CAMPUS
DE EXCELENCIA
INTERNACIONAL

Graduado en Matemáticas e Informática

Universidad Politécnica de Madrid

Escuela Técnica Superior de

Ingenieros Informáticos

TRABAJO FIN DE GRADO

Técnicas de machine learning para la detección de la negación en textos clínicos en español

Autor: Miguel Zaballa Pardo

Director: Ernestina Menasalvas

MADRID, ENERO 2016

Resumen

En los últimos años han surgido nuevos campos de las tecnologías de la información que exploran el tratamiento de la gran cantidad de datos digitales existentes y cómo transformarlos en conocimiento explícito. Las técnicas de Procesamiento del Lenguaje Natural (NLP) son capaces de extraer información de los textos digitales presentados en forma narrativa. Además, las técnicas de machine learning clasifican instancias o ejemplos en función de sus atributos, en distintas categorías, aprendiendo de otros previamente clasificados.

Los textos clínicos son una gran fuente de información no estructurada; en consecuencia, información no explotada en su totalidad. Algunos términos usados en textos clínicos se encuentran en una situación de afirmación, negación, hipótesis o histórica. La detección de esta situación es necesaria para la estructuración de información, pero a su vez tiene una gran complejidad.

Extrayendo características lingüísticas de los elementos, o tokens, de los textos mediante NLP; transformando estos tokens en instancias y las características en atributos, podemos mediante técnicas de machine learning clasificarlos con el objetivo de detectar si se encuentran afirmados, negados, hipotéticos o históricos.

La selección de los atributos que cada token debe tener para su clasificación, así como la selección del algoritmo de machine learning utilizado son elementos cruciales para la clasificación. Son, de hecho, los elementos que componen el modelo de clasificación.

Consecuentemente, este trabajo aborda el proceso de extracción de características, selección de atributos y selección del algoritmo de machine learning para la detección de la negación en textos clínicos en español.

Se expone un modelo para la clasificación que, mediante el algoritmo J48 y 35 atributos obtenidos de características lingüísticas (morfológicas y sintácticas) y disparadores de negación, detecta si un token está negado en 465 frases provenientes de textos clínicos con un F-Score del 73%, una exhaustividad del 66% y una precisión del 81% con una validación cruzada de 10 iteraciones.

Abstract

New information technologies have emerged in the recent years which explore the processing of the huge amount of existing digital data and its transformation into knowledge. Natural Language Processing (NLP) techniques are able to extract certain features from digital texts. Additionally, through machine learning techniques it is feasible to classify instances according to different categories, learning from others previously classified.

Clinical texts contain great amount of unstructured data, therefore information not fully exploited. Some terms (tokens) in clinical texts appear in different situations such as affirmed, negated, hypothetic or historic. Detecting this situation is necessary for the structuring of this data, however not simple.

It is possible to detect whether if a token is negated, affirmed, hypothetic or historic by extracting its linguistic features by NLP; transforming these tokens into instances, the features into attributes, and classifying these instances through machine learning techniques.

Selecting the attributes each instance must have, and choosing the machine learning algorithm are crucial issues for the classification. In fact, these elements set the classification model.

Consequently, this work approaches the features retrieval as well as the attributes and algorithm selection process used by machine learning techniques for the detection of negation in clinical texts in Spanish.

We present a classification model which, through J48 algorithm and 35 attributes from linguistic features (morphologic and syntactic) and negation triggers, detects whether if a token is negated in 465 sentences from historical records, with a result of 73% F-Score, 66% recall and 81% precision using a 10-fold cross-validation.

Índice general

Índice de figuras

Índice de tablas

1. INTRODUCCIÓN

1.1. Introducción

Las tecnologías de la información han supuesto cambios profundos en la manera de hacer nuestras tareas, comportarnos y comunicarnos. Estas tecnologías tienen una característica especial: nuestros trabajos, conexiones y comunicaciones generan datos, datos que se almacenan. Esto hace que la cantidad de información digital crezca sin límites. Para hacerse un ligera idea, la cantidad aproximada de información que se manejaba en Internet en 2013 era aproximadamente 1.2 zettabytes (10^{21} bytes) y la cantidad de información que se genera por segundo en Internet de unos 22000 gigabytes. Wal-Mart, una multinacional de tiendas americana maneja más de un millón de transacciones de clientes cada hora, alimentando bases de datos de más de 2,5 petabytes. Facebook maneja ya 40.000 millones de fotos. (The Economist, 2010)

Esto plantea nuevos retos en cuanto a con qué objetivos y con qué herramientas podemos utilizar esta inmensa cantidad de datos. Entre los objetivos del uso de esta información puede estar reforzar o debilitar argumentos, ayudar a la toma de decisiones o incluso predecir acontecimientos. Normalmente, los especialistas de un sector concreto no tienen el tiempo, el conocimiento o las herramientas necesarias para analizar toda esta información. Nace entonces la ciencia de los datos o *Data Science*. Pero esto no es todo, esta ciencia tiene además otros retos, por ejemplo, encontrar nuevas fuentes de datos, valorar si los datos son relevantes, extraer únicamente los datos relevantes o crear herramientas para que los especialistas obtengan información sobre estos datos. (Dhar, 2013)

Los textos escritos en lenguaje natural son una fuente de datos, por lo tanto objeto de estudio de esta ciencia. Esta información no se encuentra estructurada, es decir, está en un formato en el que es difícil de ser procesada por medios automáticos por un ordenador. Explicado con un ejemplo: si en un libro de historia quiero saber si Suárez fue presidente del gobierno español en 1980 la única opción será leer todo el libro hasta en encontrar una oración parecida a "Adolfo Suárez fue presidente del gobierno español de 1976 a 1981", si el libro está digitalizado probablemente la mejor opción será buscar la palabra "Suárez" en el buscador de la página hasta encontrar esa oración. Sin embargo, si lo que quiero es buscar cuántos presidentes del gobierno ha habido desde 1976 hasta 1993 y no existe una frase del tipo "Hubo 3 presidentes del gobierno desde 1976 hasta 1993 en España" el problema se complica.

Sin entrar en detalle en las maneras en de hacer la estructuración y extracción de la información, dado que no es el problema que nos atañe, parece lógico que si se quiere resolver el problema anterior es necesaria la "descomposición" del lenguaje para su posterior análisis. El campo del procesamiento del lenguaje natural (NLP, por sus siglas en inglés) tiene este objetivo, que la información en lenguaje natural pueda ser extraída de manera fácil y sencilla. Para ello se basa en la lingüística computacional, que utiliza principalmente el análisis morfológico, léxico, sintáctico y semántico.

Las técnicas de machine learning (o aprendizaje automático) son algoritmos de clasificación, agrupamiento, asociación o decisión en función de características o

patrones de ciertas instancias. Los ámbitos de aplicación de estas técnicas son muchos y distintos, uno de ellos es la identificación de características del lenguaje planteado en el párrafo anterior. Así, una de sus aplicaciones pude ser que aprendan a clasificar y nos digan donde se encuentran las negaciones, afirmaciones, hipótesis o históricos en un texto, estas son características intrínsecas del lenguaje que ayudan a la estructuración.

Surge entonces otra cuestión, ¿cuáles son exactamente las características o rasgos lingüísticos que permiten identificar en un texto que hay una negación? Mediante machine learning se podría obtener un modelo que permitiera partiendo de unas características predecir los elementos mencionados. Uno de los elementos claves será identificar las características del texto que harán que el modelo obtenido tenga una buena precisión.

Así, este trabajo se enmarca dentro de un proyecto de procesamiento del lenguaje natural y *Data Science*, en general. Sin embargo, su esencia consistirá en técnicas de análisis de datos para la optimización de técnicas de machine learning para la clasificación. Se considera esto el contexto científico del trabajo.

1.2. Motivación

Un campo de aplicación importante del NLP y *Data Science* es la medicina. Los textos clínicos son una fuente de información con varios rasgos a destacar: i) gran cantidad, tanto de historiales clínicos como de artículos de medicina, ii) constante crecimiento, lo que mejorará la precisión del sistema en el futuro, iii) alta fiabilidad, debido a la rigurosidad con que estos deben ser escritos, iv) la reciente predisposición de los hospitales a digitalizar los historiales clínicos.

Todas las características mencionadas hacen idónea la aplicación de técnicas de NLP sobre textos clínicos. El procesamiento del lenguaje natural en textos clínicos no es trivial en absoluto. A pesar de que el lenguaje médico intenta ser bastante preciso comparado con el lenguaje que usamos al comunicarnos o en otros ámbitos, siempre quedan ambigüedades, abreviaciones o erratas que deben ser reconocidas por el sistema y no son fáciles de detectar e interpretar.

1.3. Objetivos

Consiguientemente, el objetivo del presente trabajo es el estudio y análisis de técnicas de machine learning que optimicen la detección de la negación, la afirmación, la hipótesis o el histórico en textos clínicos en español.

Para la consecución de este objetivo proponemos los siguientes objetivos parciales:

- Analizar los algoritmos más representativos de machine learning, identificando las diferencias entre ellos y sus distintas aplicaciones.
- Entender el funcionamiento de un sistema de procesamiento de lenguaje natural.
- Determinar el conjunto de variables que afectan al reconocimiento de la negación, la hipótesis, el histórico y la afirmación de términos en textos clínicos.
- Diseñar algoritmos de machine learning que maximicen la detección de la negación, la hipótesis, el histórico y la afirmación de términos en textos clínicos.

- Diseñar los pasos para la integración del módulo desarrollado en el flujo del proceso de lenguaje natural.
- Evaluar, analizar y comparar los resultados obtenidos para poder mejorar el sistema utilizando otros algoritmos o con cambios de parámetros del mismo.
- Interpretar los resultados finales y entender dónde están las limitaciones de rendimiento del sistema.

1.4. Estructura del trabajo

El presente trabajo consta de tres partes conceptuales diferenciadas:

- Una primera parte de análisis y planteamiento del problema, en la que se introduce el problema, las herramientas necesarias para su resolución y se estudian soluciones similares realizadas por otros. En el índice corresponde a las secciones: "1. Introducción", "2. Estado de la cuestión" y "3. Planteamiento del problema".
- Una parte central, en la que se describe el diseño de la solución, correspondiente a la sección "4. Solución". Esta se divide en tres:
  - "4.1. Arquitectura del sistema para la detección de la negación".
  - "4.2.Anotación y extracción de características".
  - "4.3.Selección de atributos para el aprendizaje".
- Una tercera parte en la que se analizan y valoran los resultados obtenidos y se exponen posibles líneas futras del trabajo que se corresponde a la sección "5. Conclusiones"**.**

2. ESTADO DE LA CUESTIÓN

2.1. Trabajos relacionados

Introducimos a continuación cuatro trabajos que han sido de especial interés para el análisis del problema. Los dos primeros trabajos a mencionar tienen el mismo objetivo que el nuestro pero utilizando únicamente expresiones regulares para la clasificación, el primero para textos en inglés y el segundo para textos en español. El tercer trabajo relacionado sí utiliza técnicas de machine learning sobre características del lenguaje natural pero su finalidad última es distinta a la nuestra. El último trabajo incluido utiliza técnicas de machine learning para la detección de la negación y la especulación en textos clínicos, este último es el más similar, y por tanto relevante.

2.1.1. Detección de la negación en inglés mediante expresiones regulares

El algoritmo *ConText* (Harkema, Dowling, Thornblade, & Chapman, 2009) basado en que una palabra se encuentra afirmada por defecto. Además mediante expresiones regulares, busca disparadores (triggers), estos son palabras almacenadas en una lista susceptibles de negar, poner en pasado o especular el significado de las palabras que se encuentran en su ámbito (scope). En la implementación de *ConText*, el ámbito que se ve afectado por el disparador llega hasta el final de la oración o hasta que se encuentra un término de terminación. Sin embargo en la versión del algoritmo para la negación llamada *NegEx* e implementada por los mismos autores, el ámbito del disparador está además limitado por una longitud de seis tokens.

2.1.2. Detección de la negación en español mediante expresiones regulares

Otro trabajo (Zhang, 2014), que se enmarca dentro del mismo proyecto NLP que este TFG, presenta una implementación del algoritmo *ConText* para textos clínicos en español, por lo que sus datos de entrada y salida han estado a mi disposición. Concretamente, ha sido útil la lista de disparadores (triggers) de entrada.

2.1.3. Análisis de sentimiento

El análisis de sentimiento es el uso de NLP, análisis de texto y lingüística computacional para extracción de información subjetiva en un texto. Su objetivo es medir la negatividad o positividad de un texto de opinión. Intenta conocer en qué medida está el escritor de acuerdo sobre un tema. Esto tiene una gran aplicación en redes sociales, marketing o artículos de opinión. Lo interesante para nosotros no son los campo de aplicación de este tipo de trabajos sino los algoritmos de machine learning que utilizan para su mejora.

Le prestamos atención a un estudio (Soher, y otros, 2013) que considera que su modelo captura con bastante precisión los efectos y el ámbito de la negatividad en distintos niveles del árbol semántico. Su modelo se basa en la creación del árbol

semántico de cada oración, y la dotación de un grado de positividad a cada palabra de la oración. Después, entrena estas características mediante técnicas de machine y deep learning, concretamente redes neuronales recursivas, para conocer el grado de negatividad o positividad.

Sacamos varias conclusiones acerca de este trabajo. Primero, el análisis de la negación tiene una gran complejidad, en consecuencia, la utilización de técnicas de machine learning puede ser muy positiva. Por último, hemos de anotar una diferencia sustancial el grado de negatividad de los textos es subjetivo, comparable y relativo a otros grados de negatividad, sin embargo la negación es detectable caso a caso, salvando situaciones de ambigüedad.

2.1.4. Detección de la negación mediante técnicas de machine learning

Una vez concluido que las técnicas de machine learning serán las que mejor se aproximen a la solución, buscamos trabajos que utilicen estos métodos para la mejora de la detección de ciertas características del lenguaje únicamente mediante expresiones regulares.

La tesis de Cruz N. P. (2014) se centra en el análisis de la negación y la especulación en textos clínicos en inglés. Propone un sistema estructurado en dos fases de clasificación. Una primera que clasifica cada palabra en disparador de negación/especulación. Y una segunda fase que determina el ámbito de la oración sobre el que actúa el disparador.

Obtuvo un F-score para la primera fase de 97.3% y 94.9% para la negación y la especulación, respectivamente. El porcentaje de ámbitos correctos para tokens bien clasificados fueron de 90.9% en negación y 71.9% en especulación (Cruz, López, Vázquez, & Álvarez, 2013). Dado que consideramos estos unos buenos resultados, enfocaremos la solución de manera similar, sobretodo en cuanto al diseño, a la de esta tesis.

3. PLANTEAMIENTO DEL PROBLEMA

3.1. Marco teórico

El objetivo de este TFG es analizar las características y algoritmos de machine learning que detecten con la mayor exactitud si una frase o sintagma se encuentra en situación de afirmación, negación, hipótesis o histórico. Esto, como hemos mencionado en la introducción, es esencial en la estructuración de la información.

Concretamente en este trabajo nos centraremos en la detección de la negación, entendiendo que el diseño de una solución muy similar se podría aplicar a la afirmación, la hipótesis, el histórico o la referencia a otra persona. *Véase* "5.2.4.Detección de la afirmación, la hipótesis y el histórico".

Cuando un especialista busca aquellos casos en los que los pacientes tuvieron cáncer, como primer paso de la búsqueda, se podrían seleccionar todos los historiales donde la palabra cáncer apareciera mencionada. Sin embargo, esto solo supondría una aproximación. Es claro que si desecháramos todos aquellos historiales en los que aparecieran frases con la palabra cáncer negada con relación al paciente como: "*las pruebas mostraron negatividad en cáncer" o "no había signos de cáncer de páncreas",* nos estaríamos acercando considerablemente a la solución. Esto demuestra la importancia del módulo de detección de la negación para la estructuración de la información.

Dado un texto digitalizado en español nuestro sistema debe detectar si un token se encuentra en una situación de negación. Entendemos por token la unidad léxica mínima. En textos en lenguaje natural los tokens son palabras, números y signos de puntuación. De esta manera, clasificaremos cada palabra, número y signo de puntuación como negado o no negado.

Conociendo la aplicación de este trabajo sobre textos clínicos, de manera intencionada y cuando sea posible, el sintagma etiquetado será un término clínico. Por ejemplo, en la oración: *"Las pruebas mostraron negatividad en cáncer",* únicamente la palabra cáncer debe tener una etiqueta que indique negación. Quedará de la siguiente manera: *"Las pruebas mostraron negatividad en <negación>cáncer</negación>"*. También podría considerarse etiquetar el sintagma completo: *<negación>en cáncer</negación>*, sin embargo el etiquetado de una preposición como es "en" no aporta valor, por lo que se descarta esta idea.

Para la detección de la negación parece razonable pensar que las palabras o el sintagma que se encuentran a continuación de una palabra que indique negación como lo es "*no*" o "*negatividad*" son sintagmas negados. Esta es la manera de detectar la negación de los sistemas mediante expresiones regulares. Sin embargo, es claro que hay numerosos casos donde esto es insuficiente, por ejemplo: *"pruebas de cáncer negativas"* o *"fiebre y dolores ausentes"* contienen la palabra que indica negación después del término negado, por lo que no serían detectables.

Además, es interesante analizar la negación también en las siguientes tres frases "*nunca ha tenido cáncer*", "*no ha tenido cáncer nunca*" y "*no ha tenido cáncer*", en las

tres el término cáncer debe estar negado, las oraciones tienen estructuras distintas y curiosamente la segunda frase tiene dos elementos que indican negación, que son "*no*" y "*nunca*" y la doble negación sigue siendo negación. Sin embargo, en la siguiente frase encontrada en el corpus del sistema no ocurre lo mismo: *"no presentaba linfoma no Hodgkin"* ya que ahora el segundo *"no"* no funciona como un adverbio de negación.

Quedan todavía una serie de frases importantes a tener en cuenta. Estas son aquellas frases que contienen una negación pero el término negado no se ve afectado de la misma manera que lo harían otros. Por ejemplo: *"no se descartan otras hemorragias internas"* o *"no se negó a tomar la medicación"*. Tenemos dos escenarios posibles, considerar como negados los términos que gramaticalmente están afectados por la negación: *<negación> descartan otras hemorragias internas<\negación>* y *<negación> tomar la medicación<\negación>* o no tomarlos como una negación y considerar los términos como hipotéticos o posibles para que otro módulo los trate. Ha sido esta segunda opción la tomada en este trabajo.

Es relativamente fácil para una persona distinguir si una frase o sintagma se encuentra negada, en hipótesis o en histórico. Sin embargo, como hemos visto, debido a la complejidad del lenguaje natural este problema no es fácil de automatizar. Por estas razones introducimos métodos de machine learning que generan un modelo y aprenden de situaciones anteriores.

Los métodos de aprendizaje automático o machine learning para la clasificación aprenden de unos ejemplos previamente etiquetados para poder, ante un nuevo ejemplo no etiquetado, predecir su etiqueta. En nuestro caso a los algoritmos se les suministrará un texto dividido en tokens, estos han sido etiquetados como negación o no negación. El algoritmo deberá aprender un modelo que ante un nuevo texto sin etiquetar sepa predecir la etiqueta de negación de cada token.

Un aspecto fundamental aquí es definir cuáles son los atributos de cada token utilizados para construir un modelo, y en base a los cuales el algoritmo hará la predicción.

3.2. **Herramientas**

Se describen en este apartado las herramientas que se han utilizado en la resolución del problema.

3.2.1. **UIMA Java Framework**

UIMA (Unstructured Information Management Architecture) es una arquitectura basada en componentes e implementación de frameworks para crear, descubrir, componer e implementar distintos modos de analizar información no estructurada, así como su integración con tecnologías de búsqueda. Las aplicaciones para la gestión de información no estructurada analizan volúmenes grandes de información no estructurada para descubrir conocimiento que es relevante para un usuario final.

Concretamente se ha utilizado el Framework de Apache de Java para la plataforma Eclipse. Este framework se utiliza para procesar textos planos e identificar sus propiedades.

Un programa simple de UIMA funciona de la siguiente manera: se le pasa como entrada un texto plano, que llamaremos SofA (Subject of Analysis), el motor de análisis (anlysis engine) lo procesa y, en función de la lógica que se haya establecido, da como salida un archivo XMI que contiene enlazadas las anotaciones del texto. La combinación de varios analysis engine permite obtener varios tipos de anotaciones sobre un mismo texto y en mismo archivo de salida XMI. Cada tipo de anotación crea su propia lista enlazada de elementos. De esta manera es fácil extraer características o propiedades del texto, combinarlas y analizarlas.

En nuestro caso el SofA es el texto clínico, el elemento a analizar. Crearemos tipos de anotaciones para dividir el texto en frases y tokens, así como tipos de anotaciones que nos den características lingüísticas relevantes para la detección de la negación.

UIMA se integra en la parte del proyecto que se encarga de conectar el flujo del proyecto de procesamiento de lenguaje natural con la selección de atributos de la negación. Concretamente la sección "Creación de anotaciones".

3.2.2. **Weka**

Weka es una colección de algoritmos para tareas de data mining desarrollados en Java y de software libre. Los algoritmos pueden ser aplicados directamente a un conjunto de datos, o llamados desde otro código Java. Además, contiene herramientas para el preprocesado de datos, clasificación, regresión, agrupación, reglas de asociación y visualización. También permite desarrollar nuevos esquemas de machine learning.

4. SOLUCIÓN

Se introduce ahora la solución al problema. Se diferencian tres partes: i) la arquitectura y los elementos del proceso para la detección de la negación mediante técnicas de machine learning, ii) la creación de anotaciones y extracción de características de un texto, iii) el proceso de selección de características y algoritmos que maximizan la detección de la negación.

4.1. Arquitectura del sistema para la detección de la negación

A continuación, se explica la estructura de un sistema de machine learning para la detección y anotación de la negación:

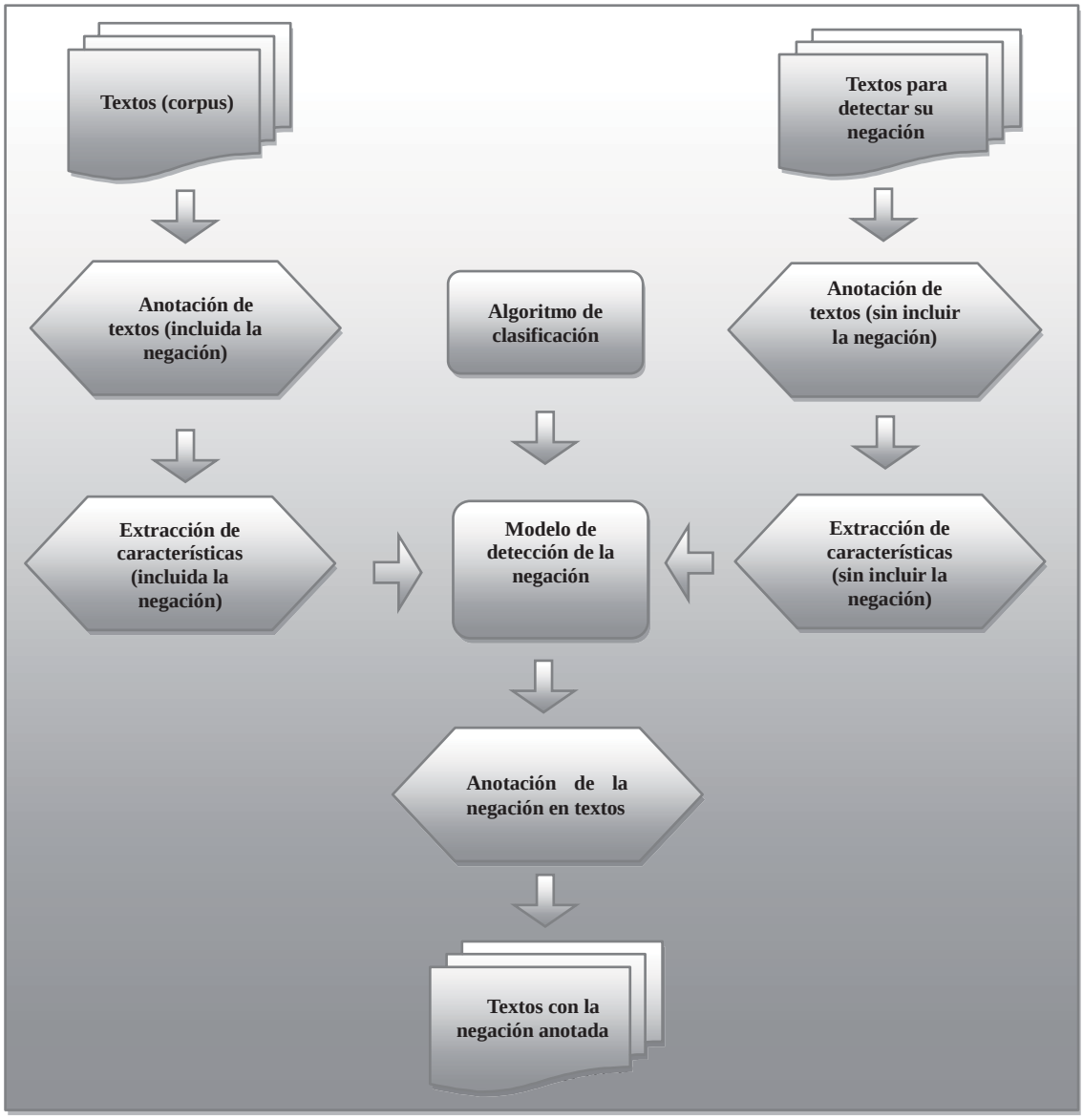

Figura 1

Los textos de entrada serán divididos en: i) el corpus, aquellos textos a los que se le hace una anotación manual de las negaciones, es decir, cada token se clasifica como negación o no negación; ii) textos para detectar su negación, son aquellos sobre los que nuestro sistema deberá anotar la negación con la mayor precisión posible.

Sobre los textos que representan el corpus se deben anotar sus características lingüísticas y la negación. Estas anotaciones, que se hacen sobre tokens o elementos del texto, se transforman en atributos de instancias, para que sean interpretables por el algoritmo. El modelo se crea a partir del algoritmo de clasificación y los atributos de las instancias obtenidas del corpus.

Por otro lado, sobre los textos para detectar su negación se anotan sus características lingüísticas. Las anotaciones sobre sus tokens se transforman también en atributos de instancias. Así, el modelo creado por el corpus puede clasificar estas instancias como negación o no negación. Para, finalmente, anotar sobre los textos la negación.

Terminada la clasificación, el sistema se puede validar o evaluar para medir su rendimiento. Este proceso se lleva a cabo viendo si cada token se ha clasificado correctamente o no.

4.2. **Anotación y extracción de características**

Se explica a continuación cómo se ha llevado a cabo el proceso de anotación de los textos con UIMA y JAVA y la extracción de las características con JAVA.

4.2.1. **Creación de anotaciones**

Se han realizado dos tipos de anotación distintas: los disparadores (Disparador) y las negaciones (NegationScope).

Una palabra en el texto será anotada como Disparador si está contenida en una lista o diccionario de disparadores. Los disparadores de negación son aquellas palabras en un texto que generan la posibilidad de que las palabras no lejanas a ella puedan estar negadas. Esta lista, inicialmente era la misma lista que se utilizó en una implementación para el algoritmo *ConText* (Zhang, 2014). La lista que se propone en este trabajo para la negación consta de 260 disparadores incluidas sus formas derivadas. Se consideran únicamente palabras, es decir, no formas compuestas. Algunas formas derivadas que se han incluido son el plural y las formas con distinto género si las tienen y para los verbos las formas del gerundio y participio. Algunas palabras de está lista son *no, sin, tampoco, ni, niega, ausente, falta, suspender…*

Las anotaciones de tipo NegationScope son las palabras en situación negada. Por tanto, son también las etiquetas que el algoritmo debe aprender a clasificar, por lo que la anotación se ha llevado a cabo de manera manual de acuerdo a lo establecido en el marco teórico del problema.

Crear una anotación en UIMA conlleva crear su tipo, su descriptor y su anotador. Para crear el tipo es necesario definir dos clases Java y un archivo XML. El descriptor es un archivo XML que tiene información de la anotación, por ejemplo parámetros. Para definir el anotador se debe crear una clase JAVA que contiene la lógica de la anotación.

Para realizar una anotación sobre un texto (SofA, subject of analysis) se debe llamar a la clase JAVA del anotador. Esta llamada hace que se recorra todo el SofA y se cree una instancia de la anotación si la lógica del descriptor lo establece. Una vez terminada la llamada se devuelve un archivo XMI que contiene el SofA y todas las instancias de las anotaciones enlazadas que se han hecho sobre él, así como los caracteres de inicio y fin de cada anotación y los parámetros, si los tiene.

En nuestro caso, para crear la anotación Disparador tuvimos que insertar un parámetro que es DisparadorName y la lista de disparadores en el archivo XML que define su clase. La lógica del anotador establece que si hay una palabra en el SofA que es igual a alguna palabra de la lista, esta se anote.

Para la creación del anotador de negación se ha utilizado la herramienta UIMA Annotation Editor que permite hacer anotaciones manualmente sobre el SofA, el texto clínico. Así, mientras se lee el texto se pueden etiquetar como NegationScope uno o varios tokens.

Además de la creación de estas dos anotaciones se han utilizado otras cuatro anotaciones realizadas en módulos previos. Creando un pipeline con las 6 anotaciones sobre un SofA que corresponde a frases de textos clínicos. A continuación detallamos las anotaciones y los resultados obtenidos.

- Sentence Annotation: establece toda una frase. Ha reconocido 465 frases e indica su carácter de inicio y fin.
- Token Annotation: es un tokenizador. Encontró 12946 tokens. Indica su carácter de inicio y fin. Permite obtener el Lemma del token (el conjunto de caracteres que conforman el token). Además, tiene como parámetro Token_Type que clasifica los tokens en:
  - WORD: si es una palabra. 11120 encontrados.
  - PUNCTUATION: para signos de puntuación. 1166 encontrados.
  - NUMBER: para números. 156 encontrados.
  - SPECIAL: para otros caracteres especiales. 4 encontrados.
- Part-of-Speech: determina el análisis morfológico de cada token. Incluye además la probabilidad de acierto de cada categoría para cada token, aunque no es utilizada en este trabajo. Hay 12946 anotaciones que se relacionan de las siguiente manera:
  - VERB: 706 verbos.
  - DET: 764 determinantes.
  - NOUN: 3965 sustantivos.
  - ADP: 2257 preposiciones.
  - ADJ: 2156 adjetivos.
  - “.”: 1656 signos de puntuación.

  - PRON: 337 pronombres.
  - NUM: 360 números.
  - ADV: 209 adverbios.
- Chunker: determina el tipo de sintagma al que pertenece el token. Se han encontrado 11263 sintagmas distintos, ya que un sintagma puede cubrir varios tokens pero todos los tokens pertenecen a un sintagma. Los sintagmas pueden ser:
  - PP: 2249 sintagmas preposicionales.
  - VP: 707 sintagmas verbales.
  - NP: 5102 sintagmas nominales.
  - O: 2531 otros (como conjunciones o símbolos de puntuación).
  - AP: 2153 sintagmas adjetivales.
  - RP: 204 sintagmas adverbiales.
- Disparador: determina si una palabra es disparador o no. Se han encontrado 233 disparadores en 158 frases, obviamente todos los tokens son de tipo WORD. De ellas, 81 son preposiciones, 67 adverbios, 41 conjunciones, 18 adjetivos, 14 sustantivos y 12 verbos.
- NegationScope: determina si una palabra está siendo afectada por la negación o no. Se han obtenido 207 ámbitos negados distintos, que comprenden 488 tokens y se encuentran en 123 frases distintas. De los 488 tokens, 482 corresponden a palabras y 6 corresponden a signos de puntuación. Respecto al PoS, se observa que 220 son sustantivos, 124 son adjetivos, 64 son preposiciones, 14 son determinantes, 6 son signos de puntuación, 5 son conjunciones, 2 son verbos y 1 es adverbio.

4.2.2. **Extracción de características**

Este módulo implementado con JAVA en Eclipse tiene una importancia fundamental, ya que extrae información del texto y la transforma en un formato de fácil análisis.

Las características o atributos que se podrán extraer están definidas por las anotaciones; de tal manera que cada token dispondrá de 7 atributos iniciales: la frase (Sentence_Annotation), el Lemma (Token_Annotation), el tipo de token (Token_Annotaion), el PoS (Part-of-Speech), el Chunker (Chunker), el disparador (Disparador), la negación (NegationScope).

Los valores de estos atributos están definidos por los parámetros contenidos en el tipo de anotación que los genera. A excepción del Lemma que toma el valor del String del token.

Se ha creado una estructura de datos basada en dos clases JAVA: i) Instancia, que contiene 6 atributos de tipo String, ii) ListaInstancias que contiene 8 atributos de tipo String, uno de tipo integer y otro de tipo ArrayList<Instancia>.

Otra de las opciones de UIMA es leer sus propios archivos XMI generados; de esta manera se pueden releer sus anotaciones. Esto se ha hecho con el XMI que contiene las anotaciones del texto y, para cada tipo anotación se ha implementado una iterador que recorre su lista enlazada de anotaciones, crea por cada frase una instancia ListaInstancias y por cada token una instancia Instancia. En cada ListaInstancia se guardan atributos de cada frase y la lista de tokens que están en la frase, mientras que cada instancia pertenece a una lista y tiene los atributos del token.

Finalmente, se ha implementado una clase que devuelve un archivo con la información de todos los tokens y todas las frases. Este archivo contiene 12946 líneas, una por cada token, y cada línea contiene 64 atributos. Estos atributos se dividen en:

- Atributos a nivel token: contienen valores únicos para cada token. Sin embargo, tienen información de los cuatro token anteriores, los cuatro tokens posteriores y el mismo token. Esta información es:
  - Lemma: es el token en sí.
  - PoS: toma los valores ADJ, ADP, NUM, VERB, ADV, NOUN, DET, NUM, CONJ, PRON, “.” y nullPoS.
  - Tipo de token: toma los valores WORD, PUNCTUATION, SPECIAL y nullType.
  - Chunker: AP, PP, VP, O, RP, NP y nullChunker.
  - Disparador: SI_DISP, NO_DISP y nullDisp.
  - Scope: SI_SCOPE, NO_SCOPE y nullScope.

Información de 9 tokens * 6 atributos de cada token dan 54 atributos a nivel token.

Los atributos “nullLemma”, “nullPoS”, “nullType”, “nullChunker”, “nullDisp” y “nullScope” solo se encuentran en la información relativa a tokens anteriores o posteriores, y se debe a que, si es la primera o última palabra de la frase la información del anterior o siguiente token no se encuentra.

- Atributos a nivel de frase: estos atributos contienen información de cada frase. Se encuentran por lo tanto los mismos valores en todos los tokens de una misma frase y son:
  - SentenceId: toma valores del 1 al 465. Es el identificador de la frase.
  - NumDisp: contiene valores de 0 a 4, indicando el número de disparadores que contiene la frase.

- StringDisp1: contiene el primer disparador de la frase. Puede además ser nullDisp.
- PoSDisp1: contiene el PoS del primer disparador de la frase. Puede además ser nullPoSDisp.
- StringDisp2: contiene el segundo disparador de la frase. Puede además ser nullDisp.
- PoSDisp2: contiene el PoS del segundo disparador de la frase. Puede además ser nullPoSDisp.
- StringDisp3: contiene el tercer disparador de la frase. Puede además ser nullDisp.
- PoSDisp3: contiene el PoS del tercer disparador de la frase. Puede además ser nullPoSDisp.
- StringDisp4: contiene el cuarto disparador de la frase. Puede además ser nullDisp.
- PoSDisp4: contiene el PoS del cuarto disparador de la frase. Puede además ser nullPoSDisp.

Esta estructura se hace pensando que añadir atributos del disparador a nivel frase puede beneficiar el aprendizaje. Y dado que no se han encontrado frases que contengan más de 4 disparadores no hacen falta más atributos del disparador.

Los nombres que se le han dado a los atributos son POS (PoS), CHUNKER (Chunker), Type (Token_Type), DISP (Disparador), SCOPE (NegationScope). Además, para establecer el atributo relativo al token anterior se ha incluido “PRE”, Por ejemplo, el atributo de PoS relativo al token anterior es PREPOS, si nos queremos referir al atributo del token anterior al anterior de Disparador diremos PREPREDISP. Análogamente, para atributos relativos a tokens posteriores se ha añadido POST, siendo el atributo de Chunker relativo al token posterior es POSTCHUNKER.

4.3. **Selección de atributos para el aprendizaje**

De cara a crear el modelo de clasificación se considera de gran importancia estudiar y analizar cuáles son los atributos que se deben incluir. Para ello, se establece un sistema de etapas. En cada etapa se aplican y comparan una serie de modelos. La figura siguiente describe el esquema que sigue cada etapa de selección de atributos:

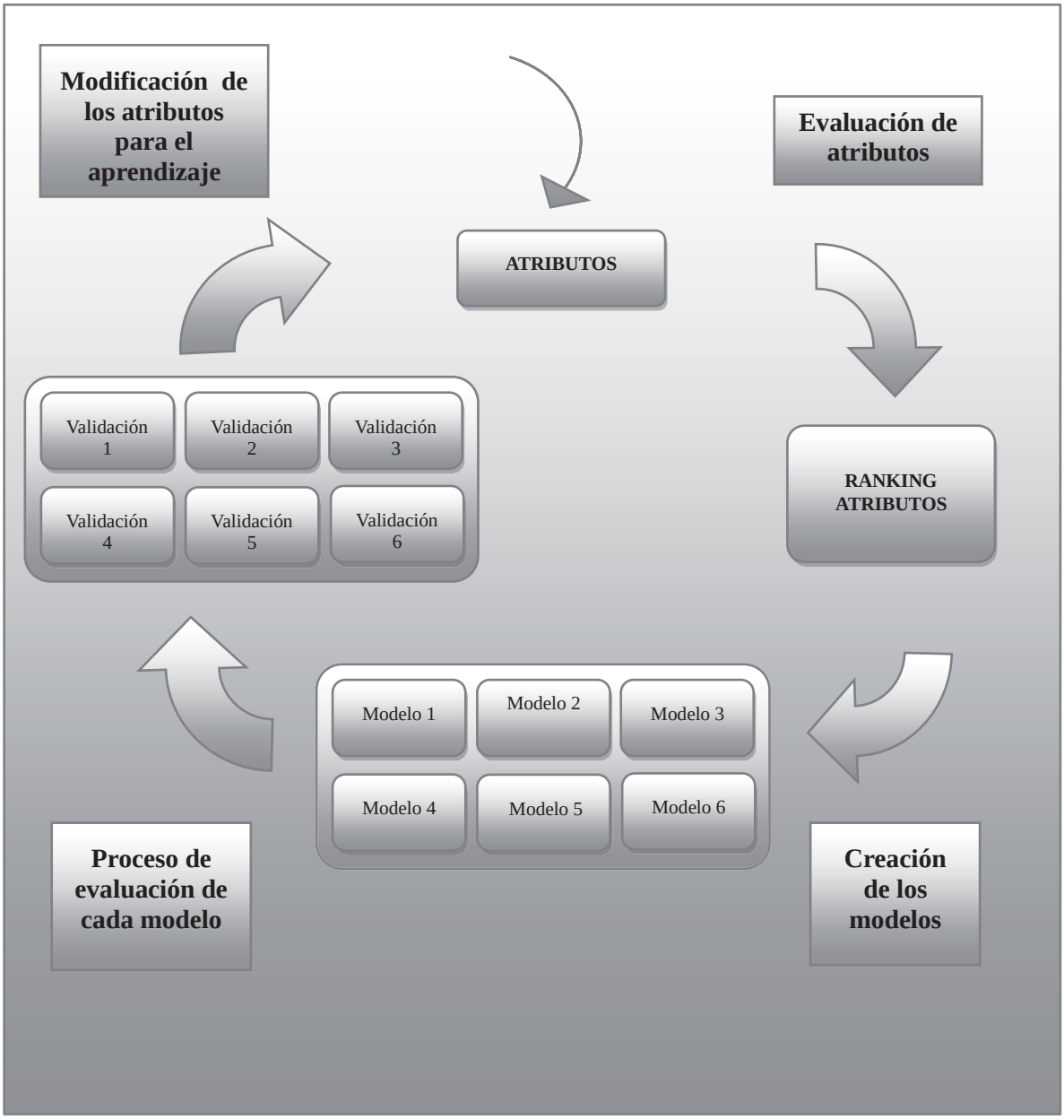

Figura 2

El sistema de selección de atributos funciona de la siguiente manera: primero se hace un ranking de atributos que más información aportan. Basándonos en el ranking se crean los modelos de clasificación y sus evaluaciones. Después se comparan los

resultados y se decide que atributos eliminar, añadir y dejar. Este proceso se puede repetir hasta que se considere que la precisión del modelo no es mejorable.

El ranking de atributos se hace con la herramienta selección de atributos de Weka. Esta ordena de mayor a menor los atributos evaluando cuales aportan mayor información al aprendizaje y nos sirve de ayuda para decidir que modelos crear o que atributos eliminar sin crear el modelo.

La creación de los modelos se ha hecho con el algoritmo J48, por su rapidez en la creación del modelo y la posibilidad que nos ofrece de ver el árbol de decisión. También se ha utilizado la implementación de Weka de un Support Vector Machine (SMO) ya que ofrecía buenos resultados, no es lento en la creación del modelo y fue utilizado en trabajos relacionados (Cruz, López, Vázquez, & Álvarez, 2013).

La evaluación de cada modelo se ha llevado a cabo con validación cruzada (cross-validation) con 10 iteraciones (10-fold). Esto quiere decir que se divide el conjunto de ejemplos en 10 conjuntos del mismo tamaño aleatoriamente. Y en cada iteración, se extrae uno de esos conjuntos, se aprende de los nueve restantes para crear el modelo y se evalúa con el conjunto apartado. La evaluación final es la media de las 10 evaluaciones obtenidas. Sigue el esquema siguiente para 4-fold cross-validation:

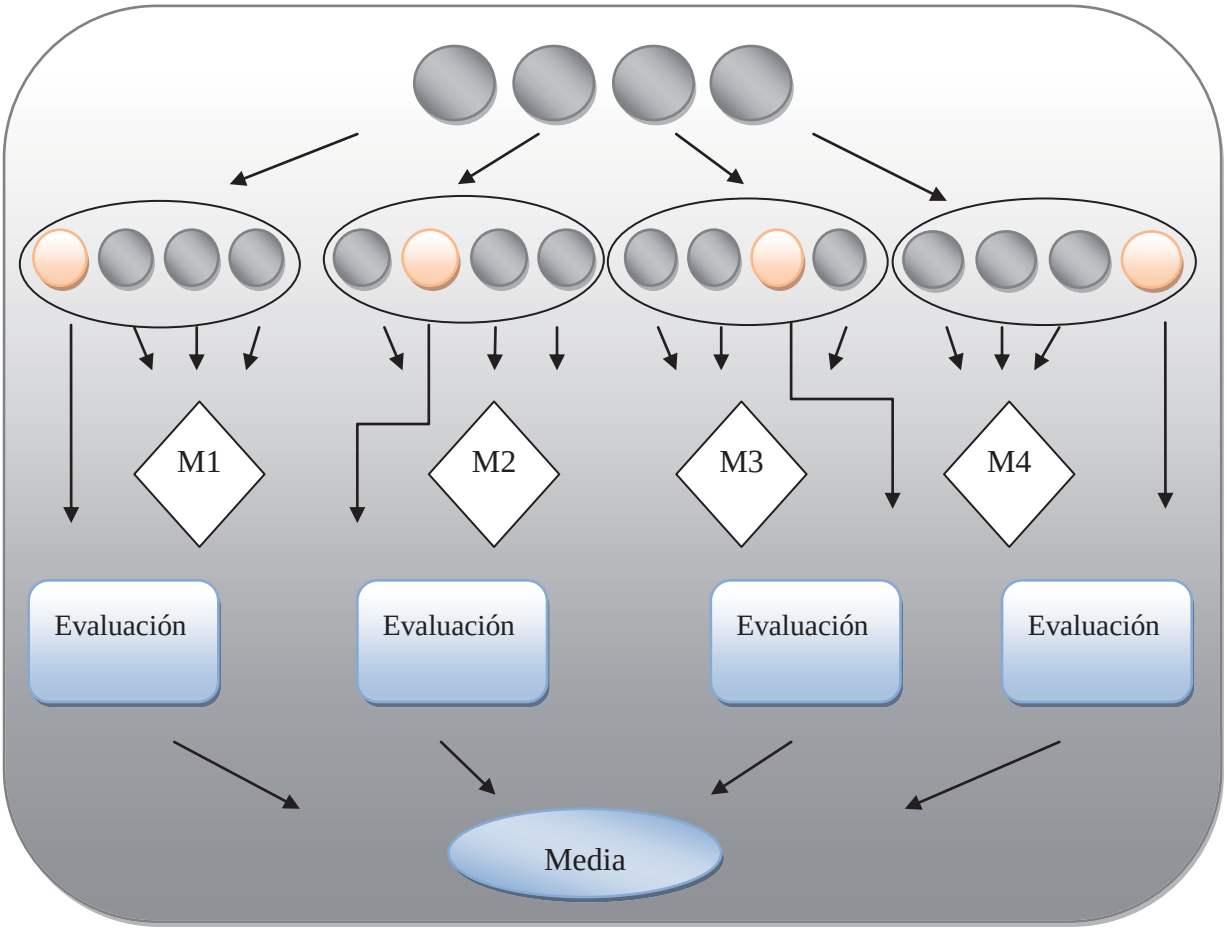

Figura 3

Las medidas que se han tenido en cuenta para la selección de los atributos tras la validación son Precision (la precisión), Recall (la exhaustividad) y el F1 (F-score). Recordamos sus fórmulas:

$$Precision = \frac{True\ positives}{True\ positives + False\ positives}$$

$$Recall = \frac{True\ positives}{True\ positives + False\ negatives}$$

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}$$

También recordamos que:

- True positive: es un acierto al clasificar con esa categoría.
- True negative: es un acierto al no clasificar con esa categoría.
- False positive: es un fallo al clasificar con esa categoría.
- False negative es un fallo al no clasificar con esa categoría.

Es importante tener en cuenta que para nuestra clasificación se hará sobre el Scope de la negación por cada token esto quiere decir que nuestro conjunto está muy desbalanceado, hay 12946 tokens y sólo 488 son SI_SCOPE. Por lo que cuando analicemos los resultados de la validación, los resultados para el NO_SCOPE serán casi siempre muy buenos, mientras que los del SI_SCOPE serán más relevantes para decidir.

Cada vez que se quiere añadir un atributo es necesario modificar el código de extracción de características, pero para eliminar un atributo no es necesario. Por lo tanto, el sistema de extracción de características se ha visto modificado a lo largo del proceso de selección de atributos.

Se explican a continuación cada etapa realizada para la selección de atributos:

4.3.1. **Primera etapa**

El planteamiento inicial era tener información del token de su anterior y su posterior está información era el Lemma, Chunker, PoS, Tipo de Token, Disparador. Teniendo 16 atributos.

Al pasarle el ranking de atributos observamos que los que más información aportan son los tokens relativos al Lemma. Dado que el Lemma puede tomar miles de valores es complicado creer que es posible el aprendizaje a partir del Lemma. Para comprobar nuestra intuición. Creamos dos modelos con J48, uno incluyendo los 16 atributos y otro eliminando los atributos relativos al Lemma. Observamos su matriz de confusión:

Clasifica como	**SI_SCOPE**	**SI_SCOPE**	**NO_SCOPE**	**NO_SCOPE**
Es	**ACIERTO**	**FALLO**	**ACIERTO**	**FALLO**
Los 16 atributos	0	488	12458	0
Los 13 atributos	106	382	12428	30

Tabla 1

Dado que si incluimos los atributos relativos al Lemma clasifica todos los tokens como NO_SCOPE decidimos a partir de este punto prescindir de todos los atributos relativos al Lemma.

4.3.2. **Segunda etapa**

Del árbol de decisión de la etapa anterior se observó que los atributos relativos a los disparadores apenas tenían relevancia en el aprendizaje ya que no se incluyen en el árbol. Para que puedan tener más relevancia se decide en este punto añadir la información que los tokens contienen de sus contiguos a tres anteriores y tres posteriores. Teniendo 28 atributos en total. Chunker, Type, PoS y Disparador por 7.

Comparamos las medidas de evaluación del modelo obtenido de la primera etapa y este con 28 características con el árbol de decisión J48:

J48	SI_SCOPE			NO_SCOPE		
Modelos	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
Los 15 atributos	0,340	0,217	0,779	0,984	0,998	0,970
Los 28 atributos	0,654	0,553	0,799	0,989	0,995	0,983

Tabla 2

Dado que el F-score del modelo con los 28 atributos es muy superior F-Score del modelo de la primera etapa existe una mejora al añadir más atributos de tokens contiguos. Además, en el ranking de atributos por ganancia de información se observó que en última posición se encontraba el atributo Disparador, que nos dice si el token es disparador o no. Pero atributos relativos al Disparador de tokens contiguos se encontraban en posiciones superiores en el ranking.

4.3.3. **Tercera etapa**

En este punto se rehízo el código para la extracción de características de tal manera que se pudiesen establecer atributos a nivel frase. Añadiendo 9 atributos: 8 atributos para el String del disparador y el PoS del disparador ya que puede haber hasta 4 disparadores en una misma frase y un atributo con el número de disparadores en la frase. Se tienen ahora 37 atributos. Para evaluar esta modificación se compararon 7 modelos:

J48	SI_SCOPE			NO_SCOPE		
Modelos	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
Los 28 atributos de la iteración 2	0,654	0,553	0,799	0,989	0,995	0,983
Los 28 atributos + (StringDisp*2)	0,646	0,557	0,768	0,988	0,993	0,983
Los 28 atributos + (StringDisp*4)	0,647	0,559	0,767	0,988	0,993	0,983
Los 37 atributos	0,635	0,543	0,764	0,988	0,993	0,982
Los 37 atributos menos el númeroDisp	0,635	0,541	0,767	0,988	0,994	0,982

Tabla 3

Estos resultados no muestran mejora por lo que se descarta la idea de incluir estos atributos a nivel de frase. A pesar de esto, el código se deja con esta estructura ya que existe la posibilidad de crear otro tipo de atributos como se explicará en líneas futuras.

4.3.4. **Cuarta etapa**

Se repite una iteración como la segunda donde se añaden atributos de tokens contiguos, esta vez hasta crear una ventana de 11 tokens en total, 5 anteriores y 5 posteriores, obteniendo 44 atributos. Se compara también con un modelo en el que cada token incluya información de 4 anteriores y 4 posteriores (36 atributos):

J48	SI_SCOPE			NO_SCOPE		
Modelos	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
Los 28 atributos de la iteración 2	0,654	0,553	0,799	0,989	0,995	0,983
Los 36 atributos	0,716	0,650	0,798	0,990	0,994	0,986
Los 44 atributos	0,715	0,652	0,791	0,990	0,993	0,986

Tabla 4

Estos resultados muestran que no existe una mejora en aumentar infinitamente la información de datos contiguos y que 4 tokens anteriores y 4 posteriores es suficiente. Se consideran a partir de ahora estos 36 atributos.

4.3.5. **Quinta etapa**

Una vez establecido que una ventana de 9 tokens nos dan unos buenos resultados podemos explorar si los tokens anteriores dan más información que los tokens posteriores o al contrario, en definitiva, si una solución donde los tokens contengan información de manera asimétrica es beneficiosa. Para llevar a cabo este estudio se ha tenido que modificar la estructura insertando en el archivo CSV entre frase y frase 8 tokens nulos por lo que el número de instancias ha cambiado. Además, para evaluar los resultados aprendemos sobre el atributo PREPRE...SCOPE o POSTPOST…SCOPE, por lo que puede tomar también el valor nullScope, pero aprende muy bien a reconocerlo. Evaluamos los siguientes modelos:

J48	SI_SCOPE			NO_SCOPE			NULL_SCOPE		
Cada token tiene atributos de:	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
8 tokens anteriores y el mismo	0,705	0,648	0,774	0,989	0,992	0,985	1	1	1
7 tokens anteriores 1 posterior y el mismo	0,711	0,660	0,770	0,989	0,992	0,986	1	1	1
6 tokens anteriores, 2 posteriores y el mismo	0,714	0,670	0,764	0,989	0,991	0,986	1	1	1
5 tokens anteriores, 3 posteriores y el mismo	0,711	0,652	0,781	0,989	0,993	0,986	1	1	1
4 tokens anteriores, 4 posteriores y el mismo	0,713	0,639	0,806	0,990	0,994	0,986	1	1	1
3 tokens anteriores, 5 posteriores y el mismo	0,660	0,564	0,797	0,989	0,994	0,983	1	1	1
8 tokens posteriores y el mismo	0,119	0,064	0,912	0,982	1	0,965	1	1	1

Tabla 5

A pesar de que el F-Score para el SI_SCOPE es mayor en el tercer modelo de la tabla comparado con el quinto, que es el que teníamos de etapas anteriores, pasaremos a la siguiente etapa con el quinto modelo, debido a que la diferencia es muy poca y trabajar con el token central nos facilitará la manipulación de los datos. Además, al haber añadido 8 tokens nulos entre frase y frase el árbol que se genera en la etapa cinco es distinto al de la etapa 6, en consecuencia las medidas de la evaluación son distintas.

4.3.6. **Sexta etapa**

Tras la quinta etapa se consideran suficientes el número de atributos y no se añadirán más. Se crean ahora distintos modelos únicamente eliminando atributos. Para ello nos basamos en los 3 últimos atributos del ranking de ganancia de información:

POSICIÓN	ATRIBUTO	GAIN RATIO
34	PREPRECHUNKER	0.00205
35	PREPREPRETYPE	0.00189
36	PREPREPREPRETYPE	0.00143

Tabla 6

Crearemos 3 modelos en los que quitaremos uno de los atributos y los comparamos con el modelo obtenido de la etapa cuatro y un modelo eliminando el atributo Disparador relativo al token en sí, ya que desde un punto de vista teórico se puede considerar irrelevante la relación entre que un token sea disparador y forme parte de una negación:

J48	SI_SCOPE			NO_SCOPE		
Cada token tiene atributos de:	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
36 atributos	0,716	0,650	0,798	0,990	0,994	0,986
36 atributos menos PREPREPREPRETYPE	0,713	0,648	0,792	0,990	0,993	0,986
36 atributos menos PREPREPRETYPE	0,713	0,650	0,791	0,990	0,993	0,986
36 atributos menos PREPRECHUNKER	0,716	0,650	0,798	0,990	0,994	0,986
36 atributos menos DISPARADOR	0,730	0,664	0,810	0,990	0,994	0,987

Tabla 7

El F-score mayor se obtiene al prescindir del atributo Disparador, por lo que a partir este punto el mejor modelo consta de los 36 atributos menos el relativo al Disparador del token.

4.3.7. **Séptima etapa**

Realizamos el ranking de ganancia de información con los 35 atributos y observamos que los últimos 4 clasificados son:

POSICIÓN	ATRIBUTO	GAIN RATIO
32	PRETYPE	0,00288
33	PREPRETYPE	0,00237
34	PREPREPRETYPE	0,00189
35	PREPREPREPRETYPE	0,00155

Tabla 8

Comparamos los 4 modelos al quitar estos atributos y el modelo de la etapa anterior:

J48	SI_SCOPE			NO_SCOPE		
Cada token tiene atributos de:	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
35 atributos	0,730	0,664	0,810	0,990	0,994	0,987
35 atributos menos PRETYPE	0,728	0,660	0,811	0,990	0,994	0,987
35 atributos menos PREPRETYPE	0,729	0,664	0,808	0,990	0,994	0,987
35 atributos menos PREPREPRETYPE	0,727	0,664	0,804	0,990	0,994	0,987
35 atributos menos PREPREPREPRETYPE	0,724	0,656	0,808	0,990	0,994	0,987

Tabla 9

Ninguno de los modelos nuevos tiene mayor F-score que el modelo anterior por lo que seguimos con los 35 atributos. Se consideran entonces los 35 atributos como la mejor solución.

4.3.8. **Octava etapa**

Una vez seleccionados los atributos compararemos los resultados que nos proporcionarían los tres algoritmos de clasificación distintos: J48, SMO (SVM), clasificador por regresión logística. Además, valoraremos las evaluaciones finales de la clasificación.

	SI_SCOPE			NO_SCOPE		
Algoritmos	**F1**	**Recall**	**Precision**	**F1**	**Recall**	**Precision**
J48	0,730	0,664	0,810	0,990	0,994	0,987
SMO	0,690	0,713	0,669	0,987	0,986	0,989
Regresión logística	0,654	0,594	0,727	0,988	0,991	0,984

Tabla 10

El algoritmo J48 es el que mejor clasifica cada token como parte de una negación.

A continuación se describen brevemente los algoritmos utilizados:

J48

Es la implementación en Weka del algoritmo C4.5. Para clasificar, este algoritmo genera un árbol de decisión. El nodo inicial del árbol corresponde al mejor atributo, el de mayor entropía (ganancia de información) de todos los ejemplos. Se divide el nodo en tantas ramas como valores tenga el atributo y para cada rama se vuelve a calcular el atributo de mayor entropía, pero únicamente entre los ejemplos que tienen ese valor. Este proceso se repite hasta que no queden atributos o todos los ejemplos se clasifiquen igual. Las hojas del árbol serán las que clasifiquen como SI_SCOPE o NO_SCOPE. Una vez se ha creado el árbol se pueden clasificar nuevos ejemplos.

Sequential Minimal Optimization (SMO)

Es la implementación de SVM que utiliza Weka. Los métodos SVM para la clasificación consideran cada atributo como una variable o dimensión, así tenemos tantas dimensiones como atributos. Su objetivo es entonces encontrar el hiperplano, es decir el vector que tiene una dimensión menos, que divide a los ejemplos en dos conjuntos y maximiza la distancia entre los puntos más cercanos entre los dos conjuntos. Además, utiliza funciones Kernel para transformar el hiperplano en formas más complejas.

Clasificador por regresión logística

La clasificación mediante regresión logística se basa en la utilización de la función logística. Primero, se han de transformar los valores nominales de los valores en valores numéricos. Después, la función logística transforma los datos de cada ejemplo multiplicados por unos parámetros a un número entre 0 y 1. Después actualiza los parámetros de tal manera que minimicen el error, que es la distancia del resultado anterior y 0 o 1. Una vez actualizados los parámetros vuelve a computar la función logística. Repite este proceso hasta que la actualización de los parámetros no haga decrecer el error.

5. CONCLUSIONES

5.1. Conclusiones

En este TFG se ha presentado un modelo para la clasificación que, mediante el algoritmo J48 y 35 atributos obtenidos de características lingüísticas (morfológicas y sintácticas) y disparadores de negación, detecta si un token está negado en 465 frases provenientes de textos clínicos con un F-Score del 73%, una exhaustividad del 66% y una precisión del 81% con una validación cruzada de 10 iteraciones.

Para su consecución se ha realizado un análisis de características lingüísticas y se ha presentado un método para la selección de los atributos más discriminantes para la detección de la negación.

Los atributos resultantes para cada token (elemento clasificado) son: su morfología (PoS), su sintagma (Chunker) y su tipo de token (Type_token); para los cuatro tokens anteriores, los cuatro tokens posteriores y el token en cuestión, además incluye atributos del disparador de negación (Disparador) para los cuatro tokens anteriores y cuatro posteriores, resultando el siguiente ranking de atributos en función de la ganancia de información:

POSICIÓN	ATRIBUTO	GAIN RATIO
1	PREPREDISP	0,02607
2	PREDISP	0,02427
3	PREPREPREDISP	0,0227
4	POS	0,01823
5	POSTPOS	0,01644
6	PREPREPREPREDISP	0,01548
7	CHUNKER	0,01343
8	POSTCHUNKER	0,01248
9	PREPOS	0,01098
10	POSTPOSTPOSTPOSTPOS	0,00908
11	POSTPOSTPOSTPOS	0,00845
12	POSTPOSTPOSTPOSTCHUNKER	0,00774
13	POSTPOSTPOS	0,00772
14	POSTPOSTPOSTPOSTDISP	0,00759
15	POSTPOSTPOSTPOSTDISP	0,00735
16	PREPREPOS	0,0073
17	POSTPOSTPOSTCHUNKER	0,0709
18	PREPREPREPOS	0,0069
19	PREPREPREPREPOS	0,0067
20	POSTTYPE	0,0067
21	POSTPOSTPOSTDISP	0,00621
22	POSTPOSTPOSTTYPE	0,00618
23	TYPE	0,00608
24	PREPREPRECHUNKER	0,0057
25	POSTPOSTCHUNKER	0,00542

26	PREPREPREPRECHUNKER	0,00534
27	PRECHUNKER	0,00513
28	PREPRECHUNKER	0,0049
29	POSTDISP	0,0048
30	POSTPOSTTYPE	0,00417
31	POSTPOSTDISP	0,00381
32	PRETYPE	0,00288
33	PREPRETYPE	0,00237
34	PREPREPRETYPE	0,00189
35	PREPREPREPRETYPE	0,00155

Nota: PREDISP es el atributo de disparador relativo al token anterior. POSTDISP es el atributo relativo de disparador relativo al token posterior. PREPREDISP es el atributo de disparador relativo al token anterior al anterior. Y así sucesivamente…

Tabla 11

Se considera este ranking coherente con el planteamiento inicial de la solución del problema ya que las primeras posiciones son ocupadas por los atributos relativos al disparador.

Se considera que el margen de mejora del modelo está limitado por: i) el número de frases utilizadas como corpus, ii) las características lingüísticas tratadas, iii) el número de algoritmos de machine learning probados y iv) el tiempo de realización del trabajo.

Por otro lado, el desarrollo del código para la anotación y extracción de características, junto con el estudio sobre la selección de atributos, hacen que la integración de este trabajo en un proyecto de estructuración de información se valore como satisfactoria,

Consiguientemente los objetivos planteados al comienzo del trabajo han sido cumplidos.

5.2. **Líneas futuras**

A pesar de haber conseguido el objetivo surgen nuevas líneas de investigación, sobretodo, orientadas a mejorar los resultados en términos de rendimiento y eficiencia del modelo.

5.2.1. **Selección de nuevos atributos**

Se considera que retomar la idea planteada en la quinta etapa sobre añadir más atributos de tokens anteriores que de tokens posteriores, puede generar buenos resultados. Esta solución debe ser valorada en aquellos sistemas donde fuese muy costoso extraer información de 4 tokens anteriores y 4 posteriores, ya que una solución asimétrica para tokens con información de 3 tokens anteriores y 1 token posterior podría aprender mejor que con uno de 2 tokens anteriores y 2 posteriores.

Otra idea a valorar es crear nuevos atributos a nivel de frase. Aunque esta idea de la manera en que se ha planteado en la tercera etapa no mejora los resultados, se podrían crear atributos que clasifiquen los disparadores en distintas categorías: una categoría para aquellos que se suelen encontrar delante la negación y otra para los que suelen

estar detrás. Esto conllevaría un estudio previo sobre qué disparadores se colocan delante y cuáles detrás.

Por último se podrían insertar atributos a nivel de frase que nos dijeran si un token se encuentra entre dos disparadores.

5.2.2. **Aprendizaje para la detección de los disparadores**

En el apartado "2.1.4 Detección de la negación mediante técnicas de Machine Learning" se comentaba el trabajo de Cruz, López, Vázquez, & Álvarez (2013) para la detección de la negación basado en dos clasificadores en serie. El primero, para determinar si un token es disparador o no lo es, y un segundo clasificador automático para detectar el Scope (ámbito) del disparador. Por tanto, es interesante para nosotros estudiar el aprendizaje que tendría el primer clasificador.

Para ello, se podría estudiar el aprendizaje de la detección de los disparadores con Weka. No sería necesario modificar el código de extracción de características. Bastaría simplemente con quitar los atributos relativos al NegationScope y relativos al disparador de tokens contiguos, para finalmente clasificar cada token como Disparador o no Disparador.

Si con nuestros atributos aprendemos a clasificar como Disparador o no Disparador, con el algoritmo J48 obtenemos un modelo que validado con 10-fold tiene los siguientes resultados: F-Score del 0,224%, Precision del 0,137% y Recall del 0,604%. Estos resultados serían solo una primera aproximación, ya que posteriormente habría que realizar una selección de atributos para mejorar el modelo.

5.2.3. **Creación de dos clasificadores**

En este apartado se plantea una solución alternativa para el sistema detección de la negación. El sistema constaría de dos clasificadores en serie: un clasificador de la negación exactamente igual al planteado en este trabajo y otro clasificador de la negación que incluyese los atributos de NegationScope obtenidos por el primer clasificador.

Creando un modelo en el que se incluyen los atributos de NegationScope de tokens contiguos se puede realizar una aproximación de los resultados que se obtendrían. Estos, a falta de una correcta selección de atributos para el segundo clasificador serían los mejores resultados obtenibles, ya que el modelo creado supone que el primer clasificador tiene una precisión del 100%. Este modelo de aproximación tiene un F-Score del 92%, un Recall del 91% y Precision del 92%.

5.2.4. **Detección de la afirmación, la hipótesis y el histórico**

Por último, se deben abrir nuevas líneas de investigación sobre la detección de la afirmación, la hipótesis y el histórico. Es importante señalar las cuestiones en las que el problema tendría diferencias notables con relación a la detección de la negación abordada en este trabajo.

Primero, en el proceso de anotación se deben definir perfectamente los casos de afirmación, hipótesis e histórico, de la misma manera que se ha hecho con los casos de

negación en el marco teórico de este trabajo. Segundo, si el problema se traslada, es necesario modificar la lista de disparadores. Por último, se precisaría realizar una nueva selección de atributos y del algoritmo que optimizara la detección de la afirmación, hipótesis, histórico, es decir, se debe crear un modelo para cada caso.

Bibliografía

Cruz, N. P. (2014). *Negation and Speculation Detection in Medical and Review Texts.* Colección de Monografías de la Sociedad Española.

Cruz, N. P., López, M. J., Vázquez, J. M., & Álvarez, V. P. (2013). *A Machine-Learning Approach to Negation and Speculation Detection in Clinical Texts.* Huelva: Dpto. de Tecnologías de la Información, Universidad de Huelva.

Dhar, V. (December de 2013). *Communications of the ACM*. Obtenido de http://cacm.acm.org/magazines/2013/12/169933-data-science-and-prediction/abstract

Harkema, H., Dowling, J. N., Thornblade, T., & Chapman, W. W. (2009). *ConText: An Algorithm for determining negation, experiencer and temporal status from clinical reports.* Pittsburgh, PA, 15260, USA: department of Biomedical Informatics, University of Pittsburgh.

Soher, R., Perelygin, A., Wu, J. Y., Chuang, J., Manning, C. D., Ng, A. Y., y otros. (2013). *Recursive Deep Models for Semantic Compositionalty Over a Sentiment Treebank.* Stanford, CA, 94305, USA: Stanford University.

The economist. (2010). *www.economist.com.* Obtenido de http://www.economist.com/node/15557443

Zhang, J. C. (2014). *Diseño e implementación de un algoritmo para la detección de la negación de textos clínicos en español.* Madrid: Trabajo de fin de grado, directora: Ernestinas Menasalvas, Facultad de Informática, UPM.

Y. Kim. (2014). Convolutional Neural Networks for Sentence Classification. New York University. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), Association for Computational Linguistics*

Salud M. Jiménez Zafra, Eugenio Martínez Cámara, M. Teresa Martín Valdivia, M. Dolores Molina González. (2015). Tratamiento de la Negación en el Análisis de Opiniones en Español. *Departamento de Informática, Escuela Politécnica Superior de Jaén Universidad de Jaén, E-23071 – Jaén.*

Ozan Irsoy. Deep Recursive Neural Networks for Compositionality in Language. (n.d.) *Department of Computer Science Cornell University Ithaca, NY 14853,Claire Cardie Department of Computer Science Cornell University Ithaca, NY 14853.*

Pierre Zweigenbaum, Dina Demner-Fushman, Hong Yu and Kevin B. Cohen. (2007). Frontiers of biomedical text mining: current progress. *Published by Oxford University Press.*

Páginas, Cursos y tutoriales

“Machine Learning Online Course” by Stanford University, Andrew Ng: https://www.coursera.org/learn/machine-learning/

"Deep Learning tutorial":
http://deeplearning.net/tutorial/
"Data mining Course":
http://www.kdnuggets.com/data_mining_course/index.html
"UIMA Overview & SDK Setup":
https://uima.apache.org/d/uimaj-2.4.0/overview_and_setup.pdf
"Logistic regression":
https://en.wikipedia.org/wiki/Logistic_regression
"Multilayer perceptron":
https://en.wikipedia.org/wiki/Multilayer_perceptron
"Convolutional neural network":
https://en.wikipedia.org/wiki/Convolutional_neural_network
"NLP":
https://en.wikipedia.org/wiki/Natural_language_processing
"Supervised learning":
https://en.wikipedia.org/wiki/Supervised_learning
"Unsupervised learning":
https://en.wikipedia.org/wiki/Unsupervised_learning
"Sentiment analysis":
https://en.wikipedia.org/wiki/Sentiment_analysis
"J48":
http://weka.sourceforge.net/doc.dev/weka/classifiers/trees/J48.html
"UIMA":
https://uima.apache.org/

Este documento esta firmado por

Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=Facultad de Informatica - UPM, C=ES
Fecha/Hora	Mon Jan 18 17:51:17 CET 2016
Emisor del Certificado	EMAILADDRESS=camanager@fi.upm.es, CN=CA Facultad de Informatica, O=Facultad de Informatica - UPM, C=ES
Numero de Serie	630
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)