Búsqueda de historiales clínicos similares

Trabajo de Fin de Grado
Curso 2019–2020

Autor
Federico Sáez Lombán[1]
Lucas Mazariegos Arraiza[2]

Director
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Grado en Ingeniería del Software [1]
Grado en Ingeniería Informática [2]
Facultad de Informática
Universidad Complutense de Madrid

Búsqueda de historiales clínicos similares

Trabajo de Fin de Grado en Ingeniería Informática y Software
Departamento de Ingeniería del Software e Inteligencia Artificial

Autor
Federico Sáez Lombán[1]
Lucas Mazariegos Arraiza[2]

Director
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Convocatoria: *Enero 2020*

Grado en Ingeniería del Software [1]
Grado en Ingeniería Informática [2]
Facultad de Informática
Universidad Complutense de Madrid

23 de enero de 2020

Search for similar medical records

Final Thesis
Course 2019–2020

Author
Federico Sáez Lombán [1]
Lucas Mazariegos Arraiza [2]

Director
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Bachelor in Software Engineering [1]
Bachelor in Computer Engineering [2]
Faculty of Informatics
Complutense University of Madrid

Search for similar medical records

Final Thesis in Computer and Software Engineering
Department of Software Engineering and Artificial intelligence

Author
Federico Sáez Lombán [1]
Lucas Mazariegos Arraiza [2]

Director
Alberto Díaz Esteban
Antonio Fernando García Sevilla

Call: *January 2020*

Bachelor in Software Engineering [1]
Bachelor in Computer Engineering [2]
Faculty of Informatics
Complutense University of Madrid

23 de enero de 2020

Autorización de difusión

El abajo firmante, matriculado en el Máster en Ingeniería en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Grado: "Búsqueda de historiales clínicos similares", realizado durante el curso académico 2019-2020 bajo la dirección de Alberto Díaz Esteban y Antonio Fernando García Sevilla en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Lucas Mazariegos Arraiza, Federico Sáez Lombán

23 de enero de 2020

Dedicatoria

A mis padres y hermanos por estar siempre ahí. Al resto de mi familia, tanto a los que están como a los que no, y a mis amigos, por convertime en lo que soy. A Cintia por acompañarme y enseñarme que las mejores cosas merecen cualquier esfuerzo. A todos, gracias.

- Lucas Mazariegos Arraiza

Se lo dedico a mi familia, a mis amigos y a mi novia, que me han apoyado en todo momento

- Federico Sáez Lombán

Agradecimientos

A Alberto Díaz Esteban y Antonio Fernando García Sevilla por su apoyo, guía y paciencia durante todo este tiempo y a todas las personas que nos han apoyado durante todo este tiempo de manera directa o indirecta. Muchas gracias.

Resumen

Los historiales médicos contienen mucha información sobre un paciente. No solo síntomas, si no tratamientos aplicados y resultados de los mismos. En este trabajo crearemos una herramienta enfocada a los médicos en forma de aplicación, con interfaz web incluida, que analice y compare diferentes historiales médicos mediante diferentes técnicas de procesamiento de texto independientemente de su formato y devuelva el grado de similitud del mismo con otros para facilitar y agilizar la labor de los médicos a la hora de descubrir causas o asignar tratamientos a diferentes enfermedades. Los resultados inciales, debido al formato simple, y cercano al lenguaje real de los historiales, reflejan bastante fielmente la similitud entre los historiales obteniéndose mejores resultados en historiales de la misma especialidad médica o de sintomática similar. Sin embargo, los segundos resultados, no fueron los esperados debido al formato de los archivos, mucho más técnico y dificil de manejar, denotando la importacia de mantener una cohesión estructural entre los archivos de una misma colección.

0.1. Palabras clave

ElasticSearch, Historiales Médicos, Análisis de Texto, Salud, Procesamiento de Lenguaje Natural, F1-Recall-Precision, Similitud.

Abstract

Medical reports contain several information about a patient. Not only Symptoms, but also applied treatments and their results, whether they are good or not. In the present work, we will create an application that analyzes and compares different medical reports by text analysis regardless of the format of the data-set applied which gives back the similarity percentage of that report with the rest, with the objective of simplify and hasten the doctor's job when having to discover the cause or assign treatments to different illnesses. The initial results taking a collection with normalized data, are close to reality, showing a higher similarity between reports of the same medical branch or with similar symptoms thanks to their simplicity, and human-like language format.The second results, however, were not as good as we expected due to the a higher complexity and technical vocabulary. This Shows the importance of keeping an internal structural cohesion between the data of the same collection.

Keywords

ElasticSearch, Medical Reports, Text Analysis, Healthcare, Natural Processing Language, F1-Recall-Precision, Similarity.

Índice

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

"Medicina sólo hay una, y es efectiva cuando tiene una evidencia científica detrás que la respalde"
— J.M. Mulet

1.1. Introducción

Según un reciente artículo publicado por la Universidad de Washington, España se convertirá en el año 2040 en el país con mayor esperanza de vida del mundo, desbancando a la hoy en día, inalcanzable Japón.

La medicina ha avanzado a pasos agigantados a medida que nuevos tratamientos han sido descubiertos y nuestro objetivo es conseguir un sistema que continúe esa tendencia facilitando la laboriosa tarea de los médicos de comparar historiales uno a uno aún más, implementando una aplicación que consiga, eficazmente, realizar una búsqueda automática de similitud de historias clínicas, para facilitar y agilizar diagnósticos, tratamientos y comparaciones.

Este proyecto sigue la línea de proyectos anteriores y para ello seguimos contando con la colaboración del Hospital Público Comarcal de Jarrio en Asturias, el cual nos proporcionó una serie de datos médicos oftalmológicos. Estos historiales fueron anonimizados y su explotación académica fue aprobada por un Comité de Ética, siempre que fueran alojados y accedidos únicamente desde el servidor que nos proporcionara la universidad. Estos datos nos servirán como base para realizar los análisis de similaridad.

Sin embargo, también contaremos con la colección de datos MEDDOCAN [1], el cual surge como respuesta a la dificultad de acceder y trabajar con historiales médicos reales. Se trata de un set de datos anonimizados y

[1] `http://temu.bsc.es/meddocan/`

bien estructurados, los cuales nos permiten trabajar con ellos y realizar una aproximación a la similitud distinta a unos datos con un formato más sencillo y natural. (N. Perez, 2009)

1.2. Motivación

Con este proyecto, queremos desarrollar un software que permita comparar historiales médicos, de forma que los doctores puedan compararlos con los de su paciente, y hacerse una idea de la causa probable de sus problemas, o que tratamientos y con qué eficacia fueron tomados en casos anteriores, agilizando la labor de los médicos, permitiendo atender a más pacientes, y aumentando el porcentaje de casos tratados correctamente.

Existen varios tipos de historiales. Cada médico los escribe de diferentes maneras y nuestra aplicación debe ser capaz de lidiar con todos ellos independientemente de su formato. Para ello, nos han facilitado dos tipos de informes médicos de pacientes. Unos previamente tratados para cumplir con la Ley Orgánica de protección de datos, para poder analizar los casos, los cuales cuentan con varias secciones, tales como: motivo de ingreso, alergias... para que sea posible un mayor filtrado de información en el caso de que queramos centrarnos en un campo concreto y no en la totalidad del mismo. Por otra parte, contamos con otro set de datos creados por la herramienta MEDDOCAN, con diferente estructura e información, más sencillos, que nos permita, además de analizarlo, compararlo con los anteriores para llegar a conclusiones respecto a la importancia de varias características de los mismos y su influencia en los resultados obtenidos.

1.3. Objetivos

El objetivo principal es crear una herramienta útil y fácil de usar, que permita comparar historiales médicos en cuestión de segundos, ahorrando así un valioso tiempo a los médicos, ya que no tendrían que ir revisando historiales uno a uno tratando de encontrar algún tipo de similitud entre ellos. Es importante tener en cuenta que los médicos, por norma general, no poseen conocimientos informáticos avanzados, por lo que debe ser una herramienta simple y amigable para el usuario.

Uno de los grandes problemas de los historiales médicos es su longitud, en ocasiones extendiéndose varias páginas, por lo que es muy difícil extraer información útil de los mismos, más aún si tienen que analizarse una gran cantidad de ellos. Es por ello, que nuestro objetivo en este trabajo es lograr una rápida comparación de historiales mediante la tecnología ElasticSearch,

que nos permite obtener el grado de similitud de un historial con todos los que pertenecen a su misma colección de documentos. Esto nos dará un porcentaje, llamado score, tanto por secciones como texto completo, según nuestras necesidades, de los $\boldsymbol{n}$ documentos más similares al dado, siendo $\boldsymbol{n}$ el número de historiales más relevantes que deseamos obtener.

El otro gran problema es el tipo de datos que vamos a analizar. Cada persona tiene una forma o un estilo a la hora de almacenar los datos dentro de un historial. No es un proceso estandarizado, por lo tanto, nos encontramos con que de una colección a otra puede haber una gran diferencia en cuanto al formato de los mismos. Es por eso que, en este proyecto, trabajaremos con dos sets de documentos independientes entre sí. Por una parte, unos documentos de oftalmología, y por el otro, documentos generados por el script de anonimización MEDDOCAN. Ambos con diferente formato, campos y estructura, para analizar los datos de ambos y comprobar hasta que punto los resultados difieren o no respecto a los esperados y cual de ellos es mejor tras realizar el análisis.

Por otra parte, crearemos una interfaz gráfica que simplifique el uso de esta aplicación, que incluya varias opciones referidas a las búsquedas de historiales y nos permita variar de colección de datos fácilmente, es decir, que la aplicación sea independiente de los mismos y nos permita variarlos sin dificultad a la hora de comprobar su similitud, aumentando su versatilidad.

1.4. Estructura del documento

La memoria estará dividida en cuatro apartados, los cuales son los detallados a continuación:

1. **Introducción:** Breve resumen de las motivaciones y objetivos de este proyecto.
2. **Estado del arte:** Descripción del estado actual de las tecnologías utilizadas para este proyecto.
3. **Arquitectura:** Contiene el resumen de la arquitectura, diagrama de componentes y el flujo de un caso de uso común.
4. **Pruebas realizadas:** Exposición del trabajo que hemos llevado a cabo. Con exposición de resultados y experimentación.
5. **Conclusiones y trabajo futuro:** Conclusiones que hemos podido extraer de nuestro trabajo y posibibles maneras de mejorarlo en un futuro.

Chapter 1

Introduction

"There is only one medicine, and it is only effective when there is scientific evidence to support it"
— J.M. Mulet

1.1. Introduction

According to an article of the university of Washington, Spain will become, by year 2040, in the country with the highest lifespan in the world, surpassing like this, to the, nowadays unreachable Japan.

Medicine, since then, has advanced in giant steps while treatments were being discovered and our objective is to create a system that continues this tendency, making the doctor's hard work task of comparing medical reports one by one easier, implementing an application that manages in an efficient way, perform an automatic search of similarity between medical reports, making it easier and faster for doctors.

This project continues the same line as the previous projects and for that, we continue our collaboration with the Public Hospital of Jarrio, Asturias, who provided us a set of ophthalmological data. This data was previously anonymized and approved by an ethics committee with the condition they were only accessible by the university server. This data will be the base for our similarity analysis

However, we were also provided with the MEDDOCAN[1] data collection, that came as an answer to the difficulty of accessing and working with real medical reports. It is an well-structured, anonymized set of data, which allow us to work with them and perform a different approach to similarity analysis with an easier and more natural format.(N. Perez, 2009)

[1] `http://temu.bsc.es/meddocan/`

1.2. Motivation

With this project, we want to create a software that compares medical reports, so doctors can compare them with other patient's ones and get information about possible causes of their medical problems or what treatments are more suitable for each patient. Saving a lot of time to doctors, improving the percentage of correct diagnosis.

There are several types of medical reports. Each doctor writes them in their own way and our application must be able to deal with them independently of its format. For that, we have been given 2 different types of medical reports. One of them previously treated following the data protection directives, which have several fields like (allergies, symptoms, treatment...) so a better sorting of results is possible in case we want to study one field in particular and not all of them. We also have another data set called MEDDOCAN, with different structure and information, quite simpler, that allow us, besides analysing them, establish the importance of these characteristics and their importance in the further obtained results.

1.3. Objectives

Our main goal is to create an useful, easy to use tool, which allows doctors to compare medical reports in just a few seconds, wasting as less time as possible because until now, doctors have to do this one by one trying to find a similarity of some extent. It is important to understand that doctors don't need to have necessarily good computer knowledge, so it has to be totally simple and user-friendly.

One of the biggest issues of this project is the length of the reports, reaching many times several pages, so it is, sometimes, a nightmare to find useful information by eye, specially if you have to analyze many of them. That it why, our goal is to create a simple and fast comparison by using ElasticSearch that give us a similarity percentage of a report between them and all the rest in the same collection. This will give us a percentage, called score, divided by sections or full-text.

The other big problem is the type of the data we are going to analyse. Each person has a unique way of storing or writing data. It is not a standardized process, that is why, between one collection and another one, it can exist a big difference. In this project we will work with two different independent data sets. On the one hand, the ophthalmological documents and in the other one, the MEDDOCAN reports, both with different format, fields and way of writing so we will be able to analyse both and discover

what in the real influence - results-wise -of these characteristics and which one is better to use.

Moreover, and in addition of these results, we will create a graphic interface that simplifies the use of this application and that includes the options referred to the report search and that allow us to change the collections easily, meaning the application must be data independent and that we could change the data set easily, making it more useful and versatile.

1.4. Document structure

The memory will be divided in four sections

1. **Introduction:** Short summary of our goals and motivations
2. **State of Art:** Description of the current state of the technologies used in this project.
3. **Architecture:** It contains the architecture summary, component diagram and the flow of a common use case.
4. **Test performed:** Work we have done, including investigation and experimentation.
5. **Conclusions and future work:** Work conclusions and possible ways to improve it in the future

Capítulo 2

Estado de la Cuestión

Para nuestro proyecto, nos basaremos principalmente en el uso del motor de búsqueda ElasticSearch, que nos permitirá realizar la comparación de similitud de los documentos. A continuación expondremos el estado de del arte de la misma.

2.1. ElasticSearch Estado del Arte

ElasticSearch[1] es un motor de búsqueda basado en la librería Lucene (librería de software libre, que aunque es útil en cualquier aplicación que requiera capacidad de indexado y búsqueda, es más reconocida por su utilidad en la implementación de motores de búsqueda de internet, en búsquedas locales en páginas web).

Proporciona un motor de búsqueda distribuido con una interfaz web HTTP y documentos JSON no esquemáticos que permite, entre otras cosas, realizar análisis de texto con el que obtener la similitud entre dos textos. Está desarrollado en java bajo la licencia de software libre de apache.(ElasticSearch, 2014)

2.1.1. Conceptos básicos

Uno de los mayores usos de ElasticSearch, es que tiene capacidad de indexado. Esto nos permite crear un índice con las palabras que aparecen en cada texto. Los índices nos permiten indicar en qué texto, o incluso en qué posición se encuentra una palabra, con lo cual podemos extraer información útil de la misma. Esto puede ser mejorado de muchas formas, por ejemplo,

[1] https://www.elastic.co/es/

quitar artículos y preposiciones de la lista, ignorar mayúsculas, usar solo la raíz de palabras en vez de las mismas conjugadas...

Esto lleva a sí mismo a un problema. Hemos conseguido extraer los documentos en los que existe una palabra o serie de palabras, pero, ¿cuál es más relevante o pertinente para nosotros? Para ello se introduce el concepto de "similitud" o *scoring* en inglés, el cual asigna un valor numérico a cada documento dependiendo de su grado de similitud.

En este contexto, debemos analizar dos conceptos de forma independiente los cuales afectan al valor final del scoring:

1. **TF (Term Frequency o Frecuencia de término)** Term Frequency hace referencia a la frecuencia con que un término aparece en nuestra búsqueda, sin embargo, esta frecuencia también se ve afectada por el tamaño de los documentos, es decir, documentos más largos tendrán más posibilidad de contener una palabra más veces que uno más corto. (Wikipedia, 2019) Es necesario realizar una normalización de la frecuencia para adecuarse a esta situación. Por ello, se introduce este concepto, cuya fórmula es la siguiente:

   $$tf(t,d) = \frac{f(t,d)}{max\{f(t,d) : t \in d\}}$$

   Siendo f(t,d) la frecuencia bruta y

   $$max\{f(t,d) : t \in d\}$$

   la frecuencia máxima del término en el documento. Sin embargo, en la mayoría de ocasiones, las palabras que más aparecen en un texto no son las que más información contienen. Por el contrario, estas palabras suelen ser palabras sin significado semántico. Palabras como conectores, artículos y preposiciones, son las más repetidas en la mayoría de idiomas y no nos aportan información relevante. Es por ello que se introduce el concepto de *Document Frequency*, que calcula si esa palabra es común a todos los documentos a analizar (Asensio, 2018) (bruno Stecanella, 2019).

2. **IDF (Inverse Document Frequency o Frecuencia de Documento Inversa)**

   Cuanto mayor sea el número de documentos en los que una palabra aparece, hace que la palabra sea menos especial. En este concepto, el valor que nos interesa no es la Frecuencia de documento, si no su inverso, cuya fórmula es la siguiente:

$$idf(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

siendo |D| el número de documentos, y

$$|\{d \in D : t \in d\}|$$

el número de documentos donde aparece el término t. Sin embargo, esta fórmula no tiene en cuenta si el documento está o no dentro de la colección, lo que produciría una división por cero en caso negativo. Es por ello que se suele ajustar esta fórmula a la siguiente:

$$idf(t, D) = \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

Finalmente, el término tf-idf define cómo de frecuente es una palabra en un término de una forma normalizada, teniendo en cuenta tanto la Frecuencia de Término como el inverso de la Frecuencia de documento, que es lo que ElasticSearch utiliza para obtener el *Score*.

Score= tf-idf(t,d,D)=tf(t,d) * idf(t,D)

o si queremos visualizarla de forma desarrollada:

$$tf - idf(t, d, D) = \frac{f(t, d)}{max\{f(t, d) : t \in d\}} * \log \frac{|D|}{1 + |\{d \in D : t \in d\}|}$$

El valor tf-idf será alto con una elevada frecuencia de término (en el documento dado) y una pequeña frecuencia del término en el conjunto de documentos. Este resultado se determina basándose en los aciertos de campos de la query que haya sido especificada. Este resultado puede ser modificado dependiendo de cualquier configuración adicional o filtro que el usuario haya estimado oportuna.

2.1.2. Características

ElasticSearch puede ser usado para buscar todo tipo de documentos. Proporciona búsqueda escalable, las búsquedas se realizan en tiempo real y suporta multi-índices. Es una aplicación distribuida, lo que significa que está dividida en varias particiones, las cuales pueden tener un número variable de réplicas o no.

Cada nodo almacena varias particiones y actúa como coordinador de operaciones, asegurándose de que vayan a las particiones correctas. El balanceo

y enrutado se hace de manera automática. Los datos relacionados se almacenan normalmente en el mismo índice. Una vez el índice ha sido creado, el número de particiones no puede variar.

ElasticSearch está desarrollado conjuntamente con un motor de colección de datos y análisis de registros llamado Logstash y una plataforma de análisis y visualización llamada Kibana. Estos tres productos juntos se denominan Elastic Stack.

ElasticSearch usa Lucene e intenta hacer todas sus características disponibles a través de JSON y de API de Java. Soporta facetting (búsqueda con filtros) y *percolating*(indexar documentos y seleccionarlos por querys), lo cual es útil para notificar si nuevos documentos encajan con querys registadas.

Otro uso es el llamado Gateway, el cual gestiona la duración de los índices, por ejemplo, un índice puede ser recuperado del Gateway en caso de que el sistema se caiga o entre en suspensión.

Soporta peticiones Get en tiempo real, lo cual lo hace apropiado como una base de dato NoSQL, pero carece de transacciones distribuidas.(González, 2019) (Cuervo, 2019)

2.1.3. Usos y ventajas

En nuestra aplicación utilizamos las ventajas que ofrece ElasticSearch como un motor de búsqueda y análisis potente basado en Lucene (Apache), el cual ofrece las herramientas más poderosas de búsqueda a día de hoy, y la capacidad de poder modificar los algoritmos de búsqueda a nuestro parecer, para así poder crear búsquedas mucho más fiables para el criterio que nos compete.

ElasticSearch nos proporciona la capacidad de realizar análisis de un texto completo, y posee las herramientas necesarias, a su vez, para darnos una similitud, utilizando el algoritmo TF/IDF.

Con todo, ElasticSearch es una gran opción si disponemos de una cantidad pequeña de datos, ya que para documentos muy grandes, puede darse una pérdida de datos o que la aplicación deje de funcionar correctamente, elementos que deben ser tenidos en cuenta si pensamos escalar el proyecto a uno mucho mayor.

En nuestro poyecto base, utilizamos la función por defecto *more_like_this* de ElasticSearch, el cual nos proporciona la capacidad de realizar búsquedas mediante índices a partir de un documento. Esto da como resultado los n

documentos que son similares al dado, analizando los cuales tienen un mayor **TF-IDF** mediante las fórmulas explicadas con anterioridad. Funciona a través de búsquedas de Lucene a las que se aplican los filtros indicados.Esto puede ser afinado de diversas maneras que estudiaremos en este proyecto, como el uso de "stopwords", poner un límite mínimo al número de documentos en los que tiene que aparecer la búsqueda, darle más peso a algunas palabras que nos interesen especialmente o diversos filtrados más.

Capítulo 3

Resumen de Arquitectura y componentes

3.1. Arquitectura

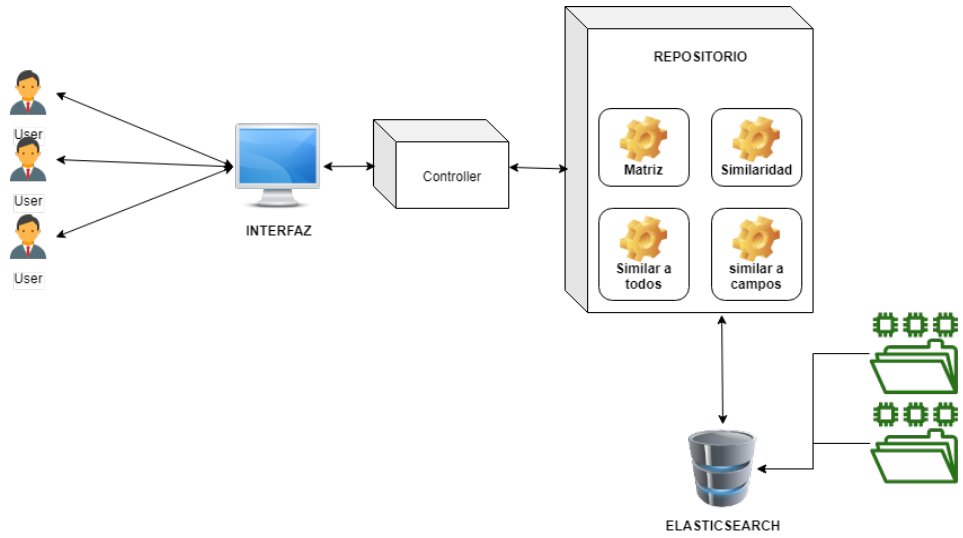

Figura 3.1: Arquitectura del sistema

La Arquitectura de nuestra aplicación está basada en diferentes elementos que interactúan entre sí, cuya funcionalidad básica es la siguiente:

- **Controlador:** Elemento clave de esta aplicación. Se encarga del funcionamiento básico de la misma. Crea el repositorio y lanza el servidor web. En general, se ocupa de ser el nexo entre el front y el back de la aplicación.

- **Repositorio:** Gestiona la base de datos ElasticSearch, así como las consultas. El procesamiento de estos datos es independiente de la colección. Contiene, además, todas las funciones de similaridad implementadas (búsqueda de similares a un caso, por secciones, por id...) serán accedidas a través del controlador y se comunican con la base de datos.

- **ElasticSearch:** Es nuestra base de datos, la cual contiene la información de los historiales procesados. Realiza el análisis del texto mediante NLP y devuelve el score de similaridad dependiendo de los filtros y la query que hayamos indicado. El resultado se carga en la interfaz a través del controlador para mostrar los datos de similitud.

- **GUI:** Para una mejor visualización de los datos, crearemos una interfaz gráfica que controle el funcionamiento del programa, ya que, pensando en un uso futuro, nuestros posibles usuarios serán personas que no tengan conocimientos informáticos. Utiliza el micro-servicio *Bottle* para comunicarse con la funcionalidad del repositorio. A su vez, recibe los datos de similitud de ElasticSearch a través del controlador para mostrarselos al usuario.

Es importante resaltar que el tratamiento de los datos se realiza de forma independiente a la implementación del programa. Es decir, el programa es capaz de trabajar y funcionar correctamente con diferentes tipos de datos (siempre en español). Esto aumenta exponencialmente la adaptabilidad de nuestro programa y permitiría su reutilización en un espectro más amplio de documentos al realizar búsquedas de similitud.

3.2. Diseño

Flujo de peticiones

A continuación pasaremos a detallar con detenimiento en flujo de datos de nuestro sistema en caso de que un usuario realice una petición a través de la interfaz. Para este caso de uso, supondremos que tanto la base de datos como el repositorio están ya creados, por lo que no se detalla en el diagrama su creación, pero si está indicado para facilitar el entendimiento del mismo.

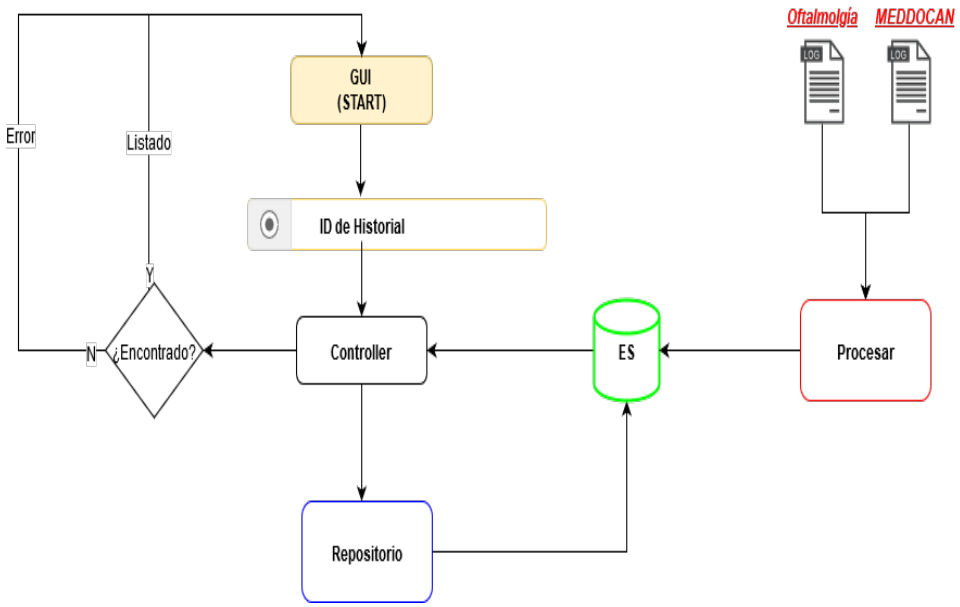

Figura 3.2: Caso de uso de búsqueda por ID

En este caso de uso, el usuario se conecta a la interfaz gráfica y elige una de las opciones que proporciona la misma, en este caso, búsqueda por ID. Esa petición llega a nuestro controlador, que se encarga de aceptar la petición y a su vez lo redirige al repositorio. Dentro del repositorio, la función lanza una query a nuestra base de datos ElasticSearch, que como podemos ver, contendrá los historiales de la colección que haya sido introducida tras procesarlos. En la base de datos se busca el ID del caso que se haya introducido, y se realiza la instrucción indicada. Este resultado se manda de nuevo al controlador, que lo redirige hasta la GUI. A partir de aquí, los resultados pueden ser dos: O bien se encuentra el ID en la base de datos y por tanto, se muestra un listado con los diez documentos mas similares junto a su score, o por el contrario no lo hace y redirige a una pantalla de error, informando del resultado.

3.3. Componentes del sistema.

Este proyecto cuenta con diferentes componentes los cuales expondremos en este punto e intentaremos explicar sus funcionalidades básicas utilizadas en nuestro proyecto.

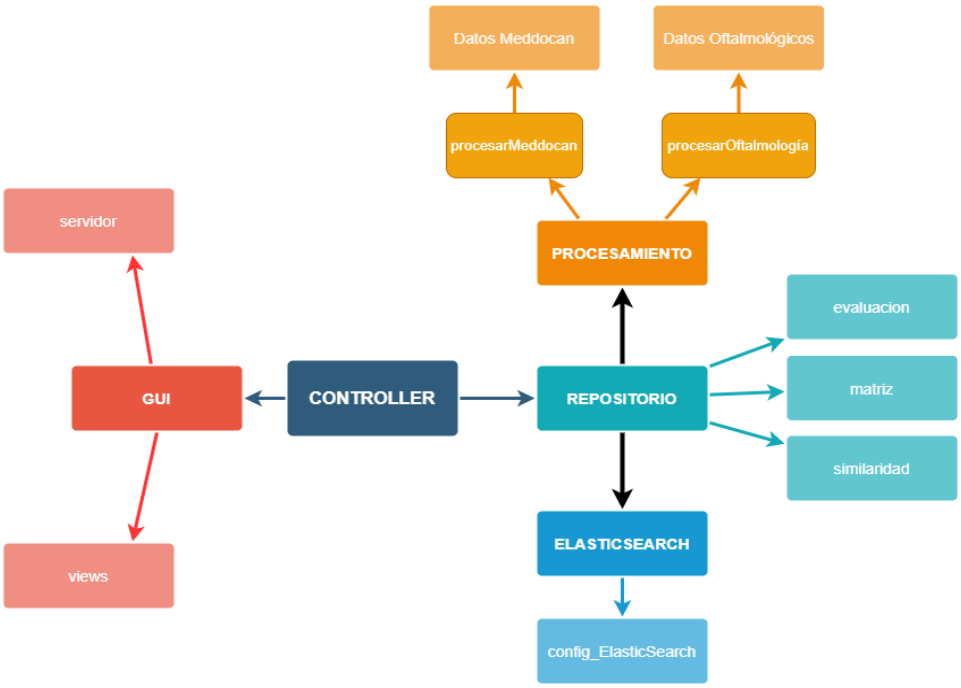

Figura 3.3: Componentes del sistema

3.3.1. Servidor

Para este proyecto, contamos con un servidor Linux en el que alojar todo nuestro proyecto. En él, contendremos toda la funcionalidad de la aplicación, el servidor web y el framework que nos permitirá realizar la comunicación entre el front y el back-end a la hora de utilizar la interfaz gráfica. Contiene también ElasticSearch y sus funcionalidades, que actúa en nuestro caso como la base de datos.

- **Lenguaje:**

  El lenguaje utilizado para este proyecto es Python. La explicación para esto es sencilla. La principal ventaja es que está diseñado para ser rápido, simple y ligero, distribuido en un solo módulo, y además con la librería estándar de Python como única dependencia. Menos tipado que Java y orientado a objetos. Nuestra dificultad con Python, pese a ser sencillo, es que nunca habíamos trabajado con él, y nos resultó

algo difícil acostumbrarnos a su sintaxis. Además, algunos módulos no son sencillos de usar e instalar y la comunicación con la interfaz web es bastante más compleja que con lenguajes como PHP o Javascript.

- **Framework:**

  Para realizar la comunicación del servidor con la interfaz web creada para este proyecto, utilizaremos el framework *Bottle* que es un micro-web framework WSGI orientado al lenguaje Python (Python, 2018).

- **Base de datos:**

  Para este proyecto usaremos a ElasticSearch como base de datos. ElasticSearch crea una base de datos vacía, y después, tras procesar todos los documentos de la colección que queramos, estos se almacenan en formato JSON, indexándolo en la base de datos. A su vez, esto se copia en un repositorio, que es una imagen del contenido de la base de datos, lo cual nos permite lanzar consultas a la misma.

3.3.2. Repositorio

En el repositorio encontramos la funcionalidad de nuestra aplicación. Contiene los métodos que acceden a ElasticSearch y realizan las consultas a la base de datos. A continuación describiremos brevemete el funcionamiento de cada uno de ellos

3.3.2.1. Búsqueda por ID

La búsqueda por similaridad se basa en el uso de la funcionalidad de Lucene ***More Like This***. Esta función requiere, primeramente, de un índice de documentos disponibles, y el documento que queremos comparar. Esto se consigue mediante el procesamiento de los datos, creando una base de datos, en principio vacía, y asignándole un índice a cada uno de los elementos de la colección.

En la búsqueda, se deben indicar los campos que se desean analizar. ***More Like This*** analiza la similitud por cada campo igual de cada documento, y despúes agrupa todos estos resultados mediante una media artimética para dar un resultado global (se realiza una media de resultados si para un paciente existe más de un informe). También es posible asignar un peso a cada campo para obtener valores más aproximados al real si se requiere.

Esto da como resultado un número que representa su grado de similitud global, *score*, respecto al caso que hemos indicado al introducir el ID. El caso a comparar seleccionado aparecerá siempre en primera posición, y a continuación los ***n*** documentos ordenados de mayor a menor según su ***score***.

3.3.2.2. Búsqueda por palabras

En caso de que no dispongamos de un ID de caso, o simplemente no nos interese buscar un historial en particular, si no algo más general, esta funcionalidad permite introducir una palabra clave y devolver los historiales en los que esa palabra sea más relevante.

Esta funcionalidad calcula, mediante la opción *cross fields*, dentro de la consulta *multi match*, si la palabra introducida está en alguno de los campos que elijamos. Es importante tener en cuenta que en caso de introducir más de una palabra, la consulta asigna el mismo peso a todas las palabras, por lo que los resultados de la frecuencia de término (TF) se ven afectados a no ser que se le apliquen diversos filtros.

3.3.2.3. Búsqueda por secciones

A la hora de tratar con historiales médicos, cobran gran importancia las diferentes secciones del mismo. Secciones tales como: motivo de ingreso, alergias, evolución, tratamiento, exploración, etc. nos ofrecen información detallada y específica encapsulada dentro de cada una de ellas. Sin embargo, cuando comparamos un historial con otro según el método anterior, como ya hemos visto en el punto anterior, se establece un media de semejanza de todos los campos, perdiéndose la encapsulación y mezclando todos los resultados en favor de un resultado global que pueda ser más rápido y sencillo de analizar. Y aquí es dónde nos surgió la siguiente pregunta:

¿Qué pasa si un campo es muy relevante pero los demás no lo son y eso hace que la media total no sea significativa? Por ello, decidimos que sería relevante llevar a cabo un análisis, no sólo del texto global, si no de todas las secciones de forma particular, de forma que el doctor pueda seleccionarlo mediante una interfaz sencilla y ajustarse mejor a sus necesidades en un momento determinado.

Para solucionar este problema, esta funcionalidad interactúa con el usuario, primeramente, preguntando qué historial médico queremos analizar, y posteriormente pidiendo a su vez, que seleccione el campo en particular que quiere analizar. Esto permite eliminar de la peticion a la base de datos todos los campos que el usuario no haya seleccionado. Como consecuencia, al hacer el análisis de similitud, ElasticSearch ignorará todos los campos excepto el seleccionado. Esto nos permite, no solo eliminar posibles falsos negativos si solo nos interesa uno de los campos del reporte, si no que facilita al usuario la posibilidad de visualizar solo aquella información que considera relevante y eliminar la tarea de ir filtrando los resultados uno a uno si resulta que el documento más similar no lo es en el campo que al usuario le interesa.

3.3.2.4. Matriz de similitud

En adición a lo anterior, y para añadir mayor utilidad para las búsquedas, creamos una matriz de similitud que compare todos los documentos con la totalidad de los documentos disponibles, creando así una matriz cuadrada, en el que las columnas y las filas son los documentos y el contenido el score que existe entre ellos.

Con esta funcionalidad crearemos una matriz de similitud de todos los documentos disponibles. Al ser una comparación de todos entre todos, la matriz será cuadrada, y sus dimensiones serán el número de documentos. A continuación, vamos recorriendo todos los documentos, y para cada uno, hallamos su similitud respecto a todos los demás de la misma colección. También, se le asignará el valor 0 a la similitud de un documento consigo mismo. Finalmente, la matriz creada, se guarda en un documento ***csv***.

Con esta mejora conseguimos que el doctor sea capaz de comparar más de un documento si así lo requiere, y ver rápidamente y de un solo vistazo de entre todos los documentos que haya introducido, cuáles y en qué grado son más similares entre ellos. Esto podría tener más utilidad que una comparación individual, por ejemplo en caso de epidemias, ya que todos los pacientes tendrán síntomas similares y así podemos darnos cuenta de que todos están afectados por la misma enfermedad.

3.3.3. Controlador

El controlador es el nexo entre el repositorio y la interfaz gráfica. Responde a eventos lanzados por el usuario en la interfaz y realiza peticiones sobre el repositorio en función de la información solicitada por el usuario. Su principal función es contener todos los métodos invocados por la interfaz y las vistas a las que se redirige tras recibir los resultados. Contiene el módulo Hisi que se encarga de crear el repositorio y lanzar el servidor web entre otras cosas.

En resumen, las principales tareas que realiza el controlador son: recibir peticiones de la interfaz gráfica, comunicar con el repositorio para comunicar una respuesta a la petición a la interfaz gráfica y lanzar el servidor web.

3.3.4. Interfaz Gráfica

Para la visualización de los datos, al tratarse de datos complejos y de difícil entendimiento para personas ajenas a la programación en general, nos vimos en la necesidad de realizar una interfaz gráfica que simplifique esta tarea y la acerque de alguna manera a los usuarios reales a los que está

dirigido, que por lo general, serán doctores o personas sin conocimientos informáticos y necesitamos que las consultas puedan ser realizadas en un entorno amigable para el usuario sin tener que recurrir al uso de la consola de comandos.

Para ello, mediante el uso de un servidor web, creamos una página utilizando las tecnologías HTML, CSS y JavaScript. (W3Schools, 2014) (Moraes, 2020) Sin embargo, se nos presentaba el problema de cómo relacionar nuestro código con la interfaz gráfica, ya que el lenguaje utilizado normalmente para las páginas es Javascript en vez de Python, que es el que nosotros utilizamos para este proyecto. La solución fue el uso de Bottle, un micro-web framework WSGI orientado al lenguaje Python, diseñado para ser simple y ligero, con la librería estándar de Python como única dependencia.

Con estas características, parecía la elección perfecta para nuestras necesidades. Aunque tiene algunas pecualiaridades que merecen ser tenidas en cuenta. Bottle no permite un acceso directo a los archivos de estilo (o los JavaScript, las imágenes...) como haríamos normalmente con las instrucción *href* de html. En su lugar, Bottle utiliza la asignación de url para hacer un bind con los métodos requeridos.

Finalmente, y teniendo en cuenta las características que hemos descrito anteriormente, creamos la interfaz en la url: https://holstein.fdi.ucm.es/tfg-hisi/ y le añadimos una serie de funciones que implementan la funciones recogidas en el repositorio automáticamente, haciéndolo todo de forma más sencilla para nuestros usuarios objetivo.

3.3.4.1. Funcionalidad de la Interfaz

La interfaz gráfica que hemos creado, contiene, por el momento, cuatro funcionalidades, que pueden ser aumentadas progresivamente en trabajos futuros. Las funcionalidades son las siguientes:

1. **Búsqueda por ID:** Realiza la búsqueda de similaridad del historial con el ID introducido por el usuario y devuelve una lista ordenada con los más parecidos junto a su score.

2. **Búsqueda por Palabras Clave:** Realiza la búsqueda de los historiales más relevantes que contengan la palabra introducida.

3. **Búsqueda por campos:** Realiza la búsqueda de similaridad del historial introducido por el ID, únicamente teniendo en cuenta el campo señalado. Esto es útil si queremos centrarnos en un solo campo en particular en vez de en la media de todos los campos del texto.

4. **Descarga de la Matriz de similitud:** Genera y descarga la matriz de similitud por ID de todos los elementos de la colección, estableciendo como 0 la similitud de un caso consigo mismo.

3.4. Ejemplo de uso

A continuación, veremos unas imagenes del aspecto de la página y comentaremos de una forma más extensa y visual la funcionalidad que hay implementada hasta este momento analizando un ejemplo de uso.

Supongamos que queremos realizar la búsqueda de similitud de un documento a través de su ID, del cual disponemos con anterioridad.

En la figura 3.4, podemos ver la portada de nuestra interfaz. Posee una barra de navegación en la que podremos incluir nuevas opciones a medida que la página vaya adquiriendo nuevas funcionalidades. También contiene una barra de búsqueda por palabras, en la que introducimos una palabra y nos devuelve los documentos más relevantes que la contengan.

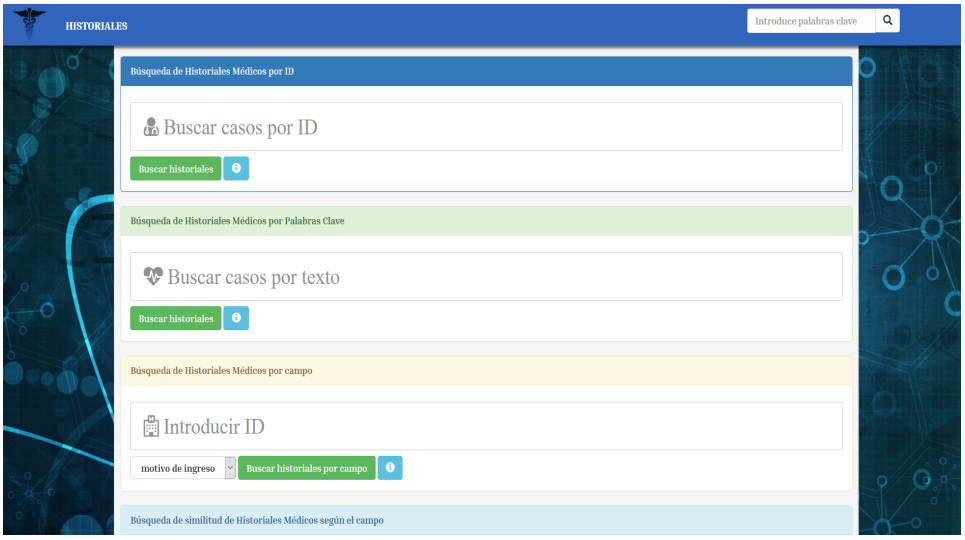

Figura 3.4: Portada de nuestra página

Después encontramos varios input o entradas de texto que suponen la funcionalidad básica de nuestra interfaz. Cada una representa una distinta y requieren la introducción de diferentes entradas, lanzando diferentes métodos. Como son: Búsqueda por palabras, por ID, por un campo, u ordenadas por cada campo. También proporcionamos la opción de descargar una matriz de similitud que contiene la información al completo de la similitud de todos los archivos de la colección.

En nuestro caso, usaremos la primera opción (figura 3.5), la cual nos permite introducir directamente el ID de un historial, en caso de que dispongamos de él y queramos buscar una aproximación más directa en lugar de buscar por palabras relevantes.

Figura 3.5: ID introducido

Una vez pulsado el botón verde, se nos mostrara un listado con los 10 historiales más similares, proporcionándonos su ID, y el score de similaridad proporcionado por ElasticSearch y la función *Buscar similares* en orden descendente (3.6).

Figura 3.6: Similaridad y Score de similitud con el ID introducido

Al pulsar pulsar en cualquiera de los historiales obtenidos, accederemos a una versión en detalle del mismo, en la cual podremos leer el documento completo (3.7).

Figura 3.7: Detalle del historial

También posee dos botones que nos permitirán o bien descargar el historial en formato ZIP o buscar de nuevo los archivos que más se le asemejen a ese historial, realizandose la misma función que en la búsqueda por ID y redirigiendo al mismo listado (3.8).

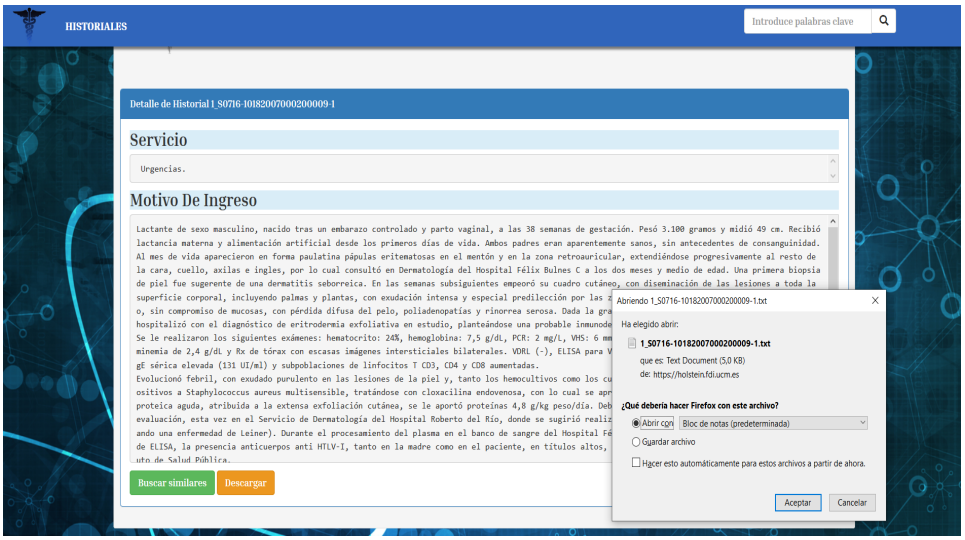

Figura 3.8: Opción de descargar el historial

Finalmente, es importante hacer notar que hemos añadido unos botones

de información que lanza una nueva ventana con información relativa al uso y funcionamiento de los inputs de la portada, los cuales pueden ayudar al usuario (3.9).

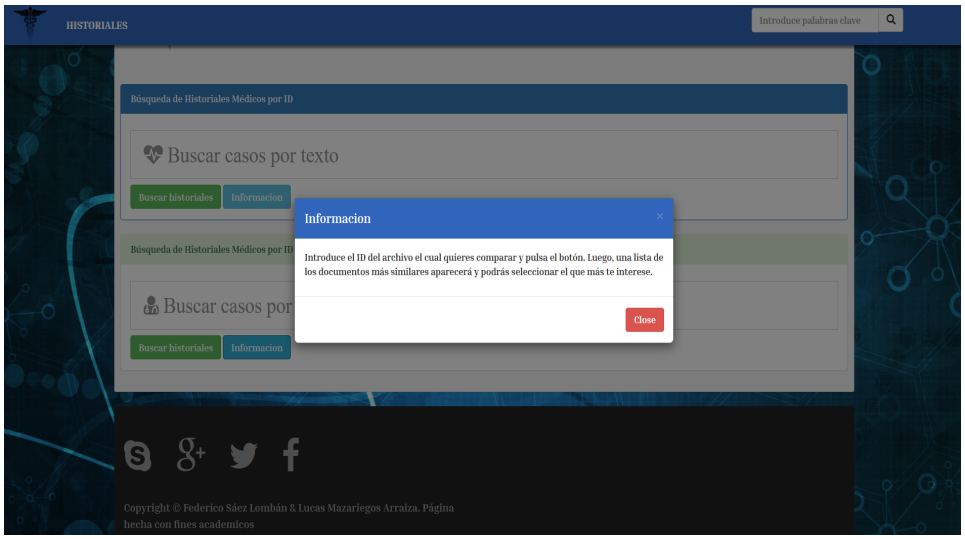

Figura 3.9: Botón de ayuda

Capítulo 4

Pruebas Realizadas

4.1. Primeros pasos

El objetivo del proyecto es obtener el grado de similitud de un historial médico dado sobre otro conjunto de historiales. Desde un primer momento tuvimos claro que nuestra aplicación tenía que ser independiente de los tipos de datos con los que trabajara, para que pudiera ser exportada y utilizada en distintas situaciones y contextos.

La única restricción que impusimos fue que los tipos de datos que se procesaran estén escritos en lenguaje natural y en español. Los diferentes formatos de texto a los que nos enfrentamos son: *.docx* y *.txt*.

Nuestros primeros pasos se dirigieron hacia realizar un análisis de los datos para su procesado. Para la realización de pruebas de la aplicación contamos con dos colecciones de datos; datos oftalmológicos (ordenados por grupos) y datos meddocan, cada una con su propio estilo y estructura que serán explicadas en el capítulo a continuación.

Era fundamental para nosotros, por otra parte, que se pudiera conmutar entre ellos de forma rápida y sencilla, es decir, que el programa debía ser independiente de la colección elegida de forma que únicamente hubiera que cambiar la ruta donde se encuentran los historiales para conseguir que funcione con cualquier colección de archivos. También fue necesario adaptar las búsquedas para las características de cada colección, principalmente adaptar los campos que requería cada historial, y adaptar el algoritmo de búsqueda para que analizara el nombre de los archivos y les asignara una etiqueta, para poder así clasificarlos. De esta forma, podemos añadir documentos al índice de forma rápida.

4.2. Historiales Médicos

A continuación se describirán los datos con los que trabajaremos en nuestra apliación.

4.2.1. Historiales MEDDOCAN

Partiremos de un dataset llamado MEDDOCAN[1] (Medical Document Anonymization Track) que contiene datos médicos ficticios para llevar a cabo una exploración de diferentes técnicas de procesamiento de texto para anonimizar los datos. Aunque nuestra tarea se basa en hacer una búsqueda de historiales similares a uno dado, hemos elegido este dataset ya que los datos presentaban un formato común para todos ellos. Aunque algunos campos en ciertos documentos no aparecen, dentro del global de todo el conjunto de datos, es una mejor opción para determinar si siguen el mismo patrón y si son aptos para trabajar con ellos.

Para la realización de dicha tarea, proporcionan un conjunto de datos de 2749 casos, en formato ".txt". Estos historiales médicos contienen datos personales, el tipo de servicio, el motivo de ingreso y el médico que lo redactó.

[1] http://temu.bsc.es/meddocan/index.php/datasets/

```
Datos del paciente.
Nombre: Luis.
Apellidos: Garrido Carrasco.
NHC: 1347609.
NASS: 28 38956749 65.
Domicilio: Calle La Cañada, 44 #3453.
Localidad/ Provincia: Valencia.
CP: 47007.
Datos asistenciales.
Fecha de nacimiento: 23/05/1998.
País de nacimiento: España.
Edad: 20 meses  Sexo: H.
Fecha de Ingreso: 23/05/2018.
Servicio: Cirugía Maxilofacial.
Médico: Jennifer Gaona Silva Carrera.  NºCol: 28 28 57031.
Paciente de 36 años que presenta desde hace 2 años masa
intraescrotal izquierda
con molestias ocasionales. En la ECO se informa como masa
intraescrotal, epidídimo
izquierdo engrosado hipoecoico y con dos nódulos.
Dirección para correspondencia:  Dra. Margarita Gimeno Aránguez
Servicio de Anatomía Patológica
HGU Gregorio Marañón  C/ Dr Esquerdo, 46  E-28007 Madrid. (España)
E-mail: mgimeno.hgugm@salud.madrid.org
```

Figura 4.1: Contenido de un historial de MEDDOCAN

Como vemos en la figura 4.1, se trata de una serie de historiales sencillos, con pocos campos que analizar y escrito de un forma muy natural. Posee poco vocabulario técnico o abreviaturas y además, está escrito de forma continua, cohesionada, sin ningún tipo de subíndice que pueda alterar los resultados. Como consecuencia, los resultados se espera que sean lo más cercanos a la realidad posible al eliminar la mayor parte de los elementos que pudieran interferir negativamente en los mismos.

4.2.2. Historiales Oftalmológicos

Esta colección de datos contiene una serie de documentos oftalmológicos organizados en 5 grupos: G1, G2, G3, G4 y G5. Cada uno de estos grupos está formado por diez casos, a excepción de G2 y G5 que tienen nueve. Haciendo la suma total, podemos definir que el total de casos del que disponemos es 48. Cada grupo a su vez, representa una especialidad oftalmológica diferente, estando organizados según ese criterio.

Como podemos comprobar en la tabla 4.2, la estructura de cada caso,

al contrario que la de los historiales MEDDOCAN, está formada por un conjunto de todos los informes que pertenecen a un mismo paciente. Es decir, dentro del caso de un paciente encontramos documentos de cada vez que ha ido a un hospital. Esto significa que hay documentos de alta de cada visita, o informes quirúrgicos dependiendo si ha tenido que pasar por quirófano o no. Esto complicará en gran medida el análisis a realizar, ya que cada visita del paciente no necesariamente tiene que ser por el mismo motivo, y al ser el resultado de similitud una media de todos los documentos presentes, van a verse influenciados por documentos que no estén relacionados.

Sin embargo, y en la medida de lo posible, los historiales de los pacientes pertenecientes al mismos grupo deberían guardar una mayor similitud, ya que son relativos a una misma enfermedad o al menos enfermedades relacionadas, vamos a realizar un modelo ideal de referencia para poder comparar los datos obtenidos con los que deberíamos obtener. Este modelo de referencia será creado según el criterio de la siguiente afirmación por lo antes mencionado:

"los resultados de la similaridad de un caso médico deberán ser mayores en aquellos que pertenezcan a su mismo grupo".

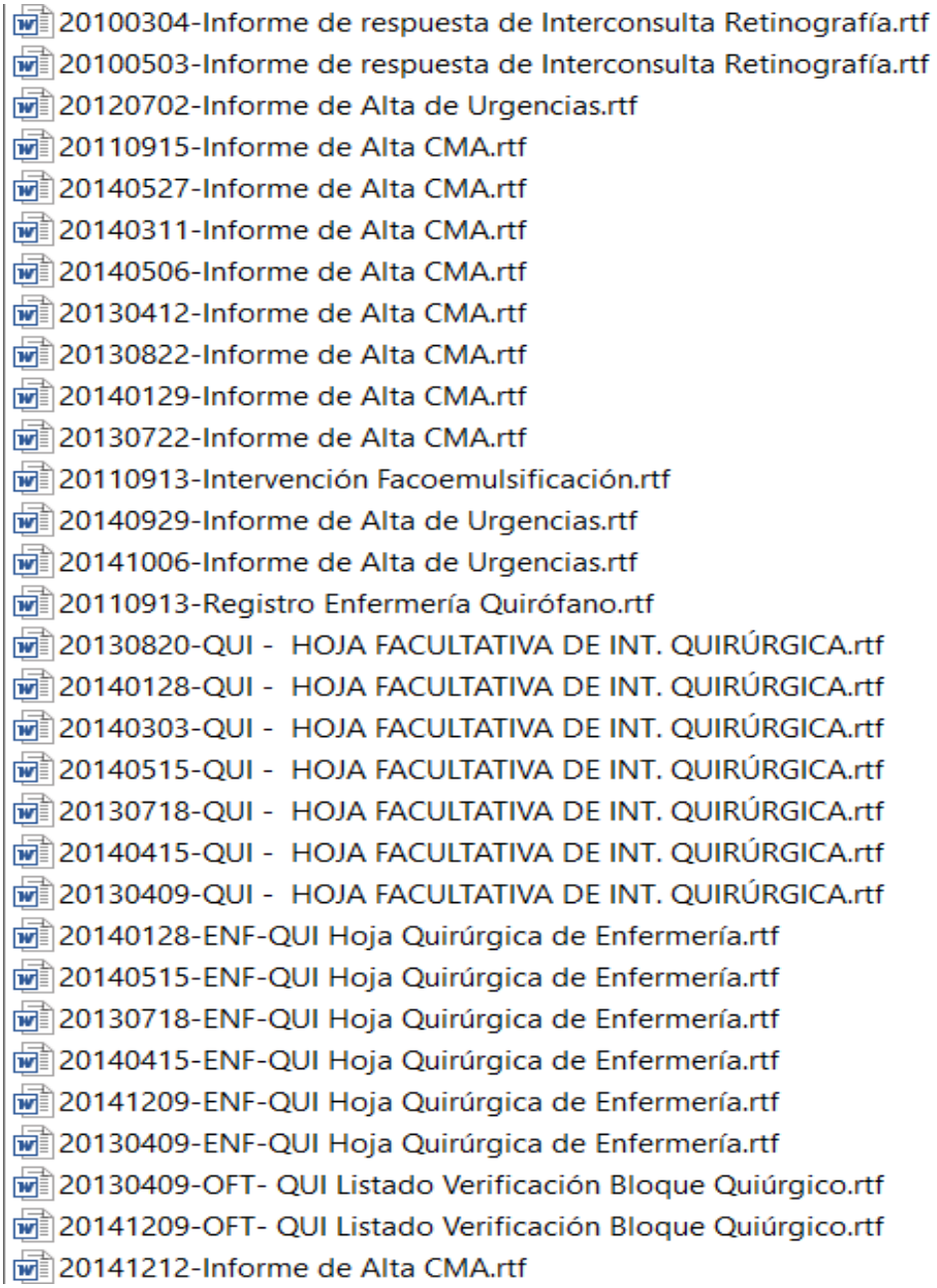

Figura 4.2: Contenido de un historial oftalmológico

Para entender la dificultad del problema vamos a exponer los problemas que encontramos en el análisis:

- Los datos no están cohesionados, es decir, no poseen la misma estructura, lo cual afecta a los resultados.

- Son datos reales e implica que están redactados por médicos. Aquí nos encontramos con errores ortográficos, erratas o nomenclaturas diferentes para un mismo término.

- A diferencia de los anteriores, estos documentos poseen un lenguaje mucho más técnico, menos natural que los de MEDDOCAN, con multitud de abreviaciones médicas y una estructura pensada para ser rápida de escribir y no sencilla de leer.

- No son uniformes. Es decir, a diferencia de MEDDOCAN, donde teníamos un solo archivo por caso médico, cada paciente posee más de un documento dependiendo de las veces que haya acudido al hospital o del tipo de consulta al que haya acudido. Esto da como resultado que puedan existir por ejemplo 4 o 5 informes de alta para un mismo paciente, cada uno con diferente información, lastrando enormemente el procesado.

- Inconsistencia. Para un mismo paciente pueden existir multitud de informes de un tipo (Informe de Alta, Informe Facultativo, Informe De Urgencias...), o por el contrario ninguno de ellos, lo cual lastra cosiderablemete cualquier búsqueda global de similitud que se quiera alcanzar, al verse afectados los resultados por un documento del mismo tipo (urgencias por ejemplo) que no sea similar en absoluto para un paciente, a pesar de que el resto de urgencias de ese paciente sí que lo sean

4.3. Procesamiento de historiales

Antes de poder analizar una colección de datos, es necesario pasar por una fase de tratamiento de los mismos. Aprovechando la abstracción que nos permite la aplicación, solo hemos de modificar levemente el archivo que procesa los historiales, de forma que se permita, de forma sencilla, evaluar diferentes colecciones de forma totalmente independiente a las mismas.

Básicamente, para cada tipo de colección de datos adaptaremos el archivo que los procesa. Utilizaremos el análisis de los datos realizado previamente para saber que campo o campos serán relevantes a la hora de comparar la similitud.

El procesamiento de los historiales es el único módulo de nuestra aplicación que no va a poder ser reutilizable. Como tenemos dos colecciones de datos diferentes, deberemos de hacer dos versiones de procesado adaptándolo a las necesidades de cada tipo de datos.

4.3.1. Historiales MEDDOCAN

Aprovechando la prácticamente uniformidad de los campos en la estructura de los diversos casos médicos, podremos decidir qué tipo de campos son relevantes para hacer diferentes experimentos.

Posteriormente realizamos un estudio sobre el total de casos (2700 aproximadamente). El estudio consiste en contar el número de campos que contienen los historiales médicos:

Campo/s	Número de registros	Porcentaje
Servicio	1340	48 %
Diagnóstico	1051	38 %
Motivo de ingreso	2301	83 %
Servicio y motivo de ingreso	804	29 %
Diagnóstico y motivo de ingreso	698	25 %

Tabla 4.1: Tabla con el número de registros que tienen cierto campo y su porcentaje.

Después de realizar, estas pruebas, decidimos realizar dos versiones diferentes del procesado, nos quedaríamos para cada versión con aquellos que tuvieran:

- Diágnostico y motivo de ingreso
- Servicio y motivo de ingreso

En ambas versiones guardaremos únicamente, para cada caso, los campos descritos arriba, eliminando así información irrelevante. Cada caso será procesado en JSON, el formato requerido por nuestra base de datos ElasticSearch. Se almacenará una copia en el servidor de los historiales procesados.

4.3.2. Historiales Oftalmológicos

Debido a la complejidad de estos datos, esta parte se antoja muy importante. Existe un archivo de configuración con formato *.yaml* en el que escribimos una serie de secciones que nos serán relevantes almacenar en el procesado. Al igual, que en MEDDOCAN, almacenaremos los historiales procesados como documentos JSON para que sean compatibles con ElasticSearch.

Se comienza leyendo por cada historial, documento a documento. Nos aseguramos de que la primera palabra/s que se encuentre en cada documento sea una sección o una parte a eliminar. El programa no guardará datos

personales, fechas ni una serie de términos que también se añaden en el archivo de configuración. Entonces, a raíz de encontrarse una palabra definida como sección, almacenará como contenido de la misma todo lo que se encuentre después hasta volver a encontrarse con otra sección.

Una parte importante de todo esto consiste en almacenar todos los posibles tipos de secciones que te puedes encontrar en los documentos de una colección. De esta forma aseguramos que el contenido que almacenemos en las mismas pertenece de manera inequívoca a dicha sección. Las secciones se pueden añadir, quitar o modificar en el archivo de configuración, de está manera resultaría fácil integrarla a posibles cambios.

Con respecto a los problemas comentados en el análisis, y al ser un volumen muy grande de documentos y diversidad, decidimos realizar otros procesados. Para ello, nos generamos nuestros propios datos quedándonos solo con: Informes de Alta, Informes de Urgencias, Hojas Facultativas e Informe de Alta de Urgencias. Esto nos dió lugar a diversas pruebas.

4.4. Ejemplos de búsquedas

4.4.1. Ejemplo de búsqueda con historiales MEDDOCAN

Para realizar la evaluación de MEDDOCAN[2] probaremos empíricamente que los resultados arrojados son correctos comprobando de manera visual que realmente un historial se parece al otro. Como el número de historiales es mucho mayor a los que creemos interesante analizar, devolveremos los 10 mejores resultados. A continuación, realizaremos un análisis de resultados para el caso ***1_S0716-10182013000500015-1*** [3] y tomaremos para el análisis el historial que proporcione un score mayor. Entonces analizaremos el texto que contienen y determinaremos si el resultado de la búsqueda ha sido existoso o no.

Desde un primer momento, las sensaciones de este experimento fueron que ha sido un éxito ya que se puede ver que analizando el ID y el del más similar (1_S0716-10182013000500015-2), ambos se identifican con la misma causa, ya que son el mismo ID cambiando el -2 del final, lo cual significa que es otra versión de la misma temática. Por ello, a pesar de pertenecer a servicios diferentes (Urgencias y Digestivo), podemos afirmar que ambos tratan el mismo tipo de problema, por lo que el experimento funciona correctamente

[2] http://temu.bsc.es/meddocan/index.php/datasets/

[3] Accesibles desde https://holstein.fdi.ucm.es/tfg-hisi/detalle/1_S0716-10182013000500015-1

analizando la temática del historial, aunque eso no significa que el texto sea el mismo.

Analizando la tabla 4.2, la cual contiene una comparación de la información más relevante contenida en cada uno de los historiales, podemos comprobar que, en efecto, se trata de casos bastante similares.

En una primera lectura vemos que el primer caso trata de un paciente que acude a urgencias por vómitos y dolores estomacales. Además detalla sus antedecentes y las pruebas realizadas y obtenidas. El caso más similar también trata sobre el ámbito digestivo, y prácticamente hablan de las mismas cosas. Además, podemos comprobar que ambos presentan exactamente los mísmos síntomas, por lo que un doctor podría usar esta información para determinar que efectivamente se trata de la misma enfermedad.

Inicial: 1_S0716-10182013000500015-1	Más parecido: 1_S0716-10182013000500015-1
de origen rural	procedente de un sector rural
vómitos	vómitos
sospecha de meningitis bacteriana aguda	meningitis bacteriana aguda
ingreso afebril	ingreso afebril
cefalea	cefalea
signos meníngeos	signos meníngeos
resultados de leucocitosis, neutrófilos y PCR	resultados de leucocitosis, neutrófilos y PCR
inició tratamiento empírico con ceftriaxona (2 g c/12 h)	iniciándose tratamiento empírico con ceftriaxona (2 g c/12 h)
tinción de Gram	tinción de Gram
agitación psicomotora	agitación psicomotora
haloperidol	haloperidol
resultados del cultivo completo	resultados del cultivo completo

Tabla 4.2: Tabla en la que comparamos términos iguales, regiones de datos sobre análisis en la que poder comparar números y frases con el mismo significado.

Para comprobar en efecto que los resultados son buenos, recurriremos a otro caso: *1_S0004-06142008000500012-1*. Analizado el texto, podemos ver que la especialidad a la que se refiere es a urología. Consideraremos entonces como acierto cada vez que uno de los historiales más similares se refieran a la misma especialidad y tengan un score de similaridad mayor de 60 para asegurarnos de que la similaridad es lo suficientemete alta como para realizar un análisis de similaridad relevante entre ellos.

ID	Score	Acierto
1_S0004-06142007000200015-1	77.48561	Sí
1_S0210-48062007000400015-1	71.05463	Sí
1_S0210-48062004000600013-1	70.9766	Sí
1_S0210-48062003000900010-1	63.899235	Sí
1_S0004-06142005000100009-1	63.548878	Sí
1_S0210-48062005000300017-3	62.334587	No
1_S0210-48062009000400019-1	61.751472	Sí
1_S0210-48062005000400016-1	61.059517	Sí
1_S0004-06142005000200004-1	60.76984	Sí
1_S0004-06142006000500016-1	60.53105	Sí

Tabla 4.3: Tabla con ejemplo de uso de MEDDOCAN

Como podemos observar en la tabla 4.3, de los 10 casos más similares, todos superiores a 60 de score, 9 se refeieren a la misma especialidad médica, urología, dándonos una gran tasa de acierto de un 90 %. Es más, teniendo en cuenta únicamente historiales que hablen de problemas renales, el acierto sería de un nada desdeñable 70 % lo cual nos muestra que en efecto, los resultados de similaridad aportados por el programa son correctos.

4.4.2. Ejemplo de búsqueda con historiales de Oftalmología

Al igual que con los datos de meddocan, realizaremos un ejemplo de uso de un caso de la colección de historiales oftalmológicos y mostraremos y analizaremos sus resultados para determinar si son correctos o no.

Primero, elegiremos un caso al azar, y comprobaremos cuáles son sus 6 historiales más similares. A continuación, analizaremos cuáles de ellos pertenecen a su mismo grupo. Como ya hemos mencionado con anterioridad, si el grupo del caso pertenece al mismo que el del caso analizado, lo contamos como un acierto. Para el caso G3_39, los resultados de similitud obtenidos son los siguientes:

ID	Score	Acierto
G3_36	24.899	Si
G3_32	23.922	Si
G3_37	19.999	Si
G3_33	19.669	Si
G5_54	16.594	No
G5_52	16.237	No

Tabla 4.4: Tabla con ejemplo de uso de oftalmología

Como podemos observar en la tabla 4.4, de 6 resultados obtenidos, los 4 más similares se encuentran en su mismo grupo, lo cual es lo que definimos como acierto a la hora de analizar los resultados. Esto nos arroja un porcentaje de acierto total del 66,6 %, lo cual, si bien no es excelente, como en los casos de meddocan, es un buen ejemplo para probar la eficacia del sistema de similitud. Estos resultados se deben, en gran parte, a la complejidad de estos datos respecto a los anteriores. Este tipo de historiales contienen una gran cantidad de informes de distinta naturaleza, y uno que no esté relacionado con los demás, lastra los resultados considerablemente.

4.5. Evaluación global

Una vez analizados los componentes de nuestro proyecto, y una vez descrita sus funcionalidades, pasaremos a realizar un estudio de las colecciones de datos de las que poseemos y evaluaremos y compararemos la similitud las mismas para llegar a una conclusión de qué set de datos es mejor para esta funcionalidad.

4.5.1. Evaluación para historiales MEDDOCAN

4.5.1.1. Método de evaluación

Para los historiales MEDDOCAN, no podemos utilizar la métrica de *F1, recall and precision* (la cúal es usada a continuación en los historiales oftalmológicos) debido a que no están ordenados por grupos, lo cual usabamos como base para realizar el análisis y lo determinábamos como acierto. Intentamos agruparlos por tipo de servicio, sin embargo, muchos historiales médicos que se parecen, no pertenencen al mismo tipo. Bien es cierto que para un número reducido de historiales si funciona bien, y podría ser aplicable pero, en el cómputo general, no. Esto se debe a que el dataset está pensado para ser anonimizado, entonces intuímos que no han prestado mucha atención en que realmente se parezca el tipo de servicio con el contenido.

A su vez, los tipos de serivicio son muy específicos, y a veces, algunos, solo aparecen una única vez, lo cual hace imposible relacionarlo con otros de su mismo tipo ya que no existe.

La forma de ver si dos historiales se parecen que hemos elegido se basa en evaluar visualmente los historiales con un score superior a 60. De esta forma podemos asegurar que hay bastante semejanza textualmente. Consideramos que a partir de un 60, los historiales tienen bastante contenido en común como para poder ser analizados posteriormente en más detalle.

4.5.1.2. Resultados con historiales MEDDOCAN

Ejecutamos la funcionalidad de buscar similares y leeremos el resultado; resaltaremos aquellas palabras o conjunto de palabras clave que coinciden en ambos textos o tienen el mismo significado. Analizaremos finalmente el número de ellas que coinciden A partir de ahí, estableceremos un número de aciertos a partir de cual, consideramos que efectivamente el documento se parece.

El objetivo final de está evaluación es la creación de una tabla de dos columnas: ID del historial a buscar e id del historial encontrado como más similar. En cada fila enfrentaremos las palabras que se hayan encontrado y su similar en el que hemos establecido como sospechoso en cuanto a similitud.

En relación a los experimentos realizados, podemos observar que normalmente, los dos o tres primeros historiales, hablan de lo mismo. Utilizan la misma terminología, temática, síntomas, etc. Pero normalmente pertenecen a tipos de servicio diferente. Es por ello que nos fiamos de estos resultados, en base a las muchas observaciones que hemos realizado. La mejor manera de comprobar si el resultado es relevante al dado, es leer el contenido, y veremos si la totalidad de las palabras o términos que son iguales son lo suficientemente útiles para el médico.

4.5.2. Evaluación para historiales de Oftalmología

En contraposición a los datos MEDDOCAN, las características de esta colección nos permite realizar una aproximación diferente en cuanto a su evaluación y realizar un método de comparación diferente que nos de otro tipo de medida y así poder analizar los resultados desde otra perspectiva.

4.5.2.1. Método de evaluación

Para evaluar nuestros resultados, y decidir así si son correctos o no, y en ese caso, averiguar en qué grado lo son, hemos recurrido a la métrica de *F1, recall and precision*, cuya definición es la siguiente:

- **Recall:** El recall o recuperación es la fracción de instancias relevantes que se han obtenido sobre la cantidad total de instancias relevantes, cuya formula es la siguiente:

  Verdaderos positivos / (Falsos negativos + Verdaderos positivos)

- **Precision:** La precisión es la fracción de todas las instancias relevantes dividido entre las instancias obtenidas, cuya fórmula es la siguiente:

  Verdaderos positivos / (Falsos positivos + Verdaderos positivos)

- **F1 score:** El Valor-F en estadística es la medida de precisión que tiene la prueba ejecutada. Se emplea en la determinación de un valor único ponderado de la precisión y el recall cuya fórmula es:

$$2 * \frac{Recall * Precision}{Recall + Precision}$$

4.5.2.2. Resultados con historiales de Oftalmología

Primeramente, utilizaremos la métrica de *F1, recall and precision*. Se considerarán como aciertos o positivos los historiales del mismo grupo que aparezcan entre los 6 resultados más similares al caso analizado y todos los demás resultados como fallos o negativos.

Utilizando estos valores hemos creado una tabla que contiene como columas el recall, la precision y el valor F1 tomando como secciones al realizar el análisis, todos los que posee cada documento, como por ejemplo: motivo de ingreso, exploración, alergias, medicación actual, evolución y tratamiento...

El valor que tendremos en cuenta a la hora de sacar conclusiones será el de F1 score, ya que realiza la media ponderada de la precision y el recall y nos da la verdadera información para relacionar los datos. Los valores obtenidos para F1 score sobre el total de campos de cada grupo son:

Grupo	Recall	Precisión	F1
G1	0,16	0,1	0,125
G2	0,23	0,18	0,19
G3	0,5	0,33	0,39
G4	0,42	0,28	0,33
G5	0,28	0,19	0,22

Tabla 4.5: Tabla de Recall-precisión utilizando todos los campos.

Son los resultados más destacables ya que en el análisis por secciones no hemos conseguido verificar alguna de las secciones del historial médico. Como podemos observar excepto G3 y G4 que rozan el 40 % de similitud, la aplicación de similitud en cualquier caso del resto de grupos arroja resultados falsos.

Con los datos obtenidos de la tabla y una validez media de un 20 % comprobamos que nuestro programa no está funcionando como debería. Es por ello que decidimos investigar el motivo de porqué está fallando.

4.5.2.3. Análisis inicial de errores en los documentos oftalmológicos

El principal y mas notorio problema de los historiales médicos es que el formato no es común para todos los casos, ni siquiera para los casos dentro de un mismo grupo, lo cual dificulta enormemente el poder sacar información útil de los mismos. Por ejemplo:

El programa analiza todos los documentos que contiene un historial médico y, cuando aparece un término asignado como 'sección', añade todo el contenido que aparece inmediatamente después hasta que se vuelve a encontrar otra 'sección'. El problema reside en qué muchos de los documentos que nos facilita el hospital tienen un formato con numerosas sub-secciones como: tablas en las que rellenar datos. Esto provoca fallos ya que el programa, al tomar esa información como relevante, observará que el contenido de las posibles tablas es igual en ambos historiales cuando en realidad es un archivo que el médico va rellenando. Hemos comprobado que dará más valor siempre a documentos que contengan esas mismas subsecciones.

4.5.2.4. Mejora en el procesamiento de los historiales oftalmológicos

Atendiendo a los fallos mencionados anteriormente, nos planteamos cómo serían los resultados de poseer todos el mismo formato exactamente y

de que sólo existiera un documento por paciente. Para ello, realizamos una mejora en el procesado de los datos. Corregimos la información contenida, eliminando subsecciones o encabezados que no fueran de utilidad para analizar los campos y dejamos únicamente la información relevante de un solo documento.

A su vez, nos dimos cuenta de que cuando se comparan dos historiales, se compara todo el contenido en conjunto. Partiendo de esta simple afirmación, nos dimos cuenta de que si añadíamos a la configuración de la aplicación un apartado llamado *stop-words*, es decir, una lista de palabras que queremos que no sean tomadas en cuenta, los resultados mejorarían ya que evitaría las palabras intermedias entre las palabras clave; como conectores, preposiciones o palabras comunes, haciendo una limpieza previa a la comparación sin modificar el contenido del historial en el repositorio y dándonos así un porcentaje de similitud mucho más cercano a la realidad.

Esta lista de palabras está principalmente formada por preposiciones o artículos que no dan ningún tipo de valor. También hemos eliminado palabras que es probable que se repitan en algunos historiales y en otros no, dando más valor a esos documentos. Estas palabras, a nuestra consideración no aportan ninguna información nueva o relevante para un caso, y podría llevar a análisis incorrectos. Por ejemplo, palabras como paciente u hospital, son esperables en un documento médico, aunque no tienen por qué estar escritas. Se entiende que todas las personas de los historiales son pacientes o han ido a un hospital, incluso si no consta en el mismo, procederemos a eliminarlas del análisis.

Existen una serie de casos que se repiten como los más similares a los demás. Todos ellos coinciden en que son los que más ficheros contienen. Partiendo de esta deducción, el grupo 3, en especial, es el que más documentos contiene y por tanto, es el grupo al que salen relacionados todos los historiales médicos. Esto se debe a varias razones, por ejemplo, estos documentos, al seguir una estructura muy parecida y forma de redactar igual, (Ejemplo: Paciente que acude para valoración y tratamiento de .... etc), repitiendo frases como esas en muchos documentos, gana mucho más puntos a la hora de calcular la similitud respecto a otros historiales pese a que es un contenido vacío que no aporta nada a la hora de analizar un texto.

Por ello, hemos utilizado únicamente un tipo de documento y un único documento, informe de Alta en particular, para minimizar el ruido proporcionado por otros documentos y por ser, además, el documento que más se repite en todos los casos.

Una vez realizados los cambios, los resultados arrojados fueron los siguientes:

Grupo	Recall	Precisión	F1
G1	0,23	0,16	0,188
G2	0,28	0,23	0,25
G3	0,58	0,62	0,599
G4	0,47	0,32	0,38
G5	0,28	0,23	0,25

Tabla 4.6: Tabla de Recall-precisión tras unificar la plantilla.

Los resultados son ligeramente superiores, aunque no los esperados. Esto pone en evidencia que, efectivamente, el formato influía en los resultados finales, pero no parece ser la principal causa de los mismos. Es por ello que buscaremos en trabajos futuros otros motivos por lo que esto puede estar sucediendo.

Capítulo 5

Conclusiones y Trabajo Futuro

5.1. Conclusiones

A la vista de los resultados obtenidos en el análisis de texto, podemos tener en cuenta varios aspectos. Primero, llegamos a la conclusión de que la forma en la que están estructurados los archivos cobra gran importancia en este tipo de análisis. Los archivos oftalmológicos, al tener diferentes formatos, generaban una gran cantidad de errores de análisis que no generaban los realizados con MEDDOCAN. El nombre de los campos de los historiales oftalmológicos estaban escritos, o de diferente manera, usando sinónimos o en diferente formato, con lo cual el sistema no los reconocía correctamente como si fueran el mismo. Algunos de los campos contienen subsecciones más específicas que ElasticSearch no reconoce como tales, de modo que tomaba esos textos como información útil para el análisis y no como lo que son, simples títulos carentes de información válida.

Es por ello, que comprobamos que muchos historiales con información, a simple vista dispar, obtienen unos valores muy altos de similitud en determinadas secciones, pese a que la información contenida en los mismos no se corresponden entre sí, simplemente porque en los casos en los que la plantilla es la misma, y al no reconocer los títulos de las subsecciones como tales, la similitud aumenta.

Tras el uso de los documentos MEDDOCAN, unos historiales médicos que comparten una misma estructura, observamos que, efectivamente, los resultados se ajustan mucho mejor a la realidad. Obteniéndose, como era de esperar, mejores resultados en aquellos historiales que comparten la misma especialidad médica o una sintomatología similar.

Esto pone de manifiesto la importancia de una cohesión interna entre los historiales de un mismo set. Si los documentos no poseen la misma estructura,

los resultados obtenidos van a ser erroneos, independientemente de cuánto se parezcan los datos entre sí, ya que el score se va a ver confundida por términos que no formen parte de la información útil, tal y como sucedió con los primeros archivos que analizamos.

Por otra parte, la sencillez de los archivos confiere mucha más seguridad a los resultados obtenidos. Mientras que los documentos oftalmológicos son de una complejidad elevada, los datos de MEDDOCAN están escritos de una forma mucho más simple, con un lenguaje más natural, y como consecuencia, el resultado del score es mucho más elevado en este set de documentos.

Finalmente, y continuando con lo mismo, el hecho de que en los documentos oftalmológicos existan más de un tipo de documento asociado al mismo paciente, y de cada tipo, más de un archivo debido a reiteradas visitas al hospital, empeora los resultados, pues no es seguro que cada visita a hospital tenga que ser por la misma dolencia, incluso si la especialidad por la que acude es la misma. Por tanto, al ser el score la media de los resultados de cada documento de forma individual, una colección con más de un documento casi siempre arrojará resultados peores que si solo existiera uno.

En conclusión, los documentos de MEDDOCAN arrojan unos valores de similitud mucho mejores que los de oftalmología debido a todos los inconvenientes de éstos previamente descritos. La simplicidad de los datos, junto a la cohesión interna, se antoja fundamental a la hora de analizarlos por texto, y una comparación uno a uno queda demostrado que es mucho mas eficiente que realizar una media de diferentes documentos del mismo tipo.

5.2. Trabajo Futuro

Es por ello que, y a la vista de lo anterior, para trabajos futuros, consideramos indispensable una unificación de la estructura de los documentos de una misma colección, de forma que estas diferencias desaparezcan. Otra opción sería realizar un tratamiento previo de los datos de manera que se extraiga únicamente la información útil de los mismos, eliminando los encabezamientos o secciones que puedan dar lugar a errores y sustituirlos por la información pura, de forma que, en este caso, se refleje más fielmente la realidad. Por otra parte, consideramos que si se va a añadir más de un documento, se hagan de forma separada a no ser que pertenezcan a la misma dolencia, ya que los resultados se ven afectados negativamente por esto.

También se podría añadir más funcionalidad a la interfaz. La aplicación tal y como está incluida es totalmente independiente de los datos introducidos, es decir, para cambiar de un data set a otro, el programa no cambia.

Sin embargo, para lanzar el servidor es necesario primero crear un repositorio y después asignarselo a la función que lo crea por consola. Esto por supuesto, no sería viable de cara a una versión final, debido a que gente sin conocimientos informáticos sería incapaz de conmutar entre colecciones de esta manera. Por lo tanto, consideramos que una buena forma de ampliar la funcionalidad de la interfaz sería permitiendo a un usuario realizar este cambio de una manera automática tan solo pulsando un botón.

Chapter 5

Conclusions and Future Work

5.1. Conclusions

Watching the obtained results in the text analysis, we can take into account various aspects. First, we conclude that the way the files are structured is vital for this type of analysis. The ophthalmological reports, because of having different formats, were generating a big amount of analysis errors that the MEDDOCAN files were not generating. Every file's fields names were whether written in a different way, using synonyms or in a different format so the program wasn't recognising them as the same. Some of the fields have subsections that ElasticSearch wasn't recognising as such. Instead, it was analysing them like useful information, not like just titles without real information.

That is the reason, we can see many reports with, obvious different information were getting high levels of similarity in some sections, even though the information contained in them was not the same. But since the structure is the same, and the information was taken wrongly, the similarity was increasing exponentially.

After analysis the MEDDOCAN files, a set of files which share the same internal structure, we could observe that, as expected, the results were adjusting way better to reality, letting us observe, better results in the reports that share the same medical branch or symptoms.

That puts into perspective the importance of an internal cohesion among the files of a same data set. If the documents don't share the same structure, our results are going to be wrong, regardless of the true similarity between them, because of useless information taken as useful, as it happened with the first data-set.

Moreover, the simplicity of the reports gives a better similarity accuracy. While the ophthalmological reports have a high complexity, the MEDDOCAN ones are written in a way simpler way, with a more natural language, and less abbreviations, and as a consequence, the similarity score is way higher in that data set.

Finally, and in line with the previous statement, the fact that the ophthalmological reports contain more than one type of file inside them associated to the same patient, and even more, more than one file of the same type due to different visits to the hospital, worsen the results because multiple visits to the same hospital doesn't they are coming for the same problem, even if the medical branch is the same. That is the reason why, being the score the mean between the scores of all the documents, a collection with more than one document will always show worse results than if only one existed.

In conclusion, the MEDDOCAN documents bring better results than the ophthalmological reports because of all the problems previously exposed. The simplicity of data and internal cohesion becomes fundamental when comparing similarities. Also, a one-to-one comparison proves itself more efficient than using the mean of several documents of the same type.

5.2. Future Work

Because of that, we consider that, for future work, it is essential a unification of the document structure, so the differences might disappear. As another option, a previous data treatment can be done, so we can extract just the useful information hidden in the files, by eliminating the section titles or other elements that can create mistakes or wrong results. We also encourage to, in case of introducing more than one document for the same patient, only introduce cases of the same illness, avoiding like this the negative effects this has on the results.

We can also implement more functionality to our graphic interface. Up to now, our application is totally collection independent, meaning you can change and use another data collection without altering the program. However, to run the server it is necessary to first create the repository and then assign it to the function that runs it. This, obviously wouldn't be possible in a final version because people without deep computer formation wouldn't be able to change the collection in this way. So, a good way to broaden this functionality could be automatizing this just by pressing a button.

Capítulo 6

Contribuciones Individuales

En esta sección analizaremos detalladamente cuáles han sido las contribuciones individuales de cada miembro al proyecto.

6.1. Lucas Mazariegos Arraiza

Al ser un trabajo realizado entre dos personas, y vistas las necesidades del proyecto que fueron identificadas desde el primer momento, quedó claro que era necesaria tanto una gran colaboración, comunicación y apoyo entre nosotros, como un reparto de tareas para agilizar el proceso. Al ser necesario un código que analizase la similitud de los documentos y una interfaz que lo implementase, desde el primer momento decidimos dividir de forma orgánica el trabajo entre front-end y back-end. Al tener más experiencia con lenguajes como HTML y CSS, decidí ser el encargado de realizarla, así como mi compañero de programar el código. Mis aportaciones podrían resumirse de la siguiente manera:

- Documentación del estado del arte de las tecnologías.
- Desarrollo y pruebas de la interfaz gráfica y sus controladores.
- Redacción de la mayor parte de la memoria y traducciones.

Antes de comenzar a realizar la interfaz, me encargué de recopilar información de diferentes métodos de análisis de texto, y de ElasticSearch en particular. ElasticSearch contenía la instrucción more like this, que nos proporcionaba la funcionalidad de similitud mediante la medida TF-IDF y nos devolvía un score con su grado de similitud.

Busqué y analicé las fórmulas matemáticas que llevaban al resultado final de score que proporcionaba ElasticSearch y decidí que era una buena medida

de los resultados que estábamos buscando y pasamos a implementarla en nuestro proyecto.

Para implementar la interfaz, primero era necesaria una manera de unir nuestro código escrito en Python con el código HTML que teníamos planeado realizar, debido a que el lenguaje comúnmente utilizado es JavaScript pero nuestro código estaba escrito en Python. Para ello, y tras mucho investigar, descubrí la existencia de la tecnología Bottle, un microservicio permitía realizar este nexo. Lo mejor de esta tecnología es que solo requería de la librería estándar de Python para poder funcionar y era sencillo de utilizar. Tras leer toda la documentación disponible, que no es demasiada y aprender a usarlo, lo implementé como controlador entre la interfaz y el código que estaba realizando mi compañero.

En cuanto a la interfaz, quería conseguir un diseño sencillo pero vistoso que implementara las funciones que habíamos creado a través de información introducida por los usuarios, anticipando cómo sería usada esta aplicación en caso de salir al mercado. Para ello, leí mucha documentación de HTML para averiguar la mejor manera de crear la opción de acceder a las funcionalidades y aprendí a usar Bootstrap 3 para darle un mejor estilo a la página y hacerla multiplataforma, permitiendo el uso de la misma en dispositivos móviles, y que realmente pareciera una aplicación que podría usarse a día de hoy en un hospital o consulta. Para asegurarme de que la página cumplía las expectativas realicé una serie de encuestas mediante una encuesta de google [1] a personas ajenas al proyecto para que me dieran sus impresiones, ideas, y si consideraban que era una interfaz intuitiva para alguien ajeno al mundo de la informática. Las impresiones fueron, en general, buenas. Leí los comentarios recibidos y realizé algunos cambios en consecuencia. También es importante resaltar que tuve que investigar cómo lanzar el servidor web para que recogiera mi código HTML y cómo conseguir que éste quedara fijo en la url tras finalizar la sesión en el servidor.

A su vez, me encargué la mayor parte de la redacción de esta memoria, aprendiendo cómo funcionaba la herramienta Latex que decidimos usar para redactarla y sus diferentes funcionalidades, aunque sin desmerecer el inestimable aporte de Federico en algunos apartados del capítulo 4 referentes a su parte del trabajo, como la evaluación de resultados de los historiales y la corrección de errores. También realicé las traducciones a inglés de la introducción, la conclusión y el resumen.

Finalmente, respecto a la parte realizada por mi compañero, le ofrecí mi apoyo a la hora de realizar el recall y la precisión de los historiales de oftalmología como nuestros tutores nos propusieron, creando el script que

[1] https://docs.google.com/forms/d/18PgGHb0XVsT3Zj_JbcHvXc9QrLfKsJ1Tg1adiY__qCg

devolvía la función F1, y realicé la unificación de campos de los informes de alta en los historiales oftalmológicos como parte del preprocesado pensado para mejorar los resultados de búsqueda.

6.2. Federico Sáez Lombán

Debido a que hemos realizado este proyecto dos personas, ha sido necesaria una estrecha comunicación con el fin de llevar a cabo la finalización del mismo.

Comenzamos con una fase de análisis, que en relación a las reuniones mantenidas con los directores, establecimos una serie de objetivos a realizar en el proyecto. Estos son los siguientes: desarrollo de código para definir la similitud entre casos, creación de una interfaz web, evaluación de resultados y desarrollo de la memoria.

A continuación, empezamos a pensar cómo llevarlas a cabo y cúal sería la mejor forma de distribuirnos el trabajo o, qué tareas podríamos realizar en paralelo. Dada mi experiencia con python y la de Lucas con front-end, decidimos que yo me encargaría del desarrollo de código y el de la interfaz web. Esto nos llevó a tener que trabajar muchas veces juntos o vía Skype para mejorar la productividad. Dado que el contenido de los historiales médicos con los que trabajabamos contiene información confidencial, teníamos que trabajar en remoto, accediendo a un servidor de la universidad que contenía una máquina de Linux.

Después de una fase previa de entender el código existente, decidí ir haciendo pequeñas modificaciones, con el fin de preparar el programa a los cambios futuros. Una tarea importante fue hacer el estudio de las posibilidades que nos ofrecía ElasticSearch sobre cómo devolver la similitud entre sus objetos en la base de datos. Es aquí dónde comienza la fase experimental haciendo pruebas, ajustes y modificaciones a la similaridad entre historiales médicos. Para la realización de estas pruebas generé unos scripts que ejecutaban todos los casos y mostraban la similitud y su grupo correspondiente, el que determina si es acierto o fallo. A su vez, también generé scripts para la evaluación que devuelve el porcentaje de éxito. Estos scripts sirven de apoyo a la experimentación y no están en la interfaz web.

Primeramente, empecé a trabajar con los datos oftalmológicos. Los primeros datos fueron bastante regulares, por lo que empecé a plantear posibles soluciones. Una de ellas era dividir los experimentos según: tipo de documento, único archivo por historial médico y análisis por sección.

Para el análisis por tipo de documento creé un conjunto de datos que contuviera solo: informes de alta o informes de urgencias o hojas facultativas.

Después de comprobar que había casos que no contenían ninguna de las anteriores y otros, muchísimos, decidí unificar el experimento analizando solo informes de alta. El experimento mejoró algo los resultados, pero no los esperados, así que dimos un paso más allá. En una tarea conjunta con Lucas, estructuramos los historiales medicos y dejamos únicamente un archivo por historial médico, evitando así un historial con 8 archivos, otro con 1 y otro con ninguno. El experimento siguió mejorando. Después añadí en la configuración de ElasticSearch una lista de palabras llamada 'stopWords'que se encargaba de no evaluar dichas palabras a la hora de la similitud. Después de más experimentos, mejoró ligeramente los resultados.

A continuación voy a detallar la evaluación. En consenso con los directores, decidimos utilizar la métrica de F1, recall y precision. A raíz del script creado, era muy fácil adaptar la evaluación al tipo de experimento, por lo que tenía que cambiar poco código para evaluar un nuevo experimento.

Los directores del proyecto nos propusieron empezar a trabajar con otros datos y ver como reaccionaba la aplicación. Desde el principio hemos tenido una máxima y es que nuestra aplicación debía de ser independiente de los datos. Tuve que cambiar algunos ajustes y parametrizar variables para conseguirlo. Me encargué de realizar el script de procesamiento, ajustado a los nuevos datos. Una vez procesados los datos, continuamos con la evaluación. Para sorpresa, en las primeras pruebas, los resultados eran bastante optimistas, ya que el grado de similitud era bastante superior a los obtenidos con los datos anteriores, procedí a inspeccionar los casos y comprobar que de verdad se parecían, obteniendo resultados bastante buenos.

Además, me encargué de generar una matriz de similaridad para todos los casos existentes. El objetivo de esto era unificarla con otra matriz de otro proyecto relacionado con los historiales médicos. Dentro de esta funcionalidad había un script que integraba ambas matrices, permitiendo asignar un peso a los datos de una y otro peso a los datos de otra. Finalmente no se decidió llevar a cabo, dada la inconsistencia de los datos. En la interfaz, para aprovechar la funcionalidad hecha, decidimos añadir una opción que generara la matriz y permitiera su descarga.

Finalmente, para el desarrollo de la memoria, hemos colaborado repartiéndonos diferentes índices.

Apéndice A

Apéndice

- En nuestro proyecto, a la hora de documentar toda la información relativa a nuestras aportaciones y adjuntarla a nuestro Github, pasamos por varias opciones, primero optamos por usar el editor de texto del bloc de notas, pero tras no estar conformes con el resultado, decidimos usar los editores de texto más estándar de github, destacando entre ellos, Markdown y Restrutured Text, finalmente decidiéndonos por el primero, por ser un modelo más estándar y más sencillo de usar.
- La función crear del makefile es la que se encarga de crear el repositorio para los historiales, y de asignarles un numero que pueda ser facilmente reconocido por el programa a la hora de realizar la matriz.

INSTALACIÓN DE ELASTICSEARCH EN EL SERVIDOR

Pasos de instalación:

1. Descargar e instalar la última versión disponible. Requisisto indispensable tener Java instalado. sudo apt-get update
2. Descargaremos la última versión a través del enlace:

   https://artifacts.elastic.co/downloads/elasticsearch/ wget y elegir la versión .deb
3. Instalamos con la siguiente intrucción:
   - *sudo dpkg -i elasticsearch-2.x.y.deb*
4. Se deberá haber instalado en: */usr/share/elasticsearch*

   cuya configuración se ubicará en */etc/elasticsearch*
   y su script añadido en */etc/init.d/elasticsearch*

5. Para asegurar que Elasticsearch arranque y apague automáticamente con el servidor usamos las siguientes instrucciones:
    - *Sudo systemctl enable elasticsearch.service*
    - *sudo systemctl start/active*

LANZAMIENTO DEL SERVIDOR WEB

Para lanzar nuestra aplicación web y que pueda ser accedida y probada, hay que realizar varios pasos:

1. Crear un repositorio. Se crea a partir del makefile, con la instrucción *make crear*
2. una vez creado, guardamos la dirección en la que está alojado nuestro repositorio
3. escribimos el siguiente código *./hisi <path/al/repo>servir –host=0.0.0.0 –port=5000 –prefix=tfg-hisi*

CÓMO CAMBIAR DE COLECCIÓN DE DATOS

Ya que nuestro repositorio es de tipo git, trabajamos con una rama para cada colección de datos.

Para cambiar de rama, habría que entrar en el servidor y posicionarse en la carpeta del proyecto, que es el directorio git:

- *git checkout meddocan* –si estuvieramos en los historiales oftalmológicos

A continuación habría que crear el repositorio y lanzar la web con:

- *make crear* –crea el repositorio
- Para lanzar el servicio web está detallado justo arriba.

Bibliografía

Y así, del mucho leer y del poco dormir,
se le secó el celebro de manera que vino
a perder el juicio.

Miguel de Cervantes Saavedra

Asensio, D. P. *Sistema de clasificación de historias cínicas con deep learning*. UCM, 2018.

Cuervo, V. *¿Qué es Elasticsearch?*. Arquitectoit.com, 2019.

ElasticSearch. *ElasticSearch, getting Started*. `https://www.elastic.co/es/webinars/getting-started-elasticsearch?elektra=home&storm=banner`, 2014.

González, P. S. *Características de la arquitectura de Elasticsearch*. Open-Webimars, 2019.

Moraes, F. *HTML For Beginners The Easy Way: Start Learning HTML and CSS*. `https://html.com`, 2020.

N. Perez, A. D. P., M. Serrax. *Vicomtech at MEDDOCAN:Medical Document Anonymization*. Vimcotech, 2009.

Python, R. *Developing with bottle*. `https://realpython.com/developing-with-bottle-part-1/`, 2018.

bruno Stecanella. *what is TF-IDF*. `https://monkeylearn.com/blog/what-is-tf-idf/`, 2019.

W3Schools. *Bootstrap 3. Css and HTML*. `https://www.w3schools.com/bootstrap/`, 2014.

Wikipedia. *TF-IDF*. `https://es.wikipedia.org/wiki/Tf-idf`, 2019.

–¿Qué te parece desto, Sancho? – Dijo Don Quijote –
Bien podrán los encantadores quitarme la ventura,
pero el esfuerzo y el ánimo, será imposible.

Segunda parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes

–Buena está – dijo Sancho –; fírmela vuestra merced.
–No es menester firmarla – dijo Don Quijote–,
sino solamente poner mi rúbrica.

Primera parte del Ingenioso Caballero
Don Quijote de la Mancha
Miguel de Cervantes