UNIVERSIDAD COMPLUTENSE
MADRID

Facultad de Informática de la Universidad Complutense
Departamento de Ingeniería del Software e Inteligencia Artificial

PROYECTO DE FIN DE CARRERA
CURSO 2011/2012
INGENIERÍA INFORMÁTICA

Procesador automático de informes médicos

Autores:

Enrique Bautista Barahona
Ignacio Salcedo Ramos
Alberto Ureña Herradón

Directores:

Alberto Díaz Esteban
Laura Plaza Morales

Fecha: 28 de junio de 2012

Enrique Bautista Barahona
Ignacio Salcedo Ramos
Alberto Ureña Herradón
Alumnos de Sistemas Informáticos, autores de este documento y
del proyecto «*Procesador automático de informes médicos*»
Ingeniería Informática
Universidad Complutense de Madrid

En Madrid, a 28 de junio de 2012

Enrique Bautista Barahona Ignacio Salcedo Ramos

Alberto Ureña Herradón

Alberto Díaz Esteban
Profesor Contratado Doctor
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

Laura Plaza Morales
Becaria del programa de Formación de Profesorado Universitario
Departamento de Ingeniería del Software e Inteligencia Artificial
Universidad Complutense de Madrid

CERTIFICAN: Que el proyecto titulado «*Procesador automático de informes médicos*» ha sido realizado por Enrique Bautista Barahona, Ignacio Salcedo Ramos y Alberto Ureña Herradón bajo nuestra dirección y constituye su Proyecto de Fin de Carrera de Ingeniería Informática.

En Madrid, a 28 de junio de 2012

Alberto Díaz Esteban
Director del proyecto

Laura Plaza Morales
Directora del proyecto

Resumen:

El acceso a la información y su intercambio es vital en el ámbito médico, tanto en la investigación como en la gestión hospitalaria. Gran parte de esta información está contenida en informes médicos escritos en lenguaje natural y, por tanto, no es fácilmente tratable por sistemas automáticos.

Esta memoria describe el proyecto de fin de carrera «Procesador automático de informes médicos», cuya finalidad es la creación de un sistema de detección de conceptos y términos médicos, representados mediante SNOMED CT, una terminología clínica de referencia. Además, y previamente a dicha extracción de conceptos, se realizan tareas de corrección ortográfica, detección y desambiguación de acrónimos y detección de negaciones.

Para la construcción de esta serie de fases, se han aplicado técnicas de procesamiento de lenguaje natural a informes médicos en castellano. Esto supone un reto, dado que la mayoría del trabajo realizado en este campo se ha realizado para lengua inglesa y los recursos para el español son bastante limitados.

Todo esto se integra en una herramienta que sirve para procesar automáticamente informes médicos y generar una representación conceptual de su contenido, útil para la gestión de dichos informes en el ámbito clínico-sanitario.

Adicionalmente, se han construido dos sistemas auxiliares para medir la eficacia de la aplicación que permiten etiquetar manualmente informes para construir un corpus de informes anotados y usar dicho corpus para evaluar los resultados del procesamiento automático.

Palabras clave:

procesamiento de lenguaje natural, informes médicos, corrección ortográfica, desambiguación de acrónimos, detección de negación, detección de conceptos, SNOMED CT

Abstract:

Accessing to and exchanging information is vital in medical settings, be it in research or in healthcare management. Most of this information is contained in clinical reports written in natural language free text and, therefore, it cannot be easily processed by automatic systems.

This document describes our final degree project, "*Procesador automático de informes médicos*", and its objective, which is the creation of a medical concept extraction system that maps texts to SNOMED CT (a standard reference terminology). Moreover, to prepare the text for the concept detection, several other tasks are performed: spelling correction, acronym detection and disambiguation, and negation detection.

In order to build the different parts of the application, we have applied natural language processing techniques to clinical reports in Spanish. This poses a challenge, given that most of the work done in this field deals with texts in English and the available resources are rather limited.

The previously described tasks are implemented in a software that automatically process medical texts, generates a conceptual representation from their contents and serves as an example of a useful application to manage clinical reports in healthcare and research settings.

Furthermore, we have built two auxiliary systems to measure the effectiveness of our tool, which allow to manually tag reports to build an annotated corpus and to use such corpus to evaluate the results of the automatic processing.

Keywords:

natural language processing, medical reports, spelling correction, acronym expansion, negation detection, concept extraction, SNOMED CT

Índice general

Índice de figuras

Índice de tablas

Índice de listados de código

Capítulo 1

Introducción

1.1. Motivación

La necesidad de tener un buen sistema informático de gestión de informes médicos está ganando mucha importancia en la actualidad. Los datos contenidos en estos informes suelen ser de vital importancia y a menudo se comparten entre diferentes profesionales del sector sanitario.

La información presente en dichos documentos suele ser de muy diversa índole: historial del paciente, dosis suministradas, prescripciones de medicamentos, etc. Muchas veces esta información aparece de manera poco estructurada y, en su mayoría, en forma de texto libre o lenguaje natural. Debido a la complejidad del procesamiento del lenguaje natural (PLN), el tratamiento e intercambio de esta información no es trivial, sobre todo si se consideran grandes cantidades de textos.

Nuestro principal reto, por tanto, es desarrollar técnicas de extracción de información y PLN para poder gestionar eficazmente la información contenida en informes médicos. Consideramos que se trata de un desafío por la complejidad intrínseca del PLN (derivada de la ambiguedad del lenguaje) y, como se detalla en la sección 1.1.1, también por nuestra decisión de trabajar con informes médicos en español.

Mediante la identificación de conceptos clínicos y su traducción a una representación estructurada y canónica, se facilitaría considerablemente su uso, ya sea por personal sanitario e investigadores o por sistemas informáticos. Con ello se mejoraría enormemente la gestión, clasificación y uso de los informes para diversos fines como

pueden ser su búsqueda y recuperación para comparar diagnósticos y tratamientos o su clasificación automática de acuerdo a estándares internacionales, como pueden ser CIE (Clasificación Internacional de Enfermedades) o SNOMED CT (*Systematized Nomenclature of Medicine – Clinical Terms*).

En este sentido, cabe destacar una serie de fases de procesamiento concretas que resultan claves al tratar con informes médicos. Son las siguientes:

Corrección ortográfica del informe

Un informe médico a menudo contiene faltas de ortografía. Para cualquier procesado posterior se necesita reducir al mínimo posible estas erratas. Al aparecer en el texto numerosos términos médicos y acrónimos, un corrector habitual no es capaz de tratarlos.

Desambiguación o expansión de acrónimos

Esta tarea consiste en detectar acrónimos en el texto y encontrar su acepción o significado, lo cual no es trivial dado que un acrónimo usado por un médico puede no tener una única acepción. Esta fase es necesaria para poder extraer posteriormente información semántica del informe.

Identificación de conceptos

En esta fase se materializa el fin último de la aplicación, el cual consiste en identificar y localizar los conceptos médicos que aparezcan en el informe. Para que esta información extraída sea de utilidad, estos conceptos deben pertenecer a una colección de terminología amplia, rigurosa y que pueda servir de referencia en el ámbito clínico y sanitario. Para ello, hemos elegido la base de conocimiento médico *SNOMED CT*.

Detección de la negación

Dado que el objetivo último al procesar el informe es detectar conceptos médicos en el mismo, resulta de utilidad saber si dichos conceptos están siendo afirmados o negados, puesto que no es lo mismo decir «el paciente no presenta fiebre» que «el paciente presenta fiebre». Para ello, esta fase se encarga de detectar en el texto del informe aquellas frases que están negadas y el ámbito o alcance de la negación.

1.1.1. El reto del idioma

Por último, hay que recordar que, aunque actualmente existen algunas herramientas para tratar con informes médicos en inglés (para alguna o varias de las tareas anteriores), nuestro proyecto se centra en informes médicos en español. Este hecho aumenta la dificultad e interés del proyecto, ya que en castellano las tecnologías de procesamiento del lenguaje natural están menos desarrolladas, existen menos herramientas o éstas deben ser adaptadas. Estas cuestiones se tratarán más a fondo en las siguientes secciones de la memoria, en especial en el apartado 2 (*Estado de la cuestión*).

1.1.2. Ejemplo de la tarea

A continuación se presenta un pequeño ejemplo simplificado para aclarar qué se quiere conseguir en las diversas fases del procesamiento.

Para el siguiente fragmento de un informe:
«*El paciente ingresó con fiebre y signs de deshidratacion. Llega a planta con REG. ACR : sin arritmias.*»
El sistema nos devolvería los siguientes resultados:

Correcciones:

signs → signos;
deshidratacion → deshidratación.

Acrónimos:

REG → «Regular estado general»;
ACR → «Auscultación cardiorespiratoria».

Conceptos:

Hallazgo - Fiebre (*afirmado*);
Hallazgo - Deshidratación (*afirmado*);
Condición - Estado general regular (*afirmado*);
Procedimiento - Auscultación cardiorespiratoria (*afirmado*);
Hallazgo - Arritmias (*negado*).

1.2. Objetivos

A continuación se exponen los objetivos del proyecto.

El objetivo principal consiste en el desarrollo de una herramienta para el análisis de informes médicos y la obtención de una representación conceptual de su contenido, distinguiendo lo negado de lo afirmado. Esta aplicación deberá constituir también un ejemplo de herramienta que sea usable en un ámbito médico o administrativo para la extracción de información de los informes.

El sistema ha de contar con una serie de módulos que respondan a las fases que se han comentado en al apartado *Motivación*. Son las siguientes:

- Corrección ortográfica.
- Expansión y desambiguación de acrónimos.
- Identificación de conceptos.
- Detección de la negación.

Se concretarán las funcionalidades de dicho sistema en el apartado 3 (*Funcionamiento general*).

Adicionalmente, para mejorar la flexibilidad y utilidad del sistema, este deberá ser configurable en la medida de lo posible, permitiendo trabajar con diferentes recursos o modificar el comportamiento de los algoritmos.

Para la tarea de construir estos módulos, realizaremos un análisis de las herramientas y algoritmos existentes para el PLN. Esto supone buscar herramientas existentes y ver cómo se pueden aplicar o adaptar para su uso con textos médicos. Dichas herramientas, además, deben estar hechas para el castellano o, al menos, se deben poder adaptar a nuestra lengua.

El último objetivo del proyecto consiste en que las herramientas y algoritmos empleados deben ser evaluables, es decir, necesitamos obtener medidas objetivas sobre su eficacia y corrección. Esto es siempre muy importante a la hora de desarrollar una herramienta que trabaje sobre algo tan heterogéneo como es el lenguaje natural. Para esto, la aplicación incluirá un subsistema capaz de anotar informes médicos con resultados correctos y otro subsistema capaz de comparar los resultados del análisis con los resultados correctos previamente anotados.

1.3. Estructura de la memoria

La memoria se estructura como a continuación se detalla. Primero, se hace un repaso de la tecnología existente sobre lenguaje natural en los campos que nos interesan para el procesamiento de informes. Este será el capítulo 2, *Estado de la cuestión*.

A continuación, se describen las funcionalidades de la herramienta desarrollada en el capítulo 3, *Funcionamiento general*, para posteriormente detallar las fases del procesamiento en las que se divide la aplicación en los capítulos *Fase de corrección ortográfica*, *Fase de expansión de acrónimos*, *Fase de detección de negación* y *Fase de identificación de conceptos*.

Después, se hará un estudio de la eficacia de la aplicación en el apartado 8, *Evaluación*.

Por último, en el apartado 9, *Conclusiones y trabajo futuro*, realizaremos una reflexión sobre lo conseguido en el proyecto y qué partes sería más interesante mejorar.

Se adjuntan además como anexos un manual de usuario de la aplicación, explicaciones sobre los archivos de configuración de la misma, un comentario sobre los formatos de entrada y salida y un glosario.

Capítulo 2

Estado de la cuestión

Antes de comenzar con el desarrollo de la aplicación se ha procedido a estudiar el estado de la cuestión que nos ocupa: el procesamiento de lenguaje natural y su aplicación al ámbito de los informes médicos. Con este estudio se pretende conocer las técnicas empleadas actualmente en este campo y determinar si alguna de las herramientas y librerías disponibles en la actualidad pueden ser de utilidad en el desarrollo del proyecto.

Asimismo, se han examinado dos módulos proporcionados por los directores del proyecto y desarrollados por el grupo NIL[1] de la Universidad Complutense de Madrid, al que pertenecen. Dichos módulos son parte del trabajo realizado en el proyecto AutoIndexer[2], que tenía como objetivo la investigación y el desarrollo de metodologías y recursos para el procesamiento de documentos clínicos. El proyecto se realizó bajo el programa AVANZA I+D del Ministerio de Industria, Comercio y Turismo, en colaboración con la empresa Indizen.

2.1. Corrección ortográfica

La mayoría de programas de corrección ortográfica analizan el texto a corregir palabra por palabra, sin tener en cuenta el contexto en el que aparece cada vocablo. Este análisis consiste en una búsqueda en un diccionario. En caso de no encontrar

[1] http://nil.fdi.ucm.es/
[2] http://nil.fdi.ucm.es/index.php?q=node/471

ninguna coincidencia, el programa considera que la palabra no está escrita correctamente, obtiene una serie de términos similares al analizado y los presenta como posibles soluciones del error.

La diferencia entre unos correctores ortográficos y otros suele residir en el método que emplean para decidir qué palabras sugerir como soluciones. Las técnicas usadas van desde la comparación entre cadenas de caracteres, al uso de reglas gramaticales más o menos complejas, pasando por comparaciones de la fonética de las palabras o la distancia en el teclado de las letras utilizadas.

2.1.1. Correctores ortográficos

A continuación, se presenta un resumen de las capacidades de algunas de las herramientas de corrección ortográfica con más difusión en la actualidad. Después, se concluye explicando cuál de ellas se integrará en la aplicación y por qué.

2.1.1.1. Ispell

Ispell[3] es un corrector ortográfico para Unix disponible bajo una licencia propia de código abierto. Originalmente construido para el inglés, actualmente cuenta con soporte para la mayoría de idiomas europeos y algunas lenguas del sur de África y del sureste de Asia.

Es uno de los correctores más antiguos y es la base de la mayoría de correctores que han surgido después. Todos sus sucesores incorporan su mecanismo de creación de sugerencias que, pese a ser simple, suele dar resultados razonablemente buenos. Cuando detecta una palabra mal escrita, solo sugiere términos que se encuentren a una distancia Damerau-Levenshtein (Damerau [1964]) de 1. Esto quiere decir que las sugerencias se obtienen realizando una sola operación sobre la palabra original, siendo las operaciones posibles el borrado, la adición y la modificación de una letra, el intercambio de dos letras y la adición de un espacio o guión.

[3] http://www.lasr.cs.ucla.edu/geoff/ispell.html

2.1.1.2. GNU Aspell

Aspell[4] es un corrector ortográfico basado en Ispell y creado con el propósito de reemplazarlo. Está escrito en C++ y se puede utilizar bajo la licencia LGPL. Proporciona mejores resultados que Ispell para el inglés gracias a la adición de un sistema de comparación fonética. En el diccionario español, sin embargo, no se adjuntan las reglas fonéticas necesarias para usar dicho sistema, aunque se pueden añadir. Además, presenta las siguientes mejoras: uso simultáneo de varios diccionarios, más facilidad de uso para codificación UTF8 y uso optimizado de diccionarios personales con varios procesos.

2.1.1.3. MySpell

MySpell[5] es un corrector ortográfico similar a Ispell, disponible bajo una licencia BSD. Fue desarrollado en C++ para ser incluido en la suite ofimática de OpenOffice.org. En su sitio web se informa explícitamente de que su funcionalidad es inferior a la de Ispell y Aspell.

También existe una implementación en Java independiente del proyecto original: JMySpell[6].

2.1.1.4. Hunspell

Hunspell[7] es un corrector ortográfico y analizador morfológico diseñado para lenguajes con una morfología y composición de palabras complejos (fue creado originalmente para el húngaro). No obstante, esto no quiere decir que sea utilizado solo con lenguajes de estas características. Desarrollado en C++, está disponible bajo licencias GPL, LGPL y MPL.

Es una herramienta popular, utilizada actualmente en numerosas aplicaciones de éxito entre las que se encuentran Google Chrome, Mozilla Firefox, Mozilla Thunderbird, las suites ofimáticas de OpenOffice.org y LibreOffice y el sistema operativo

[4] http://aspell.net/
[5] https://code.google.com/a/apache-extras.org/p/ooo-myspell/
[6] http://kenai.com/projects/jmyspell
[7] http://hunspell.sourceforge.net/

Mac OS X de Apple. Es debido a su gran popularidad que existe una gran variedad de diccionarios disponibles para Hunspell.

Los algoritmos que utiliza están basados en los de Ispell y Aspell y, de nuevo, en el diccionario español no se adjuntan las reglas fonéticas, pero se pueden añadir. Además, para mejorar el mecanismo de creación de sugerencias, permite utilizar reglas y n-gramas.

2.1.2. Métricas de comparación de cadenas

Las métricas de comparación de cadenas son métricas que miden lo parecidas que son dos cadenas de caracteres (o, en otras palabras, la distancia que hay entre ellas).

Dado el trabajo que se pretende desarrollar en este proyecto, creemos que este tipo de métricas pueden resultar útiles, por ejemplo, para puntuar las sugerencias de un corrector ortográfico.

2.1.2.1. Distancia de edición

La distancia de edición es el coste mínimo resultante de aplicar las operaciones necesarias para transformar una cadena en otra, perteneciendo estas operaciones a un conjunto de operaciones permitidas y teniendo un coste y unos efectos concretos. Variando las operaciones que forman parte de este conjunto y sus costes, se han definido varias métricas diferentes.

Distancia de Levenshtein

La distancia de Levenshtein (Levenshtein [1966]) es la distancia de edición más común. Su conjunto de operaciones válidas está formado por la inserción, el borrado y la substitución de un carácter. Todas tienen un coste de 1, por lo que el coste total es igual al número de operaciones aplicadas, p. ej. la distancia de Levenshtein entre «repostero» y «costeras» es 5, ya que son indispensables al menos 5 operaciones para transformar una en la otra:

1. **re**postero - postero (2 borrados)
2. post**e**r**o** - cost**e**r**a** (2 sustituciones)

3. costera - costera**s** (1 inserción)

Similares a la distancia de Levenshtein, existen otras métricas de coste 1, que se diferencian simplemente por el conjunto de operaciones que permiten realizar sobre las cadenas, p. ej.:

- La distancia de Hamming (Hamming [1950]), precursora de la de Levenshtein, únicamente permite la sustitución de caracteres.

- La distancia de Damerau-Levenshtein (Damerau [1964]) permite las mismas operaciones que la de Levenshtein y, además, la trasposición de dos caracteres adyacentes.

Algoritmo Needleman-Wunsch

El algoritmo Needleman-Wunsch (Needleman and Wunsch [1970]) surgió en el campo de la bioinformática para comparar secuencias de ADN, ARN o proteínas, pero también es usado en el campo del procesamiento del lenguaje natural.

Trata el problema del alineamiento global de secuencias, que consiste en alinear cada elemento de una secuencia con el mismo elemento en la otra secuencia, sin modificar el orden de los elementos. Está permitido añadir elementos vacíos (huecos) y alinear elementos que no sean iguales si es preciso, lo que sucederá cuando una secuencia no sea una subcadena o subsecuencia, respectivamente, de la otra. Ha sido demostrado que este problema es equivalente a la minimización de la distancia de edición (Sellers [1974]).

Para calcular la similitud de las secuencias, se hace uso de una matriz de similitud, que contiene para cada par de elementos alineados una puntuación distinta, y de una puntuación para los huecos. Por tanto, la diferencia con las distancias de edición expuestas anteriormente reside en los costes de las operaciones. Concretamente, los costes de inserción/borrado pueden ser distintos a los de sustitución, y a su vez, los costes de sustitución de cada par de caracteres del lenguaje considerado pueden también ser distintos.

2.1.2.2. Algoritmos fonéticos

Los algoritmos fonéticos son algoritmos que traducen una palabra a un código que la represente, de acuerdo a ciertas reglas basadas en las normas de pronunciación de un lenguaje concreto. Después, estos códigos se pueden comparar para determinar la similitud de las palabras originales.

La mayoría de algoritmos fonéticos se desarrollaron con el propósito de traducir nombres y apellidos. No obstante, también existe algún algoritmo creado con el objetivo de traducir cualquier palabra, siendo los más conocidos las diferentes versiones del algoritmo Metaphone (Philips [1990]).

En cuanto al idioma, originalmente fueron desarrollados para el inglés y, con el tiempo, fueron mejorados con soporte para extranjerismos. Existen adaptaciones a otros idiomas, aunque no tienen tanta difusión.

2.1.3. Conclusiones

Antes de continuar, se ha de decir que los textos que se van a tratar provienen de informes médicos y, por tanto, contienen una alta cantidad de términos específicos del ámbito clínico. En general, hacen uso de una jerga y unas expresiones recurrentes y las faltas de ortografía presentes en estos textos suelen ser leves. Dicho esto, se podría considerar que usando un diccionario extenso, con gran cantidad de terminología médica, cualquiera de los correctores analizados proporcionaría resultados decentes.

Finalmente, hemos decidido utilizar Hunspell, dado su diseño enfocado a lenguajes morfológicamente complejos y su capacidad para trabajar con reglas y n-gramas. Pese a que ahora no vamos a utilizar todo lo que nos ofrece, consideramos que es una funcionalidad conveniente para la aplicación, que podría ser utilizada en un futuro.

Hay pruebas que sugieren que Aspell proporciona mejores resultados para el inglés[8], pero estos resultados no son directamente extrapolables a otros idiomas, ya que dependen de la calidad de los recursos. En cuanto a las reglas fonéticas, tanto Aspell como Hunspell son capaces de aplicarlas, pero ninguno de los dos las

[8] http://aspell.net/test/cur/

proporcionan con sus diccionarios españoles. Por tanto, en ese aspecto se encuentran igualados.

Cabe decir que para conectar la librería Hunspell con la aplicación partimos del módulo de corrección de AutoIndexer, ya que sirve precisamente de envoltorio de Hunspell, simplificando el acceso desde código Java a la funcionalidad que este ofrece.

Por último, en cuanto a las métricas de comparación estudiadas, se ha decidido que se utilizarán para puntuar las sugerencias del corrector. Concretamente, se empleará una combinación de ellas para construir un sistema de puntuación que tenga en cuenta la fonética y la distancia de los caracteres en el teclado, además de la distancia de edición.

2.2. Expansión de acrónimos

Los sistemas de expansión automática de acrónimos tratan de resolver dos problemas diferentes: la detección y la desambiguación de acrónimos.

La detección es el proceso mediante el cual se localizan los acrónimos presentes en el texto que se está procesando. La desambiguación es el mecanismo que determina cuál es la expansión adecuada de cada acrónimo detectado según el contexto en el que se encuentra.

2.2.1. Sistemas de detección de acrónimos

Los sistemas de detección de acrónimos emplean lexicones, métodos basados en patrones, aprendizaje máquina o combinaciones de estos. A continuación, se presenta un breve resumen de cada técnica y algún ejemplo de su uso en sistemas reales.

2.2.1.1. Uso de lexicón de acrónimos

Este método es el más sencillo de implementar de todos. Consiste en dividir el texto a procesar en palabras y buscar cada palabra en un lexicón de acrónimos. Las palabras

que estén en el lexicón serán clasificadas como acrónimos. El proyecto AutoIndexer[9] utiliza este método.

2.2.1.2. Métodos basados en patrones

Estos métodos detectarán una palabra como acrónimo cuando la palabra cumpla una serie de patrones (que todas sus letras sean mayúsculas o que su longitud sea menor de cuatro caracteres, son ejemplos de posibles patrones).

Acronym Finder Program

Es el sistema pionero en la detección automática de siglas (Taghva and Gilbreth [1999]). Es una herramienta que combina el uso de patrones con el algoritmo de la mayor subsecuencia común para hallar todas las alineaciones posibles entre cada candidato a sigla y su forma expandida.

Los patrones que utiliza son los siguientes:

- Todas las letras de las palabras son mayúsculas.
- Las palabras tienen desde tres hasta diez caracteres.
- Cada carácter de la sigla debe coincidir con el primer carácter de cada palabra de la forma expandida.

Para evaluar la herramienta se utilizó un corpus de 17 documentos. El sistema identificó de forma correcta 398 siglas, lo que supuso una precisión del 98 %.

Three Letter Acronym

Es un sistema que tiene el principio general de que una palabra es una sigla si cumple con ciertos patrones y además se encuentra cerca de una forma expandida coincidente con las siglas (Larkey et al. [2000]). El sistema utiliza cuatro algoritmos diferentes para detectar siglas.

Algunos de los patrones que utiliza son los siguientes:

[9] http://nil.fdi.ucm.es/index.php?q=node/471

- Todas las letras de las palabras son mayúsculas.
- La palabra tiene un punto entre cada par de caracteres.
- La palabra está compuesta por al menos tres letras mayúsculas seguidas de una secuencia de letras minúsculas, pudiendo terminar con una o dos letras mayúsculas (COGSNet o AChemS son ejemplos de siglas que cumplen este patrón).
- La palabra está compuesta por letras mayúsculas y tiene algún dígito en cualquier posición de la palabra.
- La palabra puede contener algún espacio, siempre que vaya precedido por una letra mayúscula.
- La palabra está compuesta por letras mayúsculas y tiene barras o guiones en cualquier posición.

Para la evaluación del sistema se utilizó un corpus de 936 550 páginas web de instituciones militares y gubernamentales de los Estados Unidos. Los cuatro algoritmos fallaron en la detección de tan solo 16 casos.

2.2.1.3. Métodos basados en algoritmos de aprendizaje máquina

Los algoritmos de aprendizaje máquina permiten al sistema aprender a reconocer y clasificar acrónimos mediante ejemplos, atributos y valores. Son capaces de mejorar con la experiencia.

A supervised learning approach to acronym identification

Este sistema propone un sistema de detección de siglas basado en aprendizaje supervisado (Nadeau and Turney [2005]).

El algoritmo que utiliza convierte en vectores los candidatos a sigla detectados. Cada vector se compone de 17 características que describen cada palabra. Para determinar si la palabra es una sigla, el algoritmo la confronta con un corpus anotado y según el resultado de dicha comparación la palabra será clasificada como sigla o no.

Entre las características que utiliza para clasificar a los candidatos se encuentran las siguientes: el número de letras mayúsculas, la longitud, el número de dígitos o el número de letras.

Los precisión del sistema es de un 92.5 % utilizando la técnica de clasificación *Support Vector Machine* implementada en la *suite* de aprendizaje máquina WEKA con el algoritmo *Sequential Minimal Optimization*.

Acronym Recognition: Recognizing acronyms in Swedish texts

La finalidad de este sistema es el reconocimiento de siglas en textos biomédicos escritos en sueco (Dannélls [2006]). Es similar al sistema anterior aunque utiliza tan solo 10 características para clasificar cada candidato.

En los mejores resultados obtenidos a la hora de evaluar el sistema se reconocieron correctamente el 98.9 % de las siglas.

2.2.2. Sistemas de desambiguación de acrónimos

Los sistemas de desambiguación de acrónimos suelen utilizar algoritmos de aprendizaje máquina.

En primer lugar, buscan cada una de las posibles expansiones en el texto y, en caso de encontrar una, desambiguan el acrónimo con esa expansión. En caso de no encontrar ninguna, procesan el contexto del acrónimo comparándolo con los ejemplos de entrenamiento y, según el resultado obtenido, etiquetan el acrónimo con una u otra expansión.

Polyfind

Pustejovsky et al. desarrollaron un algoritmo de aprendizaje automático llamado Polyfind para la desambiguación de siglas (Pustejovsky et al. [2004]).

Para computar la medida de similitud entre los contextos de la búsqueda y cada uno de los contextos de entrenamiento se utilizó el modelo de espacio vectorial.

En la evaluación que se realizó, Polyfind alcanzó 97.2 % de exactitud en la desambiguación.

Automatic resolution of ambiguous abbreviations in biomedical texts

Yu implementó un sistema para desambiguar las siglas de los *abstracts* de la base de datos MEDLINE (Yu [2003]). Este sistema se basa en el uso de *Support Vector Machines* y en la hipótesis de que todas las ocurrencias de una misma sigla dentro de un *abstract* tienen la misma expansión. Los SVM utilizan un corpus etiquetado como corpus de entrenamiento en el que cada sigla está representada por un vector. Los vectores están compuestos por cada una de las palabras presentes en el contexto de la sigla.

En la evaluación, este sistema alcanzó una precisión del 87 %.

2.2.3. Conclusiones

Nuestro sistema parte del proyecto AutoIndexer, en el cual se utiliza un lexicón de acrónimos para la detección y un sistema de reglas para la desambiguación.

Al disponer de parte de los recursos que utiliza AutoIndexer, decidimos utilizar la misma técnica y uno de sus lexicones para detectar los acrónimos. Para la tarea de la desambiguación, decidimos implementar un algoritmo de aprendizaje automático para utilizarlo junto con el sistema de reglas de AutoIndexer. Dado que no disponemos de un corpus de informes o artículos médicos anotados donde obtener los ejemplos de entrenamiento para el algoritmo, decidimos utilizar las descripciones de SNOMED CT. En estas descripciones aparecen algunas de las formas expandidas de los acrónimos presentes en los recursos de AutoIndexer. De dichas descripciones obtendremos los contextos de las formas expandidas, trasformándolos en vectores para usarlos como ejemplos de entrenamiento.

2.3. Detección de la negación

La tarea de la detección de la negación consiste en detectar qué frases están negadas, así como el alcance o ámbito de la negación, i.e. las partes de estas frases cuyo sentido o significado está negado.

2.3.1. Algoritmos para la detección de negación

Para el análisis de la negación existen diferentes maneras de proceder, pudiendo distinguir tres tipos principales de algoritmos.

Algoritmos basados en análisis sintáctico

Estos algoritmos procesan las frases mediante un analizador morfo-sintáctico o *parser*. Después, calculan el ámbito de la negación basándose en las dependencias y funciones de las partes de la oración (Carrillo de Albornoz et al. [2012], Ballesteros et al. [2012]).

El problema de este enfoque es que requiere que los *parsers* estén entrenados para cierto tipo de textos y, como ya se ha comentado, no disponemos de un corpus de informes médicos en castellano.

Algoritmos basados en expresiones regulares

Estos son los algoritmos con mayor éxito en la actualidad. Concretamente, NegEx aparece nombrado en diversos artículos y trabajos sobre procesamiento de informes médicos (Chapman et al. [2001], Meystre and Haug [2006]). Este ha sido el método que nosotros hemos elegido como punto de partida, dado que, aunque el algoritmo original se construyó para el inglés, ya se han hecho adaptaciones a otros idiomas, como el sueco (Skeppstedt [2011]). En el capítulo 6, *Detección de negación*, se profundizará en este algoritmo y cómo se ha adaptado al castellano.

Algoritmos basados en gramáticas independientes del contexto

El principal exponente de estos algoritmos es *Negfinder*. Dicho algoritmo se vale de un analizador léxico, así como de un analizador sintáctico basado en una gramática LALR(1) para detectar las negaciones (Cruz Díaz et al. [2010]). Algunos estudios dan a este algoritmo una eficacia superior a los basados en expresiones regulares, pero una vez más, no contamos con una versión para el español, lo cual es una desventaja considerable que nos hace decantarnos por NegEx.

2.3.2. Conclusiones

Dada la inexistencia de herramientas disponibles para la detección de la negación en castellano, sumada al hecho de que solo se puede dedicar parte del esfuerzo a este módulo, se optó por adaptar NegEx para trabajar con textos en castellano.

Otros algoritmos más sofisticados y costosos de implementar no suponen una gran diferencia de eficacia con respecto a NegEx. Además, nuestro objetivo final es identificar qué conceptos aparecen negados en un tipo de texto en particular: informes médicos. Dada la naturaleza basada en patrones de NegEx, consideramos que éste es fácilmente adaptable a dicho tipo de textos.

2.4. Identificación de conceptos

Para la construcción de un sistema de identificación de conceptos, es preciso tener en cuenta tanto los algoritmos a utilizar como la base de conocimiento con la que trabajará la aplicación.

En esta sección, se presenta en primer lugar un breve resumen de algunas de las terminologías médicas más desarrolladas. Después, dado el gran parecido de las técnicas que usan los sistemas de identificación de conceptos, a modo de ejemplo se comenta exclusivamente el funcionamiento de la herramienta MetaMap.

2.4.1. Bases de conocimiento o terminologías

El punto más importante al hablar de la identificación de conceptos son las bases de conocimiento y colecciones de vocabulario médico que existen en la actualidad.

2.4.1.1. El Metatesauro UMLS

UMLS (*Unified Medical Language System*) es un compendio de vocabulario biomédico y aplicaciones informáticas. Proporciona una estructura de relaciones entre diferentes terminologías, de modo que se pueda traducir conceptos de una base de conocimiento médica a otra.

Se le llama *Metatesauro* a la base de conocimiento principal de UMLS. Un tesauro es un conjunto de palabras o términos agrupados, en el que las palabras están relacionadas por sinónimos. El Metatesauro comprende a su vez varias bases de conocimiento y permite la traducción entre ellas. En total contiene más de un millón de conceptos biomédicos. A destacar de entre sus vocabularios o terminologías: CIE-10 y SNOMED CT.

2.4.1.2. SNOMED CT

SNOMED CT (*Systematized Nomenclature of Medicine – Clinical Terms*) es una colección de términos médicos organizados sistemáticamente. Incluye definiciones, términos, relaciones y sinónimos sobre enfermedades, procedimientos clínicos, microorganismos, síntomas, sustancias y otros conceptos.

Existen otras bases de datos sobre conceptos médicos como CIE (Clasificación internacional de enfermedades), pero SNOMED CT es considerada como la mayor y más precisa colección de terminología (codificada) en la actualidad.

Además, mediante tablas de referencias cruzadas se pueden hallar equivalencias entre conceptos SNOMED CT y otras colecciones de términos como CIE. SNOMED CT incluye diversas tablas y subconjuntos diferentes, así como versiones para diferentes lenguas

Por todos estos motivos, elegimos esta base de conocimiento para la representación conceptual a obtener como salida de nuestra aplicación. En el capítulo 7, se explicará a detalle el uso que se hace de SNOMED CT por parte de nuestra aplicación.

2.4.2. MetaMap

MetaMap (Aronson [2001]) es una herramienta con funcionalidad muy similar a la que pretendemos desarrollar en nuestro módulo identificador de conceptos, desarrollada por el *Lister Hill National Center for Biomedical Communications* de la *National Library of Medicine* de Estados Unidos para textos de lengua inglesa.

Dado un texto médico de entrada, MetaMap encuentra y devuelve conceptos pertenecientes al Metatesauro UMLS. Para ello, en su primera versión se siguen en términos generales los siguientes pasos.

- En primer lugar, se utiliza un *parser* para extraer del texto los sintagmas nominales y asignar etiquetas sintácticas (sustantivo, verbo, etc.) a las palabras.
- Después, para cada fragmento de texto se obtiene el conjunto de todos los sinónimos, acrónimos, abreviaturas, palabras de la misma familia y combinaciones de estas generadas a partir de las palabras del fragmento original. La información sobre las variantes está precomputada y almacenada para mejorar la eficiencia.
- A continuación, se buscan en la base de conocimiento las variantes generadas. Las búsquedas se realizan a través de índices creados especialmente para aumentar el rendimiento.
- Por último, se puntúa la similitud del fragmento de texto original con los conceptos resultado y combinaciones de ellos, eligiendo finalmente el concepto (o la combinación) con mejor puntuación.

2.4.3. Conclusiones

Para construir el módulo de identificación de conceptos, hemos decidido partir de cero. La razón es que no hemos encontrado ningún sistema que respete las siguientes restricciones:

- no se puede utilizar un enfoque estadístico o de aprendizaje, ya que el corpus de informes disponible no sobrepasa los 10 documentos, i.e. es demasiado pequeño.
- se debe usar la colección de terminología SNOMED CT.
- se deben procesar informes en español.

Capítulo 3

Funcionamiento general

El procesamiento de informes médicos es la meta principal de este proyecto y, por extensión, de la aplicación. El objetivo de este capítulo es proporcionar una visión general de su funcionamiento que sirva como introducción al contenido presentado en el resto de este documento. Es decir, no debe verse como una descripción detallada del procesado que se realiza sobre cada informe, sino como una guía de alto nivel del mismo.

En los siguientes capítulos de este documento se explican con más detenimiento las particularidades de cada una de las fases del procesamiento.

3.1. Corrección ortográfica

La fase de corrección ortográfica es la primera de las fases del procesado. Su objetivo es detectar palabras mal escritas en el texto y corregirlas automáticamente.

Al comienzo de la fase de corrección, se recibe una estructura de datos que contiene todo el texto a procesar, dividido en cuantas secciones tuviera el informe. Este texto se trocea en tokens y, a continuación, se procesa cada uno de ellos para determinar si necesita corrección y, en tal caso, corregirlo.

Para realizar la corrección, el sistema produce una serie de sugerencias, que después puntúa, con el objetivo de determinar cuál es la sugerencia más adecuada. En el caso de que ninguna de ellas se considere apropiada, se dejan todas como posibles soluciones, pero no se aplica ninguna de ellas, i.e. se mantiene la palabra mal escrita.

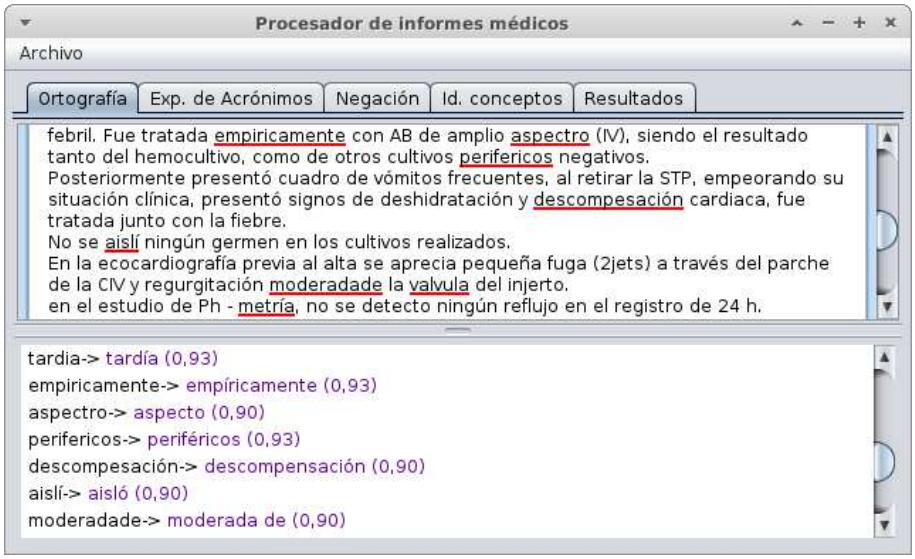

Figura 3.1: Vista de la aplicación tras la corrección ortográfica

En la figura 3.1 se muestra el resultado de aplicar la fase de corrección sobre un informe. En la parte superior se puede observar un fragmento del texto, donde las palabras subrayadas son las que se han identificado como faltas de ortografía. En la parte inferior se encuentra la lista de errores encontrados y las sugerencias de corrección. Se adjunta además la puntuación obtenida por cada sugerencia en el algoritmo de puntuación.

En el listado 3.1 se muestra un detalle de la representación interna del informe, con una de las anotaciones resultantes de aplicar la corrección ortográfica.

```
<frase>
  <br/>
  TSH y T4
  <correccion>
     <error>nirmales</error>
     <corr>normales</corr>
     <punt>0.9</punt>
  </correccion>
  .
</frase>
```

Listado de código 3.1: Representación interna de resultados de corrección ortográfica

3.2. Expansión de acrónimos

La expansión de un acrónimo es el intercambio de un acrónimo por las palabras que hacen explícito su significado. Se trata de una tarea compleja, ya que habitualmente un mismo acrónimo o abreviatura se corresponde con más de una expansión.

El objetivo de esta fase es detectar los acrónimos que hay en el texto y, mediante un proceso de desambiguación, elegir la expansión más adecuada para cada uno de ellos según el contexto en el que se encuentren.

Al comienzo de la fase se recibe el texto del informe dividido en secciones y, además, anotado con el resultado de la fase anterior. Este texto se trocea en tokens y, a continuación, se procesa cada uno de ellos para determinar si es un acrónimo y, en tal caso, expandirlo, para lo cual el sistema recupera de su base de datos sus expansiones conocidas.

Si solo se encuentra una expansión, se aplica y se termina el procesamiento, pero si se encuentran más es necesario elegir la más adecuada. Para ello, se busca en primer lugar alguna regla que indique explícitamente la expansión a realizar, dado el acrónimo y su contexto. En caso de no encontrar ninguna, se elige la expansión correspondiente al contexto más parecido al del token que está siendo procesado.

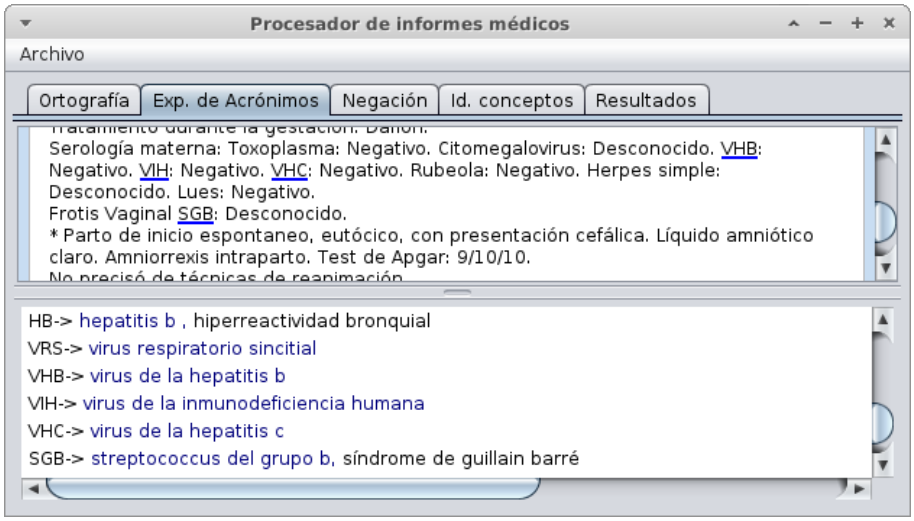

Figura 3.2: Vista de la aplicación tras la expansión de acrónimos

En la figura 3.2 se muestra el resultado de aplicar la fase de expansión sobre un informe. En la parte superior se puede observar un fragmento del texto, donde las

palabras subrayadas son las que se han identificado como acrónimos. En la parte inferior se encuentra la lista de acrónimos encontrados y las sugerencias de expansión. La primera expansión (escrita en un color más claro) es la elegida por la aplicación.

En el listado 3.2 se muestra un detalle de la representación interna del informe, con las anotaciones resultantes de aplicar la corrección ortográfica y la expansión de acrónimos.

```
<frase>
  <br/>
  <acronimo exp=" hormona tiroestimulante">
     TSH
  </acronimo>
  y
  <acronimo exp=" símbolo de la tiroxina">
     T4
  </acronimo>
  <correccion>
     <error>nirmales</error>
     <corr>normales</corr>
     <punt>0.9</punt>
  </correccion>
  .
</frase>
```

Listado de código 3.2: Representación interna de resultados de expansión de acrónimos

3.3. Detección de negación

Esta fase se encarga de detectar qué frases dentro del texto del informe son negativas o están negadas. También identifica el conjunto de tokens responsables de la negación en la frase, además del ámbito o alcance de la misma.

De nuevo, la entrada de la fase es el texto original dividido en secciones y anotado con los resultados de las fases anteriores. El contenido de cada sección se divide en frases y para cada una de ellas se comprueba si contiene alguna palabra o expresión

que denote negación y, por extensión, si la frase está negada o no. El ámbito de la negación depende de la señal de negación encontrada.

En la figura 3.3 se muestra el resultado de aplicar la detección de negación sobre un informe. En la parte superior se observa un fragmento del texto, donde las partes subrayadas son ámbitos de negaciones. En la parte inferior se encuentra la lista de negaciones encontradas. Para cada una de ellas se muestra el ámbito (*scope*) y la señal de negación (*cue*).

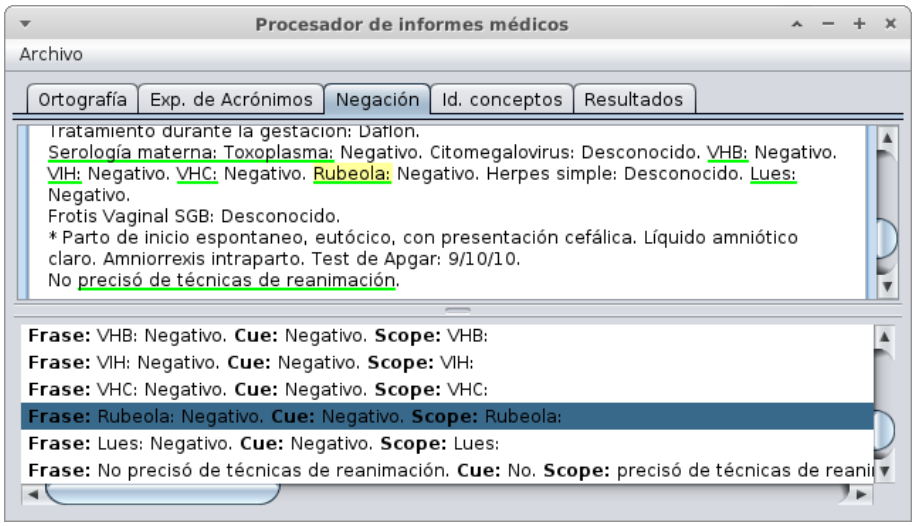

Figura 3.3: Vista de la aplicación tras la detección de negación

En el listado 3.3 se muestra un detalle de la representación interna del informe, con las anotaciones resultantes de aplicar la detección de negación y las fases anteriores.

```
<frase>
  <br/>
  <negacion cue="negativo">
     <acronimo exp=" virus respiratorio sincitial">
        VRS
     </acronimo>
     :
  </negacion>
  negativo .
</frase>
```

Listado de código 3.3: Representación interna de resultados de detección de negación

3.4. Identificación de conceptos

En esta última fase el objetivo es detectar y localizar en el texto conceptos o términos médicos. Dichos conceptos objetivo están contenidos en la base de datos de la aplicación. Para aumentar el rendimiento de las consultas se hace uso de la librería de recuperación de información *Apache Lucene*.

De la misma manera que en el resto de partes del procesamiento, se recibe el informe original dividido en secciones y anotado con los resultados de las fases anteriores. El contenido de cada sección se divide en frases y estas a su vez se dividen en unidades más pequeñas, considerando separadores las comas, las disyunciones y otros nexos.

Para cada una de estas «subfrases» se realiza una consulta a través de Lucene, que devolverá los conceptos para los cuales se han encontrado coincidencias. Finalmente, estos resultados se puntúan para elegir los conceptos más precisos y coherentes.

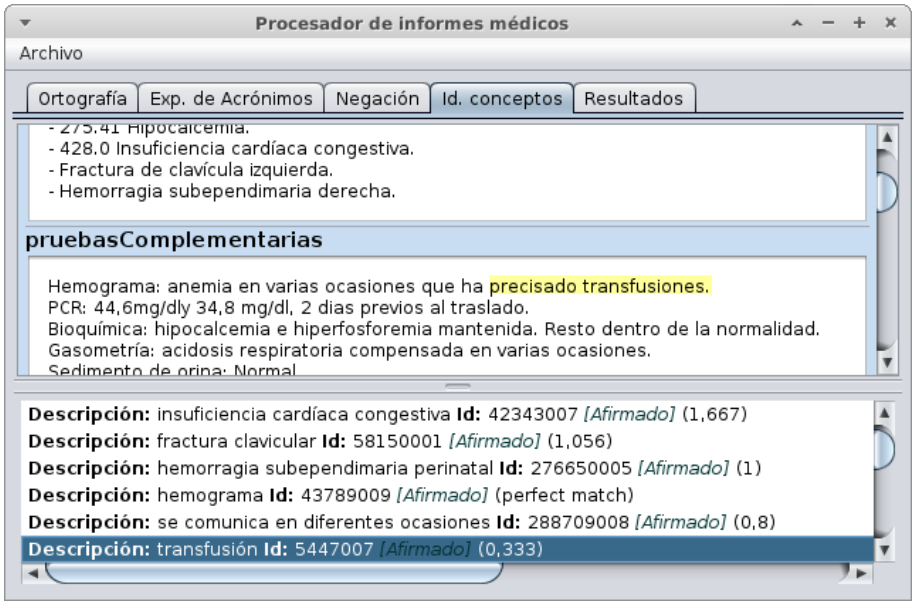

Figura 3.4: Vista de la aplicación tras la identificación de conceptos

En la figura 3.4 se muestra el resultado de aplicar la fase de identificación de conceptos sobre un informe. En la parte superior se puede observar un fragmento del texto. En la parte inferior se encuentra la lista de conceptos encontrados, con información adicional, como el identificador del concepto en la colección de términos SNOMED CT y si está negado o afirmado.

En el listado 3.4 se muestra un detalle de la representación interna del informe, con las anotaciones resultantes de aplicar la identificación de conceptos y las fases anteriores.

```
<frase>
  <br/>
  <concepto id="225386006">
     En la ecocardiografía previa al alta se aprecia
  </concepto>
  pequeña fuga ( 2jets ) a través del parche de la
  <acronimo exp=" comunicación interventricular">
     CIV
  </acronimo>
  y
  <concepto id="287312007">
     regurgitación
     <correccion>
        <error>moderadade</error>
        <corr>moderada de</corr>
        <punt>0.9</punt>
     </correccion>
     la
     <correccion>
        <error>valvula</error>
        <corr>válvula</corr>
        <punt>0.9333333333333333</punt>
     </correccion>
     del injerto
  </concepto>
  .
</frase>
```

Listado de código 3.4: Representación interna de resultados de identificación de conceptos

Además de este tipo de representación, se produce también otro tipo de salida que contiene únicamente información sobre los conceptos. De esta manera se facilita la

carga de los resultados en otros sistemas, además de omitir los datos de carácter personal que pueda contener el texto del informe.

En el listado 3.5 se presenta un detalle de la representación interna de los resultados finales.

```
<resultado>
  <id>62944002</id>
  <frec>1</frec>
  <desc>Virus de la hepatitis C (organismo)</desc>
  <seccion>antecedentes</seccion>
  <tipo>410607006</tipo>
  <estado>negado</estado>
</resultado>
```

Listado de código 3.5: Representación interna de resultados finales

En la figura 3.5 se muestra la vista de resultados, en la que se presenta la información obtenida en forma de tabla y se ofrece al usuario la posibilidad de ordenar y filtrar las filas.

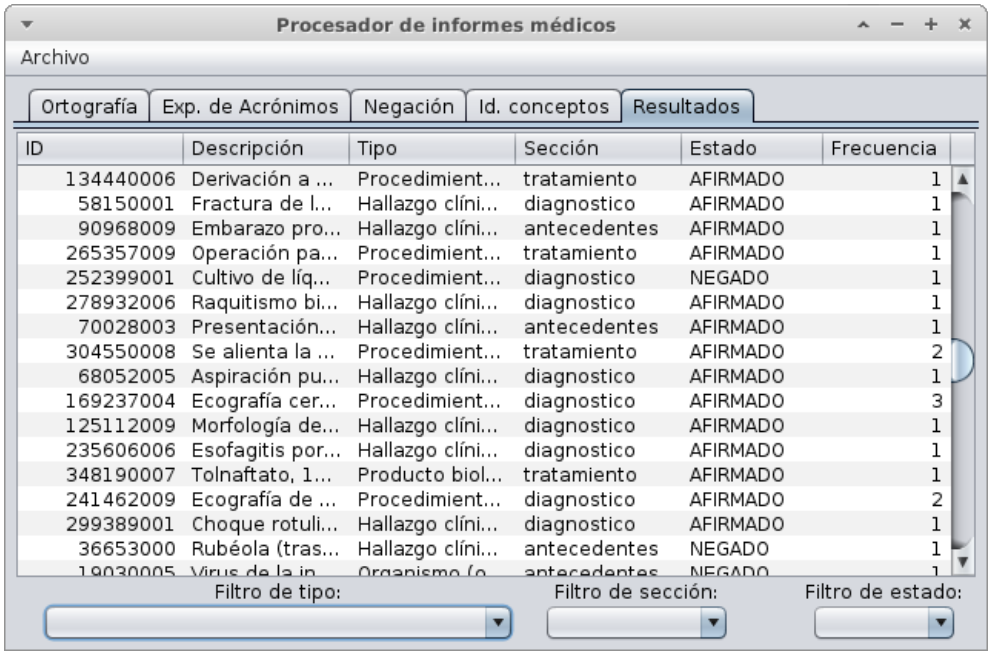

Figura 3.5: Vista de la aplicación mostrando los resultados finales

3.5. Anotación y evaluación

Para poder evaluar la eficacia de la aplicación se han desarrollado dos modos de uso adicionales.

3.5.1. Evaluación

El evaluador permite comparar informes anotados por la aplicación con informes anotados a mano por un usuario experto con el objetivo de dar una medida de la eficacia del sistema. Para ello se utilizan principalmente las métricas de *precision*, *recall* y F_1 *score*.

En la figura 3.6 se muestra la interfaz del evaluador. En la parte superior se muestra el texto de los dos informes: el procesado por la aplicación (izquierda) y el anotado por el usuario (derecha). En la parte inferior se presentan los resultados de la evaluación.

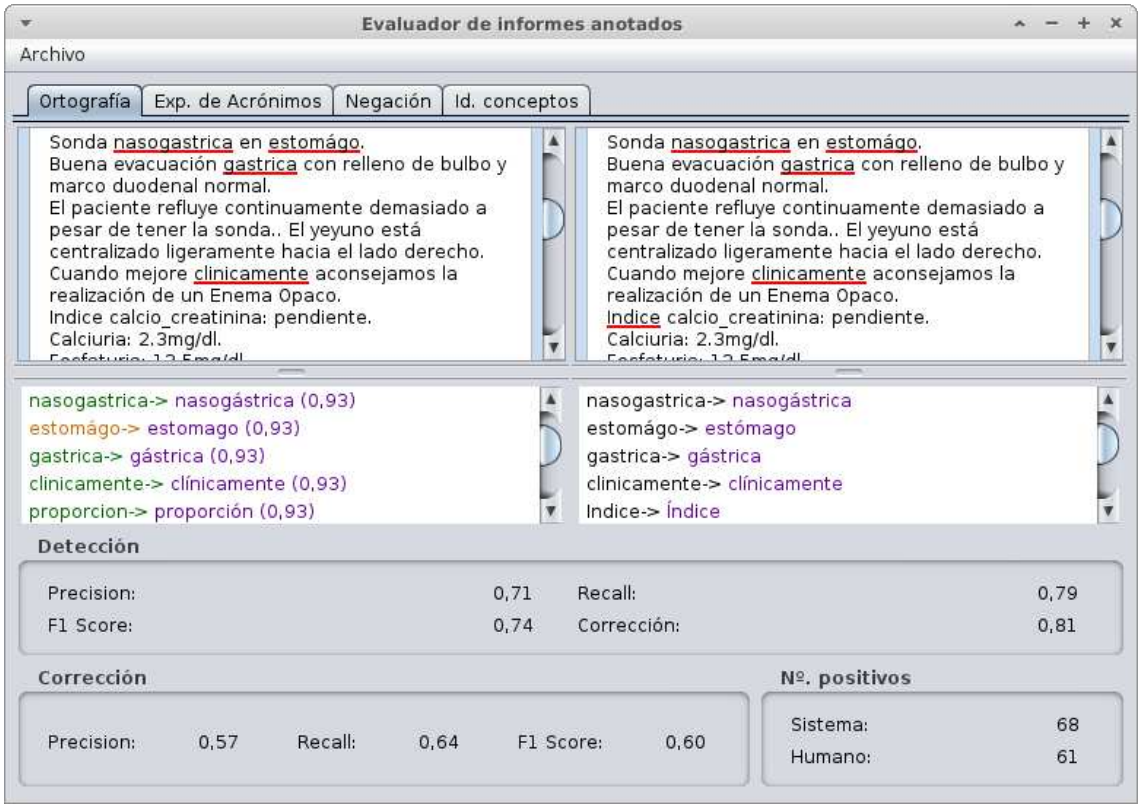

Figura 3.6: Vista del modo evaluador

3.5.2. Anotación

El propósito del anotador es facilitar la tarea de anotar informes, permitiendo añadir, modificar y eliminar anotaciones a través de una interfaz, i.e. sin tener que editar los ficheros a mano.

En la figura 3.7 se muestra un detalle de la interfaz del anotador. En la parte superior se muestra el texto del informe, mientras que en la parte inferior se listan las anotaciones. Clicando sobre palabras del texto, se pueden activar diálogos para introducir la información de las anotaciones.

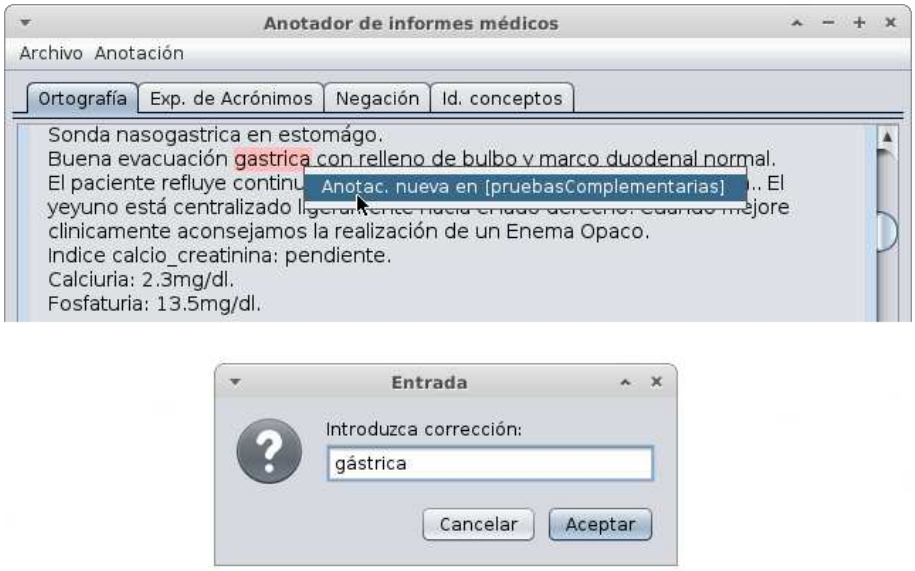

Figura 3.7: Detalle del modo anotador

Los detalles sobre el modo de uso del anotador se pueden consultar en la sección A.3 del apéndice *Manual*.

Capítulo 4

Fase de corrección ortográfica

La fase de corrección ortográfica es la primera de las fases del procesado. Su objetivo es detectar palabras mal escritas en el texto y corregirlas automáticamente. Se trata de una tarea muy importante, ya que de ella depende en parte el correcto funcionamiento de las siguientes. El resto de los módulos se basan en la búsqueda de ciertas palabras y patrones en el texto, por lo que las faltas ortográficas pueden influir negativamente en su eficacia.

La corrección ortográfica es un problema muy común, para el que se han desarrollado multitud de librerías y aplicaciones. Tras informarnos acerca de las soluciones existentes para este problema, decidimos que lo más apropiado era reutilizar alguna de ellas. En concreto, hemos basado nuestro corrector en el módulo de corrección de la herramienta AutoIndexer, que a su vez utiliza Hunspell, construido por el grupo NIL de la Universidad Complutense de Madrid y sobre el que ya se discutió en el capítulo 2, *Estado de la cuestión*.

4.1. Resumen del proceso de corrección

Al comienzo de la fase de corrección, se recibe una estructura de datos que contiene todo el texto a procesar, dividido en cuantas secciones tuviera el informe. Este texto se trocea en tokens, y para cada uno de ellos se aplica el siguiente proceso:

- Se busca el token en el diccionario de la aplicación.

- Si se encuentra el token en el diccionario, se considera que la palabra está escrita correctamente, y se procede a aplicar el proceso sobre el siguiente token.

- Si no se encuentra el token en el diccionario, se considera que la palabra contiene errores de ortografía. En tal caso, la librería Hunspell sugiere una serie de términos corregidos para reemplazar la palabra mal escrita.

- Se puntúan las sugerencias obtenidas por Hunspell comparándolas con el token original.

- Si la puntuación de una sugerencia supera o iguala un determinado umbral, entonces se considera que es la corrección adecuada. En caso de haber varias sugerencias con puntuaciones superiores o iguales al umbral, se elige la de mayor puntuación. Si todas las sugerencias tienen la misma puntuación, se escoge la primera.

- Si no existen sugerencias cuyas puntuaciones al menos igualen el umbral, entonces no se descarta ninguna sugerencia como posible. Es decir, se considera que todas las sugerencias son válidas, y no se realiza corrección automática.

- Se añade una anotación en el texto recibido originalmente adjuntando al token la corrección adecuada, o la lista de correcciones posibles, según el caso.

El valor de puntuación que se utiliza como umbral en el algoritmo se puede modificar desde uno de los ficheros de configuración de la herramienta (ver apéndice B) y deberá ser establecido empíricamente.

4.2. Detección de errores y creación de sugerencias

Para comprobar qué palabras son incorrectas y obtener las posibles correcciones se hace uso de parte del corrector ortográfico desarrollado en el proyecto AutoIndexer, que se menciona en el comienzo de este capítulo, y del que se dio una breve explicación en el capítulo 2 (*Estado de la cuestión*).

Hunspell hace uso de un diccionario para determinar si una palabra es correcta o no. El diccionario utilizado en la aplicación es un diccionario común de español, mejorado con palabras y acrónimos del ámbito médico.

En caso de no encontrar la palabra en el diccionario, se considera que existen errores en su escritura y se procede a elaborar una lista de sugerencias para la corrección de la palabra. Para derivar las posibles soluciones a partir del término original, Hunspell elige aquellos vocablos para los que se minimice la distancia de Levenshtein (ver capítulo 2). También es capaz de hacer uso de reglas fonéticas y de n-gramas para elegir las sugerencias, pero en nuestro caso no se utiliza dicha funcionalidad.

En el caso de los n-gramas, se debe a que no se dispone de un corpus apropiado del que extraer información estadística. Por su parte, en el caso de las reglas fonéticas, sí se incluyeron y se pudo comprobar experimentalmente que daban lugar a sugerencias mucho peores. Pese a la decisión de no tenerlas en cuenta en la creación de posibles correcciones, las reglas fonéticas sí que son empleadas en el proceso de puntuación de sugerencias, que se describe en la siguiente sección.

4.3. Puntuación de las sugerencias

Las sugerencias recibidas de la librería Hunspell no están ordenadas de ninguna manera, por lo que es necesario determinar su «calidad» comparándolas con la palabra que originalmente se deseaba corregir. Las sugerencias se ordenan entonces de acuerdo con esa puntuación, que representa el grado de similitud existente entre ellas y la palabra original. Este grado de similitud se representa con un número del 0 al 1, indicando una puntuación de 0 que las dos palabras no se parecen absolutamente en nada y una puntuación de 1 que son el mismo término.

La puntuación de cada una de estas sugerencias del diccionario se calcula a partir de la distancia entre dicha sugerencia y la palabra que se está procesando. En concreto, consideramos que

- $si\ \ Distancia < 10\ \ entonces\ \ Puntuación = 1 - Distancia/10$
- $si\ \ Distancia \geq 10\ \ entonces\ \ Puntuación = 0$

En cuanto a la distancia, se obtiene sumando tres medidas diferentes de distancia, que ya fueron introducidas en el capítulo 2. Los pesos que se asigna a cada una de ellas se pueden modificar en los archivos de configuración de la herramienta (ver apéndice B).

4.3.1. Distancia de Levenshtein

La distancia de Levenshtein se define como el mínimo número de operaciones necesarias para transformar una cadena en otra, siendo las operaciones posibles la inserción, el borrado y la sustitución de un carácter. En la aplicación, se ha implementado mediante el algoritmo Wagner-Fischer (Wagner and Fischer [1974]), que se basa en el principio de la programación dinámica.

Concretamente, comienza calculando la distancia entre los prefijos de menor longitud de las palabras, y progresivamente, calcula las distancias entre prefijos más largos hasta llegar a las palabras completas, usando siempre en los cálculos las distancias obtenidas previamente.

4.3.2. Distancia de teclado

Denominamos distancia de teclado a la distancia entre los caracteres de la palabra original y la sugerencia en un teclado con disposición QWERTY. Los teclados de disposición QWERTY son los más comunes en la actualidad, y se llaman así por el orden en que aparecen las letras en la fila superior: primero Q, luego W, etc. (ver figura 4.1).

Para calcular esta distancia, es preciso alinear las palabras de tal forma que coincidan el mayor número de caracteres de ambas. Es por ello que se hace uso del algoritmo Needleman-Wunsch, que ya se introdujo en el capítulo 2. Este algoritmo maximiza la similitud entre dos secuencias, alineando sus caracteres de tal manera que coincidan el mayor número de ellos que sean iguales. En caso de que las palabras a comparar no sean de igual longitud, se introducen huecos.

Una vez que se han alineado las secuencias, se calcula la similitud sumando las puntuaciones asociadas a cada par de caracteres, además de una penalización por cada hueco que se haya debido introducir. En nuestro caso, la puntuación de cada

Figura 4.1: Teclado QWERTY

par de caracteres es la distancia de Manhattan que existe entre ellos tras alinear las teclas en una cuadrícula, y el coste de cada hueco es 1.

Figura 4.2: Distancia de teclado

Por ejemplo, si consideramos el término original «docio» y las sugerencias «socio» y «bocio», tenemos que la distancia de Levenshtein es 1 en ambos casos, por la sustitución de la «d». Si solo se tiene dicha distancia en cuenta, las dos sugerencias resultan igual de lógicas, o en otras palabras, la probabilidad de que una sea la correcta es similar en los dos casos. Sin embargo, midiendo la distancia de teclado, obtenemos 1 en el primer caso y 4 en el segundo (ver figura 4.2). Este resultado apoya a la intuición de que la sugerencia más lógica es «socio», por la proximidad de la «d» y la «s».

[1]Imagenes bajo licencias GNU Free Documentation License, Versión 1.2 o posterior (http://commons.wikimedia.org/wiki/Commons:GNU_Free_Documentation_License_1.2) y Creative Commons Attribution-Share Alike 3.0 Unported (http://creativecommons.org/licenses/by-sa/3.0/deed.en). Basadas en imagen de Simo Kaupinmäki (disponible en http://commons.wikimedia.org/wiki/File:ISO_keyboard_%28105%29_QWERTY_UK.svg)

4.3.3. Distancia fonética

La distancia fonética entre la palabra original y la sugerencia se calcula comparando sus pronunciaciones. Más concretamente, en la implementación se hace uso de una adaptación al español del algoritmo Metaphone, ya mencionado en el capítulo 2. Este algoritmo utiliza en primer lugar un conjunto de reglas de pronunciación para traducir las palabras que se desea comparar a otras que las representan fonéticamente. Después, calcula la distancia de Levenshtein entre dichas palabras modificadas para determinar cuánto difieren los términos en lo que a pronunciación se refiere.

Esta técnica resulta especialmente útil en los casos en los que el término original es incorrecto pero tiene exactamente la misma pronunciación que la sugerencia. Para ilustrar esto con un ejemplo, supongamos que se desea comparar las palabras «agugeta» y «agujeta». Si consideramos la distancia de Levenshtein, la diferencia entre las dos cadenas es 1, correspondiente a la sustitución de la «g» por la «j». Sin embargo, la distancia fonética es 0, ya que las dos se pronuncian de igual manera.

Concretamente, se consideran las siguientes reglas de pronunciación en la implementación del algoritmo:

- C = Z, cuando van seguidas de E o I.
- C = K, cuando van seguidas de A, O, U o cualquier otra consonante, menos la C y la H.
- G = J, cuando van seguidas de E o I.
- Ll = Y.
- B = V.
- U = W.
- X = S, cuando son la primera letra de la palabra.
- QU = K.
- Q = K, cuando van seguidas de cualquier letra, menos la U.
- la H se omite.

Capítulo 5

Fase de expansión de acrónimos

En el ámbito médico se utilizan gran cantidad de abreviaciones a la hora de redactar informes. Estas abreviaturas en muchos casos no están homologadas y su significado (forma expandida) puede variar en función del centro hospitalario donde se redacte e incluso de las diferentes secciones dentro de un mismo centro.

Las abreviaciones son recursos para ahorrar tiempo y espacio en el lenguaje pero el uso excesivo de estas genera dificultades a la hora de comprender los textos, siendo un gran problema para los documentalistas médicos al tener que interpretar los mismos. Para facilitar la interpretación y comprensión de textos médicos se utilizan sistemas automáticos de detección y expansión de acrónimos[1].

Por tanto, el objetivo de esta fase es detectar los acrónimos que hay en el texto y, mediante un proceso de desambiguación, elegir la expansión más adecuada de cada acrónimo según el contexto en el que se encuentre.

5.1. Detección de acrónimos

Para el proceso de detección decidimos emplear el mismo sistema que se utiliza en AutoIndexer. Se divide el texto en palabras, se busca cada una de ellas en el lexicón de acrónimos y si se encuentra es clasificada como acrónimo. AutoIndexer hace uso de varios lexicones pero solamente uno de ellos se restringe al ámbito médico. Por tanto, decidimos utilizar dicho lexicón de acrónimos médicos para que el resto no produjese

[1]Para abreviar utilizaremos la palabra acrónimo para referirnos también a siglas y abreviaturas

ruido en el sistema. Este lexicón contaba con aproximadamente 1500 acrónimos pero lo completamos utilizando el diccionario de siglas médicas del Ministerio de Sanidad y Consumo[2], superando la cifra de 3000 acrónimos.

5.2. Desambiguación de acrónimos

El proceso de desambiguación se presenta como una tarea compleja y costosa de implementar, dado que ciertos estudios (Hongfang Liu [2002]) demuestran que el 81 % de los acrónimos encontrados en los *abstracts* de MEDLINE son ambiguos, teniendo una media de 16 posibles expansiones cada uno. Habitualmente, se emplean algoritmos de aprendizaje máquina para desambiguar acrónimos, mediante los cuales el sistema aprende los contextos en los que aparece cada expansión dentro de un corpus (artículos científicos de MEDLINE, por ejemplo), clasificando cada contexto por la expansión a la que acompaña.

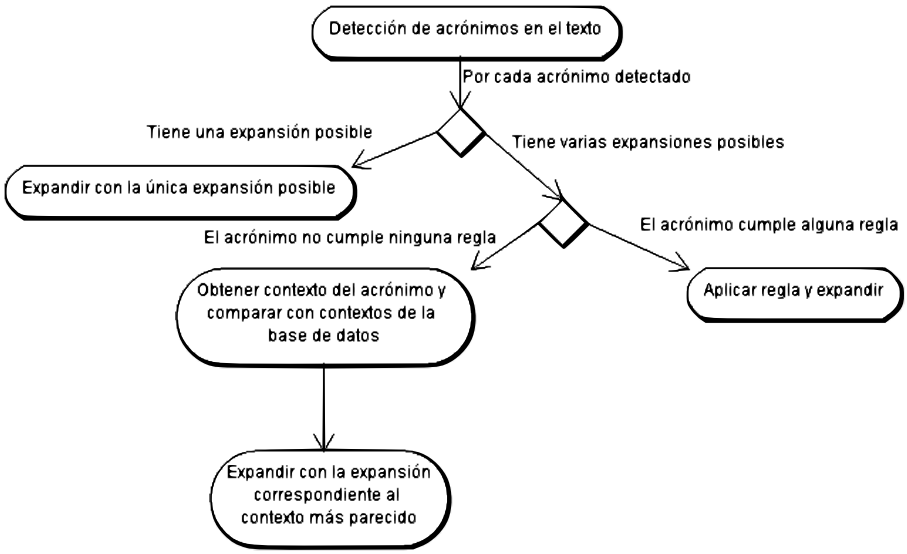

Figura 5.1: Diagrama de actividad de la desambiguación de acrónimos

Dado que no disponemos de un corpus de informes o artículos médicos anotados donde obtener los ejemplos de entrenamiento para el algoritmo, decidimos utilizar como corpus SNOMED CT. En las descripciones de conceptos contenidas en SNOMED CT aparecen algunas de las formas expandidas de los acrónimos presentes en

[2] http://www.msc.es/estadEstudios/estadisticas/docs/diccionarioSiglasMedicas.pdf

los recursos de AutoIndexer. De dichas descripciones obtenemos los contextos de las formas expandidas para usarlos como los ejemplos de entrenamiento del algoritmo.

Además, utilizamos el sistema de reglas de AutoIndexer para apoyar al método anterior, que permite que un experto inserte reglas de desambiguación, por las cuales un acrónimo se puede expandir directamente con una expansión concreta.

El proceso de desambiguación se representa en el diagrama de la figura 5.1.

5.2.1. Sistema de reglas

Dado que en las descripciones de SNOMED CT no aparecen muchos acrónimos de nuestro lexicón, decidimos utilizar un sistema de reglas para apoyar al método de aprendizaje. Las reglas deben ser establecidas por un experto para mejorar la eficacia del sistema a la hora de desambiguar los acrónimos detectados.

Las reglas que el sistema utiliza son de la forma

IF <*expresión lógica* > AND <*sección del acrónimo*> = <*sección de la regla*>
THEN < *expansión*>

El significado de cada acrónimo suele depender del contexto en el que aparece, por tanto la condición lógica de la regla se refiere a los contextos de los acrónimos. Además, para que una regla se aplique, se tiene que cumplir que la sección del informe en la que se encuentra el acrónimo coincida con la sección especificada en la regla. En la tabla 5.1 se muestran los tipos de expresiones lógicas que se pueden utilizar.

El contexto CONTIENE ALGUNA palabra del conjunto {P1... Pn}
El contexto CONTIENE TODAS las palabras del conjunto {P1... Pn}
El contexto NO CONTIENE NINGUNA de las palabras del conjunto {P1... Pn}

Tabla 5.1: Expresiones lógicas a utilizar en las reglas.

Existen acrónimos que suelen ir acompañados por un número. Se puede añadir la cadena de caracteres [numero] al contexto de la regla para indicar que el acrónimo

ha de ir acompañado de cualquier número. Así, si el número puede tomar una amplio rango de valores se evita tener que indicar todos los números en la regla.

Las reglas se almacenan en la base de datos de la aplicación y se pueden añadir, editar y eliminar a través de un editor para facilitar al experto su gestión (ver figura 5.2).

Figura 5.2: Editor de reglas

5.2.2. Sistema de aprendizaje

El objetivo del sistema de aprendizaje consiste en aprender los contextos en los que aparece cada acrónimo dentro de un corpus (SNOMED CT, en nuestro caso) para luego, en cada ejecución, comparar el contexto en el que aparece el acrónimo a desambiguar con los contextos aprendidos para ese acrónimo y elegir la expansión adecuada.

En primer lugar, el sistema busca el acrónimo en el corpus y aprende el contexto en el que aparece cada una de sus expansiones. Cabe decir que esta fase solamente se realiza una vez para un corpus dado, ya que los resultados se almacenan en la base de datos de la aplicación.

Esta tarea se realiza mediante un módulo específico de aprendizaje. Como en SNOMED CT no aparecen todos los acrónimos de la base de datos, el módulo de aprendizaje se ha desarrollado de manera que se permita obtener nuevos contextos de otros corpus de aprendizaje más completos.

Una vez que el sistema ha aprendido los contextos, se compara el contexto del acrónimo a desambiguar con los contextos aprendidos para ese acrónimo y, finalmente, se selecciona la expansión con el contexto más similar al dado.

En el siguiente cuadro se muestra cual sería el contexto de un acrónimo para una frase dada.

Texto en el que aparece el acrónimo (PCR):
PCR en sangre para CMV; adenovirus: negativo.
Contexto del acrónimo (solamente palabras importantes):
[sangre, CMV, adenovirus, negativo]

Para realizar las comparaciones entre contextos utilizamos el modelo del espacio vectorial. Este modelo se basa en la idea de que la relevancia de un conjunto de palabras dentro de un documento puede ser calculada mediante la diferencia de ángulos (usando el coseno de los ángulos) de los vectores de cada uno de los documentos respecto del vector del conjunto de palabras inicial.

En nuestro caso, se crea un vector con las palabras de cada contexto aprendido y otro con el contexto a comparar. Después, se calcula la distancia entre ellos y se elige el contexto cuya distancia sea menor.

Como ejemplo, calculamos la distancia entre el contexto mostrado en el cuadro anterior y dos de los contextos aprendidos.

Contexto del acrónimo a desambiguar:
[sangre, CMV, adenovirus, negativo]
Contextos aprendidos:
C1 (correspondiente con la expansión Reacción en cadena de la polimerasa"): [fingerprinting, detección, genética, ácido, Pneumocystis, genotipo, positiva, ***negativa****, humano, ribonucleico, digital, huella, aminotransferasa,* ***adenovirus****, secuencia, detectada, observación, persistente, Dengue]*
C2 (correspondiente con la expansión "Proteína C reactiva"): [sustancia, plasmática, determinación, normal, función, hallazgo, plasmático, reacción, observable, medición, nivel]

Para calcular el ángulo entre cada par de vectores se usa la siguiente ecuación:

$$\cos\Theta = \frac{v1 \cdot v2}{\|v1\|\|v2\|}$$

En el numerador tendremos el número de elementos comunes (dos palabras se consideran comunes si comparten la misma raíz) en ambos vectores y en el denominador el producto de las magnitudes de ambos vectores. Por tanto:

Distancia con C1:
$\cos\Theta = \frac{2}{76} = 0{,}0263$
Distancia con C2:
$\cos\Theta = \frac{0}{44} = 0$

Entonces, se seleccionaría la expansión del contexto C2 ya que es la que nos proporciona un valor del coseno del ángulo más cercano a 1 y, por tanto, una distancia menor. Hay que destacar que, aunque se seleccione la expansión con el contexto más parecido, decidimos que el sistema devuelva todas las expansiones posibles ordenadas de menor a mayor distancia ya que el recurso del que fueron obtenidos los contextos hace que los resultados no sean totalmente fiables.

Capítulo 6

Fase de detección de negación

Esta fase se encarga de identificar qué frases dentro del texto del informe están negadas. Además, su objetivo es definir el ámbito o alcance de la negación dentro de cada una de esas frases. Esto se usará posteriormente para poder etiquetar los conceptos médicos identificados como «afirmados» o «negados», según corresponda.

En la siguiente tabla se puede ver un ejemplo del resultado esperado de esta fase. La segunda columna indica el *cue*, o señal de negación, que indica que la frase contiene una negación. La tercera columna corresponde al ámbito o alcance de la negación; es decir, la porción de frase cuyo significado está siendo negado.

Frase	Señal de negación	Ámbito
El paciente no tiene fiebre	no tiene	fiebre
Resultado de la prueba negativo	negativo	Resultado de la prueba

Tabla 6.1: Ejemplo de detección de negación.

6.1. El algoritmo NegEx

Este es el método de detección de negación que se ha adaptado para el proyecto.

NegEx es un algoritmo que fue creado para la detección de la negación en inglés. Tras estudiarlo y ver que ya habían sido realizadas adaptaciones a otros idiomas como el sueco (Skeppstedt [2011]), se decidió adaptarlo al castellano.

Está basado en la detección de ciertos patrones o *triggers* mediante expresiones regulares. Se utilizan cuatro clases de *triggers*:

Pseudo-negación: Son patrones que parecen de negación, pero que en realidad no niegan ninguna condición o concepto clínico. Cuando el algoritmo los encuentra, salta hasta el siguiente término de negación.

Negación: Son términos que niegan condiciones clínicas que aparecen a continuación en la frase.

Post-negación: Términos que indican negación hacia atrás en la frase, i.e. niegan las palabras que los preceden.

Conjunciones: Conjunciones adversativas o locuciones que pueden anular o cortar una negación. Se utilizan para acotar el ámbito de la negación dentro de una frase.

Estos términos de negación se han adaptado al castellano a partir de los originales en inglés encontrados en un proyecto de código abierto sobre NegEx[1].

Algunos de los *triggers* del inglés no tienen correspondencia directa en castellano y al contrario, hemos tenido que incluir locuciones habituales en castellano sin equivalente anglosajón. Todo este proceso de traducción y adaptación de patrones se ha realizado mediante la experiencia, hasta conseguir un conjunto de ellos aceptable en cuanto a resultados del algoritmo. En la siguiente tabla aparecen unos cuantos ejemplos de los mismos.

Pseudo-negación	Negación	Post-negación	Conjunciones
sin dificultad	no presenta	es negativo	salvo
no solo	sin signos	dio negativo	pero
no se conoce	negativo para	debe ser excluido	como principio de
no aumenta	descartando	fue descartado/a	no obstante
no hay cambios	ausencia de	es negativo	a causa de

Tabla 6.2: Triggers de negación.

[1] https://code.google.com/p/negex/

En total hemos obtenido la siguiente cantidad de triggers:

Pseudo-negación: 12.

Negación: 68.

Post-negación: 19.

Conjunciones: 49.

6.1.1. Descripción del algoritmo

Al comienzo de la fase, se recibe una estructura de datos que contiene todo el texto a procesar, dividido en cuantas secciones tuviera el informe. Este texto se trocea en frases y, para cada una de ellas, se buscan los *triggers* que contiene y se aplica el siguiente proceso:

- Ir al siguiente *trigger* en la frase (Neg1).
  - Si Neg1 es de tipo Pseudo-Negación, saltar al siguiente *trigger* en la frase
  - Si Neg1 es de tipo Negación: definir el alcance de Neg1 hacia adelante, cortando dicho alcance al encontrar alguno de los siguientes:
    - Un *trigger* de tipo Conjunción.
    - Otro *trigger* de Negación o Pseudo-Negación.
    - El final de la frase.
  - Si Neg1 es un término de Post-Negación: definir el alcance hacia atrás hasta el inicio de la frase.
- Repetir por cada *trigger* que quede en la frase.

6.1.2. Expresiones regulares: ventajas e inconvenientes

El uso de este algoritmo, totalmente basado en expresiones regulares, comporta tanto ventajas como inconvenientes.

Entre las primeras se encuentra la rapidez con la que se ejecutan, mayor que en otras técnicas que acceden a bases de datos o parsean y etiquetan el texto, y que en

nuestro caso hace que esta sea la tarea más rápida, en comparación con el resto de fases.

Otro de los pros de este algoritmo es que se presta a ser mejorado de forma sencilla mediante el uso de nuevos *triggers*. En este proyecto se ha construido un conjunto básico de patrones, pero con tiempo y recursos podría elaborarse una base de patrones más completa. Concretamente, por estar el registro de los textos bastante delimitado, se podrían añadir *triggers* específicos que proporcionen buenos resultados para informes médicos.

En cuanto a los inconvenientes, surgen al considerar los errores que se producen en la detección del ámbito. Por una parte, se ha de tener en cuenta que este algoritmo, al igual que el resto de técnicas basadas en expresiones regulares, no considera ni la sintaxis ni la semántica de las oraciones. Por tanto, ciertas construcciones gramaticales resultan en una detección del ámbito poco precisa o errónea.

Por otra parte, la necesidad de contar con un *splitter* para obtener las frases que componen el texto introduce otra posible fuente de errores. Habitualmente, los *sentence splitter* son sencillos y bastante fiables; pero en ocasiones, como cualquier herramienta de procesamiento del lenguaje natural, fallan y esto perjudica enormemente el funcionamiento de NegEx al delimitar el ámbito de una negación.

Capítulo 7

Fase de identificación de conceptos

En esta última fase, el objetivo es detectar y localizar en el texto conceptos o términos médicos. Dichos conceptos objetivo están contenidos en una base de datos llamada *SNOMED CT*.

Cabe recordar, que al llegar a este apartado en el procesado del informe, se tienen ya ciertas anotaciones sobre el mismo, correspondientes a las diferentes salidas de las anteriores fases.

7.1. La base de conocimiento SNOMED CT

SNOMED CT (*Systematized Nomenclature of Medicine – Clinical Terms*), es una colección de términos médicos organizados sistemáticamente. Incluye definiciones, términos, relaciones y sinónimos sobre enfermedades, procedimientos clínicos, microorganismos, síntomas, sustancias y otros conceptos.

SNOMED CT incluye diversas tablas y subconjuntos diferentes, así como versiones para diferentes lenguas.

7.1.1. Tablas de SNOMED CT utilizadas

Las tablas a continuación descritas corresponden a la Edición en Español de octubre de 2011 y Edición Internacional de enero de 2012 publicadas en la web de la Biblioteca Nacional de Medicina de EEUU (NLM).

7.1.1.1. Tabla de descripciones

Esta tabla contiene las diferentes descripciones o definiciones (suele haber más de una) de cada uno de los conceptos médicos de SNOMED CT.

A continuación se muestra un ejemplo de varias entradas y sus campos más relevantes.

ID Descripción	Texto	ID Concepto
898593010	infarto de miocardio (trastorno)	**22298006**
898592017	ataque al corazón	**22298006**
849160018	infarto de miocardio, cicatrizado	1755008
1000794015	angina preinfarto	64333001

Tabla 7.1: Ejemplo de las descripciones de SNOMED CT

- **ID Descripción:** Identificador único asociado a cada entrada de la tabla.
- **Texto:** Frase o enunciado que define el concepto médico identificado por el contenido del campo ID Concepto.
- **ID Concepto:** Número que identifica un concepto médico. Dicho concepto puede tener varias descripciones sinónimas, como se puede apreciar en las dos primeras filas del ejemplo.

7.1.1.2. Tabla de relaciones

Esta otra tabla (7.2) contiene entradas que relacionan pares de conceptos SNOMED CT tales como los que aparecen en la columna ID Concepto de la tabla de descripciones 7.1. Una columna de la tabla indica además el tipo de relación entre los conceptos. Los tipos de relaciones son a su vez conceptos SNOMED (marcados como conceptos especiales). Cabe destacar el concepto más importante, *is a*, el cual marca relaciones de «supertipo-subtipo» o relaciones «padre-hijo», que constituyen la base de las jerarquías de conceptos de SNOMED CT.

Ejemplo:

Fractura de hueso del tarso (trastorno). Se define como:

- es un (*is a*) → fractura de pie (trastorno)
- sitio del hallazgo → estructura ósea del tarso (estructura corporal)
- morfología asociada → fractura (anomalía morfológica)

ID Rel.	ID Concep. 1	Tipo de Rel.	ID Concep. 2
85...	<fract h. del tarso>	<is a>	<fractura del pie>
34...	<fract h. del tarso>	<sitio del hallazgo>	<huesos del tarso>
33...	<fract h. del tarso>	<morfología asociada>	<fractura>

Tabla 7.2: Tabla ejemplo de relaciones SNOMED CT

7.1.2. Problemas con SNOMED CT

Además de las tablas descritas, SNOMED CT contiene otros recursos como tablas específicas para las búsquedas y las consultas, tablas de equivalencias entre palabras y más. Por desgracia, este tipo de recursos de momento solo se hallan disponibles para el idioma inglés.

Otro de los inconvenientes que surgen al trabajar con SNOMED CT es la enorme cantidad de ramas y conceptos que engloba (ver tabla 7.3). Todos estos conceptos pueden ser interesantes al definir un concepto médico, pero a la hora de localizar conceptos en un informe, es posible que no se desee hallar ciertos términos pertenecientes a algunas de estas jerarquías (por ejemplo: «localización geográfica» o «animales»).

El filtrado de estos conceptos implica realizar un procesamiento extra, ya que en principio hay que recorrer las relaciones hasta alcanzar uno de los conceptos «ancestros» que se corresponden con cada una de las jerarquías referidas arriba.

Para facilitar dicha tarea, se ha realizado una búsqueda del concepto «ancestro» para cada concepto de SNOMED CT, almacenandolo en una nueva base de datos, en la que a su vez se almacena la descripción canónica en castellano del concepto.

Conceptos superiores
Fuerza física
Procedimiento
Evento
Entidad observable
Ambiente o localización geográfica
Estructura corporal
Contexto social
Organismo
Situación con contexto explícito
Sustancia
Estadificaciones y escalas
Producto farmacéutico/biológico
Objeto físico
Espécimen
Calificador
Concepto especial
Elemento de registro
Hallazgo clínico
Concepto de enlace

Tabla 7.3: Jerarquías de SNOMED CT

Con esta nueva tabla (7.4) se puede saber a partir de un identificador de concepto, a qué jerarquía pertenece y su descripción textual en castellano.

ID Concep.	FSN (Nombre completamente especificado)	Tipo (ID de ancestro)
29...	Bajo peso corporal	<Hallazgo clínico>
34...	Enalapril	<Sustancia>
52...	Intubación	<Procedimiento>

Tabla 7.4: Ejemplo de conceptos en tabla propia

7.2. Procedimiento utilizado

En esta sección se describe el método usado en esta fase del procesado del informe para conseguir la identificación y localización de los conceptos médicos mencionados en el texto del documento.

El objetivo principal consiste en poder hallar en el texto apariciones de conceptos de SNOMED CT, lo más precisos y relevantes que sea posible atendiendo al contenido del informe. Nuestro enfoque para lograrlo consiste en intentar hacer un encaje o «matching» entre el texto y los conceptos o, más concretamente, las descripciones que conforman la tabla de descripciones de SNOMED CT mencionada anteriormente (7.1).

Por cuestiones prácticas y de eficiencia, hemos indexado el contenido de la tabla de descripciones de SNOMED CT mediante la librería Apache Lucene. En las siguientes secciones se explica el uso de dicha librería, los resultados obtenidos y los métodos y técnicas empleados para mejorar y filtrar dichos resultados.

7.2.1. Indexación de descripciones mediante Lucene

Lucene[1] es una librería de código abierto para la recuperación de información. Su principal función consiste en estructurar la información a recuperar -normalmente texto- en forma de documentos indexados para, posteriormente, poder realizar búsquedas eficientes sobre dicho texto. Cabe decir que, pese a su nombre, Lucene no crea realmente documentos en el sentido habitual de la palabra, sino estructuras de datos optimizadas para la recuperación de información.

Para nuestra misión se crea un documento por cada entrada de la tabla de descripciones o, lo que es lo mismo, por cada una de las diferentes definiciones distintas que existen en SNOMED CT. La creación de los documentos indexados no es directa: primero es necesario preprocesar estas definiciones, con el propósito de hacerlas más generales y aumentar las posibilidades de encaje, respetando en lo posible el contenido original.

En primer lugar, se eliminan las palabras que no aportan significado o son muy comunes, conocidas en el ámbito del PLN como *stop words*. La lista de *stop words* a eliminar que se ha utilizado se basa casi en su totalidad en la del proyecto *open source* Snowball[2], dedicado al desarrollo de *stemmers*.

[1] http://lucene.apache.org/core/
[2] http://snowball.tartarus.org/algorithms/spanish/stop.txt

Después, se aplica a los términos restantes un *stemmer* para español, incluido en la propia librería Lucene. Un *stemmer* es un sistema que reduce palabras a su raíz o lexema, p. ej.: el lexema de «pato» es «pat-» y el de «liberación» es «liber-».

De esta manera se reducen las definiciones de los conceptos a una representación de menor tamaño que descarta lo superfluo y conserva el significado esencial, con lo que se logra encontrar coincidencias con un mayor número de textos.

7.2.2. Del contenido del informe a búsquedas en Lucene

Una vez creado el índice de documentos, el siguiente paso es crear una *query*, i.e. una consulta, que contiene la información que se desea recuperar, la cual Lucene interpreta y para la que devuelve ciertos resultados. En nuestro caso dicha *query* es puramente textual y Lucene devuelve resultados basándose en la similitud entre el texto de la *query* y el texto de los documentos (descripciones de conceptos de SNOMED CT).

Con la *query* ya construida, Lucene ejecuta su algoritmo de búsqueda y devuelve los resultados. Al tratarse de búsquedas basadas en la similitud entre textos, Lucene puede devolver miles de resultados en la mayoría de los casos. Dichos resultados son devueltos siempre en orden de relevancia (i.e. similitud).

Ej: Para la *query* «El recién nacido fue ingresado», Lucene puede devolver:
- Recién nacido.
- Recién nacido prematuro.
- Ingreso del paciente.
- ...

En una primera versión, cada *query* era creada simplemente a partir de las frases del informe, utilizando el mismo filtrado de *stop words* y el *stemmer* que en la creación del índice.

El principal problema de usar frases completas es que estas contienen normalmente más de un término médico. Se observó que con frases largas la mayor cantidad de palabras empeoraba los resultados de Lucene, devolviendo documentos imprecisos o erróneos. Además, atendiendo únicamente al orden devuelto por Lucene, es imposible saber si hay uno o varios conceptos correctos en la lista de resultados.

Por tanto, se decidió a raíz de lo anterior el uso de ciertas palabras como separadores (nexos, disyunciones, comas...) para dividir las frases en enunciados más pequeños, aumentando así el número de búsquedas en Lucene, pero siendo los resultados devueltos más precisos.

7.2.3. Procesamiento de los resultados

Tal y como se ha dicho, Lucene devuelve N resultados ordenados por relevancia (siendo N arbitrariamente grande). Esto presenta las siguientes cuestiones.

1. Se obtienen coincidencias con documentos (descripciones de conceptos) erróneas o imprecisas.

2. En casi la totalidad de los casos se obtiene una gran cantidad de resultados. Por ello, se hace necesario establecer un criterio para decidir cual o cuales son los conceptos referidos en el texto. Por ejemplo, quedarnos con el primero y más relevante de los resultados.

3. Como ya se ha mencionado en el apartado 7.1.2, conviene tener en cuenta la categoría superior a la que pertenezcan los conceptos recuperados; ya sea para filtrar según la sección del informe o dar prioridad a unos sobre otros.

4. Por último, aunque se trate de una cuestión técnica, hay que tener en cuenta la cantidad de tiempo que conlleva realizar numerosas consultas.

Para intentar dar solución a algunos de estos problemas, se muestran a continuación las estrategias implementadas.

7.2.3.1. Eliminación de fechas y otros elementos

Se ha añadido la funcionalidad de poder filtrar cualquier fragmento de texto mediante expresiones regulares definidas en un fichero. Esto permite, por ejemplo, quitar fechas, números de cantidades y medidas, y cualquier otro tipo de cadenas de texto que no se quieran contemplar, o que con toda seguridad no van a ser un concepto y se sabe que solo aumentarán el error de esta fase.

Al poder configurarse este tipo de filtro mediante un fichero, damos la posibilidad de poder mejorar la herramienta con cierta facilidad, ya que cuanto más acertados sean estos filtros, habrá menos error y mayor precisión en la recuperación de conceptos.

El fichero de filtros a utilizar se define en el fichero de configuración conceptos.properties (ver apéndice B).

7.2.3.2. Puntuación de resultados

La librería Lucene devuelve una serie de resultados ordenados por su puntuación de similitud a la consulta. Desafortunadamente, dicha similitud no puede ser tenida en cuenta para valorar la calidad de los resultados, dado que es una puntuación local para cada una de las búsquedas.

Ya que se desea establecer una medida de la calidad o precisión de los resultados devueltos, se ha decidido calcular aparte dicha similitud. Para ello, se emplea la siguiente medida de similitud, llamada índice de Jaccard:

$$J(A,B) = \frac{|A \bigcap B|}{|A \bigcup B|}$$

siendo A y B conjuntos de *tokens* reducidos a su raíz y habiendo sido descartadas las palabras señaladas como *stop words*.

Esta medidad de similitud se realiza entre la *query* (texto del informe) y cada uno de los resultados devueltos por Lucene.

Como medida especial para reducir posibles errores provenientes del uso de raíces en las consultas, se calcula también el índice de Jaccard con las palabras en su forma completa y se suma al resultado anterior. Normalmente esta segunda medida es igual o inferior a la medida obtenida usando raíces, pero al emplearse se evita que palabras pequeñas que coincidan perfectamente pierdan relevancia tras la aplicación del *stemmer*.

Solo se consideran los conceptos cuya puntuación supere un umbral, definido en el fichero de configuración conceptos.properties (ver apéndice B).

7.2.3.3. Filtrado y reordenación según contexto

Un informe médico normalmente se divide en una serie de secciones o apartados. En cada una de las secciones, es posible que interese priorizar un tipo concreto de conceptos o descartar ciertos tipos. Al acotar y priorizar el tipo de conceptos según la sección se observa una mejora sustancial de la precisión y reducción del error (por ejemplo, priorizando «fármacos» en una sección sobre «tratamiento»).

Los tipos de conceptos posibles corresponden a las jerarquías de SNOMED CT presentadas anteriormente en la tabla 7.3.

El método empleado se basa en descartar cualquier resultado que no pertenezca a una de las jerarquías marcadas como «permitidas» en la sección. Y por otra parte, multiplicar por un factor la puntuación de aquellos conceptos de un tipo marcado como «favorito» para la sección actual del informe.

Dado que las secciones de los informes a procesar puede variar, se establece una correspondencia entre cada una de ellas y una sección maestra o «metasección». Cada metasección puede corresponder a una o varias secciones del informe, p. ej.: las secciones «Antecedentes personales» y «Antecedentes familiares» podrían corresponder a una metasección «Antecedentes».

Tras asociar cada sección de los informe con una metasección, para cada una de las metasecciones se deben definir cuáles son las jerarquías permitidas y, si se desea, una jerarquía «preferida».

Todo esto se configura desde el fichero de configuración conceptos.properties (ver apéndice B).

7.2.3.4. Uso de los resultados de las fases anteriores

Para mejorar los resultados de este módulo, se utiliza la información obtenida en las fases anteriores del procesamiento.

En el caso de la expansión de acrónimos, se usa añadiendo el texto expandido del acrónimo al propio acrónimo a la hora de realizar la *query*. Esto se hace así porque SNOMED CT contiene en sus descripciones tanto acrónimos sin expandir, como el término expandido. Además, al encontrarse un acrónimo, normalmente dicho acrónimo suele corresponderse directamente con un concepto médico.

Por su parte, en el caso de la negación cabe destacar que se usan los ámbitos detectados para etiquetar como negados los conceptos hallados en ellos.

7.2.3.5. Intentos de mejora fallidos

Aparte del procedimiento descrito, se intentaron otras vías para mejorar los resultados, las cuales no dieron el fruto esperado. A continuación se explican brevemente algunas de ellas.

Agrupación de conceptos con relación de parentesco

Un intento de mejora consistió en agrupar conceptos con relación de parentesco en la jerarquía de SNOMED CT y quedarse solo con el más general, siendo más probable que el resultado fuera correcto al ser menos específico.

La idea finalmente se descartó, ya que aparte de la evidente pérdida de precisión, se observó que eran muy pocos los casos en los que realmente se daban resultados relacionados directamente. Además, las consultas extra a la base de datos perjudicaban notablemente el rendimiento.

Uso de sinónimos y conceptos relacionados en la puntuación

Otra idea que surgió fue la de sumar a la puntuación devuelta por la función de similitud, la similitud con los sinónimos o con los conceptos relacionados (p. ej. los «padres»).

Se comprobó que esto alteraba los resultados de forma impredecible. En algunos casos, se producía una mejora positiva al aumentar la puntuación de conceptos acertados cuyos sinónimos coincidían en cierta medida con el texto de la consulta. En otros muchos casos, sin embargo, esta técnica se convertía en una fuente de errores.

Uso de frases completas en la búsqueda de conceptos

Se pensó que al aumentar el tamaño de la *query* se podría capturar mejor el contexto del concepto y, por tanto, mejorar los resultados, pero lo único que se consiguió al

usar frases completas fue reducir la precisión. Al emplear demasiadas palabras, la búsqueda se vuelve demasiado dispersa y aparecen gran numero de términos con una coincidencia parcial con la consulta, pero erróneos.

Capítulo 8

Evaluación

Como parte del desarrollo de una aplicación de procesamiento de lenguaje natural, surge la necesidad de evaluar la eficacia del sistema. Con este propósito se han creado dos modos adicionales de uso:

Anotación: facilita la tarea de anotar informes, permitiendo añadir, modificar y eliminar anotaciones a través de una interfaz, i.e. sin tener que editar los ficheros a mano.

Evaluación: compara informes anotados por la aplicación con informes anotados a mano por un usuario experto, dando una medida de la eficacia de los métodos empleados.

Este capítulo tiene dos secciones claramente diferenciadas. En la primera, *Método de evaluación*, se explican las técnicas y cálculos que se emplean para evaluar la eficacia del procesamiento automático que realiza la aplicación. En la segunda, *Resultados*, se exponen los resultados obtenidos en cada una de las fases del procesamiento.

8.1. Método de evaluación

El método de evaluación concreto que se aplica a cada una de las fases varía ligeramente por lo que en esta sección se procede a detallar los pormenores del proceso para cada una de ellas. Sin embargo, en todos los casos se cuenta con una base

común, que es la evaluación de las distintas técnicas de clasificación, entendiendo por clasificación la identificación de las distintas entidades o partes del texto como faltas de ortografía, acrónimos, negaciones, etc., dependiendo de la fase de procesamiento.

Para evaluar las tareas de clasificación, cada uno de los elementos del texto sobre los que se ha aplicado el procesamiento se separa en uno de cuatro conjuntos diferentes: verdaderos positivos, falsos positivos, verdaderos negativos y falsos negativos.

Verdadero positivo
: en adelante abreviado como VP, indica que tanto el clasificador de la aplicación como el clasificador humano han identificado el elemento como miembro de la clase que se intenta detectar (falta de ortografía, acrónimo, negación, etc. dependiendo de la fase; de aquí en adelante, la clase).

Falso positivo
: en adelante abreviado como FP, indica que el clasificador de la aplicación ha identificado el elemento como miembro de la clase, cuando en realidad no lo era.

Falso negativo
: en adelante abreviado como FN, indica que el clasificador de la aplicación no ha identificado un elemento que sí era miembro de la clase como tal.

Verdadero negativo
: en adelante abreviado como VN, indica que tanto el clasificador de la aplicación como el clasificador humano no han identificado el elemento como miembro de la clase.

Una vez que se han asignado todos los elementos a uno de estos cuatro conjuntos, es posible utilizar la cardinalidad de dichos conjuntos (es decir, la medida del número de elementos que contienen) para realizar ciertos cálculos que permitan evaluar la eficacia de la clasificación. Estas cardinalidades se suelen presentar en forma tabular, en lo que se conoce como tabla de contingencia (ver tabla 8.1).

Los dos cálculos más comunes que se realizan para evaluar las tareas de clasificación, y que son utilizados usualmente en las tareas de procesamiento de lenguaje natural, son la precisión y la cobertura (*precision* y *recall* en su forma inglesa).

		Humano	
		Miembro	«No-miembro»
Sistema	Miembro	VP	FP
	«No-miembro»	FN	VN

Tabla 8.1: Tabla de contingencia.

De manera intuitiva, la *precision* se podría entender como una medida de exactitud o calidad, mientras que el *recall* sería una medida de completitud o cantidad. Formalmente quedan definidos de la siguiente forma:

$$Precision = \frac{|VP|}{|VP| + |FP|}$$

$$Recall = \frac{|VP|}{|VP| + |FN|}$$

Para explicarlo con palabras de manera más precisa se puede decir que

Precision
: es una medida que indica cuántos de los elementos que han sido clasificados como miembros de la clase lo son realmente.

Recall
: es una medida que indica cuántos de los elementos que son en verdad miembros de la clase se han clasificado correctamente.

Una posible alternativa a estas dos medidas, que es también utilizada en ocasiones, es la *Accuracy* (escrita en su forma inglesa para distinguirla de la precisión ya mencionada). Esta medida se considera más general, ya que indica la fracción de elementos clasificados correctamente, i.e. tiene en cuenta también los elementos clasificados como «no-miembros» de la clase. Concretamente, la *accuracy* se define de la siguiente manera:

$$Accuracy = \frac{|VP| + |VN|}{|VP| + |FP| + |FN| + |VN|}$$

La razón por la que usamos la precisión y el *recall* es que la *accuracy* presenta un problema, derivado precisamente de la inclusión de los verdaderos negativos.

En muchas tareas de clasificación, es común que el número de miembros de VN sea considerablemente mayor que el del resto de conjuntos, lo cual implica la obtención de un alto valor de *accuracy*, independientemente de la distribución del resto de elementos en los otros 3 conjuntos. Esta «debilidad» se puede explotar incluso, construyendo un sistema con predisposición a clasificar elementos como «no-miembros» de la clase. En resumen, se trata de una medida no muy segura, ya que no captura correctamente los resultados realmente importantes (|VP|, |FP| y |FN|) desde el punto de vista de la relevancia y la utilidad del sistema. Por su parte, la precisión y el *recall* sí ofrecen información relevante sobre la eficacia del sistema.

Respecto a la importancia que se concede a cada una de ellas al interpretar los resultados, hay diferencias dependiendo de la tarea que se esté evaluando. En ciertos sistemas, es interesante obtener resultados más precisos, es decir, que la mayoría de los resultados obtenidos sean correctos. Un ejemplo de esto son los sistemas de búsqueda de páginas web: los usuarios prefieren recibir pocos resultados, todos acordes a lo que están buscando, antes que muchos resultados sin relación con su búsqueda. En otras ocasiones, interesa primar el *recall* y obtener todos los resultados que puedan ser relevantes, aunque a la vez se obtengan falsos positivos. Es el caso de, entre otros, los sistemas de diagnóstico médico: es mejor identificar al paciente como enfermo y descubrir mediante pruebas posteriores que el diagnóstico era incorrecto, a no identificarlo como tal y que realmente esté enfermo.

En el caso de la construcción de una aplicación genérica, en la que no se desee primar la cantidad sobre la calidad de los resultados, o viceversa, puede que estas medidas no aporten información clara sobre su eficacia. Por ejemplo, podría suceder que, tras un cambio en el método de clasificación utilizado, una de las medidas aumentara y la otra disminuyera. En tal caso, no sería posible determinar con claridad si la eficacia global de la tarea ha aumentado o, por el contrario, ha disminuido.

Para resolver este problema, existen varias medidas que combinan la precisión y el *recall* (o, en ocasiones, las cardinalidades de los cuatro conjuntos) dando como resultado un único valor. De esta manera se posee un indicador más directo de la eficacia de la evaluación, y la calidad de cualquier cambio en los métodos empleados se puede evaluar con más rapidez y seguridad.

En nuestro caso, se utiliza otra de las medidas más comúnmente empleadas para la evaluación de tareas de procesamiento de lenguaje natural: ***F₁ score***. Dicha medida se calcula como la media armónica de la precisión y el *recall*:

$$F_1 = \frac{1}{\frac{1}{2} * \frac{1}{Precision} + (1 - \frac{1}{2}) * \frac{1}{Recall}}$$

O como se representa normalmente:

$$F_1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

Cabe decir que esta medida es un caso específico de la *F score*, que permite dar más relevancia al *recall* frente a la precisión, o viceversa, y que se calcula como una media armónica ponderada:

$$F = \frac{1}{\alpha * \frac{1}{Precision} + (1 - \alpha) * \frac{1}{Recall}}$$

Que, de nuevo, se reformula normalmente de la siguiente forma:

$$F = \frac{(\beta^2 + 1) * Precision * Recall}{\beta^2 * Precision + Recall} \quad \text{con} \quad \beta = \frac{1 - \alpha}{\alpha}$$

La razón por la que se usa la media armónica en lugar de la media aritmética es que trata mejor los casos extremos en los que la precisión o el *recall* sean 0, penalizándolos más severamente. En concreto, para esa situación la media armónica es 0, independientemente del valor del otro parámetro, mientras que la media aritmética se corresponde con la mitad del valor del parámetro distinto de 0.

8.1.1. Corrección ortográfica y expansión de acrónimos

Para evaluar las fases de corrección ortográfica y de expansión de acrónimos se aplican principalmente las medidas ya detalladas anteriormente. Por una parte, se presentan bajo el título de **Detección** los resultados de la evaluación de la clasificación en sí: la identificación de cada palabra del texto como miembro o «no-miembro» de la clase (falta/acrónimo). Además se incluye una métrica que evalúa la corrección

y la expansión en sí, es decir, la idoneidad de la solución propuesta por la aplicación que sustituirá al término clasificado como error o acrónimo. A esta medida la llamamos **Coincidencias** y la definimos como el número de soluciones automáticas coincidentes con las soluciones humanas de entre todas las correcciones presentes en el conjunto VP.

Por otra parte, se presenta bajo el título de **Corrección** otro conjunto de resultados (precisión, *recall* y F_1 *score*) que evalúan a un tiempo tanto la detección como las coincidencias en la solución (corrección/expansión) propuesta. La diferencia con los resultados anteriores radica en la manera de calcular el número de verdaderos positivos. En este caso, solo se consideran como tal las palabras que sean clasificadas exactamente igual por el clasificador automático y el clasificador humano: no solo deben coincidir en señalar que se trata de una falta o un acrónimo, sino que deben proporcionar la misma corrección en un caso, y expansión en el otro.

Por último y a modo informativo, se muestra el **Número de positivos** hallados por el sistema y el anotador humano, i.e. el número de palabras que han sido clasificadas por uno y otro como faltas de ortografía y acrónimos en cada caso.

8.1.2. Detección de negación

En el caso de la detección de las negaciones presentes en el texto se presentan tres conjuntos de resultados, compuestos por precisión, *recall* y F_1 *score*.

En el primero, **Detección de *cue***, se evalúa la clasificación de las palabras en las clases «*Cue*» y «No *cue*». En el segundo, **Detección de *cue* y ámbito (a nivel de token)**, se evalúa la clasificación de las palabras en las clases «Forma parte de una negación» y «No forma parte de una negación», considerándose que una palabra forma parte de una negación si es el propio *cue* de la negación o está contenido en el ámbito de un *cue*.

Por último, en el tercero, **Detección de *cue* y ámbito (a nivel de frase)**, se evalúa la clasificación de las frases como negaciones, comprobando de forma estricta el *cue* y el ámbito identificados, i.e. para cada frase se evalúa si se ha detectado una negación en su contenido y si el *cue* y el ámbito de dicha negación se corresponden exactamente con los proporcionados en la solución humana.

Además de los resultados de la evaluación, se muestra también el **Número de positivos** hallados por el sistema y el anotador humano para cada uno de los 3 casos, de la misma forma que en la evaluación de la corrección ortográfica y la expansión de acrónimos.

8.1.3. Identificación de conceptos

Para evaluar la tarea de identificación de conceptos se presenta únicamente un conjunto de resultados (precisión, *recall* y F_1 *score*) bajo el nombre de **Identificación de conceptos** y calculado considerando las coincidencias entre los conceptos hallados en cada «subfrase» en que se parte el texto.

La comparación de dos conceptos se realiza simplemente comparando sus identificadores. Esta comparación produce un resultado binario, i.e. pueden ser iguales (se refieren al mismo concepto) o distintos (se refieren a conceptos diferentes). Es por ello que en la evaluación de esta fase no se presentan resultados parciales, como se hacía en las anteriores.

Además de los resultados de la evaluación, se muestra también el **Número de conceptos** hallados por el sistema y el anotador humano.

8.2. Resultados

En esta sección, se presenta una muestra de los resultados obtenidos al evaluar el procesamiento que realiza la aplicación. Los resultados corresponden a un mismo informe para el que se han probado diferentes configuraciones de la herramienta.

La tabla 8.3 contiene el significado de los encabezados de las tablas de resultados.

8.2.1. Fase de corrección ortográfica

Con el objetivo de ilustrar las conclusiones obtenidas tras las pruebas realizadas, se muestra en la tabla 8.2 un extracto de las mismas.

Las tres primeras filas se corresponden con la configuración de pesos por defecto de la herramienta, en la que las tres distancias tienen la misma importancia. Los

tres conjuntos de resultados que se presentan a continuación se corresponden con configuraciones en las que se da todo el peso de la puntuación a una de las distancias. Finalmente, las tres últimas filas corresponden a una configuración en la que se asigna un peso cercano a cero a las distancias fonética y de Needleman-Wunsch.

Parámetros				Detección				Corrección			Nº. positivos	
S	L	NW	F	P	R	F1	C	P	R	F1	PP	PA
3	0,33	0,33	0,33	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
5	0,33	0,33	0,33	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
7	0,33	0,33	0,33	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
3	1	0	0	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
5	1	0	0	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
7	1	0	0	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
3	0	1	0	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
5	0	1	0	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
7	0	1	0	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
3	0	0	1	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
5	0	0	1	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
7	0	0	1	0,71	0,79	0,74	0,79	0,56	0,62	0,59	68	61
3	0,9	0,05	0,05	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
5	0,9	0,05	0,05	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61
7	0,9	0,05	0,05	0,71	0,79	0,74	0,81	0,57	0,64	0,60	68	61

Tabla 8.2: Resultados de corrección ortográfica

Las conclusiones extraídas tras las pruebas realizadas son las siguientes:

- el número de sugerencias que devuelve Hunspell no parece influir en los resultados.
- los resultados de detección y los positivos encontrados se mantienen constantes en todos los casos.
- la corrección o idoneidad de las soluciones mejora en las configuraciones que asignan un peso mayor que cero a la distancia Needleman-Wunsch. Un mayor peso en esta distancia no deviene en resultados aún mejores. Cabe decir que estos resultados no son totalmente generalizables, ya que dependen del diccionario utilizado.
- para mejorar tanto la detección de faltas como la idoneidad de las sugerencias, la solución pasa por la mejora del diccionario.

P	*Precision*
R	*Recall*
F1	F_1 *score*
C	Corrección (definida en la sección 8.1.1)
PP	Positivos en el informe procesado
PA	Positivos en el informe anotado a mano
S	Número de sugerencias que propone Hunspell
L	Peso de la distancia de Levenshtein en la puntuación
NW	Peso de la distancia de Needleman-Wunsch en la puntuación
F	Peso de la distancia fonética en la puntuación
Reg	Reglas de acrónimos activadas
Cont	Comparación de contextos de acrónimos activada

Tabla 8.3: Leyenda de las tablas de resultados

8.2.2. Fase de expansión de acrónimos

Para evaluar la expansión de acrónimos realizada por la aplicación, se han realizado pruebas con todas las configuraciones posibles de la desambiguación, activando y desactivando las reglas de expansión y la comparación de contextos. Los resultados se muestran en la tabla 8.4.

Parámetros		Detección				Corrección			Nº. positivos	
Reg	Cont	P	R	F1	C	P	R	F1	PP	PA
No	No	0,83	0,76	0,79	0,56	0,47	0,42	0,44	30	33
No	Sí	0,83	0,76	0,79	0,60	0,50	0,45	0,48	30	33
Sí	No	0,83	0,76	0,79	0,68	0,57	0,52	0,54	30	33
Sí	Sí	0,83	0,76	0,79	0,72	0,60	0,55	0,57	30	33

Tabla 8.4: Resultados de expansión de acrónimos

Las conclusiones extraídas tras las pruebas realizadas son las siguientes:

- la detección se mantiene constante en todos los casos, ya que tanto las reglas de expansión como la comparación de contextos se aplican solo en la desambiguación. Para mejorar la detección, se deben introducir nuevos acrónimos en la base de datos de la aplicación.

- como era de esperar, los peores resultados de corrección se obtienen cuando no se aplican ni las reglas de expansión ni la comparación de contextos, y los mejores, cuando se aplican ambos.

- en el caso de utilizar solo uno de los sistemas, se puede observar que el sistema de reglas proporciona mejores resultados. No se trata de una conclusión generalizable, ya que la eficacia de ambos sistemas depende de las reglas y los contextos contenidos en la base de datos de la herramienta.
- por tanto, para aumentar la idoneidad de las expansiones hay dos posibles soluciones, consistentes ambas en la adición de nuevos recursos. Por un lado, el uso de un corpus de grandes dimensiones del que extraer contextos contribuiría a aumentar la eficacia del sistema de comparación. Por otro lado, la inclusión de nuevas reglas de expansión por parte de un experto mejoraría los resultados obtenidos al aplicarlas.

8.2.3. Fase de detección de negación

En el caso de la detección de negación no se han podido realizar pruebas con distintas configuraciones. Esto es debido a que NegEx -el algoritmo implementado en esta fase- no tiene opciones modificables que permitan variar su comportamiento, aparte de los recursos que utiliza (las listas de *triggers*).

Al igual que en las fases anteriores, se ha procesado un informe y se ha realizado su evaluación, comparandolo al mismo informe anotado a mano. En la tabla 8.5 se muestran los resultados.

Cues					Cues y tokens de ámbito					Neg. completas				
P	R	F1	PP	PA	P	R	F1	PP	PA	P	R	F1	PP	PA
1	0,96	0,98	23	24	0,68	0,99	0,80	148	101	0,70	0,67	0,68	23	24

Tabla 8.5: Resultados de detección de negación

Las conclusiones extraídas de los resultados son las siguientes:

- la detección de *cues* es casi perfecta, dado que la mayoría de ellos se han tenido en cuenta en las listas de *triggers*. Para mejorar la detección de *cues* y, por extensión, de negaciones, la solución pasa por mejorar los recursos del algoritmo con un mayor número de *triggers*.
- los resultados bajo el encabezado «Cues y tokens de ámbito» sirven de indicador directo de la detección de ámbito, debido a la eficacia casi perfecta de la detección de *cues*, que introduce un error despreciable en la medida.

- se observa pues que en la detección de ámbito se obtiene un muy buen *recall* y una *precision* comparativamente peor. Respectivamente, estos resultados indican que el algoritmo detecta casi todos los token de negación, pero clasifica como tales algunos tokens que no están negados. Es decir, no falta prácticamente ninguno, pero sí sobra una cierta cantidad de ellos.

- el error en la detección de los ámbitos es un resultado directo del algoritmo implementado. Mientras que para oraciones simples obtiene resultados coherentes, para frases de cierta longitud o de estructura más compleja detecta ámbitos excesivamente grandes. La solución a esta carencia consiste simplemente en la implementación de un algoritmo distinto o, al menos, en la modificación de NegEx para que utilice una heurística de detección de ámbitos más compleja.

8.2.4. Fase de identificación de conceptos

La anotación de conceptos en textos médicos requiere un conocimiento profundo de medicina y del campo de la documentación clínica, conocimiento que no poseemos. Por tanto, no se han podido anotar a mano informes completos.

No obstante, a modo de prueba se ha intentado anotar una sección de un informe. La anotación de dicha sección se consideró «viable» por su estructura: contiene una lista de pruebas y procedimientos realizados a un paciente. Los resultados obtenidos se muestran en la tabla 8.6.

Identificación de conceptos				
P	R	F1	PP	PA
0,67	0,59	0,63	15	17

Tabla 8.6: Resultados de identificación de conceptos

Por supuesto, los resultados obtenidos no son totalmente extrapolables al funcionamiento general de esta fase. Para elaborar las conclusiones sobre la eficacia de la extracción de conceptos, se ha realizado también una inspección visual de los conceptos obtenidos por la aplicación.

En general, consideramos que los resultados son aceptables, dadas las técnicas que se emplean, pero sin duda, son susceptibles de ser mejorados. A continuación, se

presentan los defectos observados más significativos y posibles soluciones para los mismos.

- En ocasiones, se obtienen conceptos con una puntuación de encaje alta, pero con un matiz ligeramente distinto al que se busca. Esto se debe principalmente a la no utilización de palabras derivadas que, por ejemplo, impide encontrar la correspondencia entre «ecografía abdominal» y «ecografía de abdomen».
- En otros casos, los problemas derivan de no utilizar sinónimos, lo que impide encontrar conceptos equivalentes semánticamente pero muy diferentes en su escritura, p.ej. «aerosolterapia» y «nebulización». Durante el desarrollo de esta fase, se intentaron utilizar sinónimos, como se expuso en el capítulo 7, y no se obtuvieron resultados significativamente mejores. Sería necesario revisitar la idea e intentar un nuevo enfoque.
- Por último, otra de las fuentes de errores es el *splitter*, que parte las frases en los fragmentos más pequeños que son procesados. El uso de una herramienta de análisis sintáctico para elegir las partes de la oración más significativas es una de las alternativas que podrían reducir este error.

Capítulo 9

Conclusiones y trabajo futuro

9.1. Conclusiones

El objetivo principal del proyecto era la construcción de una representación conceptual del contenido de informes médicos. Para obtener dicha representación se hizo necesario dividir el procesamiento del informe en diferentes fases sobre las que ir construyendo incrementalmente.

Antes de implementar cada fase tuvimos que realizar una importante labor de investigación, con el fin de analizar y estudiar las técnicas de procesamiento de lenguaje natural que podíamos emplear. Nosotros desconocíamos muchos de los métodos, pero a base de estudiar ciertos proyectos y artículos (citados en la bibliografía) logramos desarrollar las técnicas que consideramos adecuadas para cada fase.

La investigación y el posterior desarrollo no fueron sencillos porque la mayor parte del trabajo realizado en este ámbito es para la lengua inglesa. Esto junto con las limitaciones impuestas por el pequeño corpus con el que hemos contado fueron las mayores dificultades con las que hemos tenido que enfrentarnos. Por un lado, en el caso de las técnicas que emplean listas de palabras y patrones, era necesario adaptar al español los recursos y, en ocasiones, los algoritmos encontrados. Por otro lado, en el caso de los métodos estadísticos y de aprendizaje, nos encontramos con que simplemente no era posible utilizarlos: el pequeño tamaño del corpus de informes en español del que disponíamos impedía la creación de modelos lingüísticos aceptables.

Pese a los problemas que hemos tenido que confrontar, finalmente conseguimos implementar un sistema capaz de representar conceptualmente informes médicos y evaluar los resultados obtenidos en cada fase. Además, ofrece cierta flexibilidad, ya que permite configurar los recursos a utilizar, modificar el comportamiento de sus algoritmos y utilizar diferentes formatos de entrada y salida. Por tanto, consideramos que hemos cumplido con los objetivos iniciales del proyecto.

La calidad de los resultados obtenidos en cada fase son, en general, aceptables. En las fases de corrección ortográfica y expansión de acrónimos podríamos haber obtenido mejores resultados si hubiéramos dedicado más trabajo al desarrollo o la búsqueda de recursos más completos. De igual modo, en las fases de detección de negación e identificación de conceptos se podrían haber utilizado algoritmos algo más complejos para que la detección del ámbito de la negación o de la posición de un concepto hubieran sido más precisas. Pero, con el tiempo del que hemos dispuesto para desarrollar el proyecto, tuvimos que establecer ciertas prioridades sin poder profundizar todo lo que hubiéramos deseado en algunas fases del desarrollo. Aún así, en la siguiente sección se reflejan las líneas de trabajo futuro que nosotros no hemos podido desarrollar y que mejorarían los resultados del proyecto y ampliarían su funcionalidad.

En definitiva, creemos haber cumplido los objetivos propuestos en un proyecto que nos ha servido para familiarizarnos con la labor de investigación y con el que hemos aprendido a adaptar y complementar recursos y técnicas ya desarrolladas de acuerdo con nuestras necesidades, así como a tener en cuenta la complejidad del procesamiento del lenguaje natural y de la gran cantidad de técnicas que existen para ello.

9.2. Trabajo futuro

Este tipo de sistemas pueden ofrecer más posibilidades de las que hemos podido desarrollar. En el tiempo del que hemos dispuesto, hemos implementado las fases de corrección ortográfica, detección y desambiguación de acrónimos, detección de frases negadas e identificación de conceptos, además del modo evaluador y anotador. Las mejoras y ampliaciones más interesantes que se podrían hacer al sistema se presentan en las siguientes secciones.

9.2.1. Mejoras del sistema

Mejoras en la fase de corrección ortográfica

La ampliación de los recursos utilizados mejoraría los resultados de la corrección ortográfica. Existen multitud de términos, ya sean del idioma castellano o del ámbito médico, que no se encuentran en el diccionario que utiliza el corrector. Por tanto, un nuevo diccionario con más entradas, sobre todo terminología médica, mejoraría la eficacia de la fase.

Mejoras en la fase de detección y desambiguación de acrónimos

En la fase de desambiguación de acrónimos, el método de aprendizaje necesita de un corpus anotado del que poder aprender los contextos de los acrónimos. Nosotros utilizamos las descripciones de SNOMED CT como corpus, pero los contextos que el sistema aprende de este recurso son muy escuetos y muchas de las formas expandidas de los acrónimos de nuestro lexicón no aparecen en él.

Por tanto, la incorporación de un nuevo corpus anotado junto con el posterior proceso de aprendizaje mejoraría los resultados de dicha fase.

Mejoras en la fase de detección de la negación

La fase de negación ha sido implementada mediante el algoritmo NegEx. Este algoritmo está basado en el uso de expresiones regulares, por lo que no tiene en cuenta ni la sintaxis ni la semántica de las oraciones. Esto implica que el ámbito de las negaciones detectadas pueda ser erróneo o poco preciso.

La implementación de un algoritmo que trabaje con la sintaxis y la semántica de las frases permitiría una mejor detección del ámbito de las negaciones. Para ello, habría que contar con un corpus extenso de informes con el que entrenar al sistema y del que se pudiera aprender dicha información.

Mejoras en la fase de identificación de conceptos

Para mejorar el proceso de identificación de conceptos se podría construir una tabla de sinónimos. Con esto, se permitiría que se identificaran conceptos en textos en

los que, aunque no aparezcan las palabras que componen el concepto, sí aparezcan sinónimos suyos.

Implementar un algoritmo que trabaje con la sintaxis de las frases también podría resultar beneficioso para esta fase. De esta forma, se podría delimitar de forma más exacta los lugares dentro de las frases donde aparecen los conceptos. Para ello, como ya se comentó en la sección anterior, es necesario contar con un corpus extenso de informes del que aprender dicha información.

9.2.2. Ampliaciones de funcionalidad

Fase de detección de especulaciones

Además de las fases de procesamiento que hemos implementado, sería interesante agregar una nueva fase que detectara especulaciones dentro de los informes médicos. En medicina, antes de ofrecer un diagnóstico final, es normal que se hagan múltiples hipótesis y especulaciones que se verán reflejadas en los informes médicos.

Por tanto, a la hora de extraer la información de un informe médico es importante conocer que conceptos o frases son especulaciones. Para ello se puede implementar un algoritmo similar al NegEx, utilizado en la fase de detección de la negación, cambiando el tipo de patrones a detectar.

Fase de de-identificación

La de-identificación es un proceso que elimina de un informe toda la información que permita identificar al paciente, con el fin de proteger su privacidad y poder utilizar el texto en investigaciones y otros proyectos.

Todos los informes con los que hemos trabajado para desarrollar el sistema estaban anonimizados. Por tanto, no nos ha parecido prioritario desarrollar una fase de de-identificación. Pero en caso de trabajar con informes sin anonimizar, sería necesario implementar dicha fase para respetar la privacidad de los pacientes.

Recuperación de informes médicos similares a uno dado

El sistema ofrece como salida los conceptos que se han encontrado en el informe, indicando, entre otras cosas, la sección en que se encuentran, su estado (afirmado o negado) y su frecuencia de aparición en el informe.

Mediante el procesado de un corpus extenso de informes e historiales, sería posible la creación de una base de conocimiento de la que recuperar informes médicos similares a uno dado.

Esta posibilidad ofrecería diferentes aplicaciones como, por ejemplo, realización de informes estadísticos o apoyo en la valoración y diagnóstico de un paciente.

Búsqueda de informes médicos

Esta linea de trabajo futura se asemeja bastante a la anterior. La diferencia se encuentra en que en vez de recuperar informes médicos similares a uno dado se daría la posibilidad de insertar los criterios de búsqueda que se desearan. Por ejemplo, se podría hacer una búsqueda de informes en los que aparecieran ciertos conceptos en una determinada sección, con una cierta frecuencia o un cierto estado.

Adaptación a otros idiomas

Actualmente, el sistema está preparado para el procesamiento de informes en español, pero sería posible adaptar la herramienta a otros idiomas. Para ello, bastaría con modificar los ficheros de configuración con la información de los nuevos recursos a utilizar.

Apéndice A

Manual de usuario

En este apéndice se muestra la funcionalidad de la aplicación desde el punto de vista de los usuarios. Concretamente, se comienza explicando cómo se arranca la herramienta y después se presenta la forma de uso de cada uno de sus tres modos: procesamiento, anotación y evaluación.

A.1. Arranque y elección de modo

Al iniciar la aplicación, se presenta el menú principal, desde el que se puede acceder a los distintos modos de la aplicación (figura A.1): procesamiento, evaluación y anotación de informes.

Además, en el menú **Editar** se incluye el **Editor de reglas**, una herramienta auxiliar para gestionar las reglas que se usan en la fase de expansión de acrónimos.

A.2. Modo de procesamiento

En el modo de procesamiento los informes médicos son sometidos a cada una de las fases de análisis (corrección ortográfica, detección y expansión de acrónimos, detección de negaciones, etc.) generando un nuevo informe médico con las anotaciones necesarias para representar la información obtenida mediante el procesamiento.

Figura A.1: Menú principal de la aplicación

Tras hacer clic en el botón en el botón **Procesar y visualizar un informe** del menú principal (ver figura A.1) aparecerá una ventana de diálogo para la carga de archivos, en la cual se selecciona el informe que va a ser procesado. Este informe ha de tener extensión .xml (para los informes de prueba disponibles) o .txt.

Cuando se carga el informe, el sistema comienza a procesarlo. Una vez procesado aparecerá la ventana que se muestra en la figura A.2.

En la parte superior izquierda hay una pestaña para cada fase de análisis, de modo que al seleccionar una de ellas se mostrarán los resultados obtenidos en la fase correspondiente. Para ello, en la parte superior se mostrará el informe original y se resaltarán las palabras que han sido anotadas mediante subrayado y, en la parte inferior, habrá una lista con las palabras subrayadas arriba y la correspondiente información de la anotación.

Para localizar una palabra subrayada del texto en la lista basta con hacer clic derecho sobre la palabra subrayada, y, para localizar una palabra de la lista en el texto hay que hacer clic izquierdo sobre la palabra de la lista.

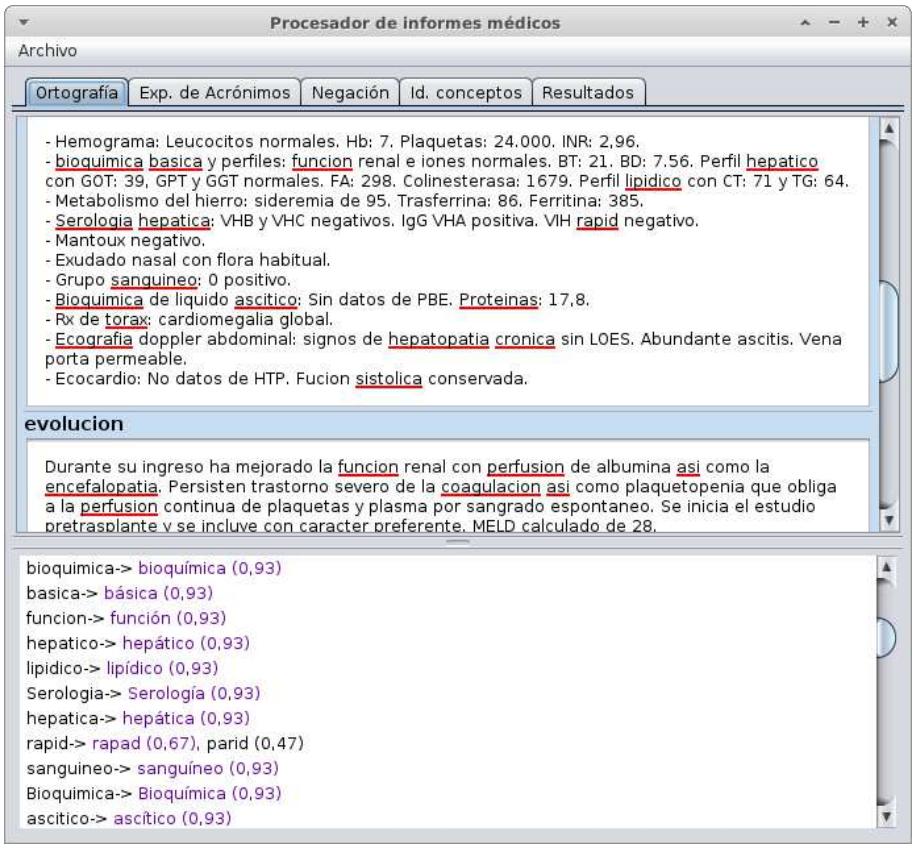

Figura A.2: Resultados obtenidos en el procesamiento del informe

A continuación, se explica la información que muestra el sistema en cada pestaña para cada fase de análisis.

Correción ortográfica: aparecerán subrayadas en rojo las palabras con faltas de ortografía y en la lista se podrán observar las palabras corregidas junto con la puntuación de confianza de la corrección.

Expansión de acrónimos: aparecerán subrayados en azul los acrónimos detectados y en la lista sus posibles expansiones ordenadas de mayor a menor confianza.

Detección de negación: aparecerán subrayadas en verde los palabras que se encuentran negadas y en la lista, la frase completa donde se encuentra cada negación, la palabra o expresión que indica negación (*cue*) y las palabras afectadas por la negación (el *scope*).

Identificación de conceptos: no aparecen subrayados los conceptos, dado que hay gran cantidad de ellos. En el panel inferior aparecen los conceptos detectados y, como siempre, haciendo clic en cualquiera de ellos se resaltará la parte del texto a la que corresponde la anotación. Entre la información de la anotación de tipo concepto, se encuentra la descripción, el ID. de SNOMED CT del concepto, la puntuación (o *perfect match* si ha habido una coincidencia perfecta) y el estado del concepto (afirmado o negado).

Resultados: se presenta en una pestaña independiente, pero se puede considerar parte de la fase de identificación de conceptos. Muestra una tabla resumen de los conceptos hallados en el informe (figura A.3). Se puede ordenar pulsando sobre las diferentes cabeceras, así como filtrar según diversos parámetros usando las cajas de selección.

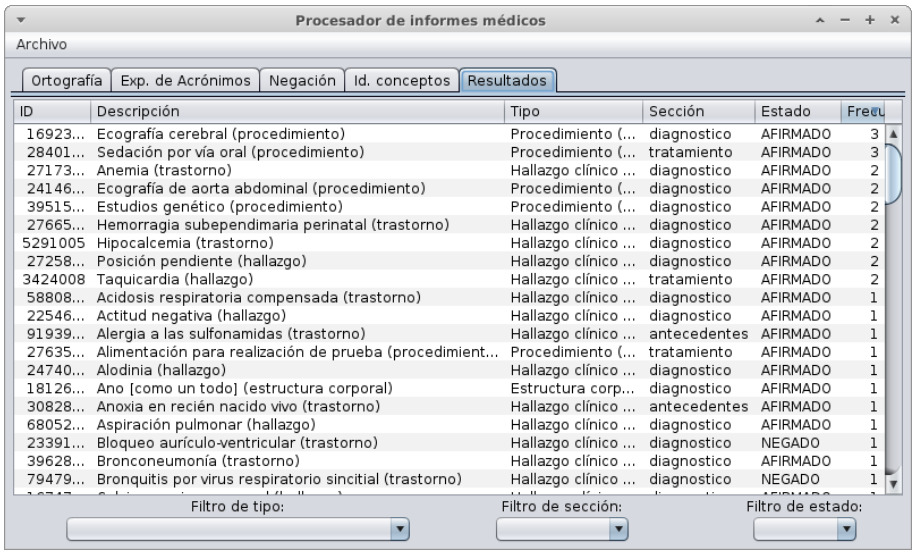

Figura A.3: Tabla resumen de los conceptos identificados del informe

Para guardar los resultados del procesamiento hay que hacer clic en el menú **Archivo** de la barra de menú y después en la opción deseada (ver figura A.4).

Guardar informe procesado genera un fichero con extensión .pxml, que contendrá un informe igual al de entrada pero anotado con toda la información obtenida en cada fase del procesamiento.

Guardar resultados genera un fichero con extensión .xml, que contendrá únicamente la información de la pestaña **Resultados**: descripción, identificador de SNOMED CT, tipo de concepto, etc.

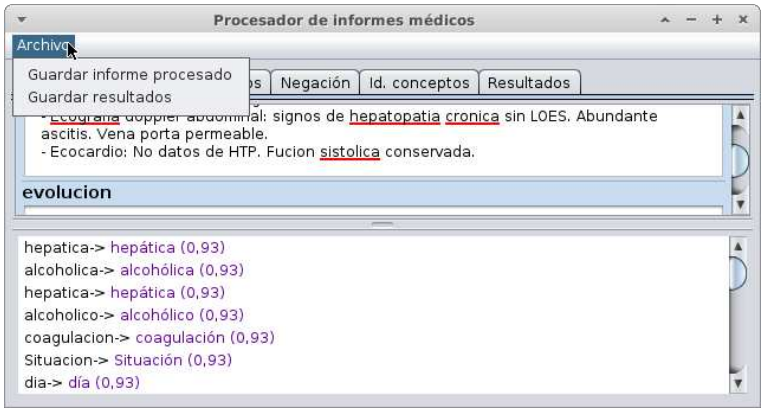

Figura A.4: Menú para guardar resultados

A.3. Modo de anotación

Este modo de funcionamiento sirve para introducir manualmente apuntes o marcas de los diferentes tipos de procesamiento de texto que trata la aplicación. Dichas anotaciones tienen como objetivo generar informes procesados «a mano», los cuales sirven para ser comparados en la evaluación con los informes procesados automáticamente por la aplicación.

Nada más seleccionar el modo anotador en la interfaz principal, se presentará una pantalla de carga, donde se puede seleccionar cualquier fichero de informe. Estos ficheros pueden ser de tres tipos, segun su extensión:

XML Originales, sin ningún tipo de procesamiento en ellos, al cargarlos aparecerá unicamente el texto de las distintas secciones del informe.

PXML Procesados por la aplicación automáticamente. Estos ficheros han sido guardados previamente tras usar la aplicación en el modo de procesamiento. Los informes procesados se pueden cargar para no partir de un informe completamente vacío como en el caso anterior, pudiendo quitar, modificar o

añadir anotaciones a los textos procesados automáticamente para generar así el informe anotado a mano deseado.

AXML Informes anotados manualmente. Son ficheros generados por este modo de la aplicación. Pueden ser cargados para añadir o modificar anotaciones.

Tras elegir el informe deseado y hacer clic en el botón **Abrir**, el contenido del informe aparece.

En las pestañas de la parte superior, se puede seleccionar cada una de las fases de análisis que trata la aplicación. Al seleccionar una de ellas, se verán las anotaciones sobre el texto correspondientes a dicha fase, al igual que en el modo principal. Si se ha seleccionado un informe no procesado ni anotado, no aparecerá ninguna marca sobre el texto (figura A.5).

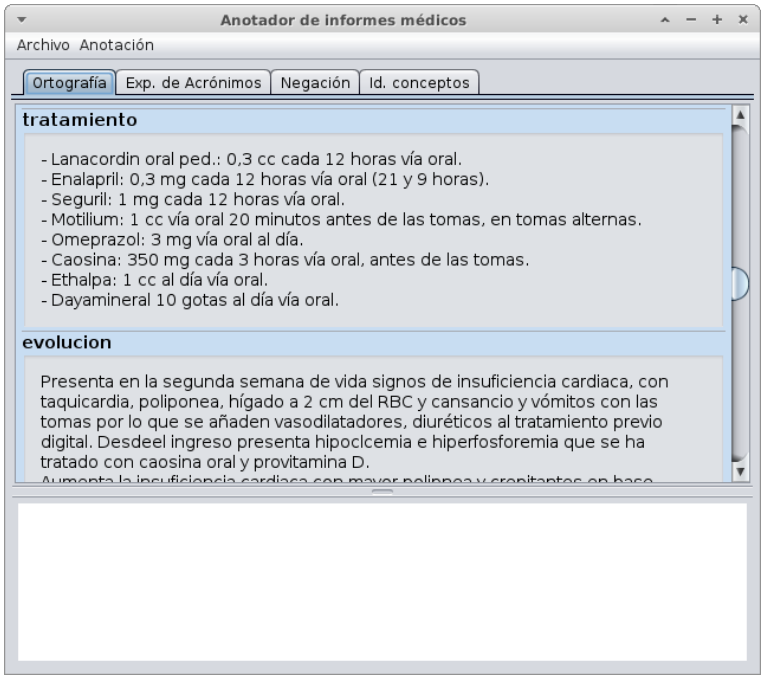

Figura A.5: Informe sin anotaciones cargado

A continuación se detalla la forma de anotar según la pestaña seleccionada, o tipo de anotaciones a realizar.

A.3.1. Modo de anotación: ortografía

Esta es la pestaña seleccionada por defecto. Para añadir una corrección al texto se debe hacer lo siguiente:

1. Seleccionar un token:
   Al pinchar en palabras dentro del texto, se irán seleccionando los tokens, apareciendo estos resaltados de color verde. Para corregir una palabra, se debe seleccionar únicamente un token. Para deseleccionar un token basta con volver a pinchar en la palabra, o clicar sobre una región distinta (no consecutiva) del texto.

2. Clic secundario en la sección:
   Dentro de la sección del texto donde queremos añadir la corrección, al realizar clic secundario aparecerá un menú contextual con la leyenda **Anotación nueva en...** (ver figura A.6). Al pincharlo pasamos al siguiente paso. Haciendo clic en cualquier otro lugar, desaparecerá el cuadro y no se añadirá la anotación.

3. Introducir la corrección:
   Tras el paso anterior aparecerá una pequeña ventana donde se debe introducir la corrección o forma correcta del token que estamos anotando y pulsar en aceptar (figura A.6). Tras esto la anotación queda creada y se añadirá al texto.

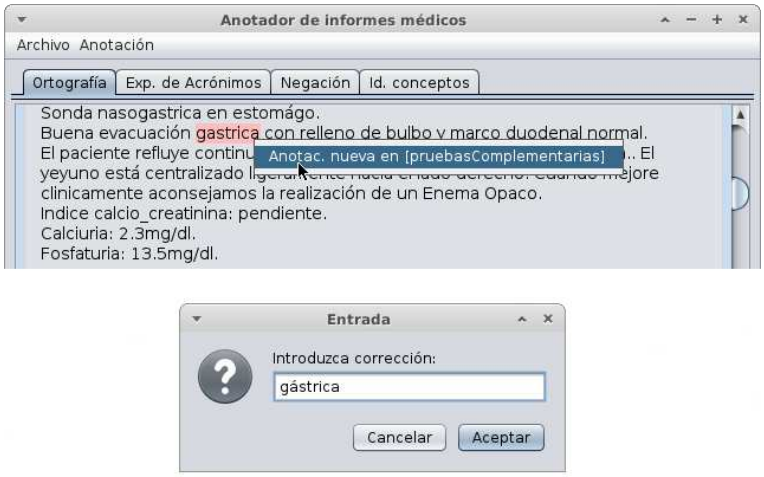

Figura A.6: Añadiendo una corrección

A.3.2. Modo de anotación: acrónimos

Este modo de funcionamiento es análogo a la introducción de anotaciones de ortografía (ver sección A.3.1). En lugar de introducir la forma correcta de un token escrito, en este caso al crear la anotación hay que introducir la expansión del acrónimo anotado.

A.3.3. Modo de anotación: negación

En este modo, a diferecia de los dos anteriores, la selección puede ser de uno o más tokens, ya que lo que se selecciona en primer lugar es el ámbito de la negación, o parte de la frase afectada por la cláusula de negación (*cue*).

Para añadir una anotación de negación se deben seguir los siguientes pasos:

1. Seleccionar el ámbito:
   Se deben seleccionar tokens como en otros tipo de anotación, con la diferencia de permitirse más de uno, y con la restricción de que dichos tokens deben pertenecer a una misma frase. La aplicación automáticamente detecta la frase de los tokens que se van seleccionando y la resalta.

2. Clic secundario para añadir anotación:
   Igual que en otros tipos de anotación, al hacer clic derecho en algún lugar de la sección, se podrá seleccionar crear una anotación nueva con la selección actual en dicha sección.

3. Seleccionar *cue*:
   Tras el paso anterior, aparecerá una ventana con una lista de tokens pertenecientes a la frase en la que antes hemos seleccionado el ámbito de la negación (ver figura A.7). En dicha lista hay que seleccionar el token o tokens correspondientes a la cláusula de negación *cue*. Haciendo clic en la lista se podrá seleccionar un token, o varios si se matiene pulsada la tecla -Ctrl- del teclado al seleccionar. Solo se permite la selección múltiple de tokens si son consecutivos.

Figura A.7: Selección de *cue*

A.3.4. Modo de anotación: conceptos

Este modo de funcionamiento es análogo a la introducción de anotaciones de ortografía (ver sección A.3.1). En lugar de introducir la forma correcta de un token escrito, en este caso al crear la anotación hay que introducir el ID del concepto para SNOMED CT que se quiere anotar. Cabe decir que no se comprueba la validez del identificador introducido.

A.3.5. Modo de anotación: eliminar anotaciones

En el panel situado bajo el texto del informe aparece la lista de anotaciones correspondientes a la pestaña seleccionada. Al pinchar una de ellas, se selecciona y queda resaltada en el texto del informe en color amarillo.

Para borrar una anotación, hay que seleccionarla en la lista y hacer clic en la opción **Eliminar Selección** que se encuentra dentro del menú **Anotación** en la barra de menú de la ventana (ver figura A.8). Se pedirá confirmación antes de borrar definitivamente la anotación seleccionada.

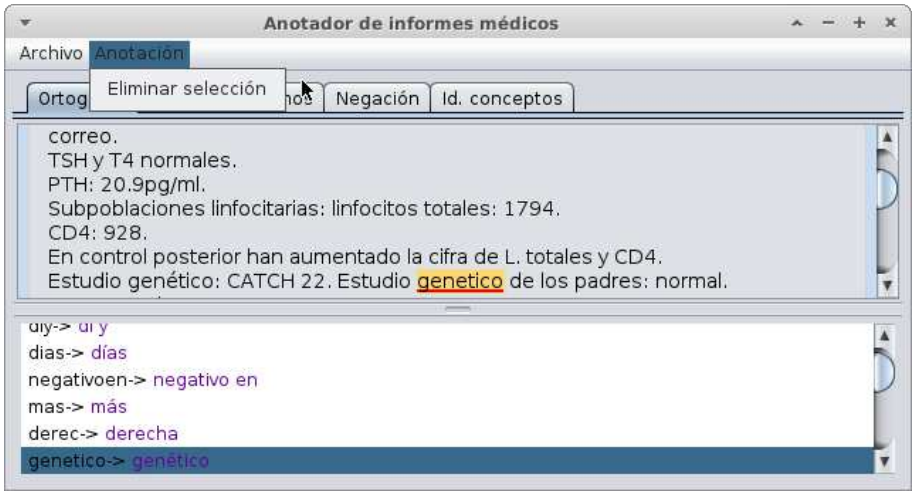

Figura A.8: Eliminando una anotación

A.4. Modo de evaluación

El propósito del modo de evaluación es el de evaluar la eficacia del modo de procesamiento. Para ello, este modo permite cargar simultáneamente pares de informes médicos anotados: unos resultado de procesar un informe original en el modo de procesamiento y otros anotados a mano por una persona usando el modo de anotación. La evaluación se realiza entonces comparando las anotaciones de los informes procesados automáticamente con las de los informes anotado a mano.

Tras hacer clic en el botón **Modo evaluador** del menú principal (figura A.1) se muestra la interfaz de carga de informes (ver figura A.9). En la parte de la izquierda se cargan los informes procesados por la aplicación (que suelen tener extensión .pxml) y en la derecha los anotados a mano (normalmente con extensión .axml). Para abrir los diálogos de carga de ficheros hay que clicar sobre los botones a la derecha de los campos de texto (etiquetados como «**...**»).

Para añadir otro par de informes hay que hacer clic sobre el botón **Añadir otro par** y para eliminar un par existente, se debe clicar el botón etiquetado con una **X**.

Al pulsar en **Evaluar** se presenta una lista de identificadores de los pares de informes (ver figura A.10) y clicando en el botón **Ver** de un par se muestra la interfaz con los resultados de la evaluación (figura A.11).

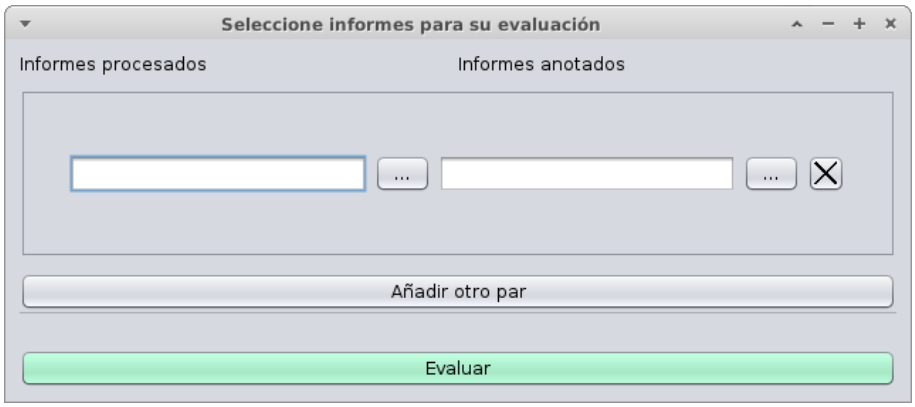

Figura A.9: Carga de pares de informes

Figura A.10: Lista de pares de informes

Pulsando sobre las pestañas que se encuentran en la parte superior izquierda de la aplicación se accede a los resultados de la evaluación de cada una de las fases.

A.4.1. Medidas empleadas

Al evaluar el funcionamiento de la aplicación, se presentan tres valores que miden la calidad del sistema de clasificación, entendiendo como tal el posicionamiento de distintos entes o elementos dentro de diferentes clases, e.g. la identificación de cada palabra como falta de ortografía o palabra bien escrita; o la identificación de cada frase del texto como negación o afirmación.

En concreto, las medidas utilizadas son *precision*, *recall* y F_1 *score*, y para poder interpretar los resultados del evaluador es necesario entender lo que representan. Toman valores reales de 0 a 1, siendo 0 el peor y 1 el mejor.

Precision indica cuántos de los elementos (palabras, frases) identificados como pertenecientes a la clase que se busca (faltas de ortografía, acrónimos, negaciones, etc.) lo son realmente. Por ejemplo, si se obtiene una *precision* de 0.7 eva-

luando la detección de faltas de ortografía, se tiene que de cada 10 faltas de ortografía identificadas por la aplicación, 7 son realmente faltas y 3 no lo son.

Recall indica cuántos de los elementos (palabras, frases) que realmente pertenecen a la clase que se busca (faltas de ortografía, acrónimos, negaciones, etc.) se han identificado correctamente. Por ejemplo, si se obtiene un *recall* de 0.6 evaluando la detección de faltas de ortografía, se tiene que de cada 10 faltas de ortografía que haya en el texto, la aplicación ha detectado 6.

F1 Score es una medida que combina la *precision* y el *recall*. Permite interpretar más claramente con una sola medida la calidad de la clasificación realizada.

A.4.2. Modo de evaluación: ortografía

Para la fase de detección de las negaciones se presentan dos conjuntos de resultados, compuestos ambos por *precision*, *recall* y F_1 *score*, pero con distintos matices (ver figura A.11).

Detección evalúa la clasificación de todas las palabras del texto en las clases «Falta de ortografía» y «Palabra escrita correctamente». Además, incluye una medida adicional llamada **Coincidencias** que indica el porcentaje de coincidencias existente entre las soluciones propuestas por el sistema y por el anotador humano para corregir las palabras que ambos clasifican como faltas.

Corrección evalúa de nuevo la clasificación de cada palabra como falta de ortografía, además de la corrección o idoneidad de la solución propuesta. En otras palabras, para contabilizar una falta y su corrección como un acierto, la palabra ha de ser considerada una falta de ortografía y la corrección debe ser igual a la proporcionada por el anotador humano.

Como información adicional, en un panel adjunto etiquetado como **Nº. positivos** se muestra el número de faltas de ortografía encontradas por el sistema y por el anotador humano.

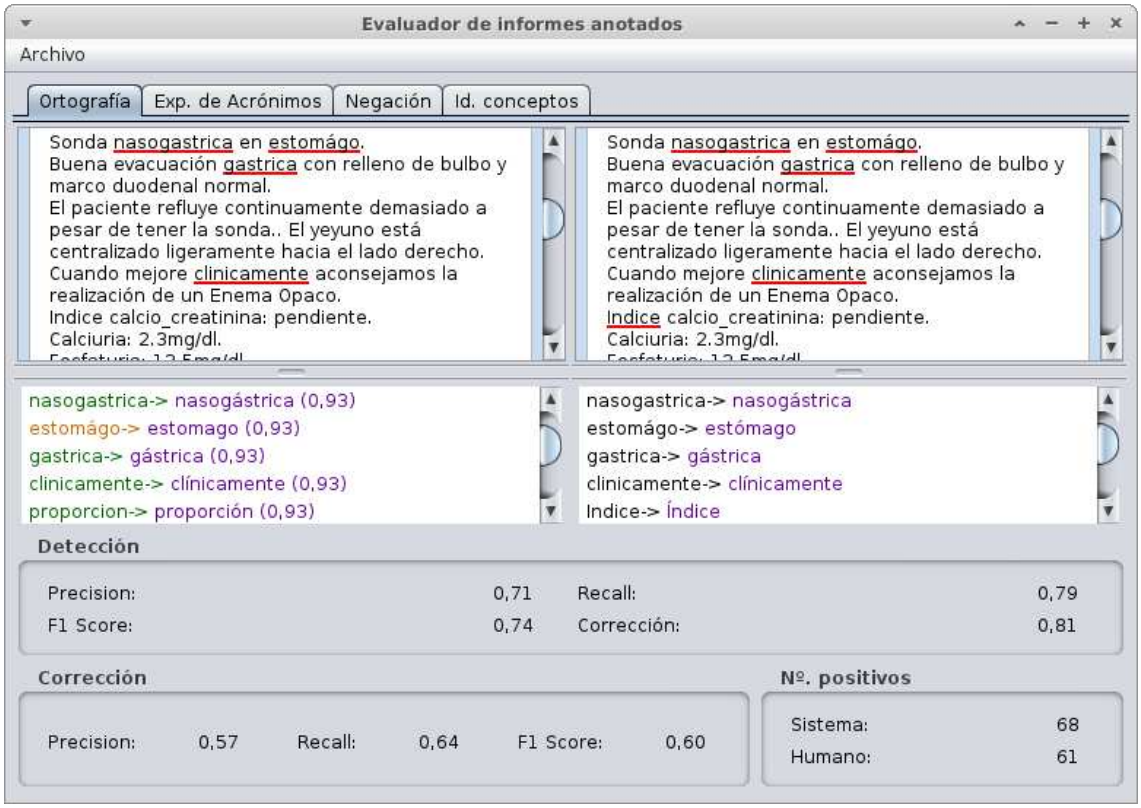

Figura A.11: Evaluación de la corrección ortográfica

A.4.3. Modo de evaluación: acrónimos

Los cálculos realizados en la evaluación de la fase de expansión de acrónimos son los mismos que para la fase de corrección ortográfica. Las únicas diferencias son que las palabras se clasifican entre acrónimos y palabras comunes (en lugar de entre faltas y palabras correctas), y que la solución propuesta es en este caso la expansión del acrónimo.

A.4.4. Modo de evaluación: negación

Para la fase de detección de las negaciones se presentan tres conjuntos de resultados, compuestos todos por *precision*, *recall* y F_1 *score*, pero que evalúan distintos matices de la tarea (ver figura A.12).

Detección de cue evalúa la clasificación de todas las palabras del texto en las clases «*Cue*» y «No *cue*». Como información adicional, en un panel adjunto

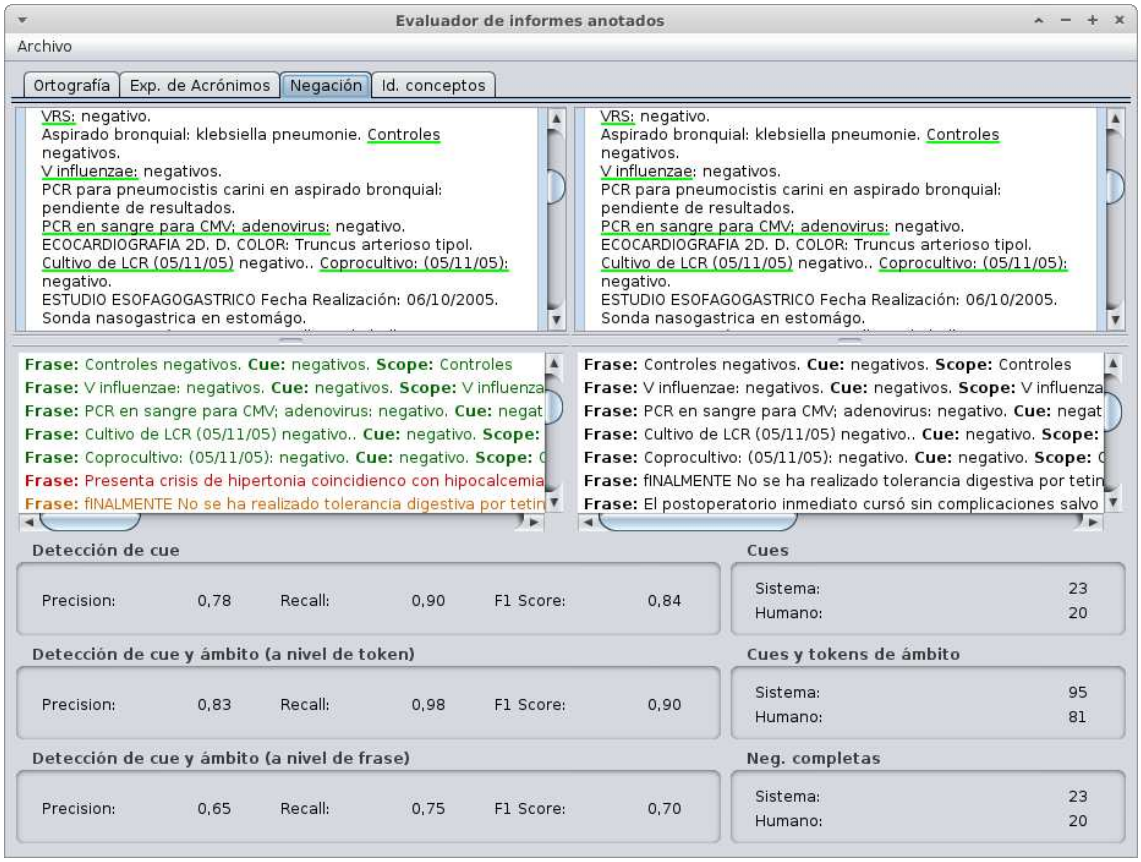

Figura A.12: Evaluación de la detección de negación

etiquetado como ***Cues*** se muestra el número de partículas negativas encontradas por el sistema y por el anotador humano.

Detección de cue y ámbito (a nivel de token) evalúa la clasificación de cada palabra del texto en las clases «Perteneciente a una negación» y «No perteneciente a una negación». Una palabra se considera perteneciente a una negación si es un *cue* o está dentro del ámbito de uno. Como información adicional, en un panel adjunto etiquetado como ***Cues* y tokens de ámbito** se muestra el número de palabras pertenecientes a una negación según el sistema y el anotador humano.

Detección de cue y ámbito (a nivel de frase) evalúa la clasificación de cada frase como negación, además de la corrección del *cue* y el ámbito identificados. A diferencia del caso anterior, se realiza la evaluación a nivel de frase, como un «todo», lo que quiere decir que, para que la clasificación de una frase se

considere acertada, tanto el *cue* como el ámbito deben coincidir exactamente con el esperado. Como información adicional, en un panel adjunto etiquetado como **Neg. completas** se muestra el número de negaciones identificadas por el sistema y por el anotador humano.

A.4.5. Modo de evaluación: conceptos

Para la fase de identificación de conceptos se presenta únicamente un conjunto de resultados, compuesto por *precision*, *recall* y F_1 *score*.

La comparación de dos conceptos se realiza simplemente comparando sus identificadores. Esta comparación produce un resultado binario, i.e. pueden ser iguales (se refieren al mismo concepto) o distintos (se refieren a conceptos diferentes). Es por ello que en la evaluación de esta fase no se presentan resultados parciales, como se hacía en las anteriores.

Como información adicional, en un panel adjunto etiquetado como **Nº. conceptos** se muestra el número de conceptos identificados por el sistema y por el anotador humano.

A.5. Editor de reglas

En el modo editor de reglas se pueden crear, editar o eliminar las reglas que se utilizarán en la desambiguación de acrónimos. Para acceder al editor hay que desplegar el menú **Editar** situado en la parte superior izquierda de la ventana principal y pulsar en **Editor de reglas**, lo que mostrará la ventana de la figura A.13.

En la mitad superior aparecen las reglas existentes en el sistema mientras que la mitad inferior está dedicada a la creación y edición de reglas.

Para crear una regla nueva basta con seleccionar el acrónimo, el tipo, la sección y la expansión de la regla y si el tipo de regla lo requiere también hay que introducir el contexto. Para ello, introducimos cada palabra del contexto separada por un espacio de la siguiente palabra. Si queremos que en el contexto aparezca cualquier número se podrá introducir la cadena de caracteres [numero] para evitar tener que escribir

todos los números. Una vez rellenados todos los campos pulsamos en **Guardar** y la regla habrá sido creada.

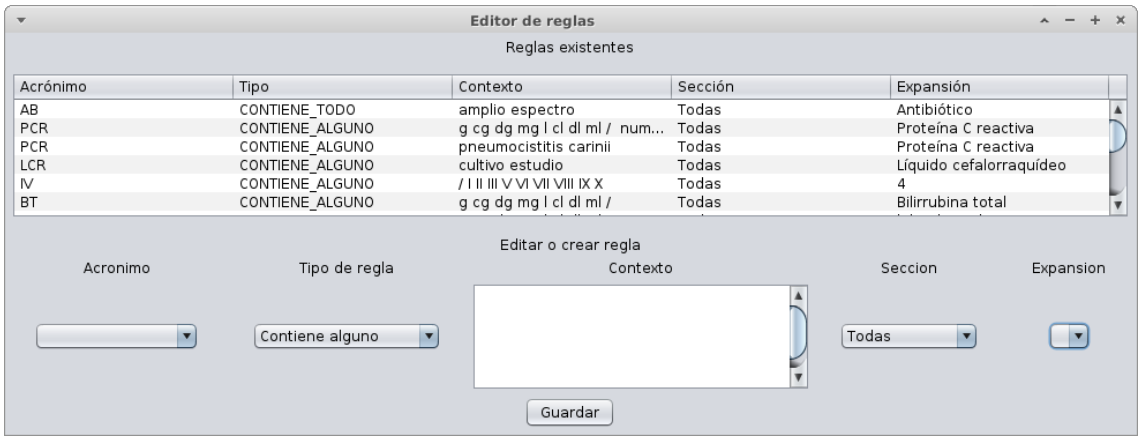

Figura A.13: Editor de reglas

Para editar una regla existente hay que hacer clic derecho sobre la regla que queramos editar y pulsar sobre **Editar regla**. Los campos de la regla aparecerán en la parte inferior para poder ser editados. Una vez terminada la edición, se pulsa sobre **Guardar** para almacenar los cambios.

Para eliminar una regla basta con hacer clic derecho sobre la regla que queramos editar y pulsar sobre **Borrar regla**. La regla desaparecerá del sistema.

Apéndice B

Configuración de la aplicación

A continuación se presentan los archivos de configuración de la aplicación, con los que se puede designar los recursos que usa el sistema.

En el caso de la configuración relativa a la identificación de conceptos, también se permite establecer correspondencias entre las secciones de entrada y las de salida y priorizar o prohibir cierto tipo de conceptos.

B.1. correccion.properties

```
1  ##########################################################
2  # Fichero de configuracion de la fase de correccion
3  ##########################################################
4  # Ruta de la libreria Hunspell
5  LIB_HUNSPELL=lib/CorrAcr/hunspell/
6  # Directorio de los diccionarios
7  HOMEDICC=resources/Diccionario/
8  # Nombre del diccionario a utilizar (sin la extension)
9  DICCIONARIO=es-ES
10 # Numero maximo de sugerencias del corrector
11 NUMERO_SUGERENCIAS=5
12 # Peso distancia Levenshtein
13 PESO_LEVENSHTEIN=0.33
14 # Peso distancia Needleman Wunsch
```

```
PESO_NEEDLEMAN=0.33
# Peso distancia fonetica
PESO_FONETICA=0.33
# Umbral que debe sobrepasar la sugerencia final
UMBRAL=0.9
```

B.2. acronimos.properties

```
############################################################
# Fichero de configuracion de la fase de acronimos
############################################################
# Recurso de acronimos que se quiere utilizar
# (xml, bbdd o thes)
SELECCION_RECURSO=bbdd
# Archivo XML de expansiones de acronimos
ACRONIMOS_XML=./resources/Acronimos.xml
# Directorio de la base de datos
URL_BBDD=resources/BBDD
# Usuario de la base de datos de SNOMED
USUARIO_BBDD=usuario
# Clave de la base de datos de SNOMED
CLAVE_BBDD=1234
# Fichero de acronimos que no hay que expandir
URL_EXCEPCIONES=resources/excepcionesAcronimos
# Tabla de descripciones de conceptos
NOMBRE_BBDD_SNOMEDCT=snomed.db
# Base de datos de acronimos
NOMBRE_BBDD_ACRONIMOS=acronimos.db
# Base de datos de reglas
NOMBRE_BBDD_REGLAS=acronimos.db
# Tabla de reglas
NOMBRE_TABLA_REGLAS=acronim_rules
# Nombre de la tabla de descripciones
NOMBRE_TABLA_DESCRIP=t_descripciones_snomed
# Nombre de la tabla de acronimos
```

```
NOMBRE_TABLA_LISTA_ACRONIMOS=acronim_list
# Nombre de la tabla de expansiones
NOMBRE_TABLA_EXPANSION_ACRONIMOS=acronim_description
# Tabla de acronimos
NOMBRE_BBDD_THESAURO=acronimos.db
# Nombre de la tabla de descripciones
NOMBRE_TABLA_THESAURO=acronim_description
# Nombre de la tabla de features de los acronimos
NOMBRE_TABLA_FEATURES=features
# Indica si se utilizan las reglas de expansion
USO_REGLAS=true
# Indica si se utiliza la comparacion entre
# contextos para la desambiguacion
USO_CONTEXTOS=false
# Tipo de la base de datos: mysql o sqlite
TIPO_BBDD=sqlite
```

B.3. negacion.properties

```
##########################################################
# Fichero de configuracion de la fase de negacion
##########################################################
# Ruta de los triggers de NegEx
RUTA_TRIGGERS_NEGEX=./resources/negExTriggers
# Fichero de triggers de negacion
NOMBRE_FICHERO_NEG=negPhrases.txt
# Fichero de triggers de pseudo-negacion
NOMBRE_FICHERO_PSENEG=pseNegPhrases.txt
# Fichero de triggers de post-negacion
NOMBRE_FICHERO_POSTNEG=postNegPhrases.txt
# Fichero de triggers de conjunciones
NOMBRE_FICHERO_CONJ=conjunctions.txt
```

B.4. conceptos.properties

```
#############################################################
# Fichero de configuracion de la fase de conceptos
#############################################################
# Tabla de conceptos
RUTA_BBDD_SNOMED_C=./resources/BBDD/concepsnomed.db
# Tabla de descripciones de conceptos
RUTA_BBDD_SNOMED_D=./resources/BBDD/snomed.db
# Tabla de relaciones entre conceptos
RUTA_BBDD_SNOMED_R=./resources/BBDD/relsnomed.db
# Nombre de la tabla de conceptos
NOMBRE_TABLA_CONCEPTOS=conceptos
# Nombre de la tabla de descripciones
NOMBRE_TABLA_SNOMED=descripciones_snomed
# Nombre de la tabla de relaciones
NOMBRE_TABLA_RELACIONES=relaciones_snomed
# Indice de Lucene
RUTA_IND_LUCENE=./resources/BBDD/snomedluceneIndex
# Ruta de stop words
RUTA_FICHERO_STOP_WORDS=./resources/stopWords.txt
# Ruta de filtros para las querys
RUTA_FILTROS_QUERY=./resources/filtrosQuery.txt
# Nombre de la columna de identificador en la tabla
# de conceptos
NOMBRE_COL_ID=concept_id
# Nombre de la columna de descripciones en la tabla
# de conceptos
NOMBRE_COL_DESC=fsn
# Nombre de la columna de tipo en la tabla de conceptos
NOMBRE_COL_TIPO=superior
# Umbral de puntuacion para aceptar un concepto
UMBRAL=0.5

#############################################################
# Mapeo de secciones del informe procesado (valores
```

```
35 # separados por comas) a secciones de salida de la
36 # aplicacion (claves)
37 # Las metasecciones definidas deben ser consistentes con
38 # su posterior uso en otras opciones de configuracion de
39 # este fichero
40
41 METASECCIONES=antecedentes|diagnostico|tratamiento
42 antecedentes=motivoIngreso|antecedentesFamiliares|
      antecedentesPersonales
43 diagnostico=juicioClinico|pruebasComplementarias|
      enfermedadActual
44 tratamiento=planTerapeutico|evolucion|tratamiento|
      intervencionQuirurgica
45
46 ############################################################
47 # IDs de SNOMED CT
48 ############################################################
49 # Relacion es-un
50 CPT_ES_UN=116680003
51 # Conceptos superiores (padres de las jerarquias)
52 FUERZA_FISICA=78621006
53 PROCEDIMIENTO=71388002
54 EVENTO=272379006
55 ENTIDAD_OBSERVABLE=363787002
56 LOCALIZACION=308916002
57 ESTRUCTURA_CORPORAL=123037004
58 FARMACO=373873005
59 CONTEXTO_SOCIAL=48176007
60 ORGANISMO=410607006
61 SITUACION_CTX=243796009
62 SUSTANCIA=105590001
63 ESCALAS=254291000
64 OBJETO_FISICO=260787004
65 ESPECIMEN=123038009
66 CALIFICADOR=362981000
67 CPTO_ESPECIAL=370115009
68 ELEMENTO_REGISTRO=419891008
```

```
HALLAZGO_CLINICO=404684003
CPT_ENLACE=106237007

############################################################
# Tipos de conceptos permitidos en cada metaseccion
# Claves del tipo cp_NOMBRE-METASECCION
cp_diagnostico=HALLAZGO_CLINICO,PROCEDIMIENTO,
   ESTRUCTURA_CORPORAL,ORGANISMO,FARMACO,SUSTANCIA
cp_antecedentes=SITUACION_CTX,CONTEXTO_SOCIAL,
   HALLAZGO_CLINICO,PROCEDIMIENTO,ESTRUCTURA_CORPORAL,
   ORGANISMO,FARMACO,SUSTANCIA
cp_tratamiento=HALLAZGO_CLINICO,PROCEDIMIENTO,
   ESTRUCTURA_CORPORAL,ORGANISMO,FARMACO,SUSTANCIA

############################################################
# Tipos de conceptos mas probables segun la seccion
# - Solo se permite uno -
# Claves del tipo cp_prob_NOMBRE-METASECCION
cp_prob_diagnostico=HALLAZGO_CLINICO
cp_prob_antecedentes=HALLAZGO_CLINICO
cp_prob_tratamiento=PROCEDIMIENTO
```

B.5. splitter.properties

```
############################################################
# Fichero de configuracion del splitter de frases
############################################################
# Codificacion de los ficheros de patrones del splitter
SPLITTER_ENCODING=UTF-8
# Ruta de los patrones del splitter
RUTA_REGEXP=./resources/sentSplitterRE
# Patrones de separacion de parrafos o bloques de texto
EXTERNAL_SPLIT_LIST=external-split-patterns.txt
# Patrones de separacion de frases de un parrafo
INTERNAL_SPLIT_LIST=internal-split-patterns.txt
```

```
12 # Ampliacion de los anteriores patrones para obtener
13 # subfrases en la fase de conceptos
14 INTERNAL_SPLIT_CONCEPTOS=internal-split-concep.txt
15 # Patrones que podrian parecer splits, pero no lo son
16 NON_SPLIT_LIST=non-split-patterns.txt
```

Apéndice C

Formato de entrada y salida

C.1. Formato de entrada

El formato de entrada y el procedimiento de carga de datos dependen totalmente de los tipos de informe de los que se disponga. Por tanto, no se va a tratar en detalle sobre el tema.

Para desarrollar la aplicación se ha utilizado un pequeño corpus de informes en formato XML. Como no se disponía de un fichero XML Schema que definiera los elementos válidos, se utilizó Trang[1] para inferirlo de los informes disponibles.

A partir de ese *schema* se utilizó JAXB[2] (Java Architecture for XML Binding) para cargar los ficheros. Dicho sistema permite mapear automáticamente clases Java a una representación XML mediante los procesos de *marshal* (para convertir objetos Java a XML) y *unmarshal* (para lo contrario, i.e. convertir XML en objetos Java). Además, proporciona la herramienta *xjc*, que se ha usado para generar automáticamente las clases Java usadas en el *marshalling* y el *unmarshalling* partiendo de ficheros XML Schema.

Aparte de lo ya mencionado, también se ha implementado carga de ficheros de texto plano, en la que se considera todo el texto del archivo como contenido de una única sección.

[1] http://www.thaiopensource.com/relaxng/trang.html
[2] http://jaxb.java.net/

C.2. Formato de salida

La aplicación genera archivos de salida con extensiones .xml, .pxml y .axml, para ficheros de resultados, ficheros de procesamiento y ficheros anotados a mano respectivamente; pero realmente son todos XML.

Los ficheros .pxml y .axml se ajustan al *schema schema_informe_procesado.xsd* y contienen informes con anotaciones de todas las fases. El hecho de que tengan extensiones diferentes se debe simplemente a la necesidad de distinguir claramente que ficheros ha anotado la aplicación y cuales ha anotado un usuario para poder realizar evaluaciones.

Por su parte, los ficheros .xml de salida se corresponden con el *schema salida.xsd* y contienen solo la información de los conceptos identificados por la aplicación. Este tipo de salida se creó para facilitar la carga de los resultados en otros sistemas y además omitir los datos de carácter personal que pueda contener el texto del informe.

Para cargar y guardar estos ficheros utilizamos JAXB, que ya se mencionó en la sección anterior.

Para cambiar la representación de las salidas, habría que seguir los siguientes pasos:

- Modificar los ficheros XML Schema mencionados anteriormente en esta sección.
- Regenerar las clases asociadas ejecutando el comando indicado en los comentarios del *schema* en cuestión y copiar el código resultante en el paquete apropiado.
- Modificar el código de la clase de gestor apropiada (*GestorSalida* o *GestorInformesProcesados*) para conectar las clases generadas con *xjc* con las clases del modelo de datos de la aplicación, realizando también en estas los cambios que sean necesarios.

C.2.1. salida.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Upon change, generate Java classes by executing -->
<!-- xjc -p modelos.salida.xml salida.xsd             -->
<!-- Si se realizan cambios en este documento, se     -->
<!-- deben regenerar las clases Java ejecutando       -->
<!-- xjc -p modelos.salida.xml salida.xsd             -->
<xs:schema targetNamespace="http://www.analizador.textual.si"
    xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns="http://www.analizador.textual.si">

  <xs:simpleType name="tipoEstado">
     <xs:restriction base="xs:string">
        <xs:enumeration value="afirmado"    />
        <xs:enumeration value="negado"      />
        <xs:enumeration value="especulado"  />
     </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="tipoInfoConcepto">
     <xs:sequence>
        <xs:element name="id"       type="xs:unsignedLong" />
        <xs:element name="frec"     type="xs:integer"      />
        <xs:element name="desc"     type="xs:string"       />
        <xs:element name="seccion"  type="xs:string"       />
        <xs:element name="tipo"     type="xs:unsignedLong" />
        <xs:element name="estado"   type="tipoEstado"      />
     </xs:sequence>
  </xs:complexType>

  <xs:complexType name="tipoResultados">
     <xs:sequence>
        <xs:element name="origen"    type="xs:string"      />
        <xs:element name="resultado" type="tipoInfoConcepto"
                    minOccurs="0"    maxOccurs="unbounded" />
     </xs:sequence>
```

```
36    </xs:complexType>
37
38    <xs:element name="resultados" type="tipoResultados" />
39
40  </xs:schema>
```

C.2.2. schema_informe_procesado.xsd

```
1   <?xml version="1.0" encoding="UTF-8"?>
2   <!-- Upon change, generate Java classes by executing -->
3   <!-- xjc -p modelos.procesado.xml schema_informe_procesado.
       xsd -->
4   <!-- Si se realizan cambios en este documento, se     -->
5   <!-- deben regenerar las clases Java ejecutando       -->
6   <!-- xjc -p modelos.procesado.xml schema_informe_procesado.
       xsd -->
7   <xs:schema targetNamespace="http://www.analizador.textual.si"
8       xmlns:xs="http://www.w3.org/2001/XMLSchema"
9       xmlns="http://www.analizador.textual.si">
10
11    <xs:complexType name="tipoCorreccion">
12      <xs:sequence>
13        <xs:element name="error" type="xs:string" />
14        <xs:element name="corr" type="xs:string"
15                    minOccurs="1" maxOccurs="unbounded" />
16        <xs:element name="punt" type*/="xs:double"
17                    minOccurs="0" maxOccurs="unbounded" />
18      </xs:sequence>
19    </xs:complexType>
20
21    <!-- Tipo auxiliar necesario para saltos de linea -->
22    <xs:complexType name="tipoBr">
23      <xs:complexContent>
24        <xs:restriction base="xs:anyType" />
25      </xs:complexContent>
```

```
26      </xs:complexType>
27
28      <xs:complexType name="tipoAcronimo" mixed="true">
29         <xs:sequence minOccurs="0" maxOccurs="unbounded">
30            <xs:element name="correccion"
31                        type="tipoCorreccion" />
32         </xs:sequence>
33         <xs:attribute name="exp" type="xs:string"
34                       use="required" />
35         <xs:attribute name="punt" type="xs:double"
36                       use="optional"/>
37      </xs:complexType>
38
39      <xs:complexType name="tipoConcepto" mixed="true">
40         <xs:choice minOccurs="0" maxOccurs="unbounded">
41            <xs:element name="acronimo" type="tipoAcronimo" />
42            <xs:element name="correccion"
43                        type="tipoCorreccion" />
44            <xs:element name="br" type="tipoBr" />
45         </xs:choice>
46         <xs:attribute name="id" type="xs:positiveInteger"
47                       use="required" />
48      </xs:complexType>
49
50      <xs:complexType name="tipoNegacion" mixed="true">
51         <xs:choice minOccurs="0" maxOccurs="unbounded">
52            <xs:element name="correccion"
53                        type="tipoCorreccion" />
54            <xs:element name="acronimo" type="tipoAcronimo" />
55            <xs:element name="concepto" type="tipoConcepto" />
56            <xs:element name="br" type="tipoBr" />-~
57         </xs:choice>
58         <xs:attribute name="cue" type="xs:string" />
59      </xs:complexType>
60
61      <xs:complexType name="tipoEspeculacion" mixed="true">
62         <xs:choice minOccurs="0" maxOccurs="unbounded">
```

```
63          <xs:element name="correccion"
64                      type="tipoCorreccion" />
65          <xs:element name="acronimo" type="tipoAcronimo" />
66          <xs:element name="concepto" type="tipoConcepto" />
67          <xs:element name="br" type="tipoBr" />
68       </xs:choice>
69    </xs:complexType>
70
71    <xs:complexType name="tipoFrase" mixed="true">
72       <xs:choice minOccurs="0" maxOccurs="unbounded">
73          <xs:element name="correccion"
74                      type="tipoCorreccion" />
75          <xs:element name="acronimo" type="tipoAcronimo" />
76          <xs:element name="concepto" type="tipoConcepto" />
77          <xs:element name="negacion" type="tipoNegacion" />
78          <xs:element name="especulacion"
79                      type="tipoEspeculacion" />
80          <xs:element name="br" type="tipoBr" />
81       </xs:choice>
82    </xs:complexType>
83
84    <xs:complexType name="tipoContenido">
85       <xs:sequence minOccurs="1" maxOccurs="unbounded">
86          <xs:element name="frase" type="tipoFrase" />
87       </xs:sequence>
88    </xs:complexType>
89
90    <xs:complexType name="tipoSeccion">
91       <xs:sequence>
92          <xs:element name="contenido" type="tipoContenido" />
93       </xs:sequence>
94       <xs:attribute name="nombre" type="xs:string"
95                     use="required" />
96    </xs:complexType>
97
98    <xs:complexType name="tipoDato">
99       <xs:simpleContent>
```

```
100         <xs:extension base="xs:string">
101            <xs:attribute name="nombre" type="xs:string"
102                          use="required" />
103         </xs:extension>
104      </xs:simpleContent>
105   </xs:complexType>
106
107   <xs:complexType name="tipoInforme">
108      <xs:sequence>
109         <xs:element name="dato" type="tipoDato"
110                     minOccurs="0" maxOccurs="unbounded" />
111         <xs:element name="seccion" type="tipoSeccion"
112                     maxOccurs="unbounded" />
113      </xs:sequence>
114   </xs:complexType>
115
116   <xs:element name="informe" type="tipoInforme" />
117
118 </xs:schema>
```

Apéndice D

Glosario

Alcance de una negación: conjunto de palabras de una frase que se ven afectadas por una negación.

Ámbito de una negación: alcance de una negación.

Base de conocimiento: base de datos para la gestión de conocimiento, usado en nuestro caso para referirnos a bases de datos contenedoras de redes semánticas de conceptos médicos.

Contexto de un token: palabras situadas alrededor de un token de las cuales depende el sentido o significado de dicho token.

Corpus: conjunto lo más extenso y ordenado posible de datos o textos, utilizado en análisis estadísticos y algoritmos para extraer conocimiento de ellos.

Cue: palabra o cadena de palabras con un significado de negación, p. ej. no, negativo, nunca.

Diccionario: lista de palabras conocidas para un sistema.

Envoltorio: adaptación de la interfaz a un sistema que hace posible el acceso a sus servicios desde un sistema cliente.

Expansión de un acrónimo: intercambio de un acrónimo por las palabras que hacen explícito su significado.

Expresión regular: expresión que describe un conjunto de cadenas sin enumerar sus elementos, usando operadores de concatenación, alternación y cuantificación sobre los caracteres de un alfabeto.

Señal de negación: *cue*.

Lexicón: diccionario.

Modelo de n-gramas: modelo lingüístico que indica la probabilidad de aparición de secuencias determinadas de N palabras en un texto.

Parser: analizador sintáctico de expresiones derivadas de una gramática concreta.

Patrón: expresión regular.

PLN: procesamiento de lenguaje natural.

Prefijo de una palabra: subcadena de una palabra que comienza con el carácter inicial, p. ej. «h», «ho», «hol» y «hola» son prefijos de «hola», pero no así «o», «ola» y «la», entre otros.

Programación dinámica: método algorítmico que construye la solución a un problema a partir de las soluciones calculadas previamente y almacenadas de subproblemas más sencillos.

Scope: alcance de una negación.

Splitter: sistema capaz de partir un texto en unidades más pequeñas, como frases o tokens.

Token: unidad elemental obtenida por un *parser* o un *splitter*. En nuestro caso, una palabra, un signo de puntuación u otros símbolos (+, /, etc).

Trigger: patrón utilizado en el algoritmo Negex.

Bibliografía

A.R. Aronson. Effective mapping of biomedical text to the umls metathesaurus: the metamap program. In *Proceedings of the AMIA Symposium*, page 17. American Medical Informatics Association, 2001.

Miguel Ballesteros, Alberto Díaz, Virginia Francisco, Pablo Gervás, Jorge Carrillo de Albornoz, and Laura Plaza. Ucm-2: a rule-based approach to infer the scope of negation via dependency parsing. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 288–293, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S12-1038.

Jorge Carrillo de Albornoz, Laura Plaza, Alberto Díaz, and Miguel Ballesteros. Ucm-i: A rule-based syntactic approach for resolving the scope of negation. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 282–287, Montréal, Canada, 7-8 June 2012. Association for Computational Linguistics. URL http://www.aclweb.org/anthology/S12-1037.

Wendy W. Chapman, Will Bridewell, Paul Hanbury, Gregory F. Cooper, and Bruce G. Buchanan. A simple algorithm for identifying negated findings and diseases in discharge summaries. *Journal of Biomedical Informatics*, 34 (5):301 – 310, 2001. ISSN 1532-0464. doi: 10.1006/jbin.2001.1029. URL http://www.sciencedirect.com/science/article/pii/S1532046401910299.

Noa P. Cruz Díaz, Manuel J. Maña López, and Jacinto. Mata Vázquez. Aprendizaje automático versus expresiones regulares en la detección de la negación y la especulación en biomedicina. 2010. ISSN 1135-5948. URL http://rua.ua.es/dspace/handle/10045/14709.

F.J. Damerau. A technique for computer detection and correction of spelling errors. *Communications of the ACM*, 7(3):171–176, 1964.

Dana Dannélls. Acronym recognition: Recognizing acronyms in swedish texts. Master's thesis, Department of Linguistics, University of Gothenburg, Gothenburg, Sweden, June 2006.

R.W. Hamming. Error detecting and error correcting codes. *Bell System Technical Journal*, 29(2):147–160, 1950.

Carol Friedman Hongfang Liu, Alan R. Aronson. A study of abbreviations in medline abstracts. pages 1–5, 2002. URL http://lhncbc.nlm.nih.gov/lhc/docs/published/2002/pub2002051.pdf.

Leah S. Larkey, Paul Ogilvie, M. Andrew Price, and Brenden Tamilio. Acrophile: an automated acronym extractor and server. In *In ACM DL*, pages 205–214, 2000.

V.I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710, 1966.

C.D. Manning, P. Raghavan, and H. Schutze. *Introduction to information retrieval*, volume 1. Cambridge University Press Cambridge, 2008.

Tara McIntosh. Unsupervised discovery of negative categories in lexicon bootstrapping. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, EMNLP '10, pages 356–365, Stroudsburg, PA, USA, 2010. Association for Computational Linguistics. URL http://dl.acm.org/citation.cfm?id=1870658.1870693.

Stéphane Meystre and Peter J. Haug. Natural language processing to extract medical problems from electronic clinical documents: Performance evaluation. *Journal of Biomedical Informatics*, 39(6):589 – 599, 2006. ISSN 1532-0464. doi: 10.1016/j.jbi.2005.11.004. URL http://www.sciencedirect.com/science/article/pii/S1532046405001140.

David Nadeau and Peter D. Turney. A supervised learning approach to acronym identification. In *In 8th Canadian Conference on Artificial Intelligence (AI'2005) (LNAI 3501*, pages 319–329, 2005.

S.B. Needleman and C.D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of molecular biology*, 48(3):443–453, 1970.

L. Philips. Hanging on the metaphone. *Computer Language*, 7(12 (December)), 1990.

James Pustejovsky, Jose Castano, Brent Cochran, Maciej Kotecki, Michael Morrell, and Anna Rumshisky. Extraction and disambiguation of acronym-meaning pairs in medline, 2004.

P.H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, pages 787–793, 1974.

Maria Skeppstedt. Negation detection in swedish clinical text: An adaption of negex to swedish. *Journal of Biomedical Semantics*, 2(Suppl 3):S3, 2011. ISSN 2041-1480. doi: 10.1186/2041-1480-2-S3-S3. URL http://www.jbiomedsem.com/content/2/S3/S3.

Kazem Taghva and Jeff Gilbreth. Recognizing acronyms and their definitions. *ISRI (Information Science Research Institute) UNLV*, 1:191–198, 1999.

R.A. Wagner and M.J. Fischer. The string-to-string correction problem. *Journal of the ACM (JACM)*, 21(1):168–173, 1974.

Zhonghua Yu. Automatic resolution of ambiguous abbreviations in biomedical texts using support vector machines and one sense per discourse hypothesis. In *In Proceedings of the SIGIR'03 Workshop on Text Analysis and Search for Bioinformatics Edited by: Brown*, pages 57–62. ACM Press, 2003.