Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA DE VALLADOLID

Grado en Ingeniería Informática

PROCESAMIENTO DE HISTORIAS CLÍNICAS USANDO TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL

Alumno: Víctor Vaquero Martínez

Tutor: Valentín Cardeñoso Payo

PROCESAMIENTO DE HISTORIAS CLÍNICAS USANDO TÉCNICAS DE PROCESAMIENTO DE LENGUAJE NATURAL

Víctor Vaquero Martínez

Índice general

Índice de figuras

Índice de cuadros

Dedicado a
mi familia

Agradecimientos

Muchas gracias a mi familia por su eterno apoyo, a mi increíble hermano Pablo por hacer posible la idea, al profesor Roberto Martin Asenjo por facilitarnos las historias clínicas y a los estudiantes Juan Pedro Garcia Navas, Roberto Comenero Rollinson y José Ángel Álvarez Vazques por dedicar las horas necesarias para sacarlas; y finalmente a mi tutor Valentín.

Resumen

La información contenida en las historias clínicas es critica para la correcta toma de decisiones por los clínicos; en este proyecto desarrollamos un prototipo de procesado de datos en texto plano para la extracción automática de este tipo de información. Usamos herramientas y algoritmos de procesado de lenguaje natural (NLP) de libre acceso para resumir la información y transformarla a un formato más sencillo e útil, llevado todo a cabo sobre un conjunto de datos reales.

Palabras claves: nlp, lenguaje natural, historias clínicas, bioinformática, python

Abstract

Information contained in clinical histories is of critical importance for the decision making of the physicians; in this project we develop a plain text data processing prototype for the automatic extraction of clinical history data. We work with natural language processing tools and algorithms to summarise the text's information into a more useful and easy to work with format, all done with a set of real-world data.

Keywords: nlp, natural language, clinical histories, bioinformatics, python

Parte I

Aspectos generales

Capítulo 1

Descripción del proyecto

Este trabajo contiene tres partes diferenciadas. Primero, un desarrollo de la motivación del proyecto, así como su integración en el marco teórico actual del Procesamiento de Lenguaje Natural o **PLN**. Segundo, una descripción de la implementación del prototipo de extracción de datos, cuyo desarrollo es el objetivo final de este trabajo; en esta parte serán desarrolladas en profundidad las herramientas usadas, la arquitectura de la aplicación y la metodología aplicada para su creación. Por último, se mostrará el producto final junto con un análisis de su capacidad y fiabilidad en la tarea de extracción de información de forma selectiva y global.

En esta primera parte sera dada una breve introducción a la temática y a las tecnologías relevantes al proyecto. A continuación mencionaremos una primera descripción abstracta de los datos, sobre los cuales se centra el procesamiento, junto con una breve mención al motivo detrás de la creación de este proyecto y la intención práctica detrás de este. Por último serán listados el conjunto de objetivos, principales y secundarios, que desean ser realizados para el fin del trabajo.

1.1. Motivación

Las historias clínicas son una parte crítica del proceso de recogida de información en la práctica clínica y aún no existe ningún método estandarizado, en español, para su extracción automática. Si bien existen diversas herramientas en otros idiomas, e incluso propuestas similares en el territorio español, aún no hay un suficiente desarrollo como para que pueda dar fruto su aplicación practica. En este proyecto deseamos analizar cuales son los obstáculos que limitan en este momento el desarrollo de los sistemas automáticos de extracción de datos, además de investigar cuales son las soluciones propuestas, en el marco general de los datos clínicos y en el caso específico del lenguaje castellano. Por último, desarrollaremos estas ideas de manera práctica en un prototipo de herramienta de procesado de lenguaje natural; posteriormente estudiaremos sus resultados, sus éxitos y, lo más importante, sus limitaciones.

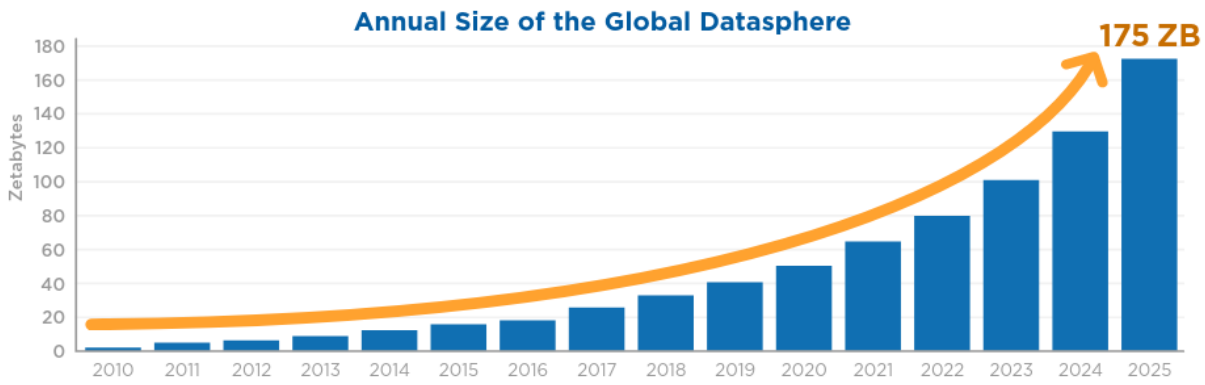

Figura 1.1: Tamaño de internet en Zetabites por año - Fuente IDC DataSphere[20]

1.2. Introducción

En la actualidad la cantidad de datos electrónicos accesibles, y por tanto explotables, ha incrementado a cotas previamente inconcebibles. En este momento existe una continua creación de información a una escala sin precedentes en la historia humana; esto ha traído el advenimiento de una nueva era de los datos, en la cual el BigData[1] y sus técnicas se han convertido en herramientas indispensables con las que acceder a todos estos nuevos documentos.

El mundo clínico no es indiferente a esta tendencia y durante los últimos años han crecido a un ritmo exponencial los documentos sanitarios en todas y cada una de las ramas. En esta situación, los facultativos se han visto sobrepasados por el número de datos que requieren de procesado manual y, debida a la especial situación del terreno médico, no es sencilla la inclusión en la práctica clínica de las nuevas formas y herramientas desarrolladas en los últimos años, debido a los estrictos requerimientos de privacidad, seguridad en el manejo de los datos sensibles y fiabilidad, ante información con múltiples requerimientos y garantías de credibilidad.

Son estas diversas complicaciones las que han dado origen a un conjunto de estándares digitales médicos, herramientas, procesos... que son locales y específicos para cada entorno, sin apenas conexión con el resto de hospitales o centros médicos, limitando así la compatibilidad y el desarrollo de sistemas más genéricos.

Por todo esto actualmente existe un movimiento para establecer diversos estándares internacionales, a través del los cuales facilitar el traspaso de información técnica, vocabulario, herramientas... requeridos para la automatización de estos procesos. Un ejemplo de esto es la propuesta **UMLS**[2] [2] de Estados Unidos que, en un principio para América, posteriormente para países de lengua inglesa y finalmente para un conjunto diverso de

[1] Algoritmos y técnicas de procesamiento de grandes cantidades de información, actualmente del orden de centenares de gigas o teras de datos

[2] Unified Medical Language System o Sistema de Lenguaje médico unificado

países e idiomas. Reúne un conjunto de software, diccionarios y estándares que permiten la interoperabilidad entre hospitales de todo el mundo. Actualmente, tras más de tres décadas de expansión y traducción de las fuentes base, está disponible para países como España, miembro que en este momento busca la digitalización de su salud pública.

Si nos centramos en la situación española, se observa que la práctica clínica española pose varias carencias en comparación con otras partes del mundo. A pesar de los esfuerzos mencionados previamente, y de muchos más que no lo han sido (como el **Plan de Impulso de las Tecnologías del Lenguaje**[18]), estamos muy por detrás del resto de los países en términos de recogida y digitalización de datos y, por ende, de su posterior uso práctico[8]. En un número muy grande de hospitales públicos (aunque menos en los privados) se siguen recogiendo las historias clínicas en papel, o se ha hecho hasta un tiempo reciente. Esto, junto con el rechazo a procesar toda esa información antigua, da problemas a la hora de modernizar la práctica clínica y retrasa en general el desarrollo y la investigación médica.

De esta manera, cada hospital español, de manera independiente, posee sus propias bases de datos, sus propios estándares, sus propios sistema de recogida, etc; de manera que, dependiendo del esfuerzo acometido y de la regulación provincial, es posible encontrar registros como el electrocardiograma[3] en papel o si se encuentran digitalizados están en formatos poco accesibles para su posterior procesamiento -como el formato PDF, útil para visualización por el personal pero no para su procesado automático–. Estos formatos "humanos", útiles para los facultativos por su sencillez y amplio uso, dificultan en gran medida (o imposibilita completamente) la recuperación y manipulación de la información sin un pesado pre-procesamiento.

Por suerte, el terreno del PLN ha avanzado de manera explosiva y permite actualmente, con relativa facilidad, solventar un número de los problemas mencionados previamente. Si está bien construido, un sistema PLN facilita el procesamiento de toda cantidad de información textual y en (casi) cualquier estructura o formato. Las herramientas para buscar conceptos, nombres, entidades en el texto existían previamente pero hasta ahora no poseíamos la capacidad computacional y las bases de datos necesarias para llevarlo acabo.

Debido a esto, han aparecido numerosas propuestas para, en el terreno clínico, aplicar estas tecnologías PLN; ejemplos de esto son: **Apache cTakes**[7], una herramienta de código libre bajo licencia Apache que facilita todas las etapas de la típica "pipeline" usada en desarrollos PLN. También existen herramientas comerciales como **Savana** [23], la cual ya incorpora diversos módulos para investigadores y hospitales en los cuales puedes, desde tan solo extraer información, hasta el completo análisis de los datos; todo esto realizado específicamente para el territorio Español y en el idioma castellano.

Así pues, no podemos sino esperar un futuro prometedor para este campo de la tecnología y creemos que es el momento para apostar por todas estas técnicas, e investigar cuales son los algoritmos que lo hacen posible.

[3]Registro de las señales eléctricas del corazón

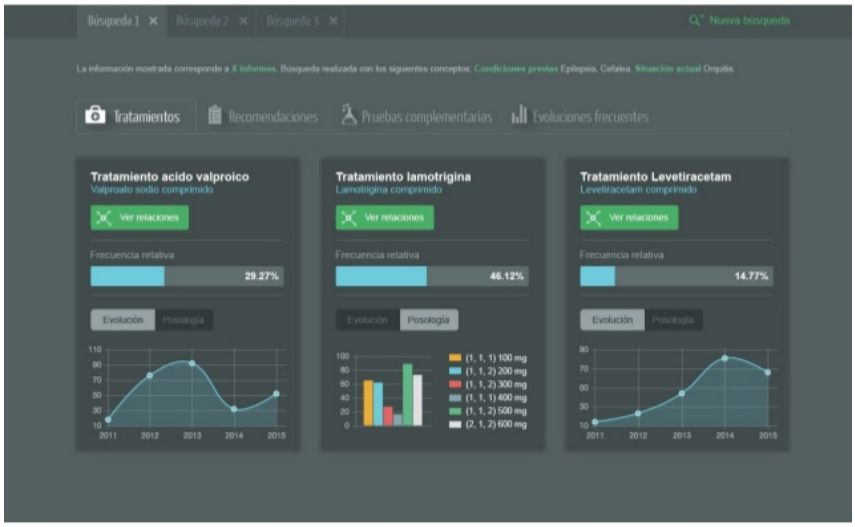

Figura 1.2: Ejemplo de un informe generado por la herramienta Savana - Fuente Savanamed[23]

Pero, antes de poder explicar cuales son las soluciones disponibles, necesitamos explicar en profundidad cual es el problema, cuales son los datos y cuales son las características que hacen el procesado de lenguaje natural en medicina distinto al resto.

1.3. Datos

El conjunto de información tratada en el ámbito médico tiene un conjunto de particularidades y restricciones atípicas en comparación con el resto de campos. Dada esta gran diferencia de estructura, contenido y forma del texto clínico respecto al resto de uso más general, no es posible, sin pasar por grandes dificultades, trasladar el conocimiento, las bases de datos, etc., de otros ámbitos a este. Si además añadimos que en medicina tan solo existen una fracción de las bases de datos que son usadas para el PLN más general (especialmente en Español), entenderemos la dificultad del desarrollo de programas y algoritmos especializados.

Esto no ha sido generado por una menor inversión en salud, sino que está provocado por el difícil acceso a estos documentos, pues están restringidos para no difundir sin permiso la información personal de pacientes y médicos. Así se ha limitado la generación de bases de datos accesibles al público con las cuales acelerar el desarrollo.

Además de la exigua información existente, estos documentos son por naturaleza muy complejos; aunque los informes suelen tener estructura, esta es dispar entre hospitales e incluso entre especialidades de un mismo centro. Por ejemplo, las secciones de sintomatología o de situación previa están organizadas de diversas formas dependiendo del objetivo

del documento, el significado de un acrónimo varía dependiendo de la especialidad (o incluso de la persona que realice el documento), las sentencias se conforman de manera acortada y de distinta manera siguiendo un patrón común, interno, de los residentes. Por esta razón, para estos tipo de procesado/extracción de información, se suelen restringir los profesionales a un conjunto de documentos más especifico, e.g. historias de altas de urgencias o únicamente aquellas de la especialidad de cardiología.

Finalmente, habiendo dado brevemente a conocer los problemas generales del PLN en medicina y la naturaleza de sus documentos, dejaremos para más adelante el desarrollo de cual es exactamente el corpus obtenido para este proyecto y las específicas restricciones del conjunto.

Informe clínico de alta

DATOS ASISTENCIALES	
Esp.:	Médico:
Fecha de Ingreso: Tipo de Ingreso: Urgente	Fecha / Hora de Alta:
Fecha de Nacimiento: País: Edad:85 años Sexo: varon	**Motivo de Alta:**

Motivo de consulta:

Ficmc.

Antecedentes:

Enfermedades familiares hereditarias:

Padre: IAM a los 60 años.
Madre: Cancer de colón 65 años.

Enfermedades previas:

Varón de 85 años,
-No alergias medicamentosas conocidas.
-Hábitos tóxicos: Fumador de 20 cigarrillos/día. No bebedor de dos litros diarios de cerveza.
-HTA. DM, No DL.
-Enfermedad tuberculosa diagnosticada a los 30 años, con instauración de varios ciclos de tratamiento que no llegó a completar; no sabe precisar los fármacos pautados, aunque refiere pauta de 9 meses. Resección de segmento apical del

Figura 1.3: Ejemplo de un informe clínico en español

1.4. Objetivos

El objetivo principal de la realización de este proyecto es la comprensión de cuales son los problemas que se encuentra un desarrollador al implementar un proceso de procesado

de lenguaje natural en el ámbito de la medicina.

Dada la promesa que ofrece este campo, deseamos explorar las capacidades y restricciones del mismo, cuales son las herramientas necesarias, cuales son las bases de datos requeridas para avanzar en el procesado y cuales son aquellas partes del proyecto que poseen la mayor dificultad para alcanzar la implementación de un producto final.

De este modo, se quiere desarrollar un prototipo funcional, con todas las características necesarias para cualificarlo como un sistema PLN completo, aunque en ningún caso se desea crear un sistema "final" capaz de resolver todos los problemas, encontrados a lo largo de la realización del mismo, o menos aún producir un producto profesional capaz de competir con las herramientas del mercado.

En conclusión, el tono que ha de tomar este proyecto es el de uno investigativo, con alas a explicar, tanto a los potenciales lectores como al mismo autor, el funcionamiento de los módulos principales que componen una aplicación real de PLN.

De esta manera, al final del desarrollo se desea obtener una aplicación capaz de, dada una historia clínica en un formato estándar, transformar la información contenida en el documento y expresarla en un lenguaje de representación intermedio, más manejable con herramientas informáticas estándar. Esto se divide en diversas etapas, primero la localización de la información en el documento, luego la desambiguación semántica y finalmente la extracción pertinente ateniéndose al contexto local.

- Transformación de los documentos a un formato de fácil acceso.
- Creación de los módulos de procesamiento inicial
- Creación de los módulos de enriquecimiento de información.
- Creación de los módulos de etiquetado y estandarizado
- Creación de un prototipo de pipeline PLN.
- Extracción práctica de un conjunto de documentos enriquecidos por PLN.
- Análisis de resultados y resumen de problemas y soluciones

Las limitaciones, por diseño, del proyecto son: la falta de una garantía de fiabilidad en los datos obtenidos, se buscara la revisión y corrección de los resultados por un experto en el ámbito médico; la restricción de dominio, dado el gran número de textos posibles solo se trataran cierto subconjunto de las historias clínicas más usadas por los profesionales; y por último, por la naturaleza del proyecto, se ha dejado como trabajo futuro el mayor desarrollo de herramientas compatibles con los datos de salida intermedios y finales (como visualizaciones e interfaces gráficas).

1.5. Entorno de aplicación

Existen un número de aplicaciones, de reciente creación, dedicadas al procesado del lenguaje natural y específicamente desarrolladas para atacar los problemas de naturaleza única encontrados en el terreno de la salud. Herramientas como Metamap[14] se centran en la resolución de un problema específico, de manera pública o comercial; este caso en particular seria el reconocimiento de conceptos en texto libre partiendo de un diccionario de vocabulario médico.

Otras herramientas como Clamp [4] o Ctakes [7] se disponen a establecer un marco general de desarrollo de software de procesamiento de lenguaje natural, incluyendo un sin número de algoritmos compatibles y preparados para usuarios no técnicos.

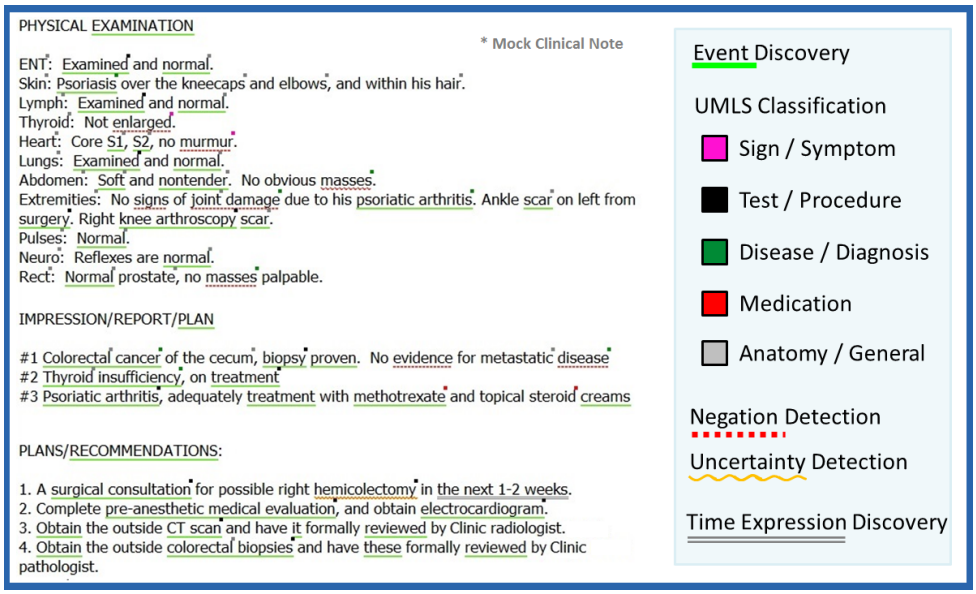

Figura 1.4: Ejemplo de la extracción realizada por
la herramienta Apache cTakes - Fuente ctakes.apache.org[7]

En resumen, y como comentamos previamente, existe un movimiento de recogida de conocimiento del ámbito clínico para el entrenamiento de algoritmos supervisados, además de una busqueda para facilitar la estandarización internacional y así posibilitar el traspaso de herramientas, con diferentes lenguajes y vocabularios. El mayor ejemplo de esto seria **UMLS**.

Más allá, a pesar de que en este momento existe un desarrollo generalizado, este se ve limitado en el terreno internacional; por suerte, recientemente se han creado iniciativas por expandir estas técnicas, vocabularios y estándares a nuestro idioma.

Un ejemplo de esto es la empresa **Savanamed**, creadora de la herramienta comercial Savana[23], la cual desea imponerse como centro del desarrollo PLN en España. Por desgracia esta es una propuesta cerrada y no existen, dentro del conocimiento del autor

de este proyecto, soluciones más abiertas y disponibles para el público en general. Dado el entorno cerrado de Savana, sin apenas estudios publicados o información sobre su funcionamiento y capacidades, es difícil saber hasta que punto están llevando a cabo sus ambiciones.

Finalmente creemos, que dadas estas carencias, actualmente es necesario desarrollar sistemas "Open source" que fortalezcan y faciliten el desarrollo del PLN en España.

Aquí es donde creemos que este tipo de proyectos como el nuestro encaja, pues se requieren para investigar las posibilidades actuales en nuestro idioma. Si bien esta herramienta no es –por el momento– de código libre, sí se desea liberarla de manera gratuita a través de la plataforma **Galena Databook**, idea ganadora del Premio CSdM Tech4futures (más sobre esto en la sección de conclusión).

Capítulo 2

Fundamento Teórico

A continuación describiremos los conceptos básicos detrás del desarrollo PLN, cuales son los módulos principales y cuales son las variantes más usadas.

2.1. Introducción

En el estudio del procesado del lenguaje natural existen múltiples problemas que, de manera independiente y sin afectar al resto, pueden ser resueltos; esta modularidad facilita el análisis y la aplicación práctica de las soluciones propuestas, además de orientar el desarrollo a una arquitectura de tipo pipeline o tubería.

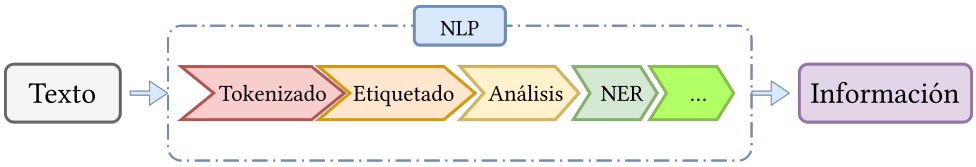

Figura 2.1: Proceso de tubería en PLN

De este modo, en un proceso PLN cualquiera se requieren de diferentes etapas dependiendo del objetivo, pero en general, se pueden organizar en una estructura común. Primero está el procesamiento a nivel de tokens[1], esta es la separación del texto en unidades más sencillas y manejables como sentencias, palabras y, finalmente, tokens. Es posible ir más allá, extrayendo las raíces de la palabra para así normalizarla (**Stemming** y **Lemmatization**) o, más comúnmente, asignar etiquetas que identifiquen dichos tokens (**POS tagging**[2]).

Tras la etapa de procesamiento a nivel de tokens se requiere extraer el significado, en su contexto, de cada uno de los elementos y su relación con el resto de ellos, para poder

[1]Termino de definición difusa, agrupa palabras, símbolos, números... son –generalmente– las unidades más básicas del PLN

[2]Etiquetado con las partes del lenguaje/habla, e.g Verbo, Determinante, etc

obtener una representación correcta de la sentencia; esta es la etapa de **Parsing** en la cual se extrae la estructura sintáctica de la frase. Aquí hay múltiples sub-etapas y niveles de profundidad, desde el más simple **Chunking** hasta el más complejo **Dependency Parsing**[3], etc.

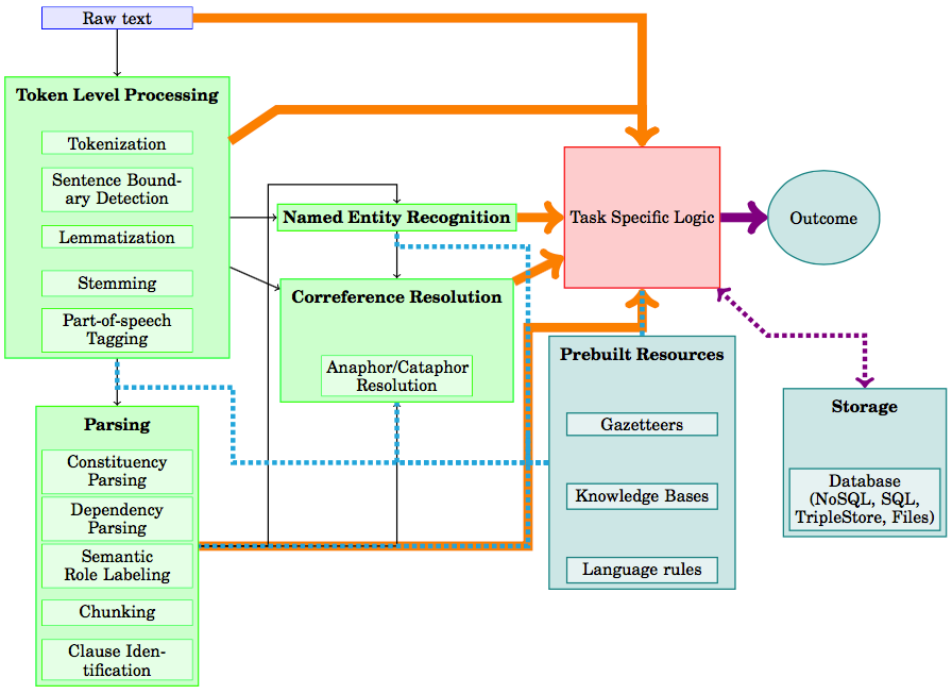

Figura 2.2: Ejemplo de los módulos más comunes en PLN

Por último, estaría la etapa más variada, con procesos como identificación de la semántica, e.g. **NER**[4], el mapeado de términos respecto a una base de datos, etc; cualquier actividad que dependa del contexto y del problema a resolver, para el cual se ha de extraer una u otra información deseada [26].

A continuación, detallaremos uno por uno los problemas más acuciantes de la literatura y las soluciones propuestas específicamente para textos en español. Además, señalaremos adecuadamente aquellas que serán aplicadas en el desarrollo del prototipo. No mostraremos todas las posibles algoritmos o etapas intermedias y nos centraremos en aquellos más útiles para el castellano y nuestro problema concreto.

2.2. Procesado a nivel token

Esta etapa consiste la separación del texto plano en unidades más simples con sentido, como frases y/o palabras (o más generalmente tokens) y categorizarlas para un posterior análisis.

[3]Análisis gramatical de la sentencia y de las dependencias entre los tokens
[4]Reconocimiento de entidades con nombre, e.g. Síntomas, Drogas, Paciente, etc

Para llevar a cabo la tarea de segmentación/tokenizado existen multitud de sistemas; dependiendo del lenguaje puede variar ampliamente de complejidad. Es mucho más sencillo, por ejemplo, para lenguajes con marcadores de limites explícitos entre las unidades básicas, como los espacios entre palabras y la puntuación entre sentencias en idiomas como el ingles o el español. En estos, los algoritmos se reducen a conjuntos de reglas o heurísticas (comúnmente **regex**[5]) que tengan en cuenta las idiosincrasias de la lengua; no obstante, esto tiene la suficiente complejidad como para existir un número de variantes.

Por otro lado, la clasificación de los tokens depende ampliamente de las bases de datos etiquetadas en el idioma correspondiente y, en particular, para el ámbito específico, ya que las técnicas y algoritmos están en general establecidas.

2.2.1. Segmentación

Este proceso consiste en la división de párrafos –o conjuntos más grandes de texto– en sentencias. Este proceso es sencillo en el Español debido a que este posee divisores explícitos de las frases, que son tres: el punto, la exclamación y la interrogación. La complejidad surge en el punto; este puede ser usado para definir acrónimos, como punto decimal, para definir fechas...; además de que es posible que dicho marcador no exista por olvido o simple fallo. Por consiguiente, para poder dividir el texto en sentencias es necesario principalmente distinguir entre aquellos puntos que significan un final de linea de los que pertenecen a acrónimos u otras ocurrencias del enunciado. También es necesario determinar finales de sentencia cuando aparecen el resto de símbolos –de interrogación y de exclamación–, pero como ya hemos comentado este proceso es inmediato.

Una aproximación razonable al problema resulta de tomar cualquier símbolo "." como punto y final. Es un sistema rápido y su implementación es trivial, pero comete errores en los casos comentados previamente, como J.R.R. Tokien, en el cual no hay sino un nombre acortado.

Es posible tener en cuenta dichas construcciones –palabras y acrónimos– usando el contexto y la estructura probabilística del texto. Un ejemplo de este tipo de algoritmos es **Punkt** [11], un sistema de detección de sentencias no supervisado, es decir, se basa en la búsqueda de acrónimos en un enunciado de entrenamiento distinguiendo aquellos tokens que poseen "." dentro del conjunto de caracteres, de aquellas que solo son continuadas por un punto de fin de sentencia (en este segundo caso habría a continuación un salto de linea). Con una estimación de máxima verosimilitud junto con varias heurísticas, extrae un conjunto de candidatos sobre los cuales se segmenta posteriormente el texto.

También es posible, si se poseen datos etiquetados, construir un clasificador (SVM, red neuronal...) que distinga automáticamente entre las distintas ocurrencias.

En nuestro caso, dado que poseemos una cantidad suficiente de documentos como para llevar a cabo el entrenamiento no supervisado, y dado que no poseemos datos etiquetados

[5]Expresiones regulares comúnmente usadas en el entorno informático

como para entrenar un clasificador, elegimos este segundo algoritmo, **Punkt**[11], por su mayor capacidad predictiva respecto al método más básico.

2.2.2. Tokenizado

Una vez se tienen separado el texto en sentencias, es necesario separar las palabras; además tenemos que tener en cuenta números, símbolos, etc entidades que aunque a priori sencillas pueden dar infinitos problemas al tener que considerar todos y cada uno de los casos límite. Esto conlleva a la definición de tokens, que como mencionamos previamente, son las unidades más sencillas del PLN.

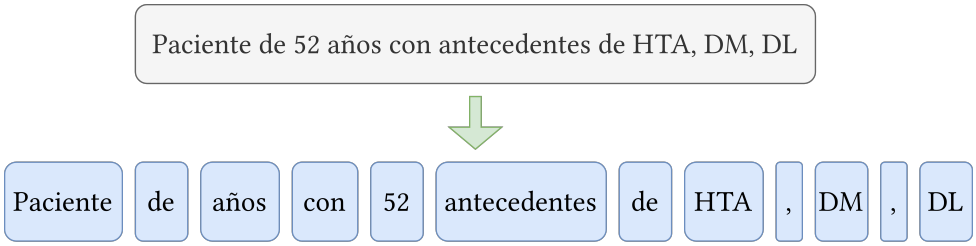

Figura 2.3: Ejemplo de tokenizado de una sentencia

Si bien hay múltiples métodos disponibles, aquellos basados en reglas, dada su relativa simplicidad y eficacia, actualmente dominan la competición. Estas reglas se definen a través de expresiones regulares, disponibles, ya optimizadas, en todos los lenguajes de programación. Para crear una nueva regla, se establece las condiciones bajo las cuales se considera que un conjunto de caracteres escritos es un nuevo token y se traspasan al lenguaje elegido. De esta manera, este método es tan bueno como lo sean las reglas construidas.

Actualmente no existen otras variantes (de uso generalizado) para llevar esta tarea; por lo tanto, y dada la simplicidad y la facilidad de implementación del método, elegimos implementarlo en nuestro proyecto.

2.2.3. Partes del habla

POS[6] es otra parte fundamental, junto con el tokenizado, de la pipeline de PLN. Es el primer paso[7] en el análisis del texto, dando a cada token su categoría gramatical, e.g. nombre, verbo, adjetivo... En casos más avanzados puede además obtener otras características de la palabra, como genero, tiempo, numero...

La utilidad de estas categorías radica, más que por sí mismas, en su capacidad para construir una base para métodos más complejos como resolución de coreferencias, extracción de entidades, etc.

[6]Parts Of Speech tagging o etiquetado de partes del habla

[7]Esto no es necesariamente verdad pero es el enfoque más usado ya que muchos algoritmos parten de esta información

Para su implementación, el grueso de los algoritmos POS parten de bases de datos con sentencias etiquetadas, palabra por palabra, con sus categorías. Por consiguiente, es posible crear a partir de estos, o de conocimientos previos, reglas para la identificación de categorías; sin embargo esto es complejo e ineficiente; como en el resto de algoritmo PLN, actualmente se usan en su métodos de Deep Learning o similares como clasificadores.

Dada la necesidad de tener en cuenta el contexto al asignar a un token su categoría (no siempre la misma palabra tiene el mismo uso), métodos basados en modelos ocultos de Markov, campos aleatorios condicionales, redes recurrentes o convolucionales... son comúnmente seleccionados. En este trabajo no entraremos en detalle sobre su diseño o implementación, pues está más allá del alcanza de este tema.

Teniendo en cuenta la capacidad y precisión del modelo, además de su disponibilidad en librerías, elegimos una versión simplificada de los modelos ocultos de Markov, conocida como **Ngram tagging** o, más específicamente, etiquetado por Bigramas; simplificando, este toma toda posible combinación de N tokens del texto (dos en el caso de bigramas) y les asigna la categoría correspondiente a la moda. Consideramos que este modelo es suficiente para nuestro proyecto.

2.3. Análisis sintáctico

El análisis sintáctico o "Parsing" consiste en examinar un texto a través de un conjunto de reglas o gramática. En el procesado de lenguaje natural esto se usa para descomponer frases en sus partes constituyentes, siendo el resultado final dependiente del tipo concreto de análisis.

Existen dos tipos principales de gramáticas/análisis[8] en PLN. Por un lado está el análisis de dependencias, que busca encontrar las relaciones entre palabras examinando la sentencia por medio de relaciones binarias entre tokens; por otro lado está el análisis de constituyentes, que con la misma sentencia genera un árbol sintáctico dando la jerarquía por la que se relacionarían los tokens; es de este segundo tipo de análisis de donde se deriva el **Shallow parsing** o análisis superficial, el cual se refiere a la práctica de tan solo extraer subconjuntos del árbol de constituyentes.

Para nuestro proyecto elegimos un tipo muy usado de **Shallow parsing** conocido como **Chunking**, el cual tan solo extrae constituyentes de manera plana, sin jerarquía alguna. E.g. Sintagmas nominales, sintagmas verbales, etc; para nuestro caso este método es suficiente y dado su popularidad existen numerosas implementaciones de código libre.

[8] Análisis sintáctico se refiere a la creación del árbol de sintaxis a partir de la sentencia. La gramática son las reglas mediante las cuales se genera dicha sentencia.

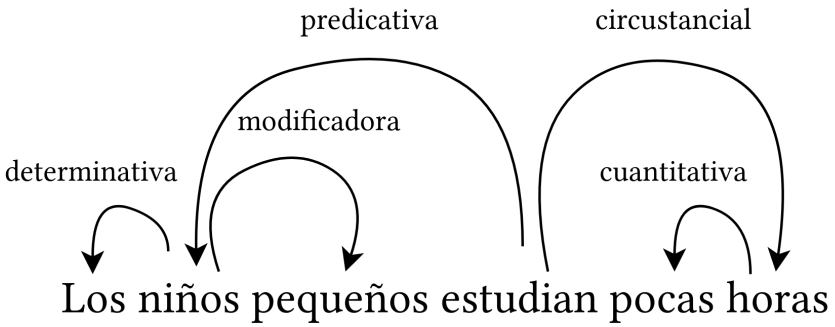

Figura 2.4: Ejemplo del análisis con una gramática de dependencias

2.4. Semántica

Tras la creación del árbol sintáctico sigue la extracción de entidades abstractas y el mapeado a terminología ya establecida en bases de datos, diferenciándose ambas por la extensión de las categorías y el método usado. La extracción de entidades usa reglas, o más comúnmente, modelos probabilísticos para "encontrar" tokens que encajen con un grupo restringido de categorías; Por otro lado, el mapeado a terminología busca coincidencias con bases de datos de manera determinista; el beneficio de esto es que el número de términos o categorías reconocibles no tiene prácticamente límite.

La extracción de entidades con nombre o **NER** consiste por lo tanto en la asignación de tokens a categorías, en nuestros caso del tipo de síntoma, droga, enfermedad, etc. Actualmente los mejores modelos parten de datos etiquetados por expertos y usan algoritmos de machine learning de tipo de **CRF**[9].

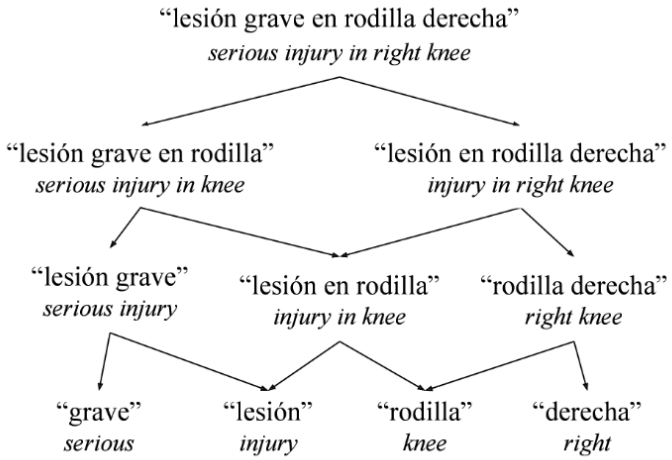

Figura 2.5: Normalización jerárquica - Fuente estudio de normalización UMLS[16]

[9]Conditional Random Fields o Campos Aleatorios Condicionales

Por otro lado, el mapeado de tokens busca encontrar tokens (simples o múltiples) relevantes de una base de datos para asignarles un código identificativo, e.g. Dislipemia ->C0123948. Si bien es simple comprobar si un termino pertenece a una lista, comprobar el conjunto de términos más cercanos a una lista de tokens requiere trabajar con distancias e incluso métodos jerárquicos. Es además importante, el hecho de que a múltiples tokens, con diferentes acentuaciones, mayúsculas o incluso diferentes palabras; se les puede asignar un mismo código identificativo.

Debida a la complejidad de este problema, decidimos trabajar siguiendo un estudio europeo [16] sobre sistemas de normalización de terminología multinivel, de manera que se parte del máximo conjunto posible de tokens, refinando hasta encontrar el mejor encuentro entre tokens y terminología de la base de datos. Esto nos permite establecer, dada una secuencia de tokens, múltiples términos relacionados en vez de solo uno.

Capítulo 3

Metodología

A continuación presentamos el método seleccionado para el desarrollo del TFG, las herramientas usadas a lo largo del proceso, por qué han sido elegidas y la arquitectura de la aplicación software. Por último, mostraremos las diferentes consideraciones temporales y materiales.

3.1. Proceso de desarrollo

Se ha seleccionado un modelo típico iterativo de desarrollo, desde la etapa inicial se usara este modelo para implementar el TFG. Por lo tanto, durante el ciclo de vida del software se llevarán a cabo múltiples etapas en las que se desarrollara un prototipo cada vez más completo y que cada vez cumpla con un mayor conjunto de especificaciones. En nuestro caso, las especificaciones principales sobre las que iterará el proyecto son, en un primer paso, los módulos independientes de la tubería PLN y, en segunda instancia, la recogida de un cada vez mayor conjunto de datos.

El razonamiento detrás de esta elección es la falta de experiencia en el tema a desarrollar, que previene una organización más clásica con un modelo en cascada, ya que no se conocen lo suficiente las tecnologías como para hacer un diseño a priori. Dado que este trabajo es de aprendizaje e investigación, el modelo de prototipo rápido se ajusta a la perfección, pues nos permite rediseñar continuamente según se vaya consiguiendo un mayor conocimiento de las tecnologías necesarias y según se vaya viendo necesario mejorar las capacidades de uno o varios de los módulos.

En cualquier caso, el software sera desarrollado a través de un paradigma orientado a objetos en el lenguaje **Python**.

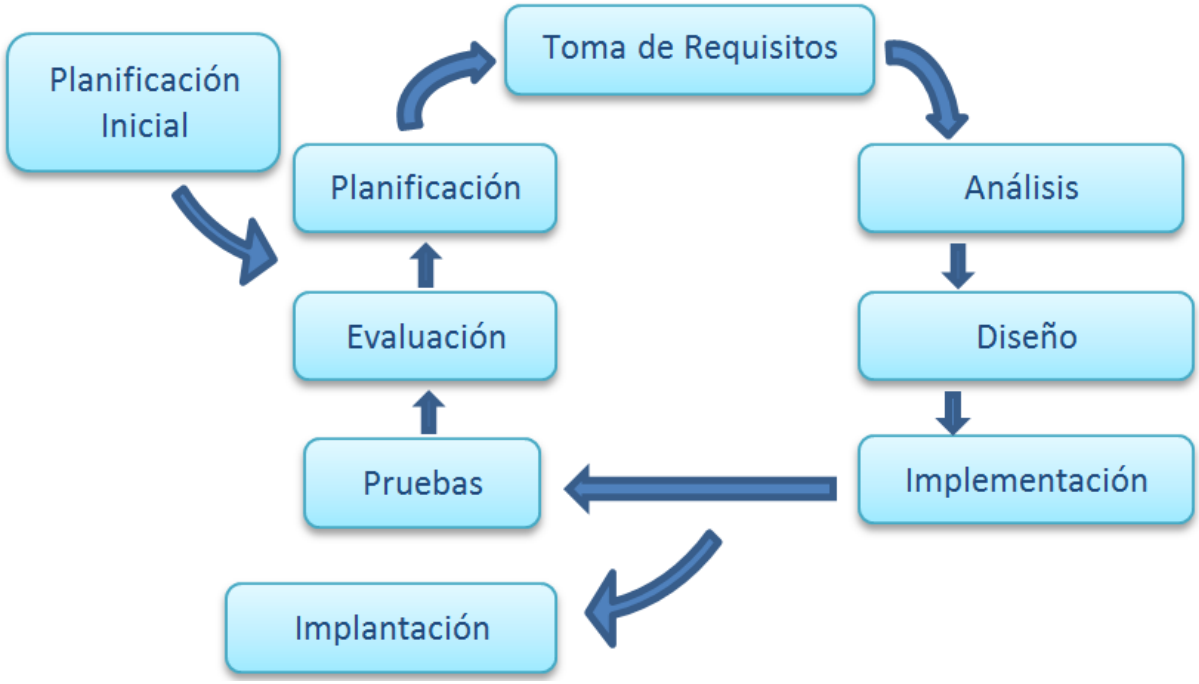

Figura 3.1: Modelo de desarrollo iterativo

3.2. Herramientas utilizadas

Para la realización del proyecto has sido necesarias varias librerías y utilidades de terceros, a continuación van a ser detalladas y comentaremos el porqué de cada una de ellas.

3.2.1. PdfToHtml

Esta es una herramienta estándar de preprocesado de pdf que facilita el traspaso de estos archivos a un formato más tratable con técnicas convencionales de procesamiento de texto. Su uso es a través de linea de comandos y se genera, a partir de un archivo pdf, un html estándar manteniendo gráficos e indentación.

Una limitación limitación de la herramienta es que la transformación de gráficos a texto no es del todo perfecta y cierta ordenación del texto y alineación de gráficos se pierde. Además algunos atributos del texto no son transformados a categorías html y son tratados como gráficos (el subrayado), lo cual hace necesario un posterior tratamiento para recuperarlos.

3.2.2. BeautifulSoup

Librería de Python que facilita el procesado y análisis de ficheros html[19]. Esta es necesaria ya que por defecto Python no posee de manera nativa herramientas para filtrar

dichos ficheros.

Su característica principal es que permite trabajar directamente sobre las etiquetas de manera jerárquica, facilitando el acceso a otras etiquetas hijas, padres y hermanas de la actual.

Se usa en el paso de los ficheros html al formato usado en el resto del procesado (xml).

3.2.3. LXML

Librería de terceros que extiende las capacidades estándar de Python en el procesado de ficheros XML[29]. Python posee por defecto un conjunto de procedimientos pensados para este tipo de parsing –ElementTree– y esta librería extiende y mejora dichas capacidades.

No solo acelera las capacidades básicas de la biblioteca sino que otorga ciertas facilidades para la programación, como la posibilidad de usar Xpath, un lenguaje de búsqueda rápida de etiquetas entre otras cosas. Su uso no es estrictamente necesario pero acelera el desarrollo del proyecto.

El uso de esta herramienta es debido a que durante todo el procesamiento se trabajara sobre un archivo XML el cual se enriquecerá a cada paso de los módulos con la información obtenida del procesado de lenguaje natural.

3.2.4. NLTK

Esta es una librería de Python centrada en facilitar el desarrollo de aplicaciones de procesamiento de lenguaje natural de una manera rápida y extensible[15]. Contiene, además de facilidades para la programación, implementaciones de algoritmos estándar de uso muy extendido en proyectos de esta índole.

Como limitación de la librería, al ser un paquete para desarrollo general contiene modelos y procedimientos optimizados para otro tipo de trabajos, sobre todo en Ingles, y para texto plano con un vocabulario de uso mas común en la población general.

El uso principal dado a esta herramienta es para el etiquetado **POS** y la segmentación de frases.

3.2.5. Conllu

Librería de terceros de Python que automatiza el procesado de ficheros con formato Conllu[6], típico en el procesado del lenguaje y usado en la base de datos de Universal Dependencies.

Capítulo 4

Planificación

Aquí desarrollamos nuestras previsiones de coste temporal, físico, económico, etc., para el proyecto completo.

4.1. Estimación del esfuerzo

Ya hemos comentado previamente que nuestra metodología de trabajo sera una iterativa, por ende todas las etapas descritas son aproximaciones temporales de las iteraciones y de las características que se quieren añadir al prototipo en cada una de ellas; si bien en todas nuestras previsiones no se puede tener en cuenta a priori todo el tiempo añadido, que podría ser necesario para mejorar y rediseñar el trabajo ya creado.

Para facilitar el análisis se separara el trabajo realizado en términos software del de recogida de datos llevada a cabo por los médicos.

Recogida de datos

Primero la recogida de datos es llevada a cabo por dos estudiantes de medicina, esta requiere de la extracción y anonimizado de las historias clínicas en forma de pdf. Dado que el anonimizado se lleva a cabo a través de herramientas automáticas solo tenemos en cuenta el trabajo de recogida de los archivos.

Para llevar a cabo esta aproximación nos basamos en el juicio experto y en una estimación por analogía de otras recogidas de datos. Concluimos que se requiere de una hora para recoger unas 30-40 historias clínicas. Por lo tanto aproximamos que se requerirá de entre 13 y 17 horas en total, repartidas entre 3 personas, para obtener los 500 pdf que conformaran la base de datos.

ID: 1 Recogida de historias clínicas
Predecesoras: Ninguna
Duración: 2 días
Recopilación del conjunto de historias clínicas

Además tenemos que tener en cuenta el tiempo requerido para desarrollar la herramienta automática de anonimizado que sera usada por los estudiantes, la cual estimamos que llevara alrededor de un día para implementar.

ID: 2 Herramienta anonimizado
Predecesoras: Ninguna
Duración: 1 día
Creación de una herramienta automática para el anonimizado de las historias

Análisis

Segundo, para la estimación de los tiempos del desarrollo del algoritmo y de la pipeline usamos un híbrido de método top-down y de analogía pues, dado que este proyecto tiene una alta componente de investigación, es prácticamente imposible establecer en detalle el software a implementar y por lo tanto el buen uso de un método de tipo paramétrico o de tipo bottom-up; de esta manera se prevé que el trabajo tendrá una duración mínima de 4 meses a tiempo completo, divididos en investigación y en el desarrollo de cada módulo.

El primer mes de trabajo se establece como introducción a la temática e investigación de las tecnologías requeridas para el suceso del proyecto, así como la recolección de estudios, herramientas similares, etc.

ID: 3 Investigación de PLN
Predecesoras: Ninguna
Duración: 20 días
Recopilación de información sobre herramientas, estudios, algoritmos... de PLN

ID: 4 Análisis de requisitos
Predecesoras: 3
Duración: 2 días
Elaboración de la lista de requisitos que deberá cumplir nuestra herramienta

ID: 5 Análisis de riesgos
Predecesoras: 4
Duración: 2 días
Elaboración de la lista de riesgos que pueden encontrarse durante el desarrollo de la herramienta

ID: 6 Diagrama de casos de uso
Predecesoras: 5
Duración: 4 días
Elaboración de la lista de actores y de los casos de uso que se daran en el sistema.

ID: 7 Planning temporal
Predecesoras: 6
Duración: 1 día
Elaboración del calendario de actividades requeridas para completar el proyecto

Diseño

Los 3 meses posteriores se han de dividir en el diseño inicial del sistema y en el desarrollo iterativo de los módulos de procesado. Primero se llevara a cabo el diseño inicial que requerirá de alrededor de una semana de trabajo.

ID: 8 Modelo de dominio
Predecesoras: 7
Duración: 5 días
Elaboración del modelo de dominio partiendo del análisis inicial de los requerimientos del sistema

ID: 8 Sistema de control de versiones
Predecesoras: 7
Duración: 1 día
Preparar el sistema de versiones para controlar el código de la herramienta

Desarrollo

A partir de aquí comenzamos la implementación en sí del software.

ID: 9 Implementación estructura inicial del diseño
Predecesoras: 8
Duración: 5 días
Desarrollar el sistema inicial de deberá permitir el subsiguiente desarrollo de los módulos PLN

Después la extracción de la estructura de las historias, que se le da 1 semana; el comienzo de la división del texto en unidades más pequeñas, segmentación y tokenización, 2 semanas; el etiquetado POS, 1 semana; el análisis/parsing, 3 semanas y la identificación con UMLS, 3 semanas.

ID: 7 Planning temporal
Predecesoras: 6
Duración: 1 día
Elaboración del calendario de actividades requeridas para completar el proyecto

ID: 8 Extracción estructura
Predecesoras: 7
Duración: 6 días
Extracción de las secciones/estructuras comunes a todas las historias clínicas

ID: 9 Segmentación
Predecesoras: 8
Duración: 4 días
Elaboración del módulo de separación de frases individuales

ID: 10 Tokenización
Predecesoras: 9
Duración: 10 días
Elaboración del módulo de tokenización

ID: 11 Etiquetado POS
Predecesoras: 10
Duración: 7 días
Elaboración del módulo de etiquetado de partes del habla

ID: 12 Parsing
Predecesoras: 11
Duración: 20 días
Elaboración del módulo de parsing

ID: 13 Base de datos UMLS
Predecesoras: 12
Duración: 10 días
Análisis, creación y despliegue de la base de datos MySQL con los datos de UMLS

ID: 14 Identificación de terminología
Predecesoras: 13
Duración: 15 días
Elaboración del módulo de identificación de términos

Damos una cantidad mucho mayor a las últimas etapas/módulos puesto que son los que poseen un mayor número de incógnitas que necesitan ser investigadas hasta poder

pasar a la implementación.

Damos además tiempo extra (4 meses es tan solo el mínimo) para las iteraciones y diversas mejoras de los módulos.

Pruebas y extracción

Por último tenemos la fase final de extracción de datos y de pruebas finales del sistema para analizar su calidad (en general también se harán pruebas a lo largo de todo el proyecto y en cada iteración).

ID: 15 Extracción de información
Predecesoras: 14
Duración: 4 días
Elaboración del módulo de extracción de información

ID: 16 Pruebas finales
Predecesoras: 15
Duración: 3 días
Pruebas finales de calidad del sistema

4.2. Planificación temporal

Como hemos comentado previamente, el desarrollo se llevará a cabo de manera iterativa. Si bien las etapas no pueden ser comenzadas fuera de orden (los módulos previos deben ser comenzados antes de poder ser desarrollados los siguientes) toda etapa puede ser reconsiderada, llevada en paralelo con otras, etc.

Primero se ha de comenzar la recogida de datos; esta se llevará a cabo de manera progresiva a lo largo de un año, incrementando así cada pocos meses el corpus disponible para el testing del algoritmo. Para facilitar la construcción de prototipo inicial, la recogida comenzara antes del inicio del desarrollo del software para así partir de un número suficiente de historias clínicas, aunque a partir de ahí se llevara en paralelo con el resto de tareas.

En todo caso, se comenzara por la investigación y el diseño inicial del sistema, para luego continuar con el desarrollo de los módulos. Si se ve necesario, se modificara el planning y/o las tareas para ajustarse al nuevo conocimiento adquirido, repitiéndose esto a lo largo de todo el proyecto.

En la figura 4.1 mostramos un ejemplo de la planificación inicial; no podemos establecer nada más concreto debido a que desde el principio está sujeto a cambios. Así pues las semanas de duración estimadas serán luego extendidas en tanto sea necesario iterar o investigar para mejorar la implementación inicial.

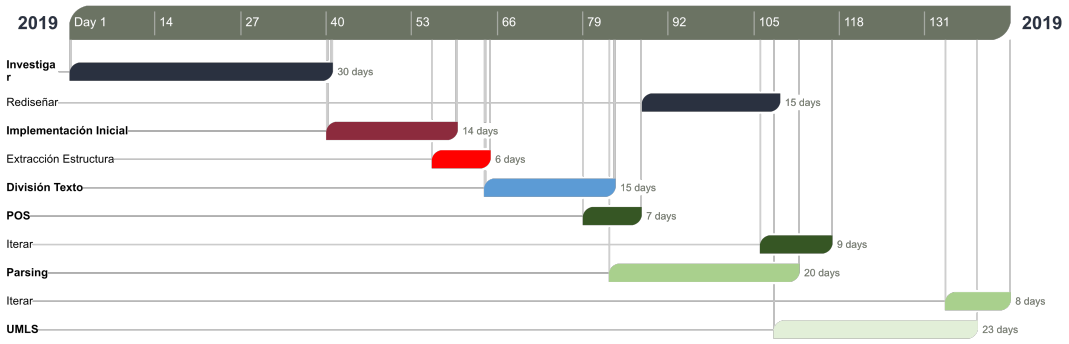

Figura 4.1: Planificación temporal del desarrollo, mostrando posibles iteracciones y re-diseños

En realidad, el proyecto comenzará durante el segundo cuatrimestre del curso 2018-2019, se mantendrá y se desarrollará durante el verano de 2019 para, a ser posible, ser terminado durante el segundo cuatrimestre del curso 2019-2020, especialmente durante los meses de mayo a julio. Este calendario discontinuo ha sido creado teniendo en cuenta el curso y otras actividades extracurriculares.

4.3. Presupuesto económico

Brevemente se comenta las necesidades, que son pocas, mínimas necesarias para la consecución del TFG.

4.3.1. Hardware y software

En nuestro caso, dada la naturaleza puramente de desarrollo software, no existen requerimientos físicos en término de herramientas o de hardware especifico. Aunque sí son usadas librerías de terceros, son todas de libre pago y, la mayoría, de código abierto, aunque esto ya se ha especificado caso por caso en la sección de herramientas.

4.3.2. Recursos humanos

Mas allá de la persona desarrollando el TFG, debido a las restricciones de las bases de datos públicas son necesarios un equipo de estudiantes médicos dedicados a la investigación para la extracción del corpus de historias clínicas.

En particular Roberto Martin Asenjo realiza la tarea de supervisión y los estudiantes Juan Pedro Garcia Navas, Roberto Comenero Rollinson y José Ángel Álvarez Vazques la de extracción de la información.

Por último está Pablo Vaquero Martinez como especialista del ámbito médico y de la documentación clínica, necesario para facilitar el procesado, el trato de las variables, etc.

4.3.3. Presupuesto total

Asumiendo los 4 meses de desarrollo descritos en la sección de planificación y contando que el sueldo medio de un informático es de 24000 euros, estimamos que el coste del sistema es de 8000 euros.

Además, sumando las 17 horas de extracción de datos llevadas a cabo por profesionales médicos y teniendo en cuenta que su salario medio es de 50000 euros, tenemos un total de coste del proyecto de **8442 euros**.

Parte II

Conjuntos de datos

Capítulo 5

Historias Clínicas

En este capitulo se describe el origen de los datos, su contenido y el trabajo llevado a cabo para adaptar las historias clínicas a las necesidades del proyecto.

5.1. Resumen

Para llevar a cabo todo el desarrollo, son necesarios un conjunto de historias clínicas facilitadas por algún centro de salud. Estos son archivos complejos, con datos semi-estructurados pero cuyo contenido principal es común a todos y cada uno de ellos. Concretamente en España se ha creado un plan por el Sistema Nacional de salud que busca establecer un marco estándar para todo el territorio (Plan de Impulso de las Tecnologías del Lenguaje[18]). Por último, dado que estos documentos son fuentes de información sensible relativa a pacientes reales, es necesario que se haya hecho un tratamiento previo para anonimizar y eliminar cualquier relación con el paciente.

5.2. Descripción genérica

El registro de la historia clínica es un documento que contiene la información pertinente al estado del paciente y la atención médica dada a este. Además de la situación presente del paciente, se incorpora toda información relativa a los antecedentes personales y familiares, así como toda su evolución durante el tratamiento y recuperación.

La estructura de esta información varia dependiendo del modelo elegido, la especialidad o el hospital entre otros factores pero, en general, siempre contienen ciertos componentes principales indispensables. Estos son los datos proporcionados por el paciente, subjetivos, sobre la causa de su llegada al centro, sobre episodios de dolor, síntomas, etc; los datos obtenidos por la exploración física y otras diversas exploraciones, cualquier prueba complementaria, resultados de laboratorio y cualquiera información extra procurada por el facultativo; el diagnóstico o juicio clínico sobre la causa o enfermedad que sufre el pacien-

te; el tratamiento, conjunto de medios a través de los cuales los médicos tratan de curar o paliar la enfermedad; y por último, la prognosis o pronostico que engloba la previsión del médico sobre la futura evolución del paciente y, si existe revisión, sobre la trayectoria a lo largo del tiempo del mismo.

Estos no son sino ejemplos de toda la información que este tipo de datos suele contener pero en ningún caso se limitan a estos, siendo las posibilidades tan variadas como lo es la medicina.

Además, no solo varia el contenido sino también el formato, estos archivos están disponibles en desde el más básico archivo word o pdf, hasta las historias clínicas integradas con el sistema electrónico de salud.

5.3. Situación en España

Los hospitales españoles no son diferentes al resto del mundo y usan estos mismos documentes, de maneras similares, para guardar la información de los pacientes. Actualmente, el formato más común en los hospitales es, sorprendentemente, el papel físico, no siendo hasta recientemente que se ha comenzado el proceso de transformarlos a formato digital, en particular en archivos pdf. Si bien dada la disparidad de las provincias españolas esto es diferente en cada zona y en cada hospital, es un hecho que para trabajar en España es necesario manipular dichos formatos, a veces incluso necesitando herramientas OCR.

Respecto al contenido, durante los últimos años se ha realizado un proceso de estandarización de la estructura y del contenido mínimo de estos documentos clínicos. Hecho surgido por la necesidad de desplazamiento de los usuarios del sistema de salud a lo largo y ancho del territorio español, ha provocado la implicación del SNS[1] para, mediante el consenso de varios grupos de trabajo, elaborar una propuesta del conjunto mínimo necesario para todos y cada uno de los informes usados por los facultativos; a saber, informes de urgencias, de alta, de especialidades, etc [21].

Esta propuesta se establece dentro del marco del proyecto español HCDSNS[2] con el mismo objetivo de facilitar la interoperabilidad de todo el sistema electrónico español de salud. Usándola, se nos facilita la comprensión automática de las historias clínicas.

5.4. Corpus

Los datos que van a ser usados provienen de un hospital madrileño, el Hospital Doce de Octubre. En este se ha llevado a cabo una asociación con varios facultativos –principalmente el jefe de cardiología Roberto Martin Asenjo– dedicados a la investigación, que son los que facilitan el acceso a los datos.

[1]Sistema Nacional de Salud

[2]Historia Clínica Digital en el Sistema Nacional de Salud

En primer lugar, para obtener las historias se ha construido un protocolo de estudio y, dentro del marco de un trabajo mayor, se ha analizado el objetivo de la investigación y la ética del mismo para su posterior análisis y validación por un cuerpo de facultativos. Además, para todas y cada una de las historias clínicas se ha a obtenido permiso por escrito para la recogida y uso de dichos datos.

Los datos consisten en documentos clínicos de urgencias del área de cardiología, concretamente informes clínicos de alta en los cuales se recogen características personales del paciente: edad, sexo,...; el motivo de ingreso, antecedentes, resultados de pruebas realizadas, etc.

Todos estos datos se encuentran separados en apartados definidos y estandarizados por el Sistema Nacional de Salud[21], dentro de los cuales se organizan las distintas informaciones referentes al paciente. Todo esto se encuentra en un formato de texto formal semi-estructurado con variaciones de forma y vocabulario.

Como se comentaba previamente, en este caso los datos son todos tratados por un profesional de manera automática y con revisión manual, siendo anonimizados antes de pasar a las posteriores etapas del análisis. Esto consiste en la eliminación del número de historia clínica, nombre del paciente, domicilio, ... Cualquier dato personal del paciente además de cualquier mención al facultativo que en su momento trato ha dicho paciente.

5.5. Pre-procesamiento

Previo uso del corpus, se ha realizado un paso de pre-procesamiento para adecuar el formato de los documentos a uno más manipulable. Para ello se transforma cada pdf a un resumen textual en xml; primero se traduce de pdf a html de manera fidedigna, es decir, manteniendo toda la información del texto en cuestión además de imágenes, cuadriculas y otros accidentes del documento. Una vez obtenido este paso intermedio, se procesa el html seleccionando tan solo la información deseada para el posterior análisis y se organiza todo ello en un xml, el cual mantiene el texto y las características más importantes de este, como la subordinación de sentencias o el subrayado de frases concretas.

Además, se mantienen índices en el archivo final que mantienen referencias al documento de origen para mejorar la trazabilidad a los documentos originales.

Capítulo 6

Bases POS

A continuación comentamos las características principales de la base de datos **SPACCC** y la base de **Universal Dependencies**. Finalmente resumimos el porqué del rechazo para este proyecto de la fuente de información **Universal Dependencies**.

6.1. Resumen

Para la asignación de etiquetas POS al texto es necesario (para usar algoritmos supervisados) bases de datos con frases con cada token etiquetado de manera fidedigna. Para ello se ha recurrido a trabajos de terceros los cuales han realizado previamente un proceso de recogida y clasificación de textos previo. Nos centramos en las bases de texto español médico, ya que cercanía con el lenguaje objetivo (las historias clínicas) mejora la precisión de los algoritmos. Esta, y los problemas de compatibilidad, es la razón de seleccionar solo el corpus **SPACCC** para la tarea de etiquetado POS (el corpus **Universal Dependencies** era con el que se iba a trabajar originalmente).

6.2. SPACCC

Este es un corpus de historias clínicas en español [12]; contiene 65000 frases anotadas de manera mixta, una parte manual y otra automática. Los datos han sido extraídos de una biblioteca virtual Scielo [24] y seleccionados de manera manual.

El formato de etiquetado es uno propio desarrollado por los investigadores, el cual une en una sola cadena todas las características de interés del token. Estas son un acrónimo descriptivo de la etiqueta POS (como AQ para adjetivo o DD para determinante) y una parte morfológica etiquetada según EAGLES - PAROLE [9], la cual contiene genero, numero, tipo, etc.

Las sentencias etiquetadas han sido guardadas en dos formatos diferentes, primero el formato estándar bratt [3] y segundo un formato propio basado en el de la base **CoNLL**

[5]; para este proyecto usaremos tan solo este segundo formato.

6.3. Universal Dependencies

Proyecto de desarrollo multilingüístico de un conjunto de corpus y anotaciones compatibles, facilitando la construcción de nuevos sistemas de aprendizaje del lenguaje [28]. Para llevar esto a cabo se han desarrollado un conjunto de etiquetas estándar a partir de las de dependencias de Stanford [13], de las etiquetas universales de Google [17] y del conjunto de etiquetas morfológicas de Interset.

Históricamente este proyecto es un intento de unir el muy usado parser de Stanford para análisis de dependencias en Ingles, con el estándar común multilenguaje de Google nacido a partir de una tarea común de análisis de McDonald y Nivre. No es sino posteriormente que se aúna este esfuerzo con otro de tipo morfológico creando el sistema actual. Es importante también la revisión del formato de entrega de datos de CoNLL-X, siendo refinado hasta el actual formato CoNLL-U.

El proyecto posee un conjunto de principios y guías de anotación para la correcta creación de nuevas bases de datos, pero una buena parte del sistema actual se compone de corpus de previa creación adaptados al nuevo sistema. De esta manera, en el momento en el que esto se escribe, el corpus posee múltiples bases de datos de un conjunto de mas de 80 lenguajes de todo el mundo.

6.3.1. Rechazo

Esta última base de datos, a pesar de su calidad –razón por la que nos intereso en un principio–, no ha sido utilizada en el sistema final. Esto es por dos razones principales: uno, la base de datos no es del ámbito médico, lo que reduce su utilidad y la limita a cuanta transferencia de conocimientos entre campos seamos capaces de realizar; esto por supuesto complica el desarrollo. Segundo, las etiquetas no son compatibles con la base de datos **SPACCC**, para utilizarlas seria necesario comparar la definición de estas y realizar un mappeado entre ambas, de nuevo complicando el desarrollo.

Por todo esto, aunque su uso seria deseable en mejores condiciones, decidimos manejar tan solo la base de datos **SPACCC**.

Capítulo 7

UMLS

Esta base de datos multipropósito es la base del sistema de extracción de información y reconocimiento de términos estandarizados. Por consiguiente, a continuación desarrollaremos paso por paso su origen, su constitución modular y su sistema de manejo de datos.

7.1. Descripción

Creada por la Librería Nacional de Medicina estadounidense (U.S. National Library of Medicine) para integrar y aunar toda información útil del terreno médico, constituye un «Sistema de distribución e integración de terminología clave, estándares de clasificación y formato, además de recursos asociados para promover la creación de servicios y sistemas de información biomédica más interoperables y efectivos, incluyendo historias clínicas electrónicas» [2].

A pesar de su inicio como iniciativa estadounidense –y por tanto de lengua inglesa– el sistema se ha visto expandido con no solo un mayor conjunto de información, sino con traducciones directas de las bases de datos a través de asociaciones con otros países. Esto permite hacer uso de todas aquellas fuentes creadas por cualquier país hispanohablante asociado, que aunque no alcanzan la amplitud original de la base de datos, si constituyen una fuente indispensable de información biomédica electrónica.

7.2. Estructura

Son tres los componentes que proporciona el acceso a este sistema: el Metathesaurus, que contiene el vocabulario, términos, códigos además de definiciones y jerarquías entre otros; la red semántica que contiene categorías y relaciones entre ellas, solo disponible en ingles y por último el léxico especialista, también solo para ingles con una colección de

léxico sintáctico biomédico además de herramientas para procesar, normalizar y generar variantes de cadenas de caracteres.

El conjunto que más nos interesa es el Metathesaurus que integra decenas de bases de datos publicas y privadas en un mismo formato para facilitar su uso. En particular para español existen un total de ocho bases de datos traducidas, con datos desde la terminología usada para los diferentes procedimientos realizados por los facultativos hasta un corpus de reacciones adversas a las drogas.

7.2.1. Semantic Network

Conjunto de categorías genéricas que clasifican de manera consistente todos los términos de Metathesaurus; además estas categorías o tipos semánticos están relacionadas entre sí en una red semántica interconectada.

Así se genera en todos los conceptos, no solo una simple jerarquía, sino toda una red que enlaza todo y cada uno de los términos facilitando varias tareas de PLN.

Dado que está en ingles no nos centraremos en el aunque existen proyectos que superar este problema a través de la relación de términos español/ingles en bases de datos traducidas [1].

7.2.2. Specialist

Conjunto de datos léxicos y otras herramientas varias para facilitar el trabajo de varias tareas de PLN. Ejemplos remarcables son herramientas de generación de variantes léxicas, de normalización de términos, de corrección de texto, etc.

Como la red semántica tan solo está disponible para ingles.

7.3. Metathesaurus

Este corpus es un conjunto de vocabulario multipropósito y multileguaje que contiene información sobre conceptos relacionados con el ámbito de la salud, sus diferentes nombres y las relaciones entre ellos. Como todo UMLS [2], esta está conformada por la integración de una multitud de bases de datos de muy diferentes orígenes cuyos términos están enlazadas, entre sí y con los otros dos módulos que componen el sistema.

Parte de la doctrina del sistema es el mantenimiento de todo dato proporcionado por los orígenes, es decir, se preserva todo significado, concepto y relación de las fuentes por lo que se tiene que tener en cuenta, como es expresado en su pagina web, «...algunos conceptos puedes ser contradictorios y ambas vistas serán presentadas en el metathesaurus».

La base del corpus está en los identificadores, existen varios y cada uno se corresponde

con una entidad diferente. Primero los identificadores de concepto o CUI[1], la principal utilidad del corpus- estos son significados, hechos, enfermedades, procedimientos, etc. Todo concepto tiene un identificador diferente y este puede tener diferentes nombres por los cuales se le reconoce, es decir, sinónimos. Estos nombres tienen un identificador único o SUI[2] el cual es único para cada cadena de caracteres del corpus. Por último para cada nombre se mantiene un identificador de variantes léxicas, LUI[3], que agrupa nombres o términos (cadenas de caracteres) con pequeñas diferencias entre si.

7.3.1. Formato y pre-procesado

Toda la base se ofrece en dos formatos diferentes, RRF[4] y ORF[5], siendo este segundo el originalmente creado para el sistema el cual fue posteriormente mejorado en el RRF, por todo esto nos centraremos en esta versión mas moderna.

En toda distribución de UMLS primero a de ser creado el subconjunto de trabajo con todos los vocabularios útiles para el desarrollo especifico de la aplicación pertinente, en nuestro caso el conjunto de conceptos en Español, el cual incluye un total de ocho orígenes de datos diferentes, en general traducciones de un original ingles.

Una vez creado el conjunto deseado con las herramientas proporcionadas, se crean un conjunto de ficheros contenedores de toda la información desde metadatos sobre el subconjunto especifico seleccionado y el conjunto mismo de ficheros, a el total de los conceptos y relaciones entre estos. A partir de aquí es posible procesar directamente los archivos pero esto se facilita mediante scripts generados automáticamente para su volcado a una base de datos relacional, que es lo que llevamos a cabo como preparación para a su uso en la aplicación.

El sistema de gestión de bases de datos elegido es MySQL debido a diferentes factores: primero, sus altas especificaciones y popularidad la convierten en no solo una herramienta rápida y eficiente sino además posee una gran compatibilidad con un sinnúmero de librerías de terceros, lo cual facilita el desarrollo. Segundo, conocimiento previo de la herramienta, al fin y al cabo es mas sencillo trabajar con un sistema sobre el que se posee experiencia previa. Por último pero no menos importante, es una de las bases de datos soportada por los scripts automáticos simplificando así el trabajo requerido para la preparación de los datos.

[1]Concept Unique Identifier o Identificador único de concepto
[2]String Unique Identifier o Identificador único de cadena
[3]Lexical Unique Identifier o Identificador único de léxico
[4]Rich Release Format o Formato rico de lanzamiento
[5]Original Release Format o Formato original de lanzamiento

Capítulo 8

Otras fuentes

A continuación describimos el resto de fuentes usadas para el proyecto, menores en tamaño pero útiles para alguna función del sistema.

8.1. Abreviaturas Médicas

En cualquier ámbito técnico es común el uso de contracciones, abreviaturas y otras partículas propias del campo en el que se trabaja. El terreno de la salud no es distinto, al contrario, posee un conjunto innumerable de términos únicos incompresibles para nadie excepto los facultativos. Esto es en parte provocado por el amplio terreno que cubre la medicina y por la exigencia de eficiencia en los centros clínicos.

Para suplir este conocimiento experto recurrimos a varios sistemas de obtención de datos, un primero con las bases de datos recogidas por expertos y publicadas de manera libre; y un segundo con métodos de extracción no supervisados sobre el corpus de historias clínicas.

Además, posteriormente usamos de un médico con conocimiento experto para corregir y especificar los acrónimos en caso de error o ambigüedad[1].

8.1.1. Bases de datos

Son dos en total las fuentes que usamos, una página web de uso muy común entre los especialistas ofrecida por la SEDOM o Sociedad Española de Documentación Médica [25] y el AbreMES-DB, un repositorio de abreviaturas recogidas junto con su definición de manera automática de publicaciones Españolas.

[1] No es posible llevar a cabo en todos los casos dicha desambiguación, e.g. DL, dependiendo del contexto, puede significar Decúbito Lateral o Dislipemia.

En total poseen, en el momento de escribir este texto, un total de 5551 y ... abreviaturas respectivamente.

8.2. NegEx-MES

Para tener en cuenta las negaciones en el texto usamos el algoritmo NegEx, el cual parte de un pequeño corpus de negaciones. Su versión traducida en español basada en el algoritmo original es NegEx-MES [22] creada gracias al Plan TL (Plan de impulso a las tecnologías del lenguaje) del gobierno español.

Este ofrece múltiples archivos en formato de texto plano con los distintos tipos de negaciones con y sin lemmas, en español e ingles[2].

Los archivos que finalmente usamos son "conjunciones.txt", "neg-phrases.txt", "post-neg-phrases.txt" y "pseudo-neg-phrases.txt" correspondiendo con las categorías definidas en el trabajo original.

[2]El sistema también ofrece herramientas en java gratuitas pero en nuestro caso no nos son útiles

Parte III

Documentación técnica

Capítulo 9

Análisis

En este apartado se van a describir los actores y los requisitos del sistema que se va a desarrollar: requisitos funcionales y no funcionales, de interfaz de usuario y de información.

9.1. Definición de Actores

En esta sección enumeramos los actores principales del sistema, dada la naturaleza del proyecto tan solo tenemos uno, el usuario de la herramienta/sistema:

Usuario

Usuario de la herramienta, técnico que aplica el procesamiento a los datos usando un módulo o la totalidad del sistema.

9.2. Requisitos

En este apartado agrupamos todos los requisitos que deberá cumplir el sistema final.

9.2.1. Funcionales

- Los usuarios podrán transformar HC en pdf a html y xml
- Los usuarios podrán anonimizar las HC de manera automática.
- Los usuarios podrán detectar la estructura de texto jerárquico de las HC
- Los usuarios podrán encontrar las secciones oficiales de las HC
- Los usuarios podrán segmentar las HC

[0]Historias Clínicas

- Los usuarios podrán tokenizar las HC
- Los usuarios podrán detectar los acrónimos de las HC
- Los usuarios podrán sustituir por su definición los acrónimos detectados
- Los usuarios podrán detectar las sentencias negadas y ambivalentes de las HC
- Los usuarios podrán extraer las partes del habla de los tokens
- Los usuarios podrán detectar las entidades[1] de las HC
- Los usuarios podrán normalizar la terminología médica de las entidades detectadas
- Los usuarios podrán extraer de manera ordenada la información generada en el procesado de las HC
- Los usuarios podrán entrenar los algoritmos del sistema sobre su propia base de datos.

9.2.2. No funcionales

- La herramienta estará desarrollada en Python 3.6
- Los archivos procesados iniciales tendrán formato pdf o html compatible con la herramienta
- Los resultados de todo proceso tendrán formato xml
- La base de datos UMLS estará disponible en un servidor MySQL
- La aplicación estará disponible en español
- La herramienta podrá ser desplegada en cualquier sistema compatible con Python
- El formato de extracción final de información sera de xml

9.2.3. De información

- El sistema almacenará las estructuras de texto comunes del conjunto de HC
- El sistema almacenará las secciones comunes del conjunto de HC
- El sistema deberá, de manera opcional, almacenar el modelo entrenado de POS
- El sistema deberá, de manera opcional, almacenar el modelo entrenado de segmentación

[1] Enfermedades, síntomas, drogas, etc.

9.3. Plan de control

Aquí desarrollamos nuestro plan para controlar todas las etapas del proyecto.

9.3.1. Gestión de requisitos

En el probable caso de que ocurran cambios en los requisitos durante el desarrollo del proyecto, se llevaran a cabo los siguientes pasos:

- Se analizaran y se priorizarán los cambios junto con sus consecuencias inmediatas y a largo plazo.
- Se incorporara dichos cambios al proyecto
- Se evaluaran los nuevos riesgos y se establecerá el nuevo calendario

9.3.2. Gestión de calendario

Para cumplir con el calendario se mantendrán hitos que nos permitirá monitorizar las desviaciones que ocurran sobre la planificación original.

9.4. Plan de gestión de riesgos

Aquí analizaremos los posibles riesgos que prevemos pueden ocurrir a lo largo del desarrollo de nuestro proyecto, su impacto y su probabilidad de suceso. Incluiremos además un plan protección para evitar los riesgos y uno de actuación en caso de fallo de la protección y subsecuente ocurrencia.

Impacto / Probabilidad	Muy Alta	Alta	Media	Baja	Muy Baja
Catastrófico	Alta	Alta	Moderada	Baja	Ninguna
Crítico	Alta	Alta	Moderada	Baja	Ninguna
Marginal	Moderada	Moderada	Baja	Ninguna	Ninguna
Despreciable	Moderada	Baja	Baja	Ninguna	Ninguna

Cuadro 9.1: Matriz de exposición a riesgos

R01	**Perdida de código y/o documentación**
Descripción	Desaparición, perdida o corrupción total o parcial del código y/o documentación que compone el proyecto.
Impacto	Crítico
Probabilidad	Baja
Plan de protección	Usar una herramienta de control de versiones y realizar copias de seguridad periódicamente
Plan de contingencia	Cambiar calendario y rehacer trabajo perdido

R02	**Interrupción inesperada del trabajo**
Descripción	Debido a una razón imprevisible se ve necesario detener el desarrollo de la aplicación.
Impacto	Crítico
Probabilidad	Media
Plan de protección	Dar a las iteraciones tiempos holgados para el caso de situaciones imprevisibles.
Plan de contingencia	En el peor caso reducir el alcance del proyecto, cambiar calendario y continuar el trabajo cuando sea posible.

R03	**Problema de estimación temporal**
Descripción	Debido a una mala estimación del tiempo de desarrollo de alguna tarea, se requiere de un mayor trabajo/tiempo para llevarla a cabo.
Impacto	Marginal
Probabilidad	Alta
Plan de protección	Analizar lo más posible las previsiones temporales del trabajo.
Plan de contingencia	Re-estimar tareas/iteraciones y actualizar calendario

R04	**Problema del diseño inicial**
Descripción	A mitad de proyecto se encuentra que el diseño no es el más adecuado para el desarrollo del proyecto
Impacto	Marginal
Probabilidad	Media
Plan de protección	Investigar lo más posible durante la etapa de análisis.
Plan de contingencia	Iterar y volver a la investigación, rehacer diseño y re-estimar tiempos y calendarios

R05	**Cambio requisitos**
Descripción	A lo largo del proyecto se ve necesario modificar los requisitos establecidos al principio del proyecto.
Impacto	Marginal
Probabilidad	Muy Alta
Plan de protección	Analizar lo más posible el problema durante la etapa de análisis e investigación.
Plan de contingencia	Modificar requisitos, cambiar estimaciones y actualizar calendario.

R06	**Módulo problemático**
Descripción	A lo largo del proyecto se encuentra un módulo que por alguna razón resulta imposible/impracticable de desarrollar.
Impacto	Catastrófico
Probabilidad	Baja
Plan de protección	Investigar a priori la pipeline del sistema nlp y establecer metas conservadoras.
Plan de contingencia	Buscar soluciones paralelas o, en el peor caso, modificar requisitos.

R07	**Capacidad final subpar**
Descripción	Al final del desarrollo se encuentra un producto con un conjunto menor de las características buscadas.
Impacto	Despreciable
Probabilidad	Alta
Plan de protección	Establecer metas conservadoras y dar tiempos holgados.
Plan de contingencia	Iterar sobre los módulos, proponer mejoras, modificar requisitos y actualizar calendario.

9.5. Modelo de Dominio

A continuación mostramos el diagrama que contiene el modelo de dominio del sistema, que muestra las clases a nivel de análisis. Dada la naturaleza exploratoria del proyecto y su creación como herramienta de linea de comandos, es relativamente sencilla.

Tras acabar con la etapa de análisis, continuaremos con la de diseño.

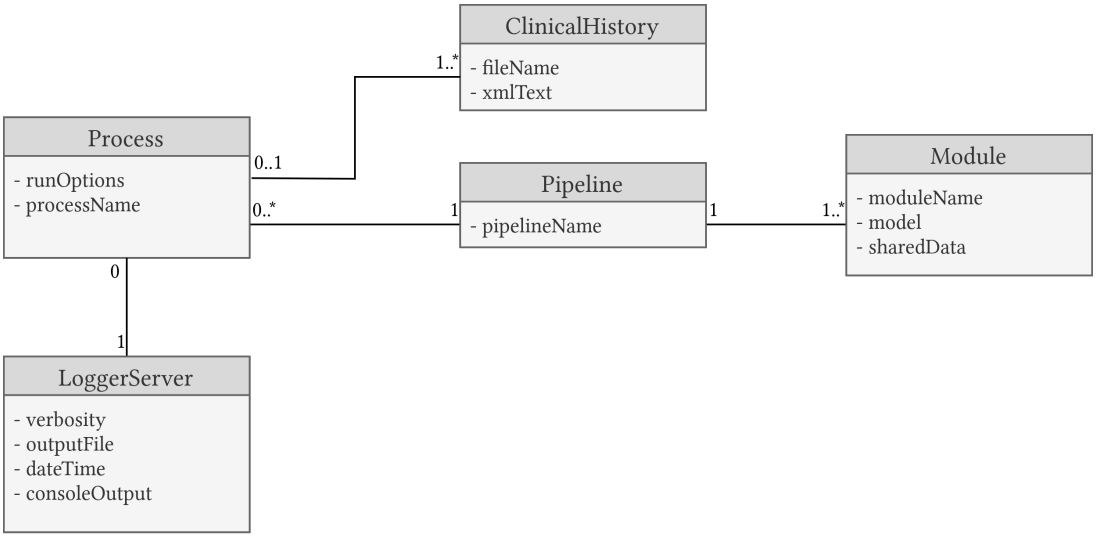

Figura 9.1: Modelo de Dominio

Capítulo 10

Diseño

En este capítulo será desarrollado el diseño de la aplicación a través de diagramas UML[1].

10.1. Patrones de diseño

Para la creación de la arquitectura del sistema se ha seguido el patrón **tubería**, en el cual las HC pasan por una cadena de módulos de procesamiento que van añadiendo nueva información. Nos hemos decantado por este diseño debido a que necesitamos pasar los archivos de datos por sucesivos procesos que, no solo son independientes, sino que están encadenados, es decir, el resultado de un proceso o módulo es la entrada del siguiente.

Tan solo una etapa se sale de la norma y es la de detección de secciones, que necesita trabajar con toda la base de datos al mismo tiempo.

10.2. Casos de uso

CU-1	**Detección de la estructura de HCs**
Actor	Usuario
Descripción	El usuario pide detectar las estructuras y secciones comunes de un conjunto de HC
Precondición	Ninguna
Postcondición	Ninguna
Secuencia Normal	**1.** El usuario pide ejecutar el algoritmo de detección **2.** El sistema toma las HC, las procesa y guarda el resultado en un archivo de configuración para posterior uso.
Excepciones	Las HC tienen un formato no compatible con el sistema

[1]Unified Modeling Language

CU-2	**Procesado de HC por uno o varios módulos PLN**
Actor	Usuario
Descripción	El usuario pide procesar un conjunto de HCs de manera secuencial con uno o varios módulos PLN del sistema.
Precondición	Ninguna.
Postcondición	Ninguna
Secuencia Normal	**1.** El usuario pide ejecutar un módulo PLN **2.** El sistema crea una pipeline, toma las HCs, las procesa y guarda los resultados finales.
Excepciones	Las HC tienen un formato no compatible con el sistema

10.3. Modelo de datos

Aunque existen varios archivos de datos necesarios para la creación de los modelos, en todo el proyecto solo hay una sola base de datos; esta es la base de UMLS que vamos mostrar en un diagrama a continuación (no mostraremos toda la base de datos UMLS sino tan solo aquella parte de la que damos uso).

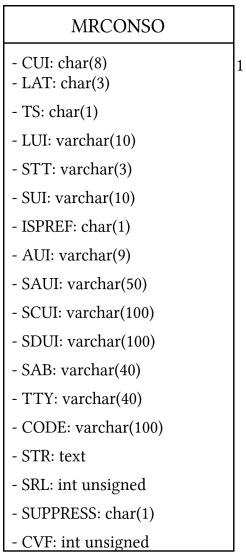

Figura 10.1: Modelo de Datos

En la figura 10.1 se observa la tabla principal **MRCONSO** que contiene la información necesaria para la identificación de la terminología.

Como en la práctica es inviable usar solo dicha tabla, hemos de incluir una auxiliar llamada **MRXW_SPA** que facilita las consultas por token (además de crear varios índices extra para los identificadores, en concreto de CUI).

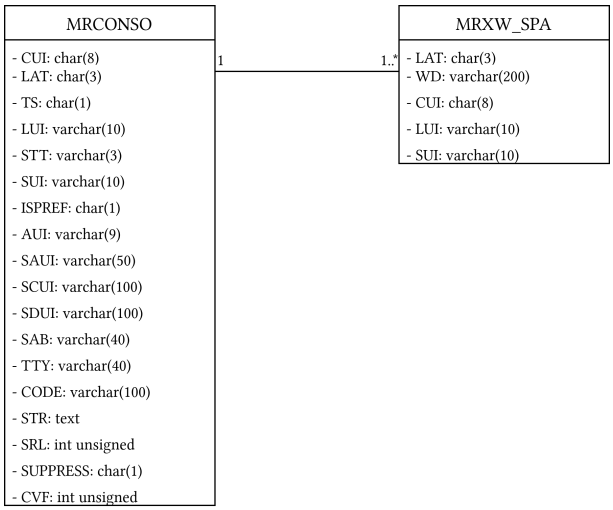

Figura 10.2: Modelo con tabla auxiliar

10.4. Diagramas de clase y de secuencia

La arquitectura del sistema está basada en el patrón pipeline. En la figura 10.3 se muestra la arquitectura global y en el resto de las figuras se da el detalle.

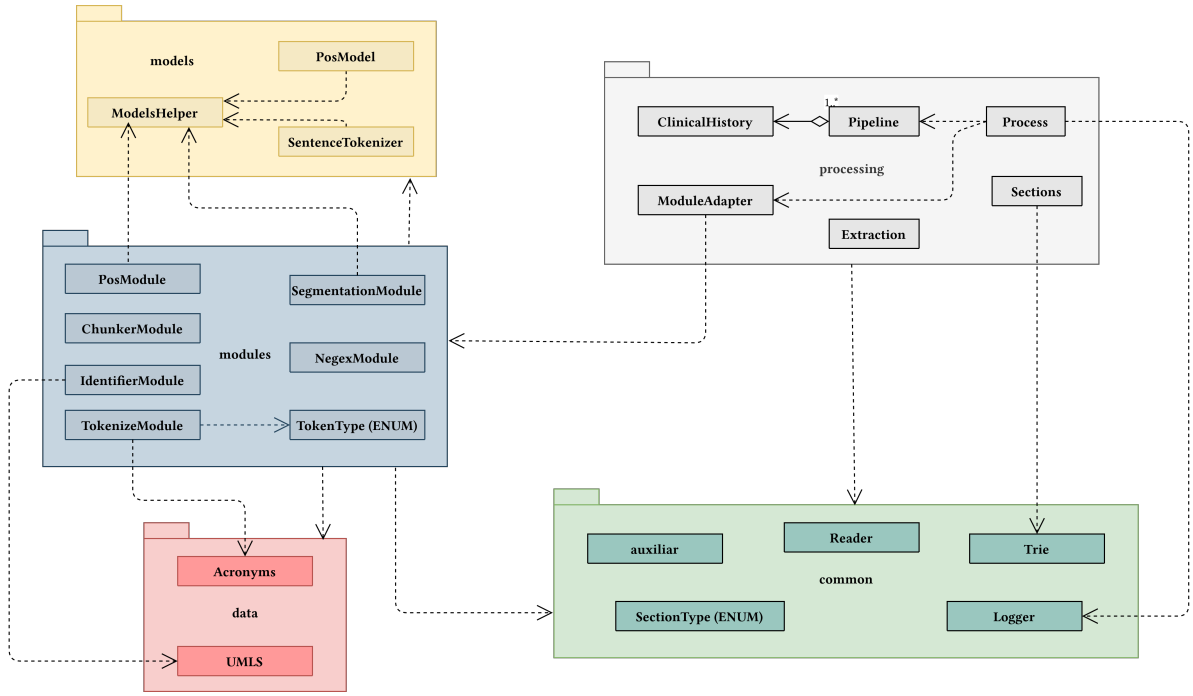

Figura 10.3: Arquitectura global del sistema

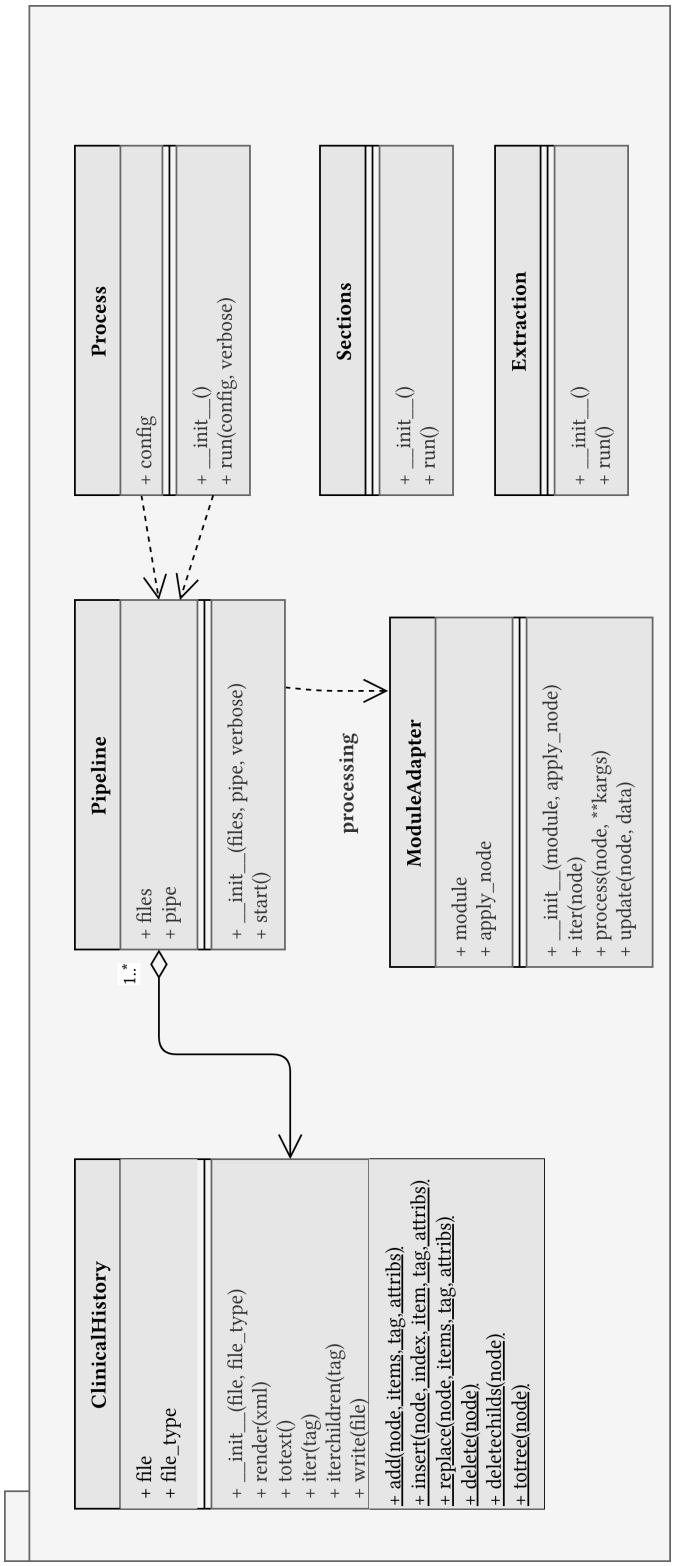

Figura 10.4: Paquete de procesado de datos

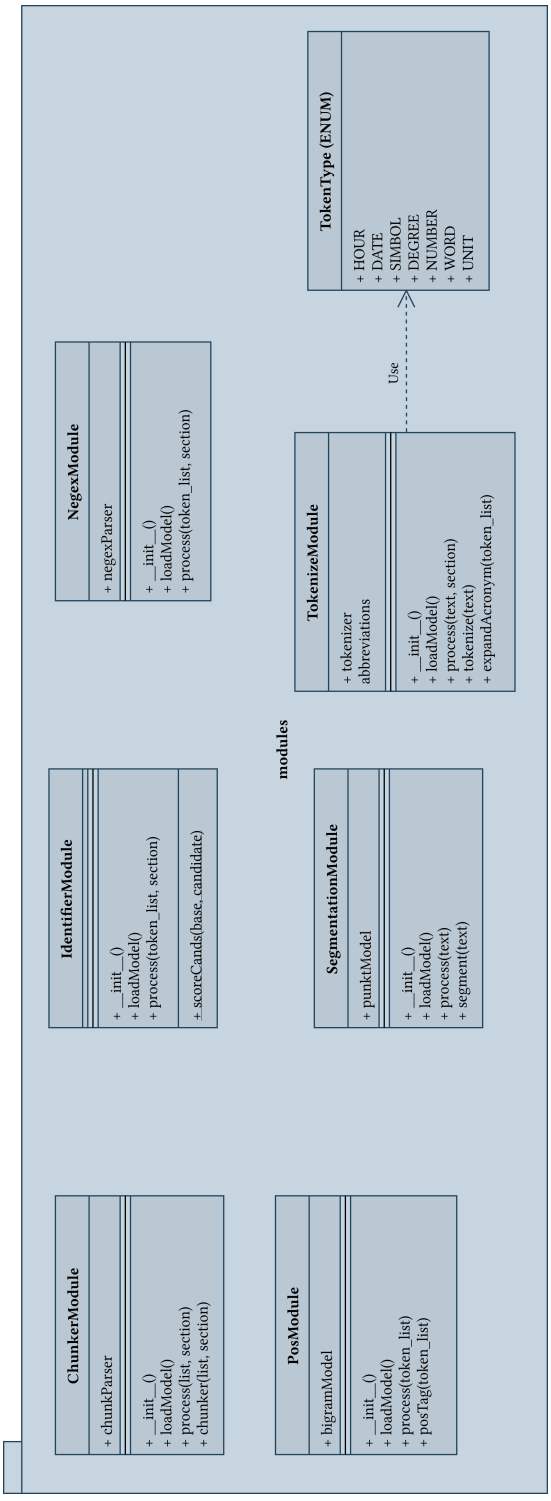

Figura 10.5: Paquete de módulos de procesado

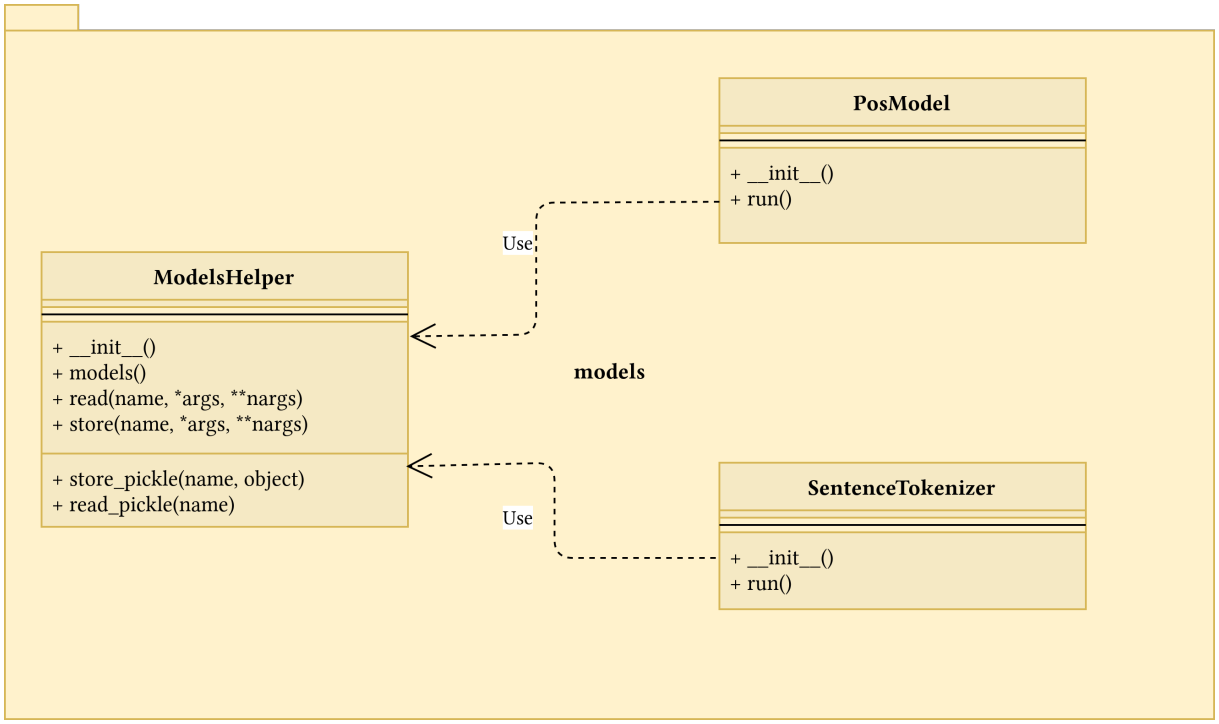

Figura 10.6: Paquete de creación de modelos PLN

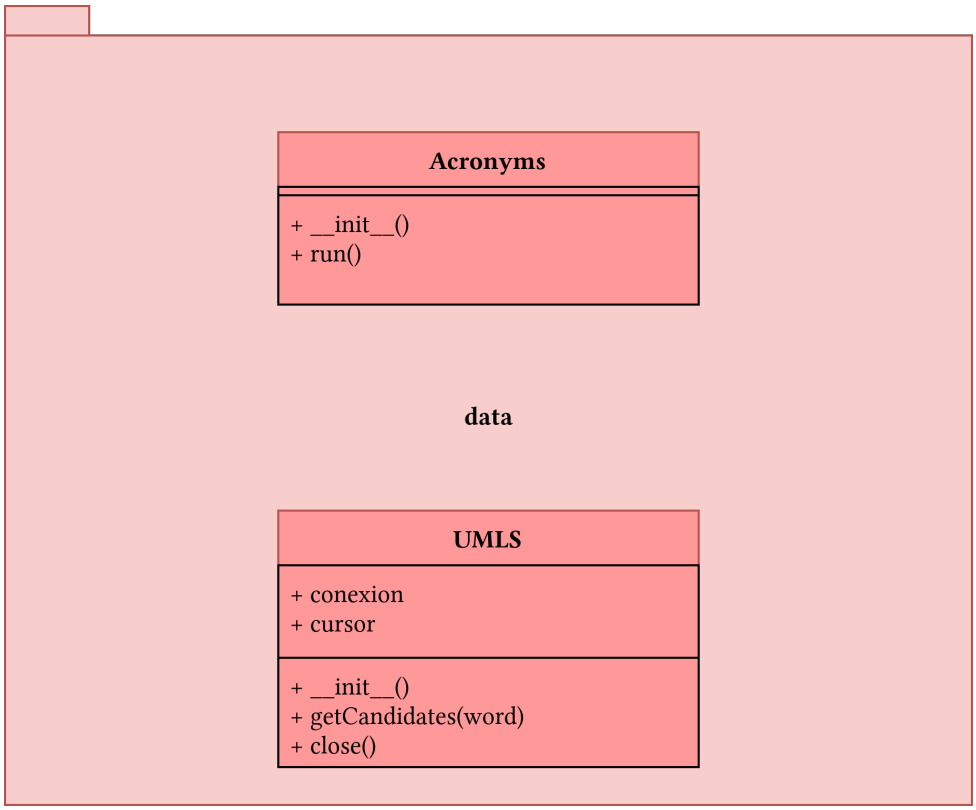

Figura 10.7: Paquete de manipulación de datos

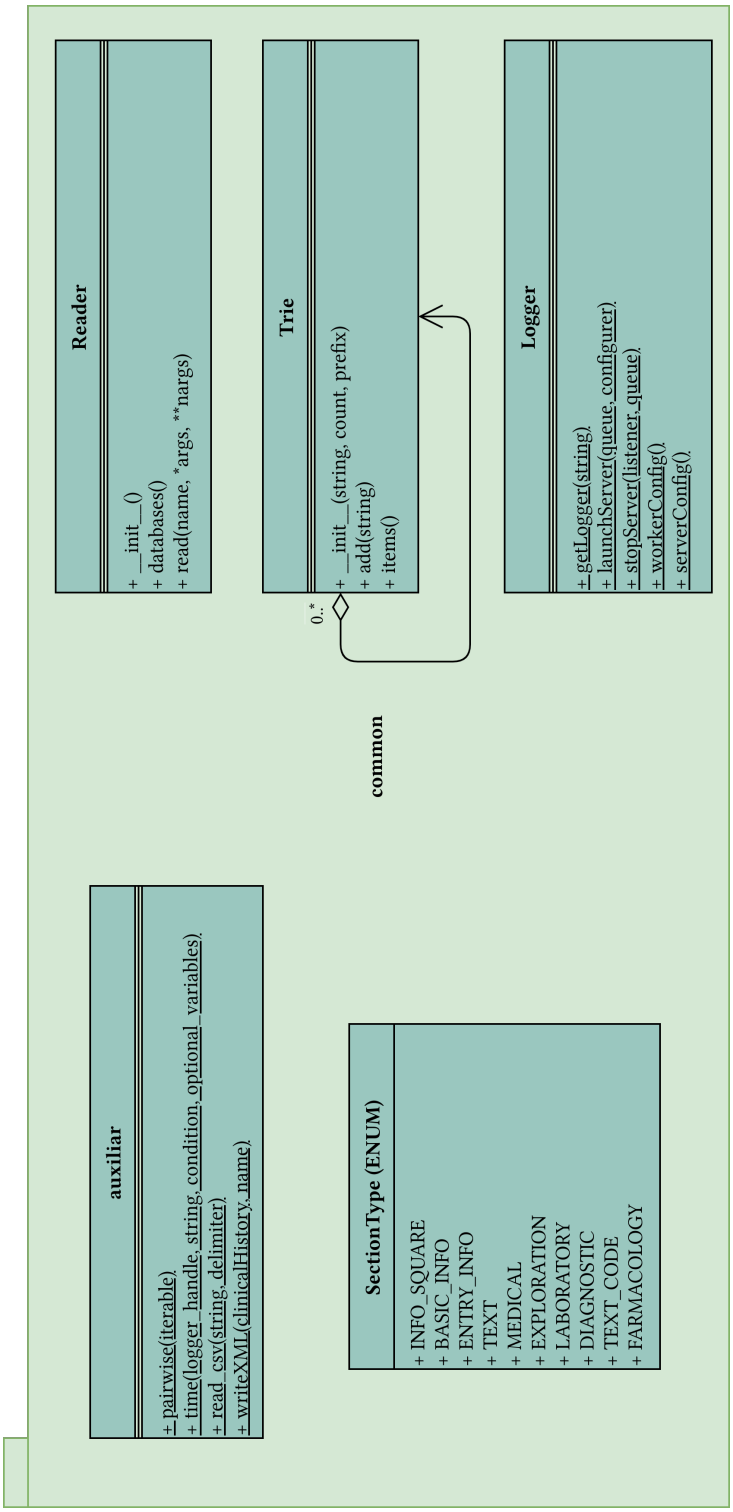

Figura 10.8: Paquete de funciones auxiliares

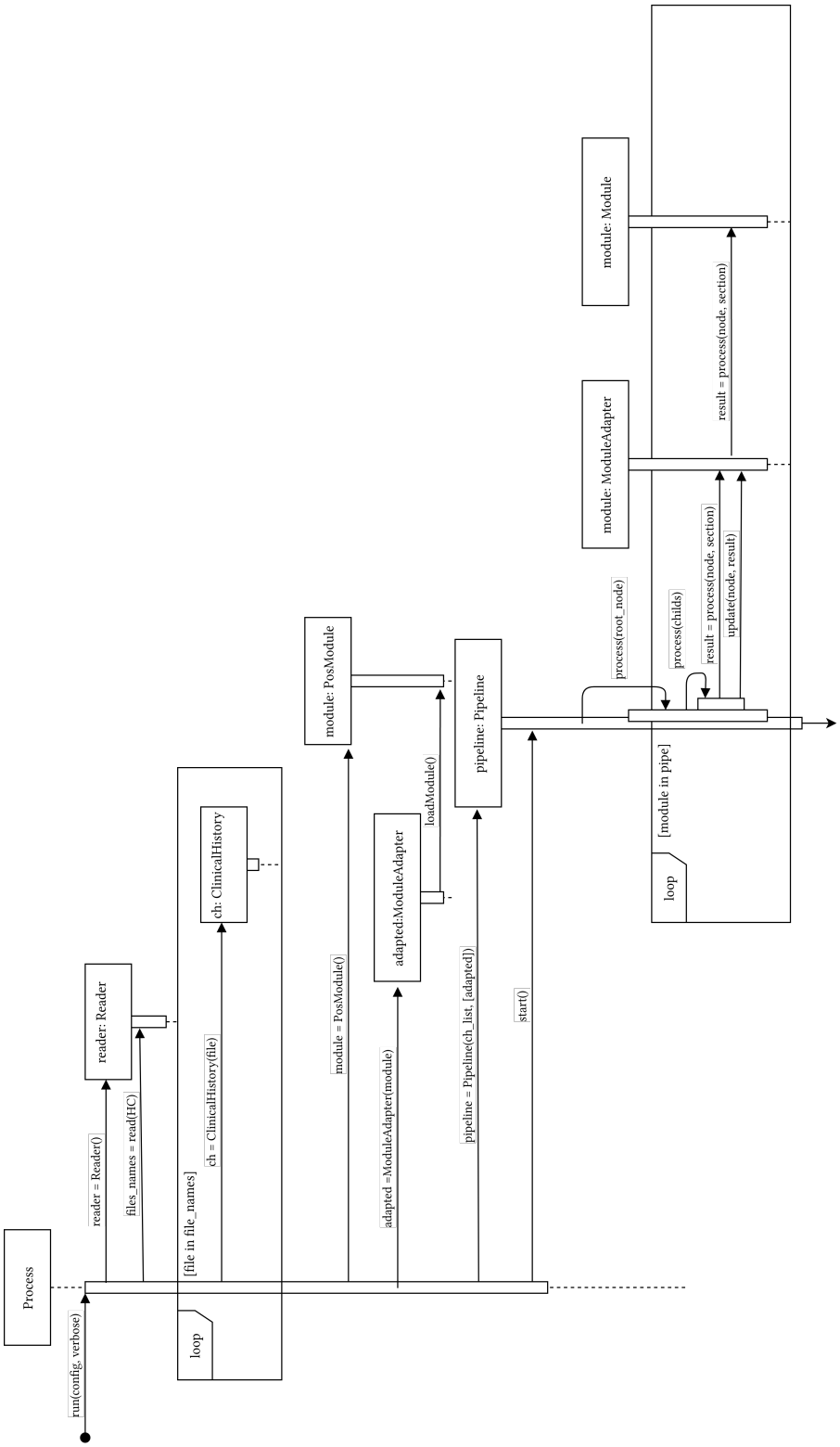

Figura 10.9: Diagrama de secuencia del CU-2

Capítulo 11

Implementación

En este capitulo serán desarrollado los detalles específicos de implementación del prototipo, los problemas encontrados a lo largo del proyecto y las soluciones propuestas. Describiremos, una por una, todas las etapas del procesado de datos y, finalmente, acabaremos con los resultados y las conclusiones del proyecto.

11.1. Entorno de desarrollo

Para el desarrollo de la aplicación se ha utilizado el editor **NeoVim**, un entorno de desarrollo moderno derivado de Vim. Se ha elegido dado la eficiencia de desarrollo en este editor, además de la experiencia previa del autor. Entra las utilidades que provee se encuentran la integración con Git y el resaltado de la sintaxis de Python.

11.2. Control de versiones

Para mantener un control del código desarrollado se ha utilizado la herramienta **Git**. Todo el código ha sido periódicamente subido a un repositorio privado de Gitlab; esto ha facilitado el desarrollo de código sin riesgo de perdida de datos, el poder cambiar de maquina con facilidad y ha facilitado el retroceso en caso de error.

11.3. Base de datos

Para el despliegue de la base UMLS se ha utilizado **MySQL**, un gestor de datos relacionales de código abierto, creada por Oracle.

Los datos necesarios se han generado a través de las herramientas y scripts proporcionados por UMLS.

11.4. Desarrollo

Como se comento previamente, se a seguido un patrón de tipo **tubería** en desarrollando módulos independientes a través los cuales los datos se procesan de manera sucesiva y en orden, hasta obtener el resultado final. Además, dichos módulos siguen un patrón **estrategia** para poder uniformizar el acceso a los algoritmos y un patrón **adaptador** para adaptarlos al sistema, que trabaja con xml, mientras que estos trabajan sobre texto plano.

Por otro lado, la estructura principal que recoge las HC implementa el patrón **iterador** para recorrer las diferentes secciones y textos de los datos.

Para mantener la facilidad de manipulación en todo momento, todos los resultados intermedios así como el producto final se mantienen en un formato xml.

Por último, para facilitar la visualización de cada etapa se incluyen ejemplos de dicho xml, estos son arquetipos fabricados de una posible historia clínica sin referencia alguna a datos reales.

11.4.1. Lectura de datos

Como se ha comentado previamente, existe una cantidad variada y heterogénea de bases de datos, con múltiples orígenes y formatos. Por ello, y para facilitar su reuso, se a elegido abstraer toda esa complejidad en un conjunto de objetos capaces de resumir de manera organizada toda la información necesaria para el proyecto y trasformarla a las estructuras óptimas para su uso.

En el listado 11.1 se muestra un ejemplo de un posible documento sobre el que se va a extraer la información. Como se observa se trata de un archivo xml en el cual se representa el texto junto con la estructura de subordinación de elementos presente en el documento original.

Este es el primer resultado del preprocesamiento, en el cual transformamos el pdf original a un xml, anonimizando y eliminando la información innecesaria[1].

11.4.2. Detección Secciones

Dado que los datos están semi organizados en categorías distinguidas por títulos, subrayados y otras características, es necesario separar, en un primer paso, aquel contenido único de la historia clínica de lo que se puede considerar el armazón o la estructura del archivo. Para realizar esto se debe, de manera eficiente, analizar que partes del texto aparecen con la mayor prevalencia.

[1]Existe un paso intermedio de transformación de pdf a html, a través de la herramienta **pdftohtml**, en el cual se lleva a cabo el anonimizado.

```
<?xml version="1.0" encoding="utf-8"?>
<raw>Informe clínico de alta
        <raw>DATOS ASISTENCIALES</raw>
        <raw>Esp.: CARDIOLOGIA</raw>
        <raw>Fecha de Ingreso: 10/02/2005 21:30 Tipo de Ingreso: Urgente</
            ↪ raw>
        <raw>Fecha / Hora de Alta: 23/04/2005 11:00</raw>
        <raw>Fecha de Nacimiento: 09/11/1960 País: ESPAÑA </raw>
        <raw>Edad: 45 años Sexo: Varón
                <raw>Motivo de alta:
                        <raw> Traslado a domicilio</raw>
                </raw>
        </raw>
        <raw>.Motivo de Ingreso:
                        <raw>Código infarto</raw>
        </raw>
        <raw>Antecedentes:
                <raw>Enfermedades previas:
                        <raw>- FRCV: HTA y DM tipo 2 en tratamiento
                            ↪ farmacológico. No DL.</raw>
                        <raw>- Tabaquismo.</raw>
                </raw>
                <raw>Alergia a:
                        <raw>El paciente no presenta alergias conocidas.</
                            ↪ raw>
                </raw>
        </raw>
</raw>
```

Listing 11.1: XML ejemplo documento base

Una vez realizado, se separara la estructura en dos tipos diferentes: cadenas que representan inicios de sección y fracción resto. Nos ayudamos en el hecho de que en toda España se han estandarizado las secciones obligatorias que han de estar presentes en toda historia clínica [21].

Trie vs Hashmap

Para el primer paso se necesita contar cadenas y para eso se necesita compararlas. Existen dos formas eficientes de comparar, de manera exacta, una gran cantidad de textos entre si: a través de una tabla hash, muy veloz pero se a de considerar el problema de las colisiones; y a través de un árbol de prefijos, que reduce la comparación de una cadena con un número indefinido de cadenas a $\mathcal{O}(n)$ siendo n el tamaño de la singular cadena a comparar.

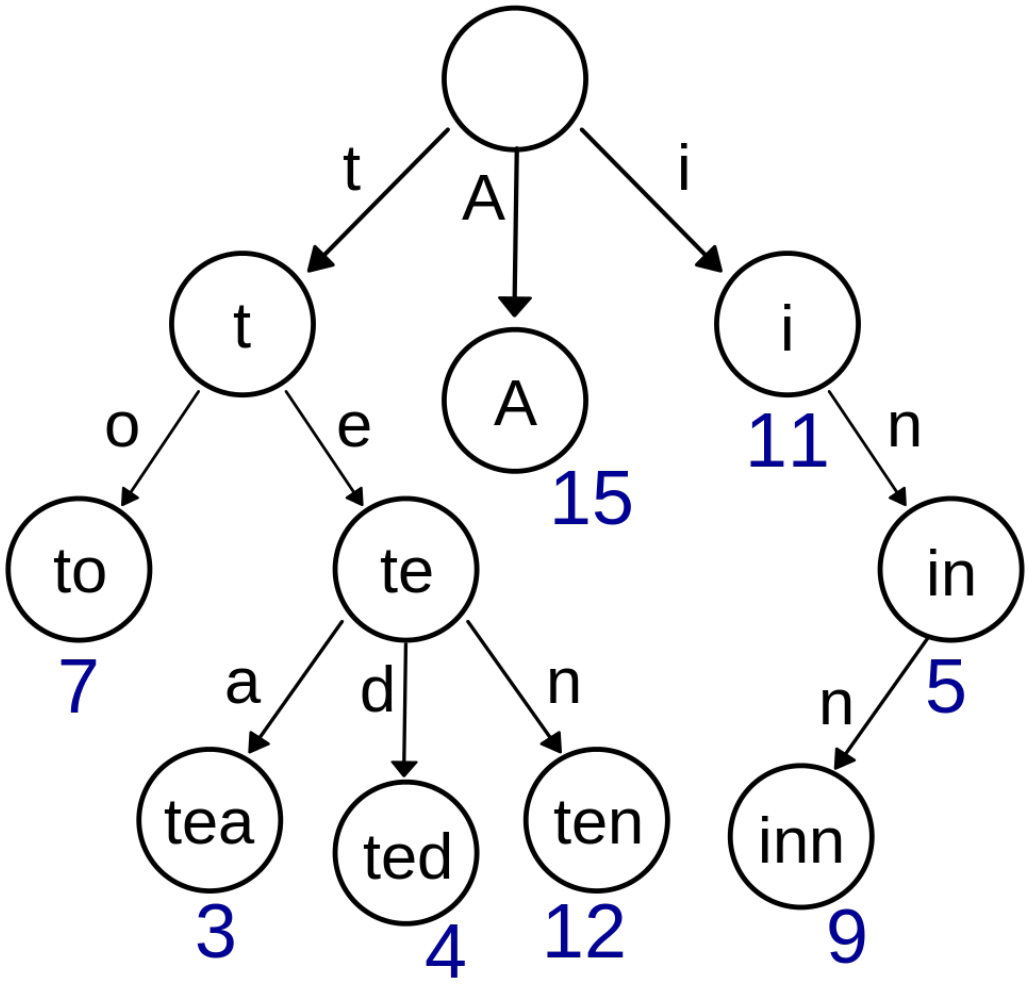

Figura 11.1: Ejemplo de Trie - Fuente Wikipedia

Este último es un **Trie**, o árbol de prefijos. Funciona manteniendo un árbol en el cual

cada nodo partiendo de la raíz contiene una cadena de texto o prefijo, el cual (el texto) se divide siempre que haya dos cadenas diferentes; es decir, si se tienen las cadenas "exacto" y "exactamente", se tendrá un primer nodo con "exact" y dos nodos hijos con "o" y "amente".

Esta estructura no solo permite la rápida comparación de cadenas exactas, sino que además facilita la comparación "difusa" o de coincidencias parciales. Es por todo esto, que nos decantamos por este segundo método.

Implementación

Modificamos la estructura del **Trie** original ligeramente para que mantenga un contador de repeticiones de cada cadena completa. Así, tan solo es necesario insertar todas las cadenas en la estructura, extraer y ordenar de mayor a menor cada cadena encontrada; las que superen un umbral son nuestros candidatos a secciones.

Para el segundo paso calculamos la distancia de las secciones oficiales a las cadenas obtenidas. Es necesario ya que, en general, las secciones no se incluyen de manera exacta, pudiendo cambiar palabras o puntuación de manera intencionada o accidental, entre otras cosas. Para esto existen varias funciones de distancia entre cadenas de caracteres y tras probar varias nos decantamos con una de uso generalizado, la distancia de Levenshtein o de edición, la cual da buenos resultados para esta tarea y es eficiente, siempre y cuando las cadenas sean relativamente cortas. Otra facilidad de esta métrica es que no es necesario implementarla pues dado su uso generalizado ya está en la biblioteca **NLTK**[15] de Python.

```
<?xml version="1.0" encoding="utf-8"?>
<struct>Informe clínico de alta
        <raw>DATOS ASISTENCIALES</raw>
        <raw>Esp.: CARDIOLOGIA</raw>
        <raw>Fecha de Ingreso: 10/02/2005 21:30 Tipo de Ingreso: Urgente</
            ↪ raw>
        <raw>Fecha / Hora de Alta: 23/04/2005 11:00</raw>
        <raw>Fecha de Nacimiento: 09/11/1960 País: ESPAÑA </raw>
        <raw>Edad: 45 años Sexo: Varón
                <section sec_name="Motivo de Alta" sec_type="Texto">Motivo
                    ↪ de alta:
                        <raw> Traslado a domicilio</raw>
                </section>
        </raw>
        <section sec_name="Motivo de Ingreso" sec_type="Texto + código">.
            ↪ Motivo de Ingreso:
                        <raw>Código infarto</raw>
        </section>
        <section sec_name="Antecedentes" sec_type="Texto">Antecedentes:
```

```
            <section sec_name="Enfermedades Previas" sec_type="Texto">
               ↪ Enfermedades previas:
                    <raw>- FRCV: HTA y DM tipo 2 en tratamiento
                       ↪ farmacológico. No DL.</raw>
                    <raw>- Tabaquismo.</raw>
            </section>
            <section sec_name="Alergias" sec_type="Texto">Alergia a:
                    <raw>El paciente no presenta alergias conocidas.</
                       ↪ raw>
            </section>
        </section>
</struct>
```

Listing 11.2: XML ejemplo documento con secciones y estructuras

El resultado de esta etapa se observa en el listado 11.2. En esta ya se han encontrado todo el texto considerado estructura (marcado con la etiqueta "struct") y toda frase que correspondiera a una sección especifica (marcado con la etiqueta "section"), además de especificar de que sección se trata y de que tipo de contenido se espera en su interior.

11.4.3. División del texto

Obtenidas las secciones, se procede a separar el texto en frases con sentido.

Segmentación

Para esto, como comentamos previamente, usamos el algoritmo no supervisado de detección de final de sentencias **Punkt**[11]. Como hablo en el apartado teórico, este algoritmo se basa en la detección de acrónimos acabados en puntos a través de heurísticas probabilísticas. Para su uso en nuestro sistema, utilizamos la implementación base ofrecida por la biblioteca **nltk** [15] y la modificamos añadiéndole nuestro conocimiento a priori sobre el vocabulario especifico de ámbito[2], gracias a la base de datos de abreviaturas recogidas previamente[10] [25].

Esto nos permite personalizar el algoritmo aprovechando tanto el conocimiento previo como el aprendizaje sobre los datos.

```
<?xml version="1.0" encoding="utf-8"?>
<struct>Informe clínico de alta
        ...
        <section sec_name="Motivo de Ingreso" sec_type="Texto + código">.
           ↪ Motivo de Ingreso:
                <lines>
```

[2]Desarrollado en el componente *SegmentationModule*

```
                    <line>Código infarto</line>
                </lines>
        </section>
        <section sec_name="Antecedentes" sec_type="Texto">Antecedentes:
                <section sec_name="Enfermedades Previas" sec_type="Texto">
                    ↪ Enfermedades previas:
                    <lines>
                        <line>- FRCV: HTA y DM tipo 2 en tratamiento
                            ↪ farmacológico.</line>
                        <line> No DL.</line>
                    </lines>
                    <lines>
                        <line>- Tabaquismo.</line>
                    </lines>
                </section>
                <section sec_name="Alergias" sec_type="Texto">Alergia a:
                    <lines>
                        <line>El paciente no presenta alergias
                            ↪ conocidas.</line>
                    </lines>
                </section>
        </section>
</struct>
```

Listing 11.3: XML ejemplo documento segmentado

Como se observa en la figura 11.3 hemos dividido los diferentes párrafos ("lines") en lineas ("line") independientes unas de otras. En todo momento mantenemos la jerarquía, con los diferentes conjuntos de lineas dentro de las correspondientes secciones.

Tokenizado

Esta etapa es una de las que, en comparación del tiempo reservado a ella, más hemos tardado. Si bien el modelo básico de expresiones regulares no a cambiado a lo largo del desarrollo de la aplicación, se ha iterado, en más de una docena de veces, sobre las reglas.

En un comienzo, se partió de un conjunto de reglas genérico obtenido a través de la red en páginas especializadas; a lo largo de la implementación del resto de módulos se encontró, numerosas veces, excepciones a las reglas, tokens demasiado estrictos o palabras directamente no tenidas en cuenta. Por consiguiente, se requirió de múltiples iteraciones refinando el sistema de manera manual hasta encontrar todos los problemas.

```
<?xml version="1.0" encoding="utf-8"?>
<lines>
```

```
        <line>- FRCV: HTA y DM tipo 2 en tratamiento farmacológico.
                <token>-</token>
                <token>factor</token>
                <token>de</token>
                <token>riesgo</token>
                <token>cardiovascular</token>
                <token>:</token>
                <token>hipertensión</token>
                <token>arterial</token>
                <token>diastólica</token>
                <token>y</token>
                <token>densitometría</token>
                <token>tipo</token>
                <token>2</token>
                <token>en</token>
                <token>tratamiento</token>
                <token>farmacológico</token>
                <token>.</token>
        </line>
        <line>No DL.
                <token>no</token>
                <token>decúbito</token>
                <token>lateral</token>
                <token>.</token>
        </line>
</lines>
```

Listing 11.4: XML ejemplo documento tokenizado

A continuación mostramos el resultado final de todo este proceso de refinamiento[3]:

```
        r'(\d\d?:\d\d?(?::\d\d?)?(?:(?=[:)]+)|$))' # Horas
        r'|(\d\d?(?P<simb>[-\\/])\d\d?(?P=simb)\d\d(?:\d\d)?(?:(?=[:)]+)|$
            ↪ ))' # Fechas
        '|([=:\*])' # Simbolos separables
        '|([º][CF])' # Grados (unidades)
        '|(\d+/\d+)' # Numeros compuestos
        '|([+-]?\d+(?:(?:[-.,_+\']|x)\d+)*[-_+\']?)'* # Numeros
        '|(‘‘)|(<<<?)|(>>>?)|(\'\')' # Comillas (Simbolos)
        '|(\.\.\.)' # Tres puntos
        '|([A-Za-z]+(?:[-\'_+][A-Za-z]+)+)' # Palabras compuestas
        '|([^\s,()]+/[^\s,()]+)|([%])' # Unidades
        '|(\w+)' # Palabra alfanumerica
```

[3]Contenido dentro del módulo de procesado de datos *TokenizeModule*

```
        '|(\W)' # Simbolos
        '|(.)' # Resto
```

Listing 11.5: Expresiones regulares de tokenizado

Como se observa, posteriormente, en la figura 11.6, además de separar el texto lo hemos clasificado (atributo "type") a través de las reglas de tokenizado. Esto es un subproducto de este módulo que nos facilitara a posteriori algunas de las tareas y que no decremente de ninguna manera la velocidad del sistema.

Por último, en esa misma figura al comienzo de la linea, encontramos la otra tarea que lleva a cabo este módulo. El acrónimo "FRCV" ha sido correctamente seleccionado y expandido a su definición –factor de riesgo cardiovascular– basándose en la revisión manual de la base de datos de acrónimos.

11.4.4. POS tagging

Una vez separado el texto en tokens pasamos a clasificarlos en su categoría gramatical. Para ello usamos, como se menciono brevemente, un modelo de bigramas entrenados sobre la base de datos **SPACCC**; además, como en la etapa de segmentación partimos de la librería **nltk** para llevarlo a cabo[4].

```
<?xml version="1.0" encoding="utf-8"?>
<line>No DL.
        <token orig="No" type="word" pos="RN">no</token>
        <token type="word" pos="VMII3S0">decúbito</token>
        <token type="word" pos="AQ0CS0">lateral</token>
        <token orig="." type="simbol" pos="Fp">.</token>
</line>
```

Listing 11.6: XML ejemplo POS

11.4.5. NegEx

Antes de pasar a la fase de sintaxis, creamos el modulo de detección de negaciones. Este es relativamente simple, nos basamos en el algoritmo NegEx que consiste en la detección de varios tipos de negaciones y palabras ambiguas para clasificar el sentido de una frase.

Se implementa como un módulo más usando el vocabulario guardado y expresiones regulares[5], cuando se encuentra un caso positivo se añade un atributo al token con la información necesaria.

[4]La construcción del modelo se llevaría a cabo en el *PosModel* y se utiliza a través del módulo de procesado de datos *PosModule*

[5]Implementado en *negexModule*

```
        <token orig="No" type="word" pos="RN">no</token>
```

Listing 11.7: XML ejemplo POS

Posteriormente estos datos son usados en la extracción de información al nivel de información negada o ambigua.

11.5. Análisis sintáctico

Continuamos con la etapa de **Chunking** en la cual obtendremos, de manera sencilla, las entidades del texto, de las cuales posteriormente extraeremos candidatos a normalizar.

Para implementarlo usamos un sistema basado en reglas; partimos de una gramática libre de contexto y, a lo largo de las iteraciones, la reducimos a expresiones regulares (pues la gramática era demasiado potente para la tarea). Aun así continuamos usando la misma herramienta (regexpparser de **nltk**) con la que iniciamos el sistema para dejar hueco a posibles mejoras del sistema.

Como con el módulo de tokenizado, esta fase nos llevo mucho más tiempo del reservado, necesitando de múltiples iteraciones para refinar el sistema. Más allá, si en un principio partimos de un sistema para todo el texto de las historias clínicas, al final del proyecto construimos múltiples diferentes reglas para cada sección.

A continuación mostramos un ejemplo de una de ellas:

```
    "explor": r"""
              XNP_CHUNK: {<N.*>+<AQ.*>*}<Fg>?<Z>
              XNP_CHUNK: {<N.*>+<.*>}<Fg><Z>
              XNP_CHUNK: {<N.*><AQ.*>*(<S.*>*<D.*>*<N.*><AQ.*>*)*}
              """,
```

Listing 11.8: Reglas de chunking para la sección de exploración

Cada linea representa una posible entidad y cada palabra dentro de un "<>" es una etiqueta POS desarrollada previamente (aquí se observa claramente como los módulos se basan en los anteriores para construir información más compleja).

A continuación mostramos los resultados (simplificados) de esta etapa:

```
<?xml version="1.0" encoding="utf-8"?>
<line>- FRCV: HTA y DM tipo 2 en tratamiento farmacológico.
        <token>-</token>
        <chunk>
                <token>factor</token>
                <token>de</token>
```

```
            <token>riesgo</token>
            <token>cardiovascular</token>
        </chunk>
        <token>:</token>
        <chunk>
            <token>hipertensión</token>
            <token>arterial</token>
            <token>diastólica</token>
        </chunk>
        <token>y</token>
        <chunk>
            <token>densitometría</token>
        </chunk>
        <token>tipo</token>
        <token>2</token>
        <token>en</token>
        <token>tratamiento</token>
        <token>farmacológico</token>
        <token>.</token>
</line>
```

Listing 11.9: XML ejemplo documento tokenizado

11.5.1. Normalización

Una vez acabada la extracción de posibles entidades continuamos con la normalización de los términos; aquí usamos el hecho de que la información relevante del texto ha sido previamente localizada (en los "chunks") para reducir el coste de cómputo.

Esto se lleva a cabo a través del módulo de identificación, el cual, dado una entidad/conjunto de términos, extrae todos los candidatos posibles de la base de datos UMLS contenida en un servidor MySQL; estos consisten en todo término de la base de datos que contenga uno o varios de los tokens de la entidad original en su definición, e.g. del término "infarto" obtenemos candidatos de la base como "Infarto hemorrágico suprarrenal". Para hacer estas peticiones rápidas tenemos una tabla (**MRXW_SPA**) con una lista de palabras relacionadas con los términos (y sus identificadores) originales, e.g. Para el término de UMLS "Infarto hemorrágico suprarrenal" tenemos tres entradas en dicha tabla correspondiendo a "Infarto", "hemorrágico" y "suprarrenal".

Esto se lleva a cabo con la siguiente petición:

```
select CUI,STR
from MRCONSO as conso
join (
    select LUI
```

```
        from MRXW_SPA
        where WD like %s limit 150) spa on conso.LUI in (spa.LUI)
    where LAT="SPA" and SUPPRESS="N" limit 150;
```

Listing 11.10: Petición SQL

Resumiendo, se extrae el identificador de token deseado de la tabla **MRXW_SPA** y se extraen (a través de una unión en *LUI*) el identificador de concepto *CUI* y el string *STR* de la tabla **MRCONSO**. Además se establece un limite a los candidatos de 150[6].

Una vez obtenida la lista de candidatos estos se ordenan dependiendo de su distancia a la entidad que buscamos normalizar[7] y de manera jerárquica; es decir, primero se busca las coincidencias sobre la entidad original completa, luego sobre un subconjunto de esta, luego sobre otro aún menor, etc. Al final del proceso retenemos toda identificación que supere un margen predefinido[8].

Por último, para acelerar las peticiones a la base de datos UMLS creamos una cache de tokens identificados que comprobamos antes de llevar a cabo solicitud al servidor.

El resultado final es el siguiente:

```
<?xml version="1.0" encoding="utf-8"?>
    <chunk>
        <token>
            hipertensión
            <info CUI="C0235222" STR="hipertensión arterial diastólica"/>
        </token>
        <token>
            arterial
            <info CUI="C0235222" STR="hipertensión arterial diastólica"/>
        </token>
        <token>
            diastólica
            <info CUI="C0235222" STR="hipertensión arterial diastólica"/>
        </token>
    </chunk>
```

Listing 11.11: XML ejemplo documento tokenizado

En este caso, dada la entidad/chunk "hipertensión arterial diastólica" (que parte originalmente en el texto del acrónimo "HTA") hemos obtenido una coincidencia exacta en la base de datos y la hemos identificado de manera única con el identificador **CUI**[9].

[6]Este límite es necesario ya que ciertos tokens comunes (como "enfermedad") pueden estar en cientos de miles de términos, lo cual ralentizaría el sistema.

[7]Usamos como distancia el número de coincidencias sin tener en cuenta mayúsculas, acentos...

[8]Como se comento en la parte de teoría, el proceso está descrito en detalle en el estudio[16].

[9]Para más información de estos identificadores ir a la sección de UMLS

11.5.2. Extracción de información

La última etapa tiene dos fases que son llevadas a cabo a través de reglas específicas para cada apartado/sección de las historias clínicas. En estas se añade información experta sobre el xml final y se extrae la información a través de la tecnología **XPath**.

En esta etapa se estructura la información generada durante todo el proceso y se organiza en formato csv con una fila por historia clínica/paciente de la base de datos. Es sobre esta base de datos sobre la que se conducen los análisis finales.

Dado que el objetivo principal de este proyecto es el estudio de las técnicas de procesado de lenguaje natural y el enriquecimiento de documentos clínicos, obviamos el resto de detalles de la extracción dada su naturaleza especifica a cada estudio clínico.

11.5.3. Otros

A parte de la funcionalidad principal, se han implementado otras características para mejorar tanto el rendimiento del sistema final como el proceso de desarrollo. En concreto, se ha establecido un sistema de multiprocesado opcional para acelerar la aplicación de los módulos a las historias clínicas además de un sistema de logs/registros que capturan el funcionamiento normal y los errores del sistema.

Multiprocesado

Para esta funcionalidad hemos trabajado con la librería estándar de python **multiprocessing** que facilita la creación de procesos. Dadas las características de la carga computacional, decidimos paralelizar al nivel de historias clínicas[10]; esto nos permite de manera sencilla crear tantos procesos como deseamos y aprovechar al máximo los núcleos de la cpu, pues no es necesaria comunicación entre procesos ya que cada historia es totalmente independiente.

Modificamos además la cache del módulo de identificación de terminología para usar un diccionario compartido y así tener una sola memoria para todos los procesos. Gracias a la librería no necesitamos mantener nosotros mismos los semáforos, ya que esta se ocupa de todo y solo es necesario pasar la referencia a donde sea necesaria.

Logger

Para tener un registro del funcionamiento del sistema y así tener una referencia en caso de fallo, creamos un logger. Dada la posibilidad de paralelizar el procesado de las historias clínicas necesitamos crear un servidor que reciba la información de cualquier proceso y centralice su tratamiento (que logs mostrar por pantalla, donde y como guardarlos en archivos...). Para esto usamos, como previamente, la librería estándar **multiproces-**

[10]Todo esto está implementado en la clase *Process*

sing para crear el servidor y la librería también estándar **logging** para el manejo de los diferentes registros[11].

Por último cabe mencionar que el paso de memoria al servidor se lleva a cabo a través de una estructura de cola compartida creada de la misma manera que el diccionario.

11.6. Resultados

Al final del desarrollo se obtuvo, además de la totalidad de historias clínicas aumentadas con la información PLN, una base de datos con un total de 688 campos de información recogidos sobre 289 pacientes. No toda esta información es útil, en general está demasiado esparcida y para llevar a cabo análisis posteriores es necesario agrupar dichos datos en variables más estándar. Asimismo, no todos los campos recogidos tienen suficiente información (por fallo del sistema o por falta de datos en las historias) como para pasar a posteriores etapas.

Por esto, con la ayuda de varios médicos, se creo una base de datos reducida con la información necesaria para llevar a cabo un estudio sobre pacientes infartados; en total contiene 78 variables curadas para 198 pacientes con la mayor información recogida.

Se añade finalmente una variable manual de diagnostico (como extra a la ya recogida automáticamente) para tener un estándar de oro sobre el que basar el análisis posterior de los datos.

Por último, dejamos la corrección de la base de datos a las pruebas realizadas en la siguiente sección.

[11]Implementado en la clase *Logger* e inicializado en *Process*

Capítulo 12

Pruebas

El conjunto de tests realizados para la comprobación final de la corrección del proyecto se pueden dividir en dos tipos, una primera batería de pruebas para la analizar el correcto funcionamiento de cada linea de código del proyecto a través de tests unitarios y de integración; y un segundo conjunto para establecer la eficacia del algoritmo final de recogida de datos.

12.1. Test unitarios

Dada la naturaleza orientada a objetos del código se han separado las diversas pruebas en conjuntos de tests unitarios para cada método de cada clase creada. En estos se comprueba el correcto funcionamiento de cada función, su salida y sus efectos secundarios sobre el estado de los objetos.

Para llevar a cabo todo esto se ha usado la librería proveída por defecto, **unittest** [27], por Python que facilita el continuo desarrollo y comprobación de tests unitarios.

En total se crearon 106 tests, unitarios y de integración, cubriendo las diferentes partes del sistema. Todos y cada uno de ellos fueron en última instancia superados por el sistema final.

12.2. Análisis de rendimiento

Para comprobar las capacidades del sistema se han calculado diferentes estadísticas que estiman de múltiples maneras el sistema final.

Coste temporal

Las primeras estadísticas obtenidas son las de velocidad por módulo por historia/s clínica/s (se calculan varias veces, se hace la media y se redondean a un decimal), primero

Módulo	1 HC	10 HC	50 HC
Segmentación	1.5 s	1.7 s	2 s
Tokenizado	1.9 s	2.1 s	3.6 s
POS	0.2 s	0.8 s	3.2 s
Negex	0.1 s	0.1 s	0.7 s
Chunk	1.6 s	2.7 s	7.4 s
Identif.	18 s	106 s	720 s (12 m)

Cuadro 12.1: Tiempos de los módulos en segundos

Módulo	1 HC	10 HC	50 HC
Segmentación	1.5 s	2.0 s	2.1 s
Tokenizado	2.3 s	2.5 s	3.4 s
POS	0.4 s	0.5 s	1.2 s
Negex	0.1 s	0.1 s	0.1 s
Chunk	1.7 s	2.4 s	5 s
Identif.	18 s	57 s	360 s (6 m)

Cuadro 12.2: Tiempos de los módulos en segundos-Multitarea

sin y luego con multiprocesado:

Observamos primero que el cuello de botella corresponde con la identificación/normalización de terminología; en este módulo no es la potencia de la CPU lo que lo limita, sino principalmente el número de peticiones a la base MySql. Aun así, el reparto de tareas es útil pues permite llevar a cabo el resto de cálculos necesarios mientras otros procesos esperan a sus respectivas peticiones, como se observa en la mejora temporal (alrededor de un 50 % con 50 historias clínicas).

El resto de módulos son relativamente veloces, sobre todo los cinco primeros; estos tienen un coste similar no importa el número de historias clínicas lo que apunta a que el cuello de botella es la preparación del módulo y no el procesado en sí. Vemos como tienen un resultado similar tanto en una como en múltiples tareas, incluso es ligeramente menor en la primera, algo esperable dado el coste adicional de creación de procesos aunque el multiprocesado mejora a partir de un número más elevado de historias.

[0]Todos los tests han sido llevados a cabo con 4 procesos en un procesador multinúcleo

Precisión

Para establecer la precisión y la capacidad del sistema creado, obtenemos un conjunto de historias clínicas, 5 en total, extraídas manualmente por un profesional para comparar con las variables generadas.

Los resultados no son concluyentes –por limitación de costes el número de prueba es bajo– pero es suficiente para estimar el comportamiento general del sistema y encontrar las mayores dificultades de este.

Las variables con la mayor precisión son, nada sorprendentemente, las de datos personales, farmacología y laboratorio con una tasa prácticamente del 100 % de acierto[1]. Estas son las secciones más estructuradas de las historias clínicas, que recogen la información general del paciente (edad, sexo, tipo de ingreso...), los biomarcadores y otras medidas comunes (glucosa en sangre, plaquetas, troponina...), etc.

Luego les siguen los antecedentes y factores de riesgo, con un acierto rondando el 90 % y un tasa de recogida del 95 %. Si bien estos valores parecen altos, es necesario considerar que las variables sobre las que se estima han sido especialmente seleccionadas por ser las más sencillas de recoger; esto incluye un sesgo en los datos pues aquellas variables sobre las que se trabaja son inevitablemente aquellas que el sistema es capaz de recoger[2].

Por último, tras la recogida se llevo a cabo un análisis de la capacidad predictiva de los datos sobre la variable manual de diagnóstico de infarto y se observo, con un SVM, una sensibilidad y especificidad de 0.75 y 0.67 respectivamente, además de una precisión del 0.7, que dada la prevalencia del 50 % de la enfermedad en los datos, demuestra el potencial del sistema.

[1] Recogemos dos tipos de errores, falta de recogida del dato y recogida del dato de manera errónea.

[2] En general la herramienta es capaz de recoger información simple, como enfermedades concretas negadas o sin negar, pero se traba con información más abstracta. Es además incapaz de obtener información temporal si no es en fechas específicas.

Capítulo 13

Conclusión

A lo largo de este proyecto hemos alcanzado los objetivos que nos propusimos al comienzo; primero, investigar sobre el procesado de lenguaje natural y estudiar como se ha de resolver con esta tecnología un problema real. Esto se ha llevado a cabo en la sección de teoría en la cual se ha explicado en detalle todas las etapas necesarias para analizar texto libre.

Segundo, desarrollar dicha solución y encontrar las dificultades con las que se encontraría cualquier proyecto de este ámbito. Así se ha llevado de esta manera un proyecto PLN sobre historias clínicas y se han encontrado los problemas generales del PLN, como la falta de datos etiquetados, las problemáticas referentes a sistemas basados en reglas, etc; además de los problemas específicos de este ámbito, como la naturaleza sensible de los datos clínicos, la diferente estructura gramática y de vocabulario del texto médico que dificulta el análisis, etc.

Y por último, hemos desarrollado una aplicación siguiendo lo aprendido en la carrera, diseñando la arquitectura, planificando el desarrollo, analizando posibles modelos de datos. Si bien esta última parte se ha visto reducida dada la naturaleza exploratoria del proyecto, esta se ha tratado de la misma manera que el resto.

- Se ha analizado las diferentes etapas de un proyecto PLN real
  - Se han analizado las distintas formas de segmentación y tokenización de un texto
  - Se ha estudiado como analizar la estructura sintáctica del texto
  - Se han analizado las diferentes formas de análisis semántico y normalización del texto
- Se ha estudiado la naturaleza de las historias clínicas, su contenido y formato
- Se ha desarrollado una herramienta PLN para procesar historias clínicas

- Se ha construido una herramienta de anonimizado
- Se ha creado un algoritmo para extraer las secciones de una HC
- Se ha creado un sistema de segmentación y tokenización del texto
- Se ha creado una base de datos de acrónimos médicos
- Se ha desarrollado un sistema de expansión de acrónimos
- Se ha creado y entrenado un modelo de etiquetado POS
- Se ha creado un sistema de clasificación de sentencias negadas
- Se ha creado una base de datos de terminología médica
- Se ha construido un algoritmo de normalización jerárquico de terminología
- Se ha implementado un sistema de extracción de variables médicas

La herramienta

Respecto a los objetivos de la herramienta, se ha observado un potencial remarcable en la idea. Si bien existen soluciones comerciales similares, tanto en Ingles como en Español, en nuestro idioma aún están en su infancia, lo que nos anima a continuar el desarrollo de estas ideas. Aquellos datos recogidos por la herramienta lo son con una alta precisión si bien es incapaz de tratar varios tipos de datos más abstractos como los temporales.

Los resultados son lo suficientemente buenos como para ser la base para los proyectos de fin de grado de dos estudiantes de medicina que participaron en este proyecto (en la etapa de recogida de datos).

Finalmente, al presentar esta idea en una competición, extendida a una plataforma –**Galena Databook**–, adquirió una gran atención e incluso gano un premio por el **mejor proyecto de salud**[1], lo que nos indica que existe interés en el mundo médico español por una herramienta/plataforma del estilo. Más allá, al presentar los compañeros médicos como TFG estudios basados en los resultados del sistema, obtuvieron los tres matricula de honor.

Futuro trabajo

Este proyecto es un prototipo, para completarlo seria necesario cambiar y mejorar un número de características. En primer lugar desarrollarlo con tecnologías más escalables, e.g. **SparkNLP**; y después sería necesario añadir más etapas al sistema, como una de corrección de gramática, otra de análisis semántico, etc. Además, para facilitar la generalización y, de nuevo, la escalabilidad del aprendizaje, seria obligado sustituir los modelos creados a través de reglas a otros con modelos de aprendizaje automático.

[1] Premio CSdM Tech4futures Hackathon Consorci Sanitari del Maresme.

Por último seria útil mejorar las bases de datos usadas, tanto las etiquetadas para entrenar los módulos PLN como la final para estimar la capacidad del sistema creado.

Resumen final

El proyecto que se ha realizado es tan solo un prototipo pero indica la utilidad real del campo del procesado del lenguaje natural. Por un lado está lleno de dificultades pero por el otro lado hay innumerables posibilidades para proyectos de este estilo; en concreto, la extracción de datos clínicos es un campo que encaja a la perfección con los problemas que el PLN intenta resolver y que en un futuro cercano cambiara la forma de investigar en medicina.

Parte IV

Apéndices

Apéndice A

Manual de Instalación

El código está completamente desarrollado en **Python 3.7.5** y está disponible en Gitlab en el repositorio privado de la UVA. Para poder usarlo es necesario instalar dicha versión de python u otra compatible además de una lista de librerías de terceros. Estas librerías, enumeradas a continuación[1], están todas disponibles a través de la herramienta **pip**; para evitar conflictos se recomienda el uso de alguna herramienta de entorno virtual como **venv**.

1. beautifulsoup4 (4.8.2)
2. bs4 (0.0.1)
3. conllu (2.2.2)
4. dnspython (1.16.0)
5. lxml (4.5.0)
6. mysql-connector-python (8.0.19)
7. nltk (3.4.5)
8. numpy (1.18.1)
9. protobuf (3.6.1)
10. python-dateutil (2.8.1)
11. pytz (2019.3)
12. regex (2020.2.20)
13. six (1.14.0)
14. soupsieve (2.0)

[1]La lista está además disponible en el repositorio en forma de un fichero "requierements.txt".

15. Unidecode (1.1.1)
16. xlrd (1.2.0)

Se ha utilizado también la herramienta **make** para facilitar la ejecución de partes del código aunque su uso es totalmente opcional.

Para el correcto funcionamiento de varios módulos es necesario la instalación de varias bases de datos, todas ellas descritas previamente; son los archivos que guardan la lista de secciones oficiales, el conjunto de acrónimos y sus definiciones, el conjunto de términos de negación y la base de datos POS. Es además requerido crear y activar una base de datos **MySql** con los datos de UMLS (como se comentó previamente, la página web de UMLS contiene scripts y guías para llevar esto a cabo), además de establecer la configuración de usuario y contraseña en el archivo "data/mysql.py".

Si se cambia el nombre o la localización de algún archivo o del proyecto en general, es necesario especificarlo en las variables globales mantenidas en el archivo "common/constants.py".

Apéndice B

Manual de Usuario

A continuación describiremos como se ha de usar la herramienta creada en este proyecto.

B.1. Manual de Usuario

Todo el proyecto está creado a través y es accesible como paquetes de Python, por lo tanto es una herramienta diseñada para usuarios avanzados o programadores. Es posible separar partes del código que se deseen utilizar –sobre todo los módulos ya que son independientes y pueden trabajar directamente sobre texto plano– y enlazarlos con otras aplicaciones cualquiera.

En total hay cinco funciones diferentes accesibles al usuario, a través del código de la clase o, más sencillamente, usando directamente el paquete. Primero, el módulo de detección de estructura y secciones, se puede acceder a través de la clase *Sections* o ejecutando el paquete directamente con **python3 -m src.processing.secciones**. Acepta htmls procesados por la herramienta pdftohtml comentada previamente.

Luego esta el sistema principal, que toma ficheros xml y los procesa de manera sucesiva con varios módulos PLN a elección del usuario. Es posible acceder a este de tres maneras, con código instanciando la clase *Pipeline* y añadiendo los archivos y módulos deseados; también es posible ejecutar varias pipelines preparadas a través de la clase *Processing* o a través de el paquete principal con **python3 -m src <clave>** y una palabra clave: "pos", "chunk" o "ident". Si se usa este último método es necesario configurar los archivos a tratar en las variables globales contenidas en "common/constants.py".

En tercer lugar es posible entrenar los modelos –de POS y segmentación– a través, de nuevo, del la clase o directamente ejecutando el paquete (y configurando las constantes necesarias). Las clases son *SentenceTokenizer* y *PosModel* y los paquetes (junto con el código necesario para ejecutarlos) seria **python3 -m src.models.sentenceTokenizer** y **python3 -m src.models.pos** respectivamente.

En cuarto y quinto lugar están el módulo de extracción y los tests del sistema, que se ejecutan de la misma manera con el código o el paquete contenidos en *src.processing.extraction* y *src.tests.xxx* respectivamente (sustituyendo xxx con el test concreto deseado).

Por último, para facilitar todo este trabajo se ha creado una makefile con atajos a los comandos más típicos, e.g. para ejecutar los tests tan solo es necesario ejecutar desde la linea de comandos "make test". La lista completa de posibilidades está en la makefile del proyecto.

Bibliografía

[1] Fernando [López Bello] y col. «From medical records to research papers: A literature analysis pipeline for supporting medical genomic diagnosis processes». En: *Informatics in Medicine Unlocked* 15 (2019), pág. 100181. ISSN: 2352-9148. DOI: https://doi.org/10.1016/j.imu.2019.100181. URL: http://www.sciencedirect.com/science/article/pii/S2352914819300309.

[2] Olivier Bodenreider. «The Unified Medical Language System (UMLS): integrating biomedical terminology». En: *Nucleic acids research* 32.Database issue (ene. de 2004). 14681409[pmid], págs. D267-D270. ISSN: 1362-4962. DOI: 10.1093/nar/gkh061. URL: https://www.ncbi.nlm.nih.gov/pubmed/14681409.

[3] *Bratt*. URL: http://brat.nlplab.org/standoff.html.

[4] *CLAMP*. URL: https://clamp.uth.edu/.

[5] *Conference on Computational Natural Language Learning*. URL: https://www.conll.org/.

[6] *CoNLL-U Parser*. URL: https://pypi.org/project/conllu/.

[7] *CTAKES, a natural language processing system for extraction of information from electronic medical record clinical free-text*. URL: https://ctakes.apache.org/index.html.

[8] *Digitalización en Salud, La Historia Clínica Digital como motor de transformación del sistema sanitario*. URL: https://cotec.es/media/InformeCotecDigitalizacionenSalud_vf.pdf.

[9] *Eagles-Parole*. URL: http://www.lsi.upc.es/%5C~nlp/tools/parole-sp.html.

[10] Ander Intxaurrondo. *AbreMES-DB*. Funded by the Plan de Impulso de las Tecnologías del Lenguaje (Plan TL). Nov. de 2018. DOI: 10.5281/zenodo.2207130. URL: https://doi.org/10.5281/zenodo.2207130.

[11] Tibor Kiss y Jan Strunk. «Unsupervised Multilingual Sentence Boundary Detection». En: *Computational Linguistics* 32.4 (2006), págs. 485-525. DOI: 10.1162/coli.2006.32.4.485. URL: https://doi.org/10.1162/coli.2006.32.4.485.

[12] Montserrat Marimon. *SPACCC$_P$OS*. Funded by the Plan de Impulso de las Tecnologías del Lenguaje (Plan TL). Nov. de 2018. DOI: 10.5281/zenodo.2560344. URL: https://doi.org/10.5281/zenodo.2560344.

[13] Marie-Catherine de Marneffe y col. «Universal Stanford dependencies: A cross-linguistic typology». En: *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*. Reykjavik, Iceland: European Languages Resources Association (ELRA), mayo de 2014, págs. 4585-4592. URL: http://www.lrec-conf.org/proceedings/lrec2014/pdf/1062_Paper.pdf.

[14] *METAMAP*. URL: https://metamap.nlm.nih.gov/.

[15] *Natural Language Toolkit*. URL: https://www.nltk.org/.

[16] Naiara Perez-Miguel, Montse Cuadros y German Rigau. «Biomedical term normalization of EHRs with UMLS». En: *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC 2018)*. Miyazaki, Japan: European Language Resources Association (ELRA), mayo de 2018. URL: https://www.aclweb.org/anthology/L18-1322.

[17] Slav Petrov, Dipanjan Das y Ryan T. McDonald. «A Universal Part-of-Speech Tagset». En: *CoRR* abs/1104.2086 (2011). arXiv: 1104.2086. URL: http://arxiv.org/abs/1104.2086.

[18] *Plan de Impulso de las Tecnologías del Lenguaje (Plan TL)*. URL: https://www.plantl.gob.es.

[19] *Python's html scraping library*. URL: https://pypi.org/project/beautifulsoup4/.

[20] David Reinsel – John Gantz – John Rydning. «The Digitization of the WorldFrom Edge to Core». En: *SeaGate* (2018).

[21] Sistema Nacional de Salud. «Historia Clínica Digital en el Sistema Nacional de Salud - Conjunto Mínimo de Datos de Informes Clínicos». En: ().

[22] Jesús Sanamaría. «NegEx-MES». En: (ene. de 2019). Funded by the Plan de Impulso de las Tecnologías del Lenguaje (Plan TL). DOI: 10.5281/zenodo.2542567.

[23] *Savana, Transform Electronic Health Records into Big Data to Accelerate Health Science*. URL: https://www.savanamed.com.

[24] *Scielo*. URL: http://scielo.isciii.es/scielo.php.

[25] Sedom. *Diccionario de Siglas Médicas*. URL: http://www.sedom.es/diccionario/.

[26] Sonit Singh. «Natural Language Processing for Information Extraction». En: (jul. de 2018).

[27] *unittest — Unit testing framework*. URL: https://docs.python.org/3.7/library/unittest.html.

[28] *Universal Dependencies*. URL: https://universaldependencies.org.

[29] *XML and HTML with Python*. URL: https://lxml.de.