hacer la etapa de prueba con las imágenes de las que no se disponía un diagnóstico fiable. Es interesante el resultado de la precisión, pues este modelo tiene un 58 % de precisión, cuando el 56 % de los sujetos del conjunto de pruebas había sido etiquetado previamente como enfermo. Aún así, no nos permite extraer conclusiones significativas.

Por último, en los resultados de la tabla 5.4 observamos un *accuracy*, un NPV y un PPV muy parejos. Sin embargo, aunque en la *specificity* nos movemos en unos valores bastante buenos, la *sensitivity* es bastante baja.

Para intentar aportar una posible explicación a estos resultados, debemos remitirnos a cómo funciona *Clinica* internamente. Como ya hemos detallado en el apartado 3.3.2, *Clinica* genera un espacio MNI normalizado y único partiendo de los resultados del procesamiento de todas y cada una de las imágenes. Esto puede provocar que, si las imágenes han sido registradas por diferentes instrumentos, en diferentes localizaciones o por diferentes técnicos, sean muy diferentes entre el conjunto de entrenamiento y el conjunto de pruebas (colores muy diferentes, ausencia de grises, brillos muy dispares, imágenes rotadas, etc.). Esto provoca que la clasificación dé peores resultados de los esperados, ya que tendríamos un espacio muy difuso y poco concreto que no se adecúa a ningún conjunto de imágenes en concreto, por lo que no se pueden extraer un número suficiente de características comunes entre todas las imágenes para poder hacer más precisas las predicciones de los modelos de aprendizaje automático.

5.4 Conclusiones

Atendiendo al trabajo realizado sobre las imágenes de TAC de tórax de los sujetos con cáncer de pulmón, se han enriquecido semánticamente las características extraídas de ellas y se ha almacenado toda esta información en una base de datos relacional de manera ordenada y estructurada. Por un lado, tener la información estructurada nos facilita las labores de búsqueda y de análisis de datos que puedan realizarse sobre los datos. Por otro lado, haber enriquecido semánticamente los conceptos médicos nos permite compartir esta información

con otros grupos de investigación o con otros hospitales sin que necesariamente utilicen el español como lengua vehicular.

En cuanto a las imágenes de MRI, los resultados no han resultado concluyentes y dan pie a continuar investigando en esa línea, ya que sería necesario investigar nuevos algoritmos que mejoraran la armonizaciónd de las imágenes. Como hemos mencionado anteriormente, esto se debe a la heterogeneidad de los datos, ya que en otros trabajos [67] donde se han utilizado imágenes de la misma procedencia, los resultados de los modelos de aprendizaje automático han resultado satisfactorios y con un alto nivel de acierto.

5 Resultados y discusión

6

Líneas futuras

Por limitaciones de tiempo y por lo que se demora *Clinica* en completar todas las etapas de su *pipeline*, hay pruebas que pudieron llevarse a cabo durante la realización de este trabajo. Por tanto, este trabajo da pie a continuar con la investigación para encontrar un modelo predictivo, partiendo de los desarrollados por Aramis Lab, que sea capaz de predecir, con una mayor precisión, si un sujeto padece o no la enfermedad de Alzheimer.

Un buen punto de partida para futuras investigaciones sería entrenar y probar el clasificador con imágenes que guarden una mayor similitud entre sí y, además de repetir las pruebas ya realizadas en este trabajo, hacer clasificadores utilizando otros algoritmos además de *Repeated Holdout*.

También cabe la posibilidad de, aprovechando que el software *Clinica* es un programa de código abierto y lo podemos modificar para adaptarlo a las condiciones del caso de uso, añadir nuevas etapas al *pipeline* o implementar nuevos algoritmos para generar modelos predictivos. Así, partiendo de la base que este *framework* nos ofrece, sería más rápido y sencillo obtener resultados que podamos aprovechar o, al menos, modificar nuestros algoritmos para hacer más precisas las predicciones generadas.

En cuanto a la librería Python de enriquecimiento semántico, está completamente abierta a modificaciones y a extensiones. Esto nos permite adaptarla a nuestro caso de uso añadiendo o modificando los modelos relacionales existentes. Además, si fuera necesario, sería posible cambiar el tipo de base de datos utilizada para el almacenamiento persistente, pudiendo elegir

otra implementación del modelo relacional o apostando por una base de datos no relacional, aunque la modificación del tipo de base de datos o de su estructura requeriría de una mayor modificación del módulo, ya que las consultas no serían reutilizables.

Durante todo este trabajo se ha trabajado en entornos Linux, pero al ser Python un lenguaje de programación interpretado y multiplataforma y al estar el servicio de la base de datos configurado para ejecutarse dentro de un contenedor de Docker, existe la posibilidad de utilizar este *framework* en entornos Windows o en entornos Unix sin tener que modificar el código fuente ni sus dependencias.

Bibliografía

[1] (Sep. de 2018). IASIS Project, dirección: http://project-iasis.eu/.

[2] (Jun. de 2019). Ontology (Computer Science), dirección: http://tomgruber.org/writing/ontology-definition-2007.htm.

[3] (Sep. de 2018). WHO | World Health Organization, dirección: http://www.who.int/.

[4] (Sep. de 2018). Cancer, dirección: http://www.who.int/news-room/fact-sheets/detail/cancer.

[5] (Sep. de 2018). Demencia, dirección: http://www.who.int/en/news-room/fact-sheets/detail/dementia.

[6] C. R. J. Prashanthi Vemuri, «Role of structural MRI in Alzheimer's disease», *BioMed Central*, 2010.

[7] (Oct. de 2018). OASIS Brains - Open Access Series of Imaging Studies, dirección: https://www.oasis-brains.org/.

[8] (Sep. de 2018). Clinica, dirección: www.clinica.run.

[9] Z. S. M. Dimitris, «Large-Scale medical image analytics: Recent methodologies, applications and Future directions», *Elsevier*, págs. 98-101, 2016.

[10] (Sep. de 2018). SNOMED, dirección: http://www.snomed.org/.

[11] (Sep. de 2018). ICD-10, dirección: https://icd.who.int/browse10/2016/en.

[12] (Sep. de 2018). UMLS, dirección: https://www.nlm.nih.gov/research/umls/.

[13] O. Bodenreider, «The Unified Medical Language System (UMLS): integrating biomedical terminology», *Nucleic Acids Research*, vol. 32, págs. D267-D270, 2004.

[14] S. S. K. M. M. M. M. S. C. A. H. M. C. Dorin, «Semantic annotation of medical images», *SPIE Medical Imaging*, vol. 7628, 2010.

[15] M. M. R. S. S. Michael, «RadSem: Semantic Annotation and Retrieval for Medical Images», *European Semantic Web Conference*, vol. 5554, págs. 21-35, 2009.

[16] F. K. W. H. W. T. N. S. J. S. S. P. T. R. Laura, «Integrating SNOMED CT into the UMLS: An Exploration of Different Views of Synonymy and Quality of Editing», *Journal of the American Medical Informatics Assocation*, págs. 486-494, 2005.

[17] U. V. C. P. I. J. H. S. V.-V. M. M. E. C. Fabio, «Semantic annotation for knowledge management: Requirements and a survey of the state of the art», *Web Semantics: Science, Services and Agents on the World Wide Web*, vol. 4, págs. 14-28, 2006.

[18] K. A. D. S. K. J. L. C. H. P. F. M. F. Dagan, «Adapting content-based image retrieval techniques for the semantic annotation of medical images», *Computerized Medical Imaging and Graphics*, vol. 49, págs. 37-45, 2016.

[19] (Oct. de 2018). MySQL, dirección: `https://www.mysql.com/`.

[20] (Oct. de 2018). DICOM, dirección: `https://www.dicomlibrary.com/dicom/`.

[21] (Nov. de 2018). NIfTI-1 Data Format, dirección: `https://nifti.nimh.nih.gov/nifti-1/`.

[22] (Jun. de 2019). Fiji: ImageJ, dirección: `https://fiji.sc/`.

[23] (Sep. de 2018). Radiomics, dirección: `http://www.radiomics.io`.

[24] (Oct. de 2018). Radiomics, dirección: `http://www.radiomics.io/pyradiomics.html`.

[25] (Ene. de 2019). Radiomics, dirección: `https://www.radiomics.io/data.html`.

[26] (Oct. de 2018). MATLAB - MathWorks, dirección: `https://www.mathworks.com/products/matlab.html`.

[27] (). supervoxel, dirección: `https://en.wiktionary.org/wiki/supervoxel`.

[28] (Oct. de 2018). Python, dirección: `https://www.python.org/`.

[29] (Oct. de 2018). JSON, dirección: `https://www.json.org/`.

[30] (Nov. de 2018). Laboratorio de Neurociencia Cognitiva y Computacional, dirección: `http://meg.ctb.upm.es/es/`.

[31] (Nov. de 2018). ARAMIS - Brain Data Science, dirección: `http://www.aramislab.fr/`.

[32] M.-O. H. S. D. T. E. O. C. Jorge Samper-González; Ninon Burgos; Sabrina Fontanella; Hugo Bertin, «Yet Another ADNI Machine Learning Paper? Paving The Way Towards Fully-reproducible Research on Classification of Alzheimer's Disease», 2017.

[33] (Dic. de 2018). Neuroimaging in Python, dirección: `https://nipype.readthedocs.io/en/latest/`.

[34] (Jun. de 2019). T1 weighted image | Radiology Reference Article, dirección: `https://radiopaedia.org/articles/t1-weighted-image`.

[35] (Jun. de 2019). DARTEL, dirección: `https://neurometrika.org/node/34`.

[36] (Jun. de 2019). MNI Space, dirección: `http://www.brainmap.org/training/BrettTransform.html`.

[37] N. T.-M. Edmund T.Rolls Marc Joliot, «Implementation of a new parcellation of the orbitofrontal cortex in the automated anatomical labeling atlas», *Science Direct*, vol. 122, págs. 1-5, 2015.

[38] (Oct. de 2018). Brain Imaging Data Structure, dirección: http://bids.neuroimaging.io/.

[39] K. J. G. T. A. V. D. C. R. C. C. S. D. E. P. D. G. F. S. S. G. T. G. Y. O. H. D. A. H. M. H. D. K. X. L. Z. M. C Poldrack, «The brain imaging data structure, a format for organizing and describing outputs of neuroimaging experiments», *Nature*, 2016.

[40] (Oct. de 2018). Gzip - GNU Project, dirección: https://www.gnu.org/software/gzip/.

[41] (Jun. de 2019). Tab-Separated Values, dirección: http://www.hep.by/gnu/gnumeric/file-format-tab.shtml.

[42] (Jun. de 2019). Dcm2Bids, dirección: https://github.com/cbedetti/Dcm2Bids.

[43] (Jun. de 2019). dcm2nii, dirección: https://www.nitrc.org/plugins/mwiki/index.php/dcm2nii:MainPage.

[44] (Nov. de 2018). heudiconv, dirección: https://github.com/nipy/heudiconv.

[45] (Jun. de 2019). Enterprise Container Platform | Docker, dirección: https://www.docker.com/.

[46] (Jun. de 2019). Scripting Language, dirección: https://www.computerhope.com/jargon/s/script.htm.

[47] (Oct. de 2018). Bash - GNU Project, dirección: https://www.gnu.org/software/bash/.

[48] (Nov. de 2018). CAPS - Clinica Documentation, dirección: http://www.clinica.run/doc/CAPS/.

[49] (Jun. de 2019). Machine Learning, dirección: https://towardsdatascience.com/machine-learning-from-first-principles-51a5e75a3c47.

[50] (Jun. de 2019). UMLS - Metathesaurus, dirección: https://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/.

[51] (Jun. de 2019). PyCharm: the Python IDE for Professional Developers by JetBrains, dirección: https://www.jetbrains.com/pycharm/.

[52] (Jun. de 2019). PEP 8 – Style Guide for Python Code, dirección: https://www.python.org/dev/peps/pep-0008/.

[53] (Jun. de 2019). Cross-validation, dirección: https://www.kdnuggets.com/2017/08/dataiku-predictive-model-holdout-cross-validation.html.

[54] (Oct. de 2018), dirección: https://gitlab.icm-institute.org/aramislab/AD-ML/blob/e45f5c469757cef8c87143fa3f2893ea1cf3a278/Paper_Specific_Versions/2018_NeuroImage/Code/experiments/1-main_classifications/ADNI_run_classifications.py.

[55] (Sep. de 2018). Alzheimer's Disease Neuroimaging Initiative, dirección: http://adni.loni.usc.edu/.

[56] (Jun. de 2019). An Overview of Object Relational Mapping & ActiveRecord, dirección: https://medium.com/@hidace/an-overview-of-object-relational-mapping-3255fa26d51f.

[57] (Jun. de 2019). SQLAlchemy - The Database Toolkit For Python, dirección: https://www.sqlalchemy.org/.

[58] (Jun. de 2019). Django ORM - Full Stack Python, dirección: https://www.fullstackpython.com/django-orm.html.

[59] (Jun. de 2019). The Web framework for perfectionists, dirección: https://www.djangoproject.com/.

[60] (Jun. de 2019). Repository Design Pattern, dirección: https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30.

[61] (Jun. de 2019). Singleton pattern, dirección: https://sourcemaking.com/design_patterns/singleton.

[62] (Jun. de 2019). Application Programming Interface, dirección: https://medium.com/@perrysetgo/what-exactly-is-an-api-69f36968a41f.

[63] (Nov. de 2018). Open Source Document Database | MongoDB, dirección: https://www.mongodb.com/.

[64] (Jun. de 2019). Git, dirección: https://git-scm.com/.

[65] (Jun. de 2019). The first single application for the entire DevOps lifecycle, dirección: https://about.gitlab.com/.

[66] (Oct. de 2018). Classification based on machine learning, dirección: http://www.clinica.run/doc/Pipelines/MachineLearning_Classification/.

[67] C. A. Ortiz Toro, «Algoritmos de segmentación semántica para anotación de imágenes», 2019. DOI: https://doi.org/10.20868/UPM.thesis.55407.

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Sat Jun 29 19:34:02 CEST 2019
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)