Herramienta para búsqueda de casos médicos semejantes

Facultad de Informática

Universidad Complutense de Madrid

Departamento de Ingeniería del Software e Inteligencia Artificial

Curso 2015/2016

Agustín Pastore Burgos

Director:

Alberto Díaz Esteban

Agustín Pastore Burgos, autor del presente documento y del proyecto "Herramienta para búsqueda de casos médicos semejantes" autoriza a la Universidad Complutense a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente al autor, tanto la propia memoria, como el código, los contenidos audiovisuales incluso si incluyen imágenes del autor, la documentación y/o el prototipo desarrollado.

16 de Junio de 2013

Agustín Pastore Burgos

He de agradecer a un gran número de personas, sin las que este trabajo no hubiera sido posible.

En primer lugar, a Alberto Díaz, director del proyectoto, por su colaboración y entusiasmo.

También a los profesores que me han guiado todos estos años, formándome, y dándome las herramientas que me han permitido llegar a donde estoy.

Finalmente, a todos los amigos y familiares que me han apoyado, proporcionándome la ayuda que he necesitado y soportandome en las peores épocas.

A todos, gracias.

Resumen

Una de las tareas más comunes a las que se enfrentan los médicos es buscar historiales médicos, una tarea lenta y laboriosa que les arrebata tiempo útil. Este proyecto intenta reducir el tiempo dedicado a esa búsqueda permitiendo que, a partir del historial médico de un paciente, se encuentren otros casos similares dentro de la base de datos.

Por eso, la base de datos con los documentos clínicos, en lenguaje natural en castellano, ha de ser procesada con las herramientas producidas por este proyecto. La aplicación esta dividida en tres partes: la primera y la segunda se encargan de procesar los informes, dividiendo en campos y hallando los conceptos médicos respectivamente; la tercera parte es la que realiza las búsquedas de informes médicos similares.

Palabras clave: Procesamiento del lenguaje natural, recuperación de información, informe médico, detección de conceptos, buscador, ontología, UMLS, MetaMap, traductor.

Abstract

One of the most common tasks that the doctors face is to look up in medical histories, a slow and tedious task that take usefull time away from them. This project aims to reduce the time dedicated to those searches, allowing the search of similar medical cases inside the database.

For this reason, the database with the clinical documents, in natural language in spanish, have to be processed with the tools developed by this project. The aplication is divided in three parts: the first and second one take care of processing the reports, dividing by fields and finding medical concepts respectively; the third part is the one that make the searches of similar medical reports.

Keywords: Natural language processing, information retrieval, medical report, concepts detection, searcher, UMLS, MetaMap, translator.

Índice general

Capítulo 1

Introducción

1.1 Motivación

La medicina es una de las ciencias más antiguas. A lo largo de toda su historia, ha asimilado los avances de otras ciencias para crear nuevas herramientas, métodos y realizar descubrimientos que le han permitido avanzar y desarrollarse a un ritmo asombroso.

La informática ya le ha proporcionado herramientas a medida que evoluciona. Desde software para codificar genes, a algoritmos para calcular las dosis de medicamentos. Pero la informática evoluciona a un ritmo sorprendente, y día a día se crean oportunidades para aplicar nuevas ideas a la medicina.

Entre las nuevas aplicaciones a la medicina se incluye el tema principal de este proyecto, la búsqueda de informes médicos similares en una base de datos.

En esta ocasión, este proyecto planea centrarse en simplificar la útil pero laboriosa tarea de comparar informes médicos, ya sea para realizar estadísticas, o, en caso de una base de datos mayor, ayudar con el diagnóstico. Ya existen incluso hospitales interesados en el resultado de este trabajo, tal es la naturaleza del problema.

Este trabajo surgió como una colaboración entre el hospital Asturiano "Comarcal de Jarrio" con la Facultad de Informática de la Universidad Complutense de Madrid. El hospital mostró interes en una herramienta que les permitiera, a partir de un informe, buscar otros similares en su base de datos.

El hospital ha colaborado mucho, su aportación más importante, los informes médicos que han guiado el desarrollo de este proyecto. Estos informes cumplen las leyes de protección de datos y recibieron aprobación por un Comité de Ética para poder ser explotados de esta forma.

Los ejemplos que nos ha proporcionado el hospital siguen una estructura compleja que se compone de varios niveles:

1. El primer nivel está formado por un grupo de carpetas, cada una conteniendo pacientes de una enfermedad concreta. Es decir, cada carpeta

contiene a los pacientes que han sufrido la misma enfermedad. Por ejemplo: Cataratas.

2. El segundo nivel esta formado por los casos médicos. Cada uno de ellos representa el historial médico de cada paciente.

3. El tercero y último está formado por todos los documentos médicos pertenecientes a cada paciente. Este nivel es el que tiene la mayor dificultad para digitalizar, pues hay que unificar todos los documentos en un solo archivo estructurado antes de poder continuar con las demás funcionalidades. Existen varios tipos de documentos, con distinta estructura y formato (varios formatos de texto, incluyendo Microsoft Word, y Excel) y que contienen distinta información médica (Cirugía, Informe de Alta, Registro de Enfermería, etc). Además, documentos de un tipo se pueden encontrar un número variable de veces o no haber ninguno de ese tipo. Todas estás dificultades han de superarse antes de poder realizar cualquier tipo de actividad con la información de los casos médicos.

Por otro lado, este proyecto aplica el desarrollo conseguido con el Trabajo de Sistemas Informáticos de 2012-2013, "Extracción de información de informes médicos" realizado por Irene Sánchez Martínez, Víctor Martínez Simón y Lucía Hervás Martín bajo la tutela de Alberto Díaz Esteban. En ese proyecto se centraron en extraer correctamente la información de un solo informe, mientras que en este el propósito es más complejo, pues se pretende comparar los resultados tras analizar un gran número de informes médicos, buscando casos parecidos en enormes bases de datos médicas.

También ha sido importante para realizar este proyecto, un artículo de la Universidad Europea de Madrid de 2008[1], que pretendía desarrollar una herramienta para conseguir que un servicio de procesado de informes médicos en inglés fuera compatible con informes médicos en castellano. "In the Development of a Spanish Metamap" fue realizado por Francisco Carrero, José Carlos Cortizo, José María Gómez y Manuel de Buenaga. Y aunque no lograron su propósito de desarrollar una herramienta que les permitiera usar el servicio de procesado de informes médicos en castellano, si fueron capaces de descubrir que utilizando un traductor prácticamente no perdían precisión al procesar los informes médicos.

1.2 Objetivos

El objetivo principal es crear una herramienta útil y fácil de usar, que permita ahorrarles cientos de horas de trabajo superfluo a los médicos y demás expertos relacionados, evitando que tenga que revisar informes médicos en busca de coincidencias.

Uno de los problemas más importantes es conseguir extraer la información con éxito de los informes médicos, a veces separados en varios archivos. Para

[1] http://www.esp.uem.es/jmgomez/papers/cikm08.pdf

esta tarea el objetivo es lograr conseguir unificar toda la información en un solo archivo.

A la hora de realizar el programa cobra importancia que sea capaz de realizar la misma labor para los posibles distintos formatos que use un informe médico. También es importante en este caso, que la configuración para alterar dicho formato sea simple y no necesite de la ayuda de un especialista en medicina y/o informática.

Un tema también importante es el idoma en el que se encuentre la información del informe. Inglés es el idioma más hablado, y en el que funcionan las herramientas de analisis médicas, pero el castellano es necesario para las posibles aplicaciones locales de la aplicación, es decir, para el hospital que se ha interesado por esta herramienta. Para derribar las barreras del idioma se usa un traductor, con un impacto muy reducido en la precisión de la aplicación.

No hay que olvidar que los datos pueden estar almacenados en distintos formatos, y, para simplificar el uso del programa, sería recomendable que fueran compatibles el mayor número de ellos, dándole prioridad a los populares .doc y .docx, además de al más especializado .xml.

Por tanto el objetivo central de este proyecto es conseguir organizar los datos para crear una base de datos que permita buscar informes parecidos.

Para conseguir este objetivo, primero se intentará crear un prototipo que haga uso de alguna herramienta de indexación simple para probar la viabilidad de la aplicación, a la vez de proporcionar un programa de prueba para posibles interesados. Tras esto, el objetivo pasará a ser un programa más avanzado que anteponga fiabilidad a velocidad.

Dado que el usuario medio de la aplicación carece de conocimientos informáticos avanzados, es necesario incluir en los objetivos una interfaz simple y clara.

Se dividira la funcionalidad de la aplicación en tres fases, las dos primeras para procesar el texto y la tercera para realizar las búsquedas:

1. Pre-procesar la información en los casos médicos para obtener una referencia estructurada de su contenido.

2. Procesar los campos de los documentos médicos pre-procesados, traduciendo la información al inglés y usando un servicio externo para obtener los conceptos médicos en el texto.

3. Búsqueda entre los distintos casos médicos, formados por conceptos médicos, para hallar los más similares.

1.3 Estructura del documento

Esta memoria está dividida en diversos apartados, explicados en detalle a continuación:

1. **Introducción**: Resume los motivos que han influenciado la creación de este proyecto, además de los objetivos que se planean resolver con su realización.

2. **Estado de la Cuestión**: Describe el estado actual de desarrollo en las áreas a las que afecta el proyecto, incluyendo las tecnologías usadas en el mismo.

3. **Procesado de Informes**: En este apartado se explica en profundidad el procesado de los informes médicos y las distintas etapas de las que esta compuesto.

4. **Indexación y búsqueda en informes procesados**: Se explica detalladamente como funciona la búsqueda de informes médicos similares.

5. **Conclusiones y trabajo futuro**: Aquí incluyo mis conclusiones tras terminar de desarrollar el proyecto, incluyendo posibles desarrollos futuros que podría tener la aplicación.

Capítulo 2

Estado de la Cuestión

2.1 Introducción

En este capítulo muestro el estado actual de las tecnologías relacionadas con este proyecto, tanto las que barajé en las primeras etapas de la aplicación, las que están presentes en el resultado final y aquellas aplicaciones con funcionalidades parecidas.

Se pueden dividir en las siguientes categorías:

- Procesado

Ontologías: Siendo que el primer requisito de la aplicación es ser capaz de procesar informes médicos para poder hacer comparaciones entre ellos y buscar similares, las ontologías juegan un papel clave en la aplicación. La extracción y representación de la información médica en un formato comprensible es indispensable, por lo que investigue las mejores alternativas y las funcionalidades de cada una. SNOMED-Clinical Terms (SNOMED-CT) y Unified Medical Language System (UMLS) son algunas de ellas.

Metamap: es una una herramienta que permite identificar los conceptos de UMLS presentes en un texto, una funcionalidad crítica. Además, sumado a otras funciones útiles, como la posibilidad de expandir acrónimos o la desambiguación de conceptos médicos, le confieren gran importancia en este proyecto. Tratándose de informes médicos, la extracción de información ha de ser lo más precisa posible. Para ello estudié la posibilidad de incluir funcionalidades para detectar fallos ortográficos, negaciones y especulaciones.

Traductor: Para poder utilizar la herramienta MetaMap es necesario que el texto de entrada esté en inglés o crear una base de datos médica que funcione en la versión local de MetaMap y que esté en castellano. Siendo que MetaMap funciona reconociendo conceptos médicos, el número de errores por usar un texto traducido es mínimo.

ApachePOI: Para poder simplificar el uso de la aplicación, incluí la funcionalidad de poder leer ficheros .doc, el formato en el que estaban guardados los informes médicos. Para ello utilice la API de Apache POI.

- Búsqueda

Lucene: La indexación de informes médicos y la búsqueda por similaridad es el propósito final de este proyecto y la razón por la que es tan importante la precisión de la información extraída. En este apartado se incluyen las tecnologías disponibles y el uso de cada una durante la evolución de la aplicación.

2.2 Ontologías y terminologías biomédicas

En el campo de la informática, la inteligencia artificial y los sistemas basados en conocimiento podemos definir una ontología como una representación formal de un conocimiento compartido que aporta un mayor nivel de información sobre un dominio determinado que un simple diccionario o conjunto de términos o definiciones. Para una definición más concreta proponemos a continuación algunas de las cuáles han sido dadas a lo largo del tiempo:

- Una ontología define los términos básicos y las relaciones que componen el vocabulario de un área temática, así como las reglas para combinar dichos términos y relaciones con el fin de definir extensiones del vocabulario.
- Es una especificación explícita de una conceptualización.
- Un conjunto de axiomas lógicos diseñados para explicar el significado pretendido de un vocabulario.
- Una terminología es, dentro de las diferentes disciplinas científico-técnicas, el conjunto de las unidades de expresión y comunicación que permiten transferir y comunicar el pensamiento especializado. Una terminología persigue el objetivo de fijar unas unidades terminológicas como formas normalizadas y de referencia que descartan las demás variantes para denominar un mismo concepto con el fin de alcanzar una comunicación profesional precisa, moderna y unívoca.

2.2.1 SNOMED-CT

SNOMED-CT o Systematized Nomenclature of Medicine Clinical Terms[1] es una extensa terminología clínica de atención médica, disponible en varios idiomas y usada en la actualidad por más de cincuenta países. Nace de la fusión entre SNOMED RT desarrollada por el College of American Pathologists (CAP) y el Clinical Terms Version 3 (CTV3) desarrollada por el National Heath Service (NHS) de Reino Unido.

En 2007 los derechos de propiedad intelectual fueron transferidos a la International Health Terminology Standards Development Organisation (IHTSDO) quien se encarga de su mantenimiento y distribución hoy en día.

[1] http://www.ihtsdo.org/snomed-ct/

2.2.1.1 Componentes

Los principales componentes de SNOMED-CT son los conceptos, las descripciones y las relaciones.

- Conceptos

Los conceptos representan como su propio nombre indica conceptos o ideas clínicas. Cada uno tiene un identificador numérico único (ConceptID). Están organizados en jerarquías que van de lo general a lo específico a medida que se desciende en ellas.

- Descripciones

Un concepto puede tener más de una descripción asociada, cada una representando un sinónimo que describe la misma idea.

- Relaciones

Las relaciones permiten asociar a un concepto otros conceptos cuyo significado está relacionado. La relación más importante es la llamada "es un" que define la mencionada jerarquía entre conceptos. Otros ejemplos son "agente causal", "sitio de hallazgo" o "morfología asociada". Existen cuatro tipos de relaciones: definitorias, calificadoras, históricas y adicionales.

2.2.2 UMLS

UMLS o Unified Medical Language System [2] es un conjunto de archivos y software que reúne gran cantidad de vocabularios biomédicos y de salud, y los estándares que dan la posibilidad de que distintos sistemas informáticos operen entre sí. Fue desarrollado por la National Library of Medicine (NLM) tratando de superar dos importantes barreras relativas a la recuperación efectiva de información por parte de máquinas:

1. La primera de ellas es la variedad de formas en que unos mismos conceptos se expresan en diferentes fuentes legibles por máquinas y personas.

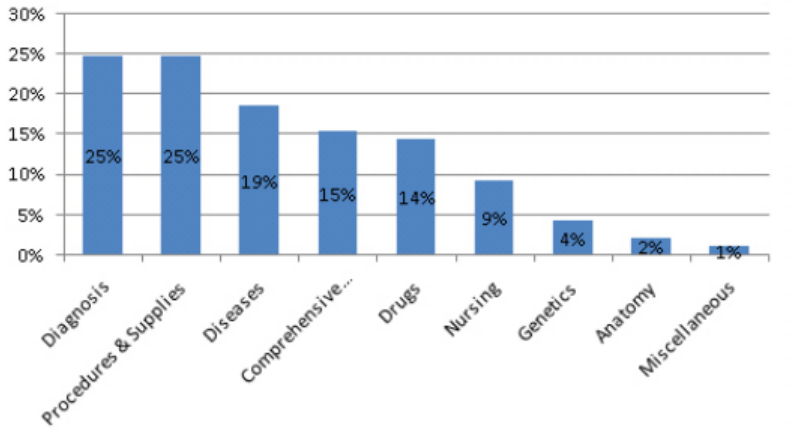
2. La segunda es la falta de un formato estándar para distribuir y difundir terminologías.

2.2.2.1 Estructura

Existen tres principales fuentes de conocimiento de UMLS: el Metatesauro, la Red Semántica y el Lexicón Especializado y Herramientas Léxicas.

- Metatesauro

[2] http://www.nlm.nih.gov/research/umls/



Fuente www.nlm.nih.gov

Figure 2.1: Porcentaje que representa cada una de las categorías del Metatesauro UMLS.

El Metatesauro [3] es una base de datos de vocabulario multi-propósito y multi-lenguaje que contiene información sobre conceptos relacionados con la biomedicina y la salud, sus distintos nombres y las relaciones entre ellos. Engloba más de un millón de conceptos biomédicos y más de 100 fuentes de vocabulario, entre las que se encuentran MeSH[4], RxNorm[5] y SNOMED-CT[6] y sirve de apoyo a la hora de crear asignaciones entre ellos, pero no pretende reemplazarlos. Existen varias categorías para clasificar los vocabularios. Algunos pueden pertenecer a más de una categoría.

Cuando un concepto es añadido al Metatesauro recibe un identificador único y es situado en su estructura. Esta estructura tiene cuatro niveles de especificación:

1. Concept Unique Identifiers (CUI) o identificadores únicos de concepto. Estarán formados por la letra C seguida de siete dígitos. Un concepto es un significado. Un significado a su vez puede tener diferentes nombres. Los objetivos clave son comprender el significado del nombre en cada fuente de vocabulario y vincular todos los sinónimos. Un CUI por tanto se asocia con los denominados conceptos, cada conjunto de conceptos sinónimo estará agrupado bajo el mismo CUI.

2. Lexical Unique Identifiers (LUI) o identificadores únicos de léxico. Estarán formados por la letra L seguida de siete dígitos. Enlazan cadenas que son variantes léxicas. Dichas variantes léxicas son detectadas utilizando

[3] http://www.nlm.nih.gov/research/umls/knowledge_sources/metathesaurus/index.html
[4] http://www.nlm.nih.gov/mesh/meshhome.html
[5] http://www.nlm.nih.gov/research/umls/rxnorm/index.html
[6] http://www.nlm.nih.gov/research/umls/Snomed/snomed_main.html

el Lexical Variant Generator (LVG) una de las herramientas de UMLS. Un LUI se asocia con los llamados términos, un conjunto de nombres normalizados tendrá asignado el mismo LUI.

3. String Unique Identifiers (SUI) o identificadores únicos de cadena. Estarán formados por la letra S seguida de siete dígitos. Cada nombre de concepto único en cada lenguaje del Metatesauro tendrá asociado un SUI. Cualquier variación en el conjunto de dichos caracteres supone una cadena diferente con un SUI diferente.

4. Atom Unique Identifiers (AUI) o identificadores únicos de átomos. Estarán formados por la letra A seguida de siete dígitos. Son el componente básico del la estructura y cada vez que se presenta una nueva cadena en un vocabulario, a esta se le asigna un AUI. Si la misma cadena aparece varias veces en el mismo vocabulario como un nombre alternativo para diferentes concepto, un AUI único será asignado para cada caso. Un AUI se asocia con cada nombre de concepto de cada fuente determinada.

A continuación proponemos un ejemplo, acompañado por la figura 2.2, para facilitar la compresión de lo anteriormente explicado.

En el ejemplo propuesto contamos con cinco cadenas de caracteres aportadas por cuatro fuentes de vocabulario (BI, SNOMED, MeSH y DX). Tal como es de esperar, se puede comprobar cómo a cada nombre de concepto de cada fuente de vocabulario determinada, se le asocia un AUI.

Anteriormente hemos mencionado que cada nombre de concepto (teniendo en cuenta cualquier variación en los caracteres) tendrá asignado un SUI. En este caso, contamos con cuatro nombres diferentes: headaches, Headache, Cranial Pain y HEAD PAIN CEPHALGIA, obteniendo así cuatro identificadores únicos de cadena.

Así pues, pasamos al nivel inmediatamente superior en la estructura, el LUI. Dado que bajo un mismo LUI se agrupa un conjunto de nombres normalizados, tendríamos tres LUI. El primero será para la normalización de los nombres head y Headache, el segundo para la normalización de Cranial Pain y el último para la normalización de HEAD PAIN CEPHALGIA.

Por último hemos dicho que bajo un CUI se agrupan conceptos sinónimos. Así pues para los dos LUI mencionados anteriormente seria asignado un único CUI, Headache.

Por otro lado es importante destacar que, debido a su volumen, para poder utilizar de manera efectiva el Metatesauro en una aplicación local, debemos personalizarlo limitando los vocabularios, lenguajes relaciones o atributos a usar, ya que no será habitual que requiramos los servicios de todos ellos. Para este fin debemos utilizar MetamorphoSys.

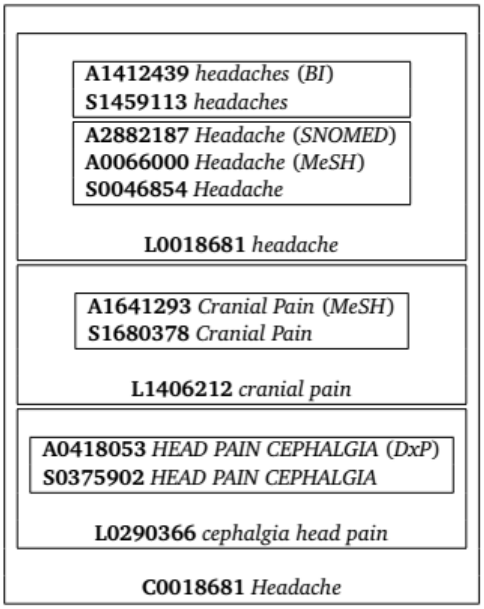


Figure 2.2: Identificadores a lo largo de la estructura del Metatesauro para el concepto Headache Fuente: www.nlm.nih.gov

2.2.2.2 MetamorphoSys

MetamorphoSys[7] es una herramienta multi-plataforma que la NLM actualiza e incluye en cada publicación de UMLS. Su finalidad es dar al usuario la posibilidad de creación de un subconjunto personalizado del Metatesauro, siendo además necesaria para instalación de manera local de las fuentes de conocimiento de UMLS.

Podemos estar interesados en trabajar con subconjuntos de Metatesauro debido a dos motivos principales:

1. El primero se deriva del tamaño del propio Metatesauro: no será habitual que requiramos el uso de toda la información almacenada originalmente en él. Por otro lado, ciertas fuentes de vocabulario requieren licencias especiales para usos específicos y puede que no estemos interesados en adquirir dichas licencias.

2. El segundo es que el usuario puede querer modificar el formato de los datos de salida o aplicarles diversos filtros.

La salida proporcionada por MetamorphoSys consistirá en un conjunto de archivos Rich Release Format (RRF) u Original Release Format (ORF). Además al crear

[7] http://www.nlm.nih.gov/research/umls/implementation_resources/metamorphosys/index.html

un conjunto del Metatesauro o instalar la Red Semántica se podrán generar scripts para ser cargados por MySQL, Oracle o Microsoft Access.

2.3 Metamap

MetaMap[8] es un software desarrollado por el Dr. Alan Aronson en la NLM de Estados Unidos que cuenta con múltiples opciones de configuración y permite mapear o asociar términos que aparecen en un texto biomédico. Los términos se asocian con conceptos del Metatesauro UMLS.

Se utiliza enfoque basado en el PLN y en técnicas lingüísticas. Además de ser aplicado tanto para recuperación de información (IR) y aplicaciones de minería de datos es uno de los fundamentos del Medical Text Indexer (MTI) del NLM el cual se utiliza para indexar de manera semiautomática o automática literatura del NLM.

2.3.1 Funcionalidades

MetaMap ofrece diversas funcionalidades, de las cuales, las más importantes para la aplicación son:

- Desambiguación

Uno de los problemas principales del PLN es la ambigüedad del lenguaje, y una de las mayores debilidades de MetaMap es su incapacidad para resolver la ambigüedad del Metatesauro en las situaciones en la que dos o más conceptos comparten un sinónimo. Para solucionar este problema se incluyó un sistema de desambiguación (Word Sense disambiguation (WSD)) activable mediante la opción -y. De esta forma se favorecen las alternativas que tienen un tipo semántico más probable en función del contexto.

- Detección de negaciones

MetaMap es capaz de detectar cuando un concepto está negado mediante el uso de una versión extendida del algoritmo NegEx[9]. Para que sea legible (human-readable), es necesario usar la opción -negex.

- Detección de acrónimos y abreviaturas definidos por el autor

En los documentos técnicos aparecen con frecuencia acrónimos y abreviaturas (Acronyms and Abbreviations (AA)) y suelen ir acompañados de definiciones o extensiones. Interesa que después de que un acrónimo o sigla haya sido definido, se asigne la misma definición en futuras apariciones. Para detectar AA MetaMap aplica un algoritmo idéntico al descrito en Schwartz and Hearst . Se trata de asociar la expansión del AA, con la sigla o acrónimo, que deberá estar escrito entre paréntesis y situado después de la expansión. Existen ciertas reglas que es necesario cumplir para evitar errores:

[8] http://metamap.nlm.nih.gov/
[9] http://code.google.com/p/negex/

- Los AA no pueden contener más de 20 caracteres.
- Las expansión deben ser mayores que los AA correspondientes.
- Una expansión no puede contener texto entre paréntesis.
- Cada palabra considerada como AA debe contener como mucho 12 caracteres.
- Los AA no pueden comenzar por "such", "also" o "including".

- Conceptos

Pero por supuesto la parte esencial de MetaMap es aquella que lo define el mapeo de términos de un texto biomédico a conceptos del Metatesauro UMLS. El algoritmo que sigue para llevar a cabo el proceso de mapeo de términos de un texto biomédico a conceptos del Metatesauro UMLS engloba diferentes fases que son detalladas por el creador, el Dr. Aronson:

1. Parsing: Haciendo uso del Léxico Especializado realiza un primer análisis sintáctico siendo capaz de detectar diferentes elementos textuales, como la palabra principal de una frase.
2. Variant Generator: Para cada frase se genera una variante utilizando de nuevo el Léxico Especializado y en esta ocasión, de manera complementaria, una base de datos de sinónimos. Dicha variante consistirá en un sintagma nominal junto con variantes ortográficas, sinónimos, acrónimos o abreviaturas entre otros.
3. Candidate Retrieval: Formar el conjunto candidato de todas las cadenas del Metatesauro que contienen al menos una de las variantes.
4. Candidate Evaluation: Asignar a cada candidato un sintagma nominal, evaluarlo o puntuarlo y ordenar los candidatos por puntuación.
5. Mapping Construction: Combinar los candidatos que están involucrados en partes disjuntas del sintagma nominal. Calcular de nuevo la puntuación y seleccionar en base a dicha puntuación.

2.3.2 Salida generada

MetaMap puede generar archivos con diferentes formatos de salida:

- Human-Readable: formato de salida por defecto. Muestra para cada frase del texto de entrada la propia frase, una lista de conceptos candidatos del Metatesauro asociados a cada parte de la frase, el mapeo de realizar las combinaciones de candidatos asociados a partes disjuntas y datos complementarios como la puntuación otorgada, u otros opcionales como el CUI del concepto (-I), los tipos semánticos (-s), o las fuentes (-G).

- MetaMap Matching Output (MMO): incluye un súper-conjunto de la información de salida Human-Readable y tiene formato de términos Prolog. Permite realizar un pos-procesamiento por parte de aplicaciones Prolog. Opción -q.
- XML: también se puede obtener la salida MMO en formato XML. Para esto se puede usar una las siguientes opciones: –XMLf, –XMLf1, –XMLn o –XMLn1. La f indica que el .xml tendrá formato y la n que no. El 1 indica que se generará un .xml para un archivo de entrada, o, si no se incluye, uno por cada entrada citada. Tiene el inconveniente del peso añadido de los archivos en el disco.
- Colorized MetaMap Output (MetaMap 3D): este formato proporciona información mediante colores para los conceptos mapeados por MetaMap en un texto.
- Fielded MetaMap Indexing (MMI) Output: Mismo contenido que la salida MMO, pero representado en la opción múltiples líneas que contienen campos delimitados por tabulaciones. Opción -f. Este tipo de salida es utilizado principalmente por el Medical Text Indexer (MTI).

2.3.3 MetaMap API

MetaMap API[10] es una librería en Java desarrollada por Willie Rogers que permite la comunicación de un proyecto desarrollado en Java con el servidor de MetaMap para servirse de toda su potencia. Para su instalación y uso es imprescindible contar con una versión instalada de MetaMap y con Java 1.7 SDK o una versión superior. Además, para la correcta utilización de la librería, los servidores de MetaMap deberán estar en ejecución. Cabe señalar que el motor está escrito principalmente en SICStus Prolog y por lo que para facilitar su uso por programas Java se utiliza Prolog Beans.

2.4 Traductor

2.4.1 Introducción

La tecnología y precisión de los traductores ha evolucionado mucho en los últimos años. Aun así las traducciones están lejos de ser perfectas. Aunque en la traducción de palabras sueltas, o vocabulario, es poco común encontrar errores, al traducir frases, aplicando la gramática, lo normal es que el traductor haya cometido algún error.

Para este trabajo, en el que la precisión es un factor tan importante, parecería imposible poder usar un traductor y que los datos resultantes siguieran siendo válidos. Pero para este programa, que procesa el texto usando MetaMap, es decir, analizando conceptos médicos, el número de errores se mantiene al mínimo.

[10] http://metamap.nlm.nih.gov/README_javaapi.html

2.4.2 Fuentes

La decisión de usar un traductor para poder procesar con MetaMap los informes médicos en castellano, proviene de un artículo de la Universidad Europea de Madrid[11] . Fue realizado por Francisco Carrero, José Carlos Cortizo, José María Gómez y Manuel de Buenaga. Este trabajo comenzó intentando desarrollar una versión en castellano de MetaMap y acabó concluyendo que era preferible usar un traductor, apoyándose en los estudios que habían realizado comparando la precisión entre usar MetaMap con informes en inglés o con informes en castellano traducidos. Estos estudios demostraron que, incluso con las imperfecciones de los traductores, las diferencias eran mínimas. Gracias a que MetaMap es una herramienta para encontrar conceptos médicos y que la traducción moderna es precisa en la traducción de palabras, aunque la frase resultante no lo sea tanto. Además el proyecto es de finales de 2008, y desde entonces tanto los traductores, como MetaMap, han mejorado enormemente, reduciendo errores y aumentando la precisión.

2.4.3 Windows Traductor

Desde que se realizó el estudio de la UAM, han desaparecido todas las APIs de traductores gratuitos, el cese de Google traductor en 2011[12] como el más destacado.

En la aplicación se utiliza el traductor de Microsoft, por ser el único que tiene una tarifa base, de 2 millones de caracteres al mes, gratuita. Teniendo en cuenta que el procesado es una etapa previa al funcionamiento de la aplicación y que incluye la traducción de toda la base de datos (sólo debe realizarse una vez), esta es la limitación más importante del proyecto.

2.5 ApachePOI

Apache POI[13] es una API para Java, diseñada para poder acceder a la información contenida en ficheros de Microsoft (1997-2008). Además de poder leer el contenido de los ficheros, también es capaz de crear los propios.

Fue creada en 2001, como un subproyecto del Jakarta Project[14]. En esta aplicación solo se usa para leer Word, uno de los tipos de documentos de Microsoft Office, por lo que solo es necesario usar HWPF (Horrible Word Processor Format).

[11] http://www.esp.uem.es/jmgomez/papers/cikm08.pdf
[12] http://www.webcitation.org/5z1B8xoUj
[13] https://poi.apache.org/
[14] https://en.wikipedia.org/wiki/Apache_POI

2.6 Lucene

Toda la representación de informes médicos está construida para poder implementar un sistema de búsqueda por similaridad. La herramienta utilizada ha sido Lucene, que permite la indexación y búsqueda de informes, además de permitir realizar las búsquedas por similaridad necesarias para el proyecto.

Lucene es una librería para Java, desarrollada por Doug Cutting, y orientada para la realización de búsquedas de manera eficaz, rápidamente y con una sintaxis muy variada. Nos permite realizar un indexado sobre documentos estructurados y luego realizar búsquedas en las tokens generadas. Es utilizado para realizar búsquedas en un gran número de páginas web, entre las que se encuentran Twitter, IBM, Apple, Wikipedia o LinkedIn.

En la actualidad se encuentra en la versión 5.3 y se distribuye bajo la licencia Apache Software License. Cabe destacar que existen versiones para otros lenguajes de programación como Delphi, Perl, C#, C++ , Python, Ruby y PHP.

2.6.1 MoreLikeThis

La función más interesante para este proyecto es MoreLikeThis, que permite realizar busquedas ena la base de datos indexada usando un documento como referencia. De esta forma los resultados son los documentos más similares al elegido.

Funciona realizando cientos de búsquedas normales de Lucene y valorando los resultados usando un gran número de criterios.

A cada token que aparece en un documento le da un valor variable que depende de la frecuencia en la que aparece. Una palabra muy repetida perderá valor, y una palabra poco común tendrá más peso en la valoración final.

Además se puede fijar el mínimo de veces que una palabra (token) tiene que aparecer para que empiece a valorarla, o un número máximo de apariciones para eliminarla si es demasiado común. También se pueden usar estas opciones en frecuencia de documentos en vez de tokens, teniendo en cuenta una palabra solo si esta en un número mínimo de documentos distintos.

También tiene funcionalidades para darle más valor a ciertas palabras marcadas de antemano.

Chapter 3

Procesado de Informes

3.1 Introducción

En este capítulo explico el funcionamiento, estructura y problemas asociados a la parte del proyecto orientada a procesar los informes médicos. También aprovecho para explicar la estructura de los casos médicos y como se almacenan en la aplicación.

El programa está dividido en tres aplicaciones claramente diferenciadas. Cada una encargada de realizar una función distinta y dependiente de las demás:

1. Pre-Procesado: La primera se encarga de leer los informes en texto simple, soportando .doc y .txt. Usando los ficheros de configuración definidos por los usuarios organiza el texto en categorías y como salida tiene un documento .mxml, un fichero .xml con secciones médicas. Es el preprocesado de la aplicación y solo ha de realizarse con los nuevos informes.

2. Procesado: La segunda aplicación es la más compleja y la más importante para conseguir una gran precisión más adelante. Este programa se encarga de leer los documentos .mxml producidos por la primera aplicación y procesarlos para obtener .pxml, es decir, un documento .xml con los conceptos médicos del texto (tokens). Solo ha de usarse con los informes Pre-Procesados, después de que hayan sido separados en secciones. Requiere conexión a internet, una cuenta UMLS y una cuenta Azure para funcionar.

3. Búsqueda: La tercera aplicación implementa el principal objetivo del proyecto y consiste en usar Lucene para realizar búsquedas de similaridad (MoreLikeThis) entre los .pxml. También ofrece soporte para funcionar sin conexión usando texto simple (.mxml), sin necesidad de la conversión a tokens.

3.2 Informes Médicos

Es muy difícil procesar la información contenida en un informe médico. Esto se debe a varios motivos:

- Para empezar, los documentos se encuentran en lenguaje natural.
- También existen varios tipos de documentos, con distinta estructura, formato (varios formatos de texto, incluyendo Microsoft Word, y Excel) y que contienen distinta información médica (Cirugía, Informe de Alta, Registro de Enfermería, etc).
- Además, documentos de un tipo se pueden encontrar un número variable de veces o no haber ninguno de ese tipo.
- A todo esto hay que sumar la posibilidad de error humano, o que un médico haya alterado la estructura de alguno de los tipos de documento.

A raíz de esto, gran parte del tiempo de desarrollo del proyecto ha sido dedicado a poder procesar los informes y a unificarlos, para poder realizar las búsquedas.

3.2.1 Almacenamiento

Para almacenar los datos se ha creado una estructura que almacena la información del informe en los distintos estados que atraviesa.

Tras la primera parte de la aplicación se utiliza para para guardar la información dividida en campos en un fichero .xml con la extensión .mxml.

En la segunda parte, primero se leen los resultados del programa anterior de los ficheros .mxml. Después, tras realizar el procesado del informe traducido con MetaMap, almacena el resultado en otros ficheros .xml pero con extensión .pxml.

Por último, en la tercera parte del programa, se leen los ficheros .pxml y se almacenan y unifican todos los documentos de los casos en uno. Luego indexa el contenido de estos informes unificados y se puede empezar a realizar búsquedas.

3.2.2 Restricciones y problemas

Es importante destacar que los posibles secciones que usa internamente la aplicación están definidos de antemano. Por eso, todos los nombres que pueden tener los campos de InformeProcesado se encuentran dentro de los siguientes: Motivo ingreso, Alergias, Mediacion actual, Antecedentes familiares, Antecedentes personales, Anamnesis, Exploracion, Pruebas complementarias, Evolucion, Intervencion quirurgica, Juicio clinico, Tratamiento y Plan terapeutico.

3.3 Pre-Procesado

3.3.1 Funcionamiento

El objetivo de la primera aplicación, es el primer paso necesario para el procesado de la información, es dividir los informes médicos en los campos que lo forman. De esta forma, más adelante, se podrá especializar la búsqueda, teniendo en cuenta sólo algunos de los campos.

Estos campos siguen una estructura estándar y varía de un tipo de documento a otro, por lo que es imprescindible poder cambiar la estructura en función del documento que se esté procesando. Esto se realiza gracias a unos archivos de configuración definidos por el usuario antes de poner la aplicación en funcionamiento.

Por otro lado los informes médicos se encuentran almacenados en texto simple, por lo que la aplicación ha sido preparada para leer archivos en formato .doc y .txt.

La aplicación procesa los informes independientemente, para que sea más simple la modificación o añadido de informes a posteriori. Es decir, en caso de añadir un nuevo documento a alguno de los casos, sólo es necesario procesar ese documento, no todos los demás.

Necesita dos datos para funcionar: la ubicación de los informes sin procesar y la ubicación donde quieren guardarse los informes procesados. Se pueden proporcionar a través de la interfaz o usando el fichero de configuración para proporcionar automáticamente la última ruta usada.

3.3.2 Tecnologías utilizadas

Solo necesita usar la librería Apache POI para poder leer documentos con formato .doc, que es el formato más común entre los informes.

3.3.3 Restricciones y problemas

Aunque debía ser la parte más simple, es la que ha tenido el problema más destacado, la falta de uniformidad entre los informes médicos. Además siempre hay pequeños detalles que complican en gran medida la lectura correcta de los datos.

Fue muy difícil solventar este problema, incluso aprovechando parte de la solución que ya se encontraba en un trabajo previo.

Otro problema fue el formato en el que estaban almacenados los informes, el formato de Microsoft Word, .doc. Por lo que, antes de poder ocuparme de la funcionalidad principal de la aplicación, implemente la capacidad de leer este formato, para lo cual me serví de la librería Apache POI.

3.4 Procesado

3.4.1 Funcionamiento

El objetivo de la segunda parte es procesar los informes médicos para conseguir mejores resultados cuando se realicen las búsquedas. Como los informes están en castellano, para poder procesar el texto es necesario primero traducirlos al inglés. Para esto se usa la API de Windows Traductor. Con el texto en inglés, ya podemos comenzar a usar MetaMap, una herramienta para analizar textos médicos relacionada con el Sistema de Lenguaje Médico Unificado (UMLS).

El primer paso es obtener la información producida en el programa de la primera etapa, es decir, los informes médicos divididos por campos. Esto se consigue leyendo los archivos del tipo .mxml.

Con la información de los documentos médicos, sometemos cada campo a un proceso complejo. Primero lo traducimos al inglés, el único idioma soportado por MetaMap. Después usamos esta herramienta, MetaMap, para analizar el texto para obtener términos médicos, buscar desambiguaciones y detectar negaciones.

Esta nueva información, se almacena en ficheros del tipo .pxml (.xml Procesado), que almacenan la información necesaria para la tercera etapa de la aplicación. Requiere la ubicación de los ficheros .mxml para funcionar. También necesita una cuenta Microsoft Azure y una cuenta UMLS válidas para funcionar (Apéndice A)

3.4.2 Estructura

La clase Execute se encarga de ejecutar la aplicacción. En esta clase se llama a la clase ConfigurationFile para obtener los datos necesarios para el funcionamiento de la aplicación (carpeta de casos pre-procesados, datos de aplicación de Microsoft Azure y datos de cuenta UMLS).

Sabiendo la ubicación de los casos pre-procesados, podemos empezar a recorrerlos, traduciendo cada campo y enviándolo como petición a UMLS a traves de MetaMap. Cuando obtenemos todos los campos procesados, es decir, todos los conceptos médicos, los guardamos divididos por campos en un archivo .pxml en la misma carpeta que los archivos pre-procesados.

3.4.3 Tecnologías utilizadas

Para implementar esta sección han sido necesarias dos tecnologías distintas: Windows Traductor y MetaMap.

Windows Traductor se encarga de traducir los informes médicos para poder ser procesados por MetaMap.

MetaMap se usa a traves de la WebAPI, realizando una consulta del tipo Batch a los servidores de UMLS por cada campo de cada documento de cada caso médico.

3.4.4 Restricciones y problemas

El procesado de los informes depende de una aplicación externa, MetaMap, por lo tanto, aunque la conversión de los informes lleva mucho tiempo, no es posible reducirlo. Aún así se ha hecho todo lo posible para no llevar a cabo este costoso proceso más veces de las necesarias, comprobando qué informes ya estaban procesados para no volverlos a convertir.

Además MetaMap es una herramienta muy especializada, que cuenta con un gran número de opciones, por lo que es muy difícil usarla. Es necesario una investigación en profundidad antes de poder empezar a utilizarla.

También a causa de que MetaMap es una herramienta privada es necesaria una cuenta UMLS para poder usarlo. Por esto es necesario que el usuario tenga que crear una cuenta antes de poder iniciar la aplicación.

El problema más grave que hay en esta parte de la aplicación son las limitaciones intrínsecas al traductor. Desde que Google Traductor se volviera un servicio de pago para aplicaciones en 2011, se ha vuelto cada vez más difícil encontrar un servicio de traducción gratuito. Windows Traductor, el servicio usado en la aplicación, tiene una limitación de 2 millones de caracteres por mes. Para poder aumentar el volumen de traducciones, una parte esencial del procesado, sería necesario invertir en una suscripción en alguno de los servicios de traducción de pago. En este caso, con usar una cuenta Microsoft Azure con mayor volumen de traducción sería suficiente (Apéndice A).

Chapter 4

Indexación y búsqueda en informes procesados

4.1 Introducción

En este capítulo el buscador de informes implementado.

Una vez obtenido el conjunto de archivos .pxml, tras ejecutar las dos primeras partes de la aplicación. El siguiente paso consiste procesarlos para poder sacar provecho de la información que contienen. De esta manera se pretende desarrollar un buscador de informes médicos que realice una búsqueda descartando términos no relevantes y sólo teniendo en cuenta los conceptos médicos. Para consultar la estructura de estos archivos revisar el apartado 3.2 **Informes Médicos**.

Dicho buscador le permite al usuario seleccionar un informe en formato .pxml e intentar localizar aquellos similares a él. También se puede realizar con informes en formato .mxml, aunque la precisión será menor pues no habrán sido procesados para eliminar las palabras que no guarden relación con el tema.

Esta aplicación comienza unificando todos los documentos procesados de un caso en un solo documento. Este a su vez está dividido en campos, y cada campo es la suma de todos los campos del mismo tipo dentro del caso.

Una vez que se unifican todos los documentos del historial médico en un solo archivo, se empieza a realizar la búsqueda por similaridad, usando sólo los campos marcados por el usuario, o, en su defecto, todos.

4.2 Indexación

Inicialmente, durante la etapa de prototipado, el programa utilizaba Lucene con un solo index, es decir, todos los documentos se indexaban juntos para la búsqueda.

Una vez estuvieron implementadas las funcionalidades básicas, con el propósito

de aumentar la precisión de la búsqueda, se dividió el index por campos. En vez de utilizarse un sólo index, se utilizaba uno por cada campo implementado. Esto además permite modificar la búsqueda por similaridad para utilizar solo los campos en los que se quiere buscar.

Con los datos resultantes de unir todos los documentos de un caso se crean los índices de Lucene. Cada uno de ellos esta formado por todos los campos del mismo tipo en la base de datos, cada bloque de información con un identificador para saber a que caso pertenece. Usando estos índices especializados se podrán realizar las búsquedas.

4.3 Búsqueda

La búsqueda por similaridad funciona realizando búsquedas MoreLikeThis de Lucene. Esta funcionalidad requiere de varios pasos previos para poder utilizarla, el más importante la indexación de los documentos entre los que se quiere buscar y cual quiere usarse como base en la búsqueda.

El resultado consiste en el identificador de cada documento indexado, acompañado por un número que representa su valoración en la comparación con el documento base. Están ordenados de mayor a menor, es decir, de más similar a menos. Como el documento base, aquel del que hay que buscar informes similares, ha de indexarse para realizar la búsqueda, siempre encabeza la lista de resultados. Para evitar ser redundante y no confundir al usuario, se muestra a partir del siguiente resultado.

Se ejecuta MoreLikeThis para todos los campos disponibles y se van unificando los resultados de las distintas búsquedas. Una vez terminado este proceso, contamos con una lista de resultados desordenada. Una vez ordenada, tendremos todos los casos ordenados por similaridad, encabezados por el caso base elegido por el usuario.

Con los datos resultantes de tantas búsquedas de similaridad como campos se hayan elegido, se crea un valor numérico por cada caso médico. Este valor, o "score", es la media de los resultados de todas las búsquedas por campo de todos los documentos de un mismo caso. Representa la similaridad que hay entre ese caso y el caso que se ha usado como base en la búsqueda de similaridad.

Por último, para mostrar los resultados es necesario ordenar los casos de mayor a menor usando la "score" (el grado de similaridad), de esta forma los casos estarán ordenados de mayor a menor similaridad.

4.4 Tecnologías Utilizadas

Para realizar este sistema se utilizó la librería de Java, Lucene que permite realizar la indexación y hacer búsquedas sobre el índice creado de manera eficaz. Un inconveniente que podría presentar esta librería es la gran cantidad de trabajo que realiza durante el proceso de indexación y el enorme consumo de recursos y tiempo que puede costar, en función del tamaño de la base de

datos. En cambio, a la hora de realizar una búsqueda, esta se lleva a cabo en un período muy corto de tiempo.

4.5 Restricciones y problemas

En esta parte del programa el problema más destacado es la unificación entre documentos procesados de distinto tipo. Para poder permitir que se siguieran comparando los casos, hubo que sacrificar un poco de precisión. Igualmente los campos internos que usa el programa están diseñados para incluir todas los campos presentes en un informe médico, de tal forma que, la pérdida de precisión, aunque inevitable, se reduzca al mínimo.

Hay otro problema menor relacionado con la librería que se encarga de la búsqueda por similaridad, Lucene. En las últimas distribuciones de la función más importante para este trabajo, la búsqueda por similaridad, MoreLikeThis, ha perdido funcionalidades y actualmente el número de opciones disponibles es muy reducido.

Chapter 5

Conclusiones y trabajo futuro

5.1 Introducción

Ahora, habiendo terminado de desarrollar el proyecto a lo largo del curso académico, puedo concluir que se han cumplido los objetivos planteados. El programa, dividido en tres fases, es capaz de leer, procesar los informes médicos para separarlo por campos, traducirlos al inglés y obtener la terminología médica utilizando MetaMap. Una vez realizado el procesado, se pueden buscar informes parecidos en la base de datos generada. El programa principal tiene un gran número de usos y aplicaciones, desde ayudar en estudios estadísticos a facilitar diagnósticos. Pero, desde mi punto de vista, creo que la primera parte del programa, encargada de procesar los informes médicos en castellano, tiene mucho más potencial a la hora de ser aplicada en trabajos futuros.

La aplicación de búsqueda puede ser útil tanto para profesionales como para investigadores, ya que gracias a ella podrían localizar informes médicos con unas características concretas en menos tiempo de lo que podrían haberlo hecho manualmente. Por estar procesado, las búsquedas tendrán más precisión, ya que el buscador no se verá afectado por el ruido causado por las palabras desdeñables para la búsqueda porque no tienen relación con los conceptos médicos.

5.2 Trabajo futuro

Para empezar, antes de poder darle una nueva aplicación al proyecto, creo que sería recomendable centrarse en corregir las limitaciones de la parte de procesado, la más compleja y útil. Existen dos graves problemas:

- Limitaciones del traductor.
- El tiempo consumido por MetaMap.

Además se me ocurren algunas usos posibles para el programa y la parte de procesado.

5.2.1 Cambiar el traductor

El problema más acuciante impide procesar una base de datos completa y está causado por motivos económicos. Para el procesado de informes es necesario el uso de un traductor. El traductor que utiliza actualmente la aplicación tienen una limitación de uso de dos millones de caracteres por mes. Mi decisión de usar el traductor de Microsoft se debió a que ofrecía una tarifa gratuita que me permitió desarrollar la aplicación. Pero, teniendo en cuenta que la base de datos sólo ha de procesarse una vez sería recomendable, cambiar a algún traductor que cobre por número de caracteres (por ejemplo Google Traductor). En este caso el proyecto tendría que contar con una inversión económica para ser viable.

5.2.2 Optimizar el uso de MetaMap

El uso de MetaMap con informes en castellano, y no en inglés, supone un problema que he podido resolver gracias al uso del traductor. Aún así el tiempo que tarda MetaMap en procesar los textos es extremadamente largo, aún más con documentos fraccionados, que aunque no tengan más texto, si tienen que realizar muchas más llamadas al servidor. Como MetaMap sólo se utiliza para procesar la base de datos la primera vez, no es un cambio urgente. Aún así pienso que el tiempo necesario para optimizar esta función compensaría a la larga. Por ejemplo, sólo consiguiendo que cada petición al servidor de MetaMap aprovechara los diez mil caracteres máximos por envío, y no sólo unos cientos, se reduciría en gran medida el tiempo gastado en esta función.

5.2.3 Soporte para un mayor número de tipos de documentos

La aplicación se ha centrado en procesar los informes de texto plano, pero exiten otros documentos que componen el historial médico y que no han sido incluidos. También existen documentos en Excel, con las anotaciones del médico en relación a un paciente a lo largo de los años. Estos documentos tienen cientos de entradas y añaden mucho ruido a las busquedas, pues hacen mención de todas las enfermedades y complicaciones que ha sufrido un paciente a lo largo de los años, la mayoría de las veces no relacionado con el enfermedad principal que se está tratando.

En un desarrollo futuro se podría implementar soporte para estos documentos. Incluso se podría intentar predecir las complicaciones derivadas de un tratamiento a partir de las notas de otros pacientes.

5.2.4 Soporte para distintos idiomas

Con una pequeña modificación, el proyecto podría adaptarse para funcionar con informes médicos en cualquier idioma. Sólo sería necesario cambiar las configuraciones del traductor. Incluso sería posible que el traductor reconociera automáticamente el idioma de los informes y actuara en consecuencia.

5.2.5 Soporte para multiples tipos de documento médico

Actualmente la aplicación procesa todos los documentos médicos como si fueran del mismo tipo de cara a la division en campos (el único momento del proceso en el que la estructura del documento no esta todavía almacenada correctamente en la base de datos procesada). Se podría mejorar la aplicación para que reconociera el tipo de documento y usara el fichero de configuración de la estructura correcto.

5.2.6 Representación gráfica

Gracias a la búsqueda por similaridad, la obtención de datos y resultados estadísticos se convierte en una tarea sencilla. Sería posible ampliar el funcionamiento del buscador de forma que se pudiesen generar gráficas y cuadros con datos reunidos previamente. Así sería viable la recopilación de información referente a la distribución de una determinada enfermedad, en hombres y en mujeres, a lo largo de los años, o sobre ciertos rangos de edad, por ejemplo.

5.2.7 Diagnósticos automáticos

Un posible uso que podría tener la aplicación sería intentar crear una inteligencia artificial capaz de realizar diagnósticos mediante aprendizaje automático. En función de los síntomas del paciente, podría buscar casos similares en la base de datos médica. A más grande fuera la base de datos, mayor sería la precisión del médico virtual.

Appendix A

Manual

A continuación explico los pasos necesarios para hacer funcionar las distintas partes de la aplicación.

A.1 Ejecución del Pre-Procesado

En esta parte de la aplicación solo es necesario decidir las localizaciones de dos carpetas:

- La carpeta con la base de datos médicos, organizada como una lista de casos médicos.
- La carpeta en la que quieres almacenar todos los casos procesados, ya sean .mxml o .pxml. (Es la misma que va a utilizarse en las dos fases siguientes).

Para decidir estas dos carpetas hay dos posibles modos.

- Por defecto se usa las ubicaciones en el archivo de configuración config.txt (figura A.1) dentro de la carpeta config, que se debe encontrar junto al programa. Se almacenan por fuera de la aplicación y puede usarse en repetidas ocasiones.
- El otro modo es a traves de la aplicación de Pre-Procesado, usando las opciones que te ofrece la interfaz (figura A.2). Utiliza los datos que selecciones solo para la siguiente ejecución. Los dos primeros botones seleccionan ubicaciones que tendrán prioridad sobre las almacenadas en el archivo de configuración, que son los valores por defecto.

A.2 Ejecución del Procesado

En esta parte de la aplicación es necesario decidir la localización de una sola carpeta y los datos de dos cuentas:

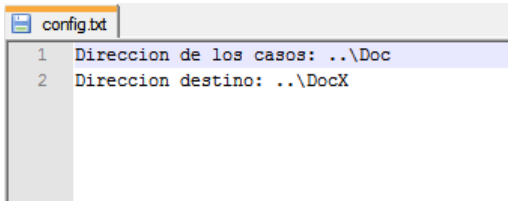


Figure A.1: Ejemplo de un archivo de configuración para la fase de Pre-Procesado.

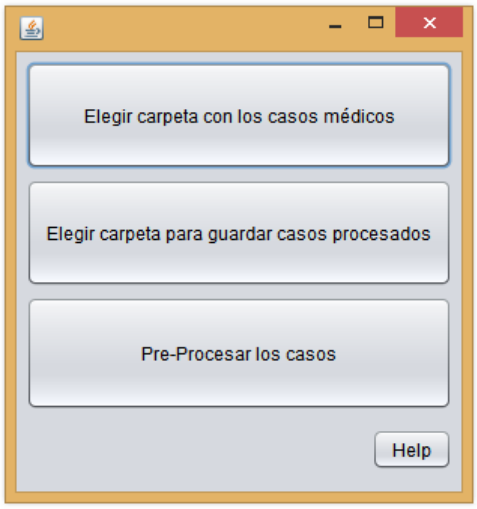


Figure A.2: Interfaz para la fase de Pre-Procesado.

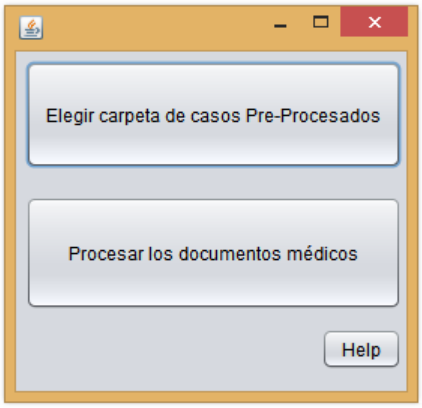


Figure A.3: Interfaz para la fase de Procesado.

- La carpeta con la base de datos médicos procesada por la aplicación, organizada como una lista de casos médicos procesados. (La misma que hayas usado para las dos fases anteriores).
- Además hay que incluir los datos de las dos cuentas necesarias (UMLS y Azure) para la ejecución de la aplicación.

En esta ocasión es más dificil introducir los datos.

- A traves de la interfaz de la aplicación de Procesados (figura A.3) solo se puede asignar la ubicación de los casos procesados para una ejecución. El primer botón selecciona una ubicación que tendrán prioridad sobre la almacenadas en el archivo de configuración, que son los valores por defecto. Tras empezar a procesar los documentos (segundo botón) saldrá un aviso informando cuántos documentos hay que procesar y el tiempo estimado. Después comenzará a trabajar en segundo plano y cuando termine saldrá un nuevo aviso y terminará la ejecución.
- La única manera de introducir las cuentas de UMLS y Azure es a través del fichero de configuración config.txt (figura A.4) dentro de la carpeta config que hay junto a la aplicación. También se puede almacenar por defecto la ubicación de los casos procesados. Como se almacenan por fuera de la aplicación, puede usarse en repetidas ocasiones.

A.2.1 Cuenta UMLS

Antes de poder empezar a usar esta etapa del procesado es necesario crear una cuenta de UMLS. Solo hay que seguir los pasos de esta página *https://uts.nlm.nih.gov//license.html*.

Es necesario crearla con anterioridad pues hay un período de dos a tres días laborables antes de recibir respuesta, ya sea para conceder la cuenta o no.

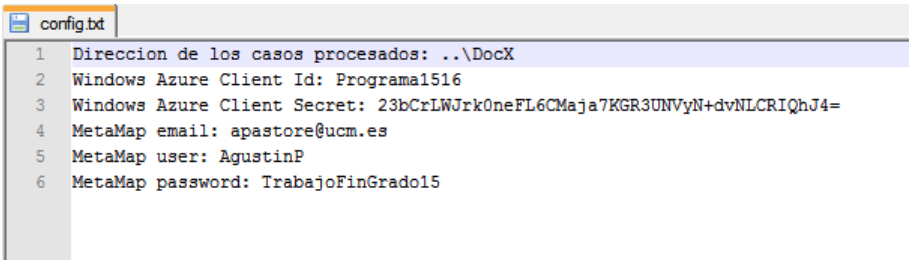


Figure A.4: Ejemplo real de un archivo de configuración para la fase de Procesado.

Si se tienen dificultades para crearla puede consultarse este video tutorial guiado[1], creado por la misma organización.

Hay tres valores a tener en cuenta a la hora de rellenar el archivo de configuración de la fase de procesado (figura A.4):

- La dirección de correo usada durante el registro.
- El nombre de usario seleccionado.
- La contraseña.

Los tres valores han de ser introducidos después de los dos puntos (':'), cada uno en su linea correspondiente para que el programa pueda funcionar.

A.2.2 Cuenta Azure

Aparte de crear una cuenta Microsoft Azure en esta dirección, también es necesario registrar una aplicación y suscribirse al servicio de traducción para esa aplicación.

Paso a paso sería como sigue:

1. Crear una cuenta Microsoft Azure[2].
2. Suscribirse al Traductor de Microsoft[3]. Sería recomendable seleccionar la opción de dos millones de caracteres al mes porque no tiene coste.
3. Registrar una aplicación en Microsoft Azure[4]. Los apartados que se necesitan para hacer funcionar la busqueda son los marcados en rojo en la figura A.5. El "Client secret" ya está completado automáticamente, mientras que en "Client ID" tiene que elegirse arbitrariamente. En el apartado "Redirect URI' se puede completar igual que en la figura A.5, con "https://www.microsoft.com".

[1] https://www.nlm.nih.gov/research/umls/user_education/quick_tours/requestLicense.html
[2] https://datamarket.azure.com/register?redirect=%2Faccount
[3] https://datamarket.azure.com/dataset/1899a118-d202-492c-aa16-ba21c33c06cb
[4] https://datamarket.azure.com/developer/applications/register

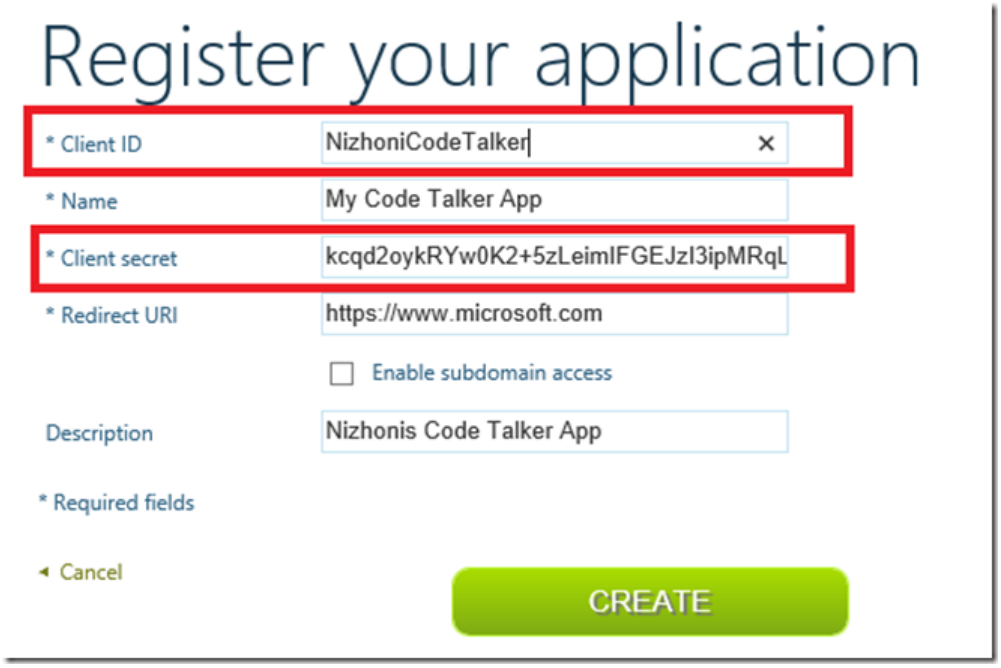


Figure A.5: Interfaz para la fase de Procesado.

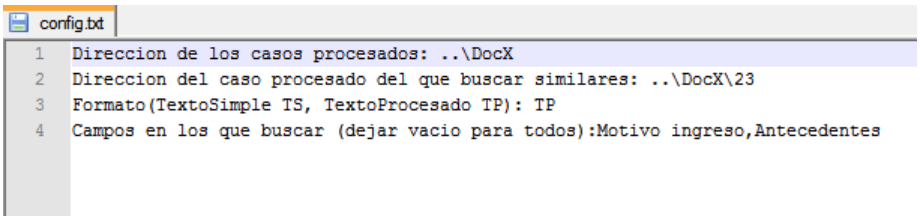


Figure A.6: Ejemplo de un archivo de configuración para la fase de Búsqueda.

En caso de surgir problemas al seguir estos pasos, tambien se puede consultar este tutorial más detallado[5].

A.3 Ejecución de la Búsqueda

En esta parte de la aplicación también es necesario decidir las localizaciones de dos carpetas:

- La carpeta con la base de datos médicos procesada por la aplicación, organizada como una lista de casos médicos procesados. (La misma que se haya usado para las dos fases anteriores).
- La carpeta con el caso médico procesado que va a utilizarse para la búsqueda.
- Además pueden seleccionarse los campos que se quieran utilizar en la búsqueda.

Para decidir estos datos hay dos posibles modos.

- Por defecto se usa las ubicaciones en el archivo de configuración config.txt (figura A.6) dentro de la carpeta config, que se debe encontrar junto al programa. Como se almacenan por fuera de la aplicación, puede usarse en repetidas ocasiones. Para los campos solo se pueden usar los siguientes (es necesario respetar la ortografía): Motivo ingreso, Alergias, Mediacion actual, Antecedentes familiares, Antecedentes personales, Anamnesis, Exploracion, Pruebas complementarias, Evolucion, Intervencion quirurgica, Juicio clinico, Tratamiento y Plan terapeutico.
- El otro modo es a través de la aplicación de Búsqueda, usando las opciones que te ofrece la interfaz (figura A.7). Utiliza los datos que selecciones solo para la siguiente ejecución. En este caso el primer botón elige la carpeta de casos médicos procesados que se utiliza El segundo botón elige el caso que se va a usar como base para realizar la búsqueda. El tercer botón inicia la búsqueda por similaridad y, al terminar, muestra los resultados. El botón

[5] https://blogs.msdn.microsoft.com/translation/gettingstarted1/

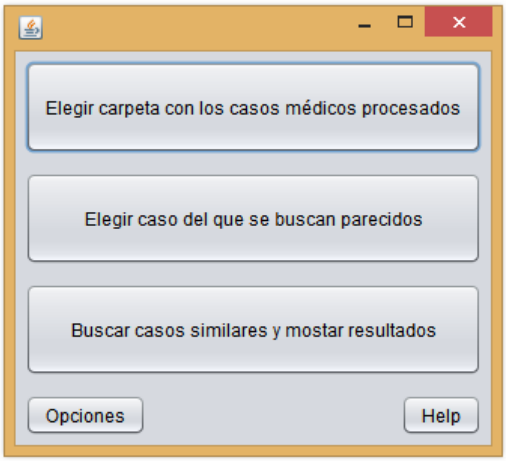


Figure A.7: Interfaz para la fase de Procesado.

de "Opciones" muestra las opciones disponibles (figura A.8), que incluyen: los campos sobre los que se va a realizar la búsqueda; y los ficheros entre los que se va a realizar la búsqueda: procesados (.pxml) o texto simple (.mxml). Todas las elecciones utilizando la interfaz tienen prioridad sobre la información almacenada en el fichero de configuración.

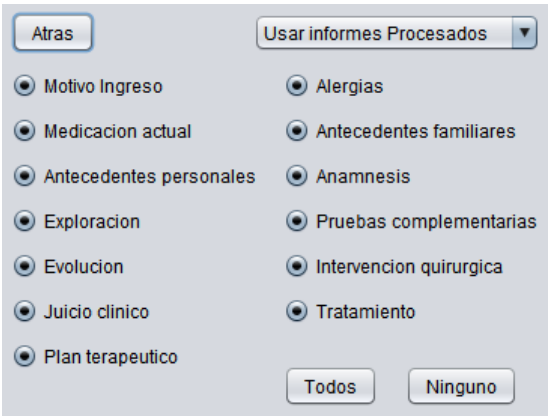


Figure A.8: Interfaz para las opciones de la fase de Procesado.