PROYECTO FIN DE CARRERA

Estudio de la explotación de información sintáctica para la extracción de interacciones farmacológicas en textos biomédicos

Autor: Mª del Mar Castro Peláez

Tutor: Isabel Segura Bedmar

Titulación: I.T.T. especialidad Telemática

Leganés, Octubre de 2011

AGRADECIMIENTOS

A Isabel, por ofrecerme la oportunidad de realizar este proyecto, por su ayuda siempre que la he necesitado, por ser una guía en un momento tan importante y por confiar en mi trabajo.

A mis hermanos, por sus consejos aunque a veces no los haya seguido, por demostrarme que siempre podre contar con ellos y por apoyarme en todo momento.

A mis padres, por quererme, por creer en mí, por mostrarme el camino y porque me siento feliz de ser vuestra hija.

A Jorge, por estar siempre a mi lado y devolverme la ilusión y la alegría de vivir.

Gracias a todos vosotros por ver en mí a esa ingeniera en la que pensaba que nunca me convertiría.

RESUMEN

Para la correcta administración de fármacos es necesario saber de antemano si los fármacos interaccionan entre sí, ya que las consecuencias pueden ser perjudiciales si la interacción causa un aumento de la toxicidad del fármaco o la disminución de su efecto, pudiendo provocar incluso la muerte del paciente en los peores casos.

Actualmente, el personal sanitario tiene a su disposición varias bases de datos sobre interacciones que permiten evitar posibles interacciones a la hora de administrar tratamientos, pero estas bases de datos no están completas. Por este motivo se ven obligados a revisar una gran cantidad de artículos científicos e informes para estar al día pero el gran volumen de información al respecto hace que estén desbordados ante tal avalancha; Todo esto hace necesario un método automático de extracción de la información de estas fuentes de datos para la detección de interacciones entre fármacos.

Motivados por estos problemas de gestión de la información, en este proyecto se desarrolla un sistema para la extracción de interacciones farmacológicas sobre textos biomédicos planteando una alternativa al sistema desarrollado por Isabel Segura en (1), en el que se plantean dos aproximaciones (una basada en patrones y uso de información sintáctica superficial, y otro basado de aprendizaje automático, en particular en métodos kernels basados en el uso de información sintáctica superficial).

El objetivo general de este proyecto es el estudio de la aportación de información sintáctica completa (obtenida con el analizador Stanford (2)) en un sistema basado en clasificadores clásicos (NaiveBayes, HypePipes, JRip, RandomForest, etc.) para resolver el problemas de extracción de interacciones. Como objetivo final, por tanto, compararemos el sistema basado en kernels propuesto en (1) con nuestro sistema, además de comparar la aportación de la información sintáctica completa (árboles sintácticos) frente a la superficial (solo sintagmas) usada en los kernels.

El resultado de esta combinación de información será analizado con distintos algoritmos de aprendizaje automático de WEKA (Waikato Environment for Knowledge Análisis) para su posterior comparación.

INDICE

INDICE DE TABLAS

INDICE DE ILUSTRACIONES

INDICE DE ECUACIONES

1 INTRODUCCIÓN

1.1 MOTIVACIÓN

Hablamos de interacciones farmacológicas (IF) cuando un fármaco modifica los efectos de otro al administrase conjuntamente pudiendo aumentar la toxicidad, disminuir su eficiencia e incluso inhibir un fármaco al otro. Estas alteraciones además de ser peligrosas para la salud del paciente también ocasionan un importante incremento del gasto médico (3) (4) (5).

Aunque actualmente hay diversas bases de datos sobre interacciones entre medicamentos (entre ellas Bot-plus (6), Stockley (7) y Drug Interactions Facts (8)), éstas no están todo lo completas y actualizadas que deberían. Esta situación provoca que los profesionales sanitarios deban consultar infinidad de documentación sobre IFs para obtener información completa y actual. Además, la gran cantidad de información sobre fármacos (ya sean en bases de datos o en textos no estructurados) hace prácticamente imposible que un profesional pueda buscar la información sobre interacciones que necesita en un tiempo razonable, provocando que dichos profesionales se vean desbordados y sin conocimiento sobre ciertas interacciones.

Debido a ello, sería de gran utilidad un sistema de extracción de información que pueda procesar dichos documentos y extraer la información necesaria para identificar una interacción entre dos fármacos. Diversos trabajos basados en el uso de algoritmos de aprendizaje automático y en el uso de características (*features-based methods*) han demostrado obtener buenos resultados en la extracción de relaciones, como los expuestos en (9), (10), (11) y (12). En estos sistemas, los ejemplos o instancias de pares de entidades que mantienen o no una relación son representados por medio de un conjunto de características útiles para entrenar los algoritmos. En nuestro problema, las características podrían ser los nombres de los fármacos, el verbo entre ambos, las palabras entre ambos, sus etiquetas morfosintácticas, el camino sintáctico entre los fármacos, etc. El hecho de contar con un sistema con las funcionalidades anteriores mejoraría de forma considerable el tiempo utilizado en la detección de IFs ayudando así a disminuir los inconvenientes que provocan y el coste asociado a la atención de los pacientes que sufren una IF.

Continuando con los estudios desarrollados por Isabel Segura en (1), por Víctor Manuel Méndez González en "*Un sistema para la extracción*

de interacciones farmacológicas basados en Support Vector Machines (SVM)" (13) y el realizado en *"Aplicación de técnicas de aprendizaje automático para la extracción de información en textos farmacológicos"* (14) por Beatriz Nombela Escobar, estudiaremos los efectos que producen en los resultados la aportación de información sintáctica utilizando algoritmos de aprendizaje automático. Estos trabajos anteriores al presente proyecto no hacen uso de información sintáctica completa, utilizando solamente el uso de información sintáctica superficial.

1.2 OBJETIVOS

Continuando los trabajos realizados (13) y (14), el objetivo de este proyecto es introducir información sintáctica obtenida gracias al uso del analizador Stanford sobre el corpus DrugDDI desarrollado en (1), en formato texto, con la intención de mejorar los resultados obtenidos anteriormente. Además, también compararemos los resultados con los obtenidos en el trabajo (1), así como con los resultados obtenidos de los distintos sistemas presentados en el workshop DDIExtraction 2011 (http://labda.inf.uc3m.es/DDIExtraction2011/) que se explicarán en el apartado 2.7.

1.2.1 OBJETIVOS ESPECÍFICOS

Los objetivos específicos de este proyecto son los siguientes:

- Estudio de trabajos anteriores: realizar un amplio estudio sobre el trabajo realizado por Isabel Segura en (1) para conocer el problema en cuestión y los enfoques propuestos posteriormente en (13) por Víctor Méndez y en (14) por Beatriz Nombela.
- Uso del analizador Stanford (2): estudiar el analizador Stanford y los principales formatos de salida que devuelve para obtener información sintáctica.
- Estudio de los formatos CoNLL (http://ifarm.nl/signll/conll/): conocer los distintos formatos y elegir de entre ellos el que mejor se adapte a nuestra tarea de extracción de relaciones. Creación de nuevo formato para el corpus DrugDDI. Este nuevo formato debe integrar la información sintáctica obtenida con el analizador Stanford.
- Estudiar la tecnología Castor (15) y XSD (16): conocer Castor y XSD (XML Schema), lo cual nos permitirá extraer las características a utilizar por los algoritmos del corpus DrugDDI de una manera más sencilla.

- Extracción de características: extraer las características que puedan enriquecer el conjunto de datos a partir del nuevo formato. Entre ellas se encontrarán la representación del contexto de cada fármaco mediante N-gramas, ventanas de diferentes tamaños en torno a los fármacos o el camino de los tipos de dependencias sintácticas entre palabras desde una palabra hasta la raíz de la oración. Estas características se describen con más detalle en el aparatado 3.4. Uso de ficheros "*CSV*": las instancias o ejemplos de pares de fármacos representados por medio de un conjunto de características son representadas en ficheros CSV.
- Composición de ficheros "*arff*": transformación de ficheros *CSV* en ficheros *arff* para poder utilizar posteriormente la herramienta Weka.
- Experimentación con Weka; aprender a utilizar la herramienta Weka y aplicar distintos algoritmos (NaiveBayes, VFI, IBk, ClassificationViaClustering, HyperPipes, RandomeTree, JRip, RandomForest, MultiBoostAB, etc.) de aprendizaje para la comparación de resultados.
- Comparación de resultados: comparar los resultados obtenidos con los distintos algoritmos utilizados en Weka para determinar qué algoritmo obtienen mejores resultados.
- Comparar los resultados con los resultados obtenidos en los trabajos realizados (1) , (13) y (14), como con los sistemas presentados en la tarea DDIExtraction2011.

1.3 ESTRUCTURA DE LA MEMORIA

Este apartado pretende ofrecer una breve visión global de los contenidos expuestos en este documento. Se explicará de forma resumida cada uno de los capítulos que conforman el escrito:

- Capítulo 1 'Introducción': se trata de una introducción general del proyecto, donde en primer lugar se indican las motivaciones que han llevado al desarrollo del proyecto, seguidamente se fijan los objetivos marcados y por último se da una visión a grandes rasgos de los temas que se van a tratar en cada capítulo.
- Capítulo 2 'Estado del Arte': en este capítulo se expone el estado de la cuestión, es decir los trabajos relativos y fundamentos teóricos que han dado lugar al desarrollo del proyecto.
- Capítulo 3 'Descripción del sistema': en primer lugar se va a describir la arquitectura del sistema y el corpus a partir del cual se va a extraer la información de los fármacos y sus interacciones. A continuación, describiremos el conjunto de

características que se utilizarán para entrenar los clasificadores. Por último se presentan los diferentes experimentos realizados y los resultados obtenidos.

- Capítulo 4 'Discusión de los resultados': en este capítulo se procederá a la comparación de los resultados obtenidos con los de trabajos anteriores.
- Capítulo 5 'Conclusiones y líneas de trabajo futuro': se realizará un breve resumen del proyecto y describiendo las razones por las cuales los resultados de este proyecto mejoran o no los resultados de los trabajos anteriores. También se detallará la línea futura del proyecto y posibles mejoras y/o evoluciones.
- Capítulo 6 'Anexo': contiene la planificación del proyecto y el presupuesto del desarrollo, incluyendo todos los trabajos realizados relativos a este, así como todos los gastos directos o indirectos que ha ocasionado.

2 ESTADO DEL ARTE

2.1 APRENDIZAJE AUTOMÁTICO

El aprendizaje automático es una rama de la Inteligencia Artificial cuyo objetivo es el desarrollo de técnicas de inducción del comportamiento: permiten a una computadora mejorar, de manera automática, un comportamiento a través de la experiencia, dándosele por ello el nombre de Aprendizaje Automático (AA) o Aprendizaje de Máquinas. Siendo más concretos, un algoritmo de AA es capaz de extraer características y patrones comunes de un conjunto de datos (datos de entrenamiento) y aplicarlas a nuevos conjuntos (datos de prueba). Esta disciplina se utiliza en diferentes campos tales como diagnósticos médicos, análisis de mercado, robótica, reconocimiento del habla o clasificación de secuencias de ADN.

Existen distintos tipos de aprendizaje automático entre los que destacamos el aprendizaje supervisado, en el que la base del conocimiento está basada en función de la etiquetación de problemas anteriores (el que usaremos en este proyecto ya que disponemos de estas anotaciones), y el aprendizaje no supervisado, en el que sólo disponemos de un conjunto de ejemplos sin anotar su categoría intentando el sistema agruparlos por similitud (17).

2.1.1 APRENDIZAJE SUPERVISADO

Este tipo de aprendizaje recibe el nombre de “supervisado” ya que deduce una función a partir de un conjunto de datos de entrenamiento que ya están etiquetados correctamente. Se usa fundamentalmente para la extracción de relaciones entre múltiples atributos y es el que usaremos en este proyecto.

En el aprendizaje supervisado nos encontramos con distintos métodos para encontrar un resultado satisfactorio, como los basados en árboles de decisión, en reglas, en redes neuronales, sobre algoritmos estadísticos… En estos algoritmos se proporciona un conjunto de datos de entrenamiento en forma de un vector. El algoritmo, tras analizar los vectores de entrada, infiere una función conocida como clasificador o función de regresión dependiendo de si la salida es discreta o continua. La función inferida debe ser capaz de predecir la clase para cualquier vector de entrada.

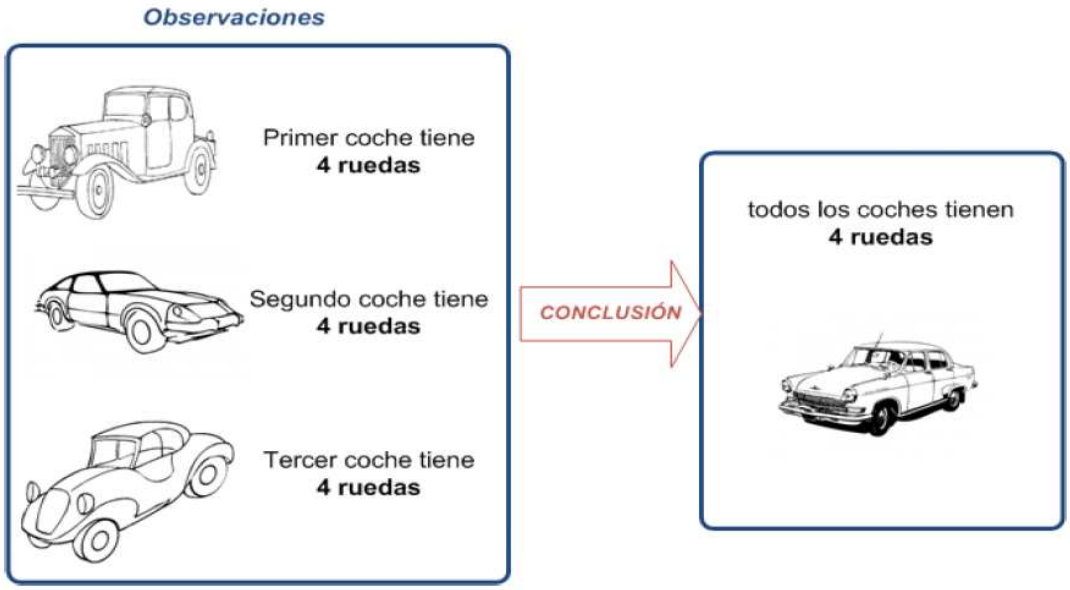


Ilustración 1: Inducción del conocimiento

En primer lugar se lleva a cabo la fase de entrenamiento, en la que a partir de un corpus se extraen los atributos relevantes para el clasificador. Estos atributos se habrán clasificado previamente, por lo que a priori ya se conoce su clase. De la extracción de atributos sale un conjunto de vectores que formarán la entrada al algoritmo, el cual a partir de los datos y sus clases generará un modelo que generalice las características extraídas del corpus. En la siguiente fase, la de predicción, se realizará el mismo proceso de extraer atributos para generar un vector, pero ya no conocemos la clase de estos ejemplos. El algoritmo, como ya habrá sido capaz de aprender un modelo, podrá aplicarlos a los datos y de esta manera predecir las clases de estos ejemplos.

2.1.2 APRENDIZAJE NO SUPERVISADO

En este tipo de aprendizaje (el cual recibe el nombre de no supervisado debido a que no se conoce nada a priori) se intenta determinar la estructura de los datos, es decir, se intentan agrupar grandes cantidades de datos en función de las características que comparten. Es una manera de encontrar jerarquía y orden en un conjunto de datos sin estructura.

Uno de los principales métodos de aprendizaje no supervisado es el clustering. Conceptualmente se suele representar viendo qué elementos forman grupos en un eje (x,y), e introduciéndolos en un círculo, tal como se muestra en la siguiente ilustración.

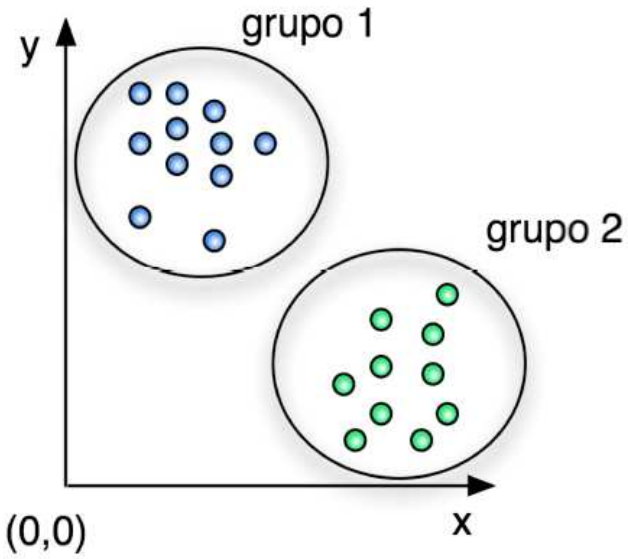


Ilustración 2: Aprendizaje no supervisado

Lo más importante en el clustering es encontrar una función que sea capaz de cuantificar las similitudes entre dos datos como un número, como por ejemplo, la distancia Euclídea entre dos puntos.

2.2 WEKA (18) (19) (20)

Weka, que proviene de Waikato Environment for Knowledge Analysis, es un programa de libre distribución y difusión que implementa algoritmos de aprendizaje automático y herramientas de procesamiento y filtrado para minería de datos desarrollado en Java por la Universidad de Waikato.

En un principio la herramienta fue pensada para analizar datos procedentes del dominio de la agricultura, sin embargo actualmente es usada en diferentes ámbitos con motivo de diferentes investigaciones. En su actual versión ofrece un poderoso entorno con una colección de algoritmos para el análisis de datos y modelado predictivo, ofreciendo además una interfaz gráfica de usuario que permite acceder cómodamente a sus funcionalidades.

Las tareas que soporta son pre procesamiento de datos, *clustering*, clasificación, regresión, visualización y selección de atributos pudiéndose ejecutar todas ellas a partir de una interfaz gráfica. También tiene una interfaz más potente, ya que proporciona acceso a todas las funcionalidades de Weka, con línea de comandos pero que resulta más compleja. Weka está compuesto por los siguientes módulos:

1. *Explorer:* está compuesto por un grupo de paneles que permite acceder a las distintas tareas de minería de datos que soporta Weka. Es el componente que más se suele utilizar. Se compone de las siguientes partes:

- *Preprocess:* este panel permite definir el origen de los datos, que puede ser importando ficheros en diferentes formatos, mediante consultas a una base de datos SQL o a partir de una URL. Una vez que se han obtenido los datos se pueden pre procesar aplicando uno o varios algoritmos de filtrado que ofrece Weka.
- *Classify:* mediante este panel se pueden aplicar algoritmos de clasificación y regresión sobre los datos para poder generar un modelo predictivo, estimar la precisión del modelo generado y así visualizar errores en las predicciones. Una vez seleccionado un algoritmo se puede elegir el modo de entrenamiento especificando si se quieren usar todos los datos para entrenar, otros datos, un porcentaje o hacer *cross-validation*. En el mismo panel se inicia la ejecución del algoritmo y una vez finalizado muestra la información referente a su desarrollo, tal como la matriz de confusión, los errores de clasificación, gráficas, el modelo generado o información sobre el dataset. También proporciona meta-clasificadores, los cuales son combinaciones de clasificadores.
- *Cluster:* en esta parte se pueden utilizar técnicas de clustering de Weka. Al igual que en el panel de clasificación, se puede aplicar un algoritmo de clustering y personalizarlo. También permite mostrar de forma gráfica la asignación de las muestras en clusters.
- *Associate:* este panel permite utilizar las reglas de asociación aprendidas para identificar asociaciones entre datos.
- *Select attributes:* mediante este panel se pueden utilizar algoritmos para identificar los atributos más relevantes en los datos, es decir, los atributos que tienen más peso a la hora de clasificar.
- *Visualize:* en este modo se muestra gráficamente la distribución de todos los atributos. De esta manera se pueden ver asociaciones entre los atributos.

2. *Experimenter:* en este módulo se pueden aplicar varios algoritmos de clasificación sobre un conjunto de datos y así poder analizar los resultados y determinar cuál de los esquemas es mejor estadísticamente. Resulta útil para realizar comparaciones de rendimiento entre varios esquemas de aprendizaje automático. Permite

configurar cada experimento especificando los datos involucrados, los algoritmos a usar, el tipo de evaluación… También se puede realizar la ejecución en modo distribuido entre varios nodos mediante RMI.

3. *KnowledgeFlow:* este módulo es una alternativa al Explorer, mostrando de una forma más explícita el funcionamiento interno de una ejecución. Permite crear flujos de datos partiendo de componentes Weka que se enlazan y combinan formando un flujo de conocimiento para procesar y analizar datos. En este módulo se proporcionan las siguientes funcionalidades:

   - Procesamiento de los datos en batch o incrementalmente (según los algoritmos lo permitan).
   - Encadenar filtros de pre procesamiento de datos.
   - Procesar flujos de datos en paralelo.
   - Visualizar los modelos intermedios generados por los clasificadores para cada fold en la evaluación cross-validation.
   - Visualización del performance de los clasificadores incrementales durante el procesamiento.

4. *Simple CLI:* el módulo Simple CLI (Simple Command-Line Interface) es una interfaz de línea de comandos a través de la cual se pueden ejecutar todas las operaciones que soporta Weka, pudiendo invocar a todas sus clases ya sean clasificadores, filtros, clusters… Por ello es bastante potente pero también más complicada de usar ya que mientras que en la interfaz gráfica es muy intuitivo realizar cualquier operación, mediante línea de comandos se requiere un conocimiento más profundo de toda la herramienta.

El acceso a los datos para su estudio se produce mediante un fichero plano *Attribute-Relation File Format* (ARFF) en el que cada registro de datos se describe por un número prefijado de atributos. No obstante, se proporciona acceso a diferentes fuentes como a bases de datos vía SQL mediante conexión JDBC procesando las consultas realizadas a la base de datos, ficheros *Comma Separated Values* (CSV) y otras fuentes de información.

2.2.1 EL FORMATO ARFF

Como hemos comentado anteriormente el formato nativo de Weka es el formato ARFF, en el que se definen los registros mediantes instancias con sus atributos. Consideramos instancia a cada grupo de características pertenecientes a un ejemplo a evaluar. Cada instancia se clasificará en función de sus atributos, que son las características que

pueden ayudar a categorizar a la instancia. La estructura del formato ARFF es la siguiente:

- *Cabecera:* donde se define el nombre de cada relación. El nombre deberá ir entrecomillado si existen espacios entre sus caracteres. La sintaxis sería la siguiente:

```
@relation <nombre-de-relación>
```

- *Declaraciones de atributos:* declaración de los atributos que conforman el archivo y el tipo correspondiente a estos atributos. Weka considera varios tipos como pueden ser NUMERIC (números reales), INTEGER (números enteros), DATE (fechas en las que se define el patrón mediante una etiqueta entrecomillada siguiendo la nomenclatura Java para las definiciones), STRING (cadenas de texto que deben ir entrecomilladas si contienen espacios), ENUMERADO (se predefinen los valores aceptados del mismo modo que se declaran arrays de cadenas en java, es decir, el tipo serían el conjunto de valores aceptados entre corchetes). Se define la sintaxis de la siguiente forma:

```
@attribute <nombre-de-atributo> <tipo>
```

- *Sección de datos:* se declaran los datos que componen la relación separando entre comas los atributos y entre saltos de línea las relaciones. Existe también un formato abreviado útil para cuando hay muchos atributos en una relación que contienen el 0, de forma que estos se ignoran y sólo se escriben los relevantes precedidos por su posición. La sintaxis para una relación con varios atributos en ambos formatos (formato completo a la izquierda y abreviado a la derecha) es la siguiente:

```
@data

8,0,0,0,0,0,2
```

```
@data

0 8,6 2
```

En el caso de querer transformar los datos, por ejemplo eliminar atributos que no sean relevantes para la clasificación, podemos pre procesar los datos recurriendo al uso de filtros que modifiquen nuestros

datos y eliminen aquellos datos que no se consideren relevantes o que solo aporten ruido.

2.2.2 FILTROS

Dentro del pre procesado nos encontramos con la aplicación de diversos filtros que nos permiten realizar transformaciones de todo tipo sobre los datos.

Entre ellos podemos mencionar algunos relacionados con el pre procesado de los atributos como el *AddExpression* que permite agregar un atributo que se corresponde con el valor de una función que se puede calcular a partir de otros atributos, la cual nos sería útil si quisiéramos usar una ecuación que fusionara varios atributos; y *StringToNominal* que convierte un atributo de tipo cadena en un tipo nominal.

El filtro *Discretize* discretiza un conjunto de valores numéricos en rangos de datos pudiendo así simplificar un atributo que recogiera valores analógicos convirtiéndolos en digitales, de manera que convertimos un atributo que podría tomar infinitos valores en un valor que tome un numero finito de valores.

Mediante el filtro *NominalToBinary* se produce una transformación de los valores nominales de un atributo en un vector cuyas coordenadas son binarias; este filtro resulta útil ya que algunos algoritmos no admiten datos nominales.

Por otra parte tenemos también filtros relacionados con las instancias, como por ejemplo el filtro *NonSparseToSparse* que convierte una muestra de un modo completado al modo abreviado reduciendo el tamaño del fichero; o el filtro *RemoveMisclassfied* que elimina el conjunto de instancias mal clasificadas, lo cual dejaría en el sistema sólo instancias bien clasificadas como ejemplo.

2.2.3 ALGORITMOS DE CLASIFICACIÓN

Weka dispone de diferentes herramientas para el análisis de datos y aprendizaje automático entre las cuales está el clasificador que agrupa tanto algoritmos de clasificación como de regresión. El más antiguo de estos modelos es el *ZeroR*, siguiéndole el *OneR* que también forma parte de los algoritmos basados en árboles de decisión, basado en un árbol de un nivel de profundidad como única regla de decisión y selecciona la clase más frecuente para el atributo con menor error cuadrático medio.

NaiveBayes en cambio implementa el clasificador probabilístico Naive Bayesian. IBk, que se basa en la implementación de los k vecinos más cercanos, es un algoritmo de aprendizaje tardío o perezoso ya que no se realiza ninguna abstracción en forma de reglas o árboles de decisión, de esta forma la instancia se clasifica en la clase más frecuente a la que pertenecen sus k vecinos más cercanos. También nos encontramos con j48 que es una implementación de C4.5 y produce árboles de decisión incorporando la poda del árbol de clasificación una vez éste se ha construido, de manera que se eliminan las ramas que aportan menor cantidad predictiva. SM0 implementa el algoritmo de optimización mínima secuencial para máquina de soporte vectorial (SVM).

2.3 LA EXTRACCIÓN DE RELACIONES: UNA TAREA DE EXTRACCIÓN DE INFORMACIÓN

Para poder realizar la detección de relaciones semánticas entre entidades de un texto hacemos uso de la extracción de relaciones. La mayoría de los sistemas de extracción de relaciones están basados en extracción de relaciones binarias (relaciones uno a uno) pero algunos sistemas pueden detectar múltiples relaciones semánticas como en la extracción de relaciones en el análisis de videos, en el que la compleja estructura de un video puede necesitar la extracción de relaciones semánticas más complejas. En el dominio biomédico la extracción de relaciones binarias se puede aplicar para detectar relaciones como la interacción proteína-proteína o fármaco-fármaco.

Ilustración 3: Ejemplo de relación

Para dicha extracción de relaciones son necesarios varios procesos de análisis del lenguaje como la extracción de raíces de las palabras y el análisis sintáctico y semántico de las frases, entre otros. Algunos de los recursos más útiles en la extracción de relaciones (21) son:

- **Información del contexto:** los elementos que rodean a las entidades o están entre ellas pueden aportar información sobre la existencia de una relación entre ambas entidades.
- **Part-of-speech tagging (POS):** la anotación de los elementos de la oración en su respectiva categoría facilita la identificación de las entidades (normalmente se corresponden con elementos etiquetados como nombres o sintagmas nominales) y las relaciones entre las entidades (que suelen etiquetarse como verbos).
- **Árboles de análisis sintáctico completo:** aportan más información que el POS ayudando a entender las relaciones entre las entidades mediante la agrupación de las palabras en una jerarquía (como sintagma nominal, sintagma verbal y sintagma preposicional); este proceso es muy costos y consume mucho tiempo.
- **Grafos de dependencia:** son la alternativa al análisis sintáctico. Representan las relaciones entre las palabras de una frase generando un grafo que una cada palabra con las que dependen de ella.

2.4 EVALUACIÓN DE SISTEMAS DE EXTRACCIÓN DE INFORMACIÓN EN BIOMEDICINA

Para poder evaluar los resultados obtenidos debemos seleccionar unas métricas claras, reproducibles y de fácil comprensión. En nuestro sistema partimos de una colección de datos que dividimos en un conjunto de entrenamiento y en un conjunto de test; a partir del conjunto de entrenamiento el sistema deberá inferir las categorías de los ejemplos del conjunto de test. Puesto que las categorías de estos ejemplos son conocidas, podremos evaluar las inferencias del sistema tomando como base la matriz de confusión en las que nos encontraremos cuatro posibilidades a las que puede optar la anotación del sistema (por encontrarnos en un problema de clasificación binaria):

- **Verdadero positivo:** aquellos elementos identificados correctamente como positivos.
- **Falso positivo:** aquellos elementos que el sistema detecta como positivos aunque en realidad sean negativos.
- **Verdadero negativo:** son aquellos elementos identificados correctamente como negativos
- **Falso negativo:** aquellos elementos que, aun siendo positivos, se anotan incorrectamente como negativos.

		RESULTADO INFERIDO	
		Positivo	**Negativo**
RESULTADO CORRECTO	**Positivo**	**VP** (Verdadero Positivo)	**FN** (Falso Negativo)
	Negativo	**FP** (Falso Positivo)	**VN** (Verdadero Negativo)

Tabla 1: Matriz de confusión

La tabla 1 muestra la estructura de una matriz de confusión, la cual es una tabla en la que cada columna representa el numero de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real; cada elemento muestra el número de ejemplos para los que la clase actual es la fila y la clase predicha es la columna.

Los falsos negativos y los verdaderos positivos están relacionados ya que ambos son positivos en la realidad; lo mismo ocurre con los falsos positivos y los verdaderos negativos al ser ambos negativos en la realidad. Por tanto denominaremos como N^+ al número de positivos reales en el sistema, es decir, a la suma de los verdaderos positivos con los falsos negativos:

$$N^+ = VP + FN$$

Ecuación 1: Número de positivos reales

Así mismo podemos definir N^- como el número de negativos reales en el sistema, o lo que es lo mismo, la suma de los verdaderos negativos con los falsos positivos:

$$N^- = VN + FN$$

Ecuación 2: Número de negativos reales

A partir de estas dos ecuaciones nos encontramos con una serie de funciones que nos sirven como medidas de evaluación de un sistema de categorización:

- **Sensibilidad o Cobertura o Recuerdo o Tasa de Verdaderos positivos (TVP o Recall):** porción de verdaderos positivos predichos entre todos los positivos posibles. Es decir, es la probabilidad de que si una instancia pertenece a una categoría, ésta instancia se clasifique con la categoría correcta. Medida muy usada en test médicos.

$$TVP = Cobertura = C = \frac{VP}{VP + FN} = \frac{VP}{N^+}$$

Ecuación 3: Tasa de verdaderos positivos o cobertura

- **Precisión:** es la proporción de instancias clasificadas como positivas correctamente frente al total de predichos para esa categoría, es decir, la probabilidad de que si una instancia sea clasificada dentro de una categoría y realmente pertenezca a ella.

$$Precisión = P = \frac{VP}{VP + FP}$$

Ecuación 4: Precisión

- **Exactitud:** porcentaje de entradas en el conjunto de evaluación que el clasificador clasifica de forma correcta.

$$Exactitud = \frac{VP + VN}{VP + FN + VN + FP} = \frac{VP}{N^+ + N^-}$$

Ecuación 5: Exactitud

Para evaluar un sistema de clasificación las medidas que se suelen utilizar son la cobertura y la precisión. La precisión mide lo bueno que puede ser un sistema clasificando mientras que la cobertura muestra la habilidad del sistema para detectar las clasificaciones correctas existentes. Un sistema perfecto sería aquel que tuviera FP = 0 y FN = 0. Nos encontramos con que la precisión y la cobertura están en contraposición y lo normal es que cuando ascienda una la otra descienda. Por ello, para poder evaluar el sistema armónicamente necesitamos una medida que combine tanto la precisión como la cobertura y para ellos contamos con la medida-F.

$$F(\beta) = \frac{(1+\beta^2)PC}{\beta^2 P + R}$$

Ecuación 6: Medida-F

Dentro de esta fórmula nos encontramos con β que es un valor positivo que aporta mayor peso o a la cobertura o a la precisión. Si β>1 se da mayor importancia a la cobertura y si β<1 se da mayor peso a la precisión. En nuestro caso usaremos β = 1 para que ambos adquieran la misma importancia.

Resumiendo, las tres medidas típicas biomédicas que tomaremos para evaluar nuestro sistema de clasificación serán precisión, cobertura y medida-F.

2.5 STANFORD

Un analizador (*parser*) de lenguaje natural es aquel que calcula la estructura gramatical de las frases, como por ejemplo qué grupo de palabras van juntas y qué palabras son el sujeto o los complementos del verbo. Los analizadores probabilísticos obtienen el conocimiento adquirido a través de frases anotadas manualmente para así intentar producir el análisis más común para nuevas frases. Estos *parsers* siguen cometiendo errores de análisis pero suelen realizar su trabajo considerablemente bien.

Las ideas técnicas básicas sobre el funcionamiento de estos parsers están presentes en "*Fast Exact Inference with a Factored Model for Natural Language Parsing*" (22) y "*Accurate Unlexicalized Parsing*" (23), ambos escritos por Dan Klein y Christopher D. Manning. En estos artículos los autores presentan un modelo de generación de estructuras arbóreas del lenguaje natural en el cual las estructuras semánticas y sintácticas están diferenciadas en modelos separados. También indican que al factorizar el modelo de esta forma, éste admite un algoritmo A* extremadamente eficiente. El parser también proporciona como salida relaciones tipadas, conocidas como relaciones gramaticales.

En la página web de este analizador (2) (*The Stanford Natural Language Processing Group*) además de la información del proyecto, está presente una demo online en la cual se permite al usuario introducir una oración, seleccionar el idioma entre inglés, chino y árabe, y realizar el análisis. Éste análisis consta de información acerca de la gramática

(apartado parse), la anotación (apartado tagging) y las dependencias entre los diferentes tokens (apartado typed dependencies).

2.5.1 DEPENDENCIAS DE STANFORD

Las dependencias de Stanford ofrecen una representación de las relaciones gramaticales entre las palabras de una oración. Han sido diseñadas para ser fáciles de entender y utilizar con eficacia. Estas dependencias se representan por el tipo la relación, la palabra gobernante y la dependiente.

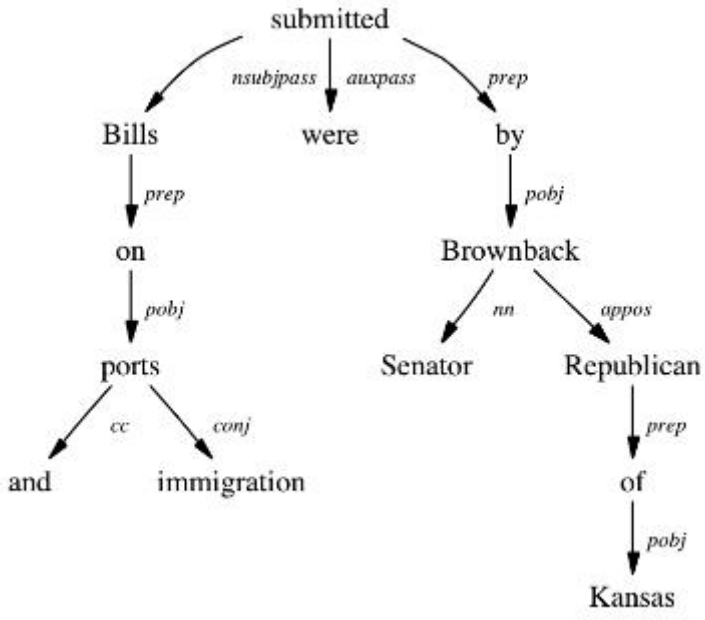


Ilustración 4: Dependencias básicas

La versión inglesa de las dependencias de Stanford ha sido desarrollada por Marie-Catherine de Marneffe, Bill MacCartney y Christopher Manning. En *"Stanford Dependencies manual"* (24) se describen todas las relaciones gramaticales existentes en la representación, y explicando las diferencias entre los cinco tipos de representación disponibles que describiremos a continuación y cómo se pueden obtener. Las representaciones siguen el mismo formato. En el formato de texto plano, una dependencia es escrita como la abreviatura del nombre de una relación (gobernante, dependiente) donde el gobernante y el dependiente son palabras de la sentencia con un número que indica la posición de la palabra en la sentencia en cuestión. El analizador también provee un formato XML que captura la misma información. Las diferencias entre los 5 formatos consisten en que van desde una representación más orientada a la superficie donde cada

token aparece como un dependiente en un árbol, a una representación más interpretada semánticamente donde ciertas relaciones de palabras, como preposiciones, son representadas como dependencias, y el conjunto de dependencias llega a ser un posible gráfico cíclico.

En la práctica las dependencias se pueden obtener de dos maneras: ya sea usando el analizador Stanford con la opción *–outputFormat typedDependencies* en texto sin formato, o directamente en los árboles de estructura de la frase con la clase *EnglishGrammaticalStructure* disponible en el paquete del analizador (nosotros hemos utilizado en este proyecto la segunda opción).

Dependencias Básicas

Las dependencias básicas forman una estructura arbórea. Esto es, no hay dependencias cruzadas o también llamada estructura de dependencias proyectivas. Cada palabra en la sentencia (excepto la raíz de la sentencia) es dependiente de otra palabra. Para la sentencia "*Bell, a company which is based in LA, makes and distributes computer products*" las dependencias básicas son las siguientes:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep(based-7, in-8)
pobj(in-8, LA-9)
root(ROOT-0, makes-11)
cc(makes-11, and-12)
conj(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

Ilustración 5: Ejemplo de dependencias básicas de Stanford

Dependencias colapsadas

En la representación colapsada, las dependencias que involucran a preposiciones, conjunciones, así como la información referente de cláusulas relativas se colapsan para obtener dependencias directas entre palabras de contenido. Este "colapso" es usado a menudo en patrones simplificados en aplicaciones de extracciones de relaciones.

Por ejemplo, las dependencias que involucran la preposición *"in"* en el ejemplo anterior serían colapsadas en una simple relación:

Ilustración 6: Ejemplo dependencias colapsadas

Además, se consideran dependencias adicionales, incluso aquellas que rompen la estructura de árbol (convirtiendo la estructura de dependencia en un gráfico dirigido) así como dependencias no proyectivas. Gracias a esto, en el ejemplo anterior se añade la siguiente relación: "*ref(company-4, which-5)*". Dicha relación no aparece en la representación básica ya que crea un ciclo con las relaciones *rcmod* y *nsubjpass*. Las relaciones que rompen la estructura de árbol son las que toman en cuenta elementos de cláusulas relativas y sus antecedentes, las relaciones de control, y la relación en el caso de preposiciones. La representación colapsada del ejemplo anterior es la siguiente:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, company-4)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep_in(based-7, LA-9)
root(ROOT-0, makes-11)
conj_and(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

Ilustración 7: Representación de dependencias colapsadas

Dependencias colapsadas con propagación de dependencias de conjunción

Cuando hay una conjunción, también se pueden tener propagación de las dependencias que incluyen conjunciones. Esta representación es una extensión de las dependencias colapsadas y no garantiza una estructura de árbol. En la sentencia que está siendo analizada, esta

propagación debe añadir dos dependencias a la representación colapsada; debido a la conjunción entre los verbos "*makes*" y "*distributes*", las relaciones de sujeto y objeto que existen en la primera parte de la conjunción ("*makes*") deben ser propagadas a la segunda parte de la conjunción ("*distributes*"):

```
nsubj(distributes-13, Bell-1)
dobj(distributes-13, products-15)
```

Ilustración 8: Ejemplo de dependencias colapsadas con propagación

Dependencias colapsadas conservando estructura de árbol

En esta representación, las dependencias que no preservan la estructura de árbol son omitidas. Esto concierne a relaciones entre elementos de una cláusula relativa y sus antecedentes, así como la relación de sujeto de control (*xsub*) y el objeto de preposición (*pobj*). Tampoco permite la propagación de dependencias de conjunción. Para nuestra oración de ejemplo la representación sería la siguiente:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep_in(based-7, LA-9)
conj_and(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
```

Ilustración 9: Ejemplo de dependencias colapsadas conservando estructura de árbol

Dependencias no colapsadas

Esta representación ofrece las dependencias básicas así como algunas dependencias extra (que rompen la estructura de árbol), sin ningún colapso o propagación de conjunciones. Hay opciones para obtener las dependencias extra separadas de las dependencias básicas. En nuestra sentencia de ejemplo esta representación sería la siguiente:

```
nsubj(makes-11, Bell-1)
det(company-4, a-3)
appos(Bell-1, company-4)
nsubjpass(based-7, which-5)
auxpass(based-7, is-6)
rcmod(company-4, based-7)
prep(based-7, in-8)
pobj(in-8, LA-9)
root(ROOT-0, makes-11)
cc(makes-11, and-12)
conj(makes-11, distributes-13)
nn(products-15, computer-14)
dobj(makes-11, products-15)
======
ref(company-4, which-5)
```

Ilustración 10: Ejemplo de dependencias no colapsadas

2.6 CONLL

CoNLL (Conference of Natural Language Learning) (25) da nombre al encuentro anual de SIGNLL. Los encuentros comenzaron en 1997 en Madrid, continuando anualmente en distintas ciudades del mundo hasta la más reciente realizada en Junio de 2011 en Portland, Oregón. En febrero de 2009, CoNLL fue nombrada la séptima conferencia más importante en el campo de Inteligencia Artificial en el sitio web cs-conference-ranking.org

Además de los temas especiales anuales, CoNLL acepta contribuciones sobre temas de aprendizaje del lenguaje como modelos computacionales de adquisición del lenguaje humano, modelos computacionales de los orígenes y evolución del lenguaje y métodos de aprendizaje automático aplicados a tareas de procesamiento del lenguaje. Desde 1999 CoNLL ha propuesto tareas específicas proporcionando conjuntos de datos para entrenamiento y evaluación que permiten a los sistemas participantes ser evaluados y comparados de una manera sistemática. A continuación resumiremos las principales tareas realizadas temas a tratar en cada conferencia:

- *CoNLL-97 (Madrid, España)* (26)*:* en 1997 se celebró el primer encuentro en el que se intentó cubrir en general tantos temas como fue posible. El rango de temas a tratar se extendió por la comprensión de mensaje, a través de la categorización de palabras, resolución de ambigüedad y segmentación de texto para la aplicación de redes neuronales en el aprendizaje del habla y fonología.

- *CoNLL-98 (Sydney, Australia)* (27)*:* los temas que se trataron durante esta conferencia fueron tales como el aprendizaje automático, representaciones arbóreas, métodos de extracción del conocimiento, aspectos semánticos, análisis multilingüe, fonología, pronunciación y morfología.
- *CoNLL-99 (Bergen, Noruega)* (28)*:* los temas a tratar este año fueron, entre otros, la conveniencia de diferentes enfoques de aprendizaje máquina (ILP, Lazy Learning, Redes Neuronales, métodos de estimación estadística, etc.) para la solución de varias tareas de NLP (fonología, morfología, sintaxis, semántica; análisis, generación, traducción; tecnología del habla), estudios de casos de tareas NLP solucionados con técnicas ML, resultados teóricos en la teoría de la facilidad de aprendizaje computacional pertinente para el aprendizaje de idiomas. Además, CoNLL invito a los grupos a tomar parte en una tarea compartida cuyos objetivos fueron la anotación de frases con información de límites NP (*noun* phrase) y la anotación de NPs reconocidos con un análisis interno.
- *CoNLL-2000 (Lisboa, Portugal)* (29)*:* En este encuentro además de los temas generales a tratar anualmente se propuso la siguiente tarea compartida: la identificación de sintagmas (constituyentes sintácticos) con métodos de aprendizaje automático, una tarea llamada fragmentación. El objetivo de esta tarea fue presentar los métodos de aprendizaje automático que después de una fase de entrenamiento pudieran reconocer la segmentación en fragmentos de los datos de prueba tan bien como fuera posible. Los datos de entrenamiento y prueba consisten en tres columnas separadas por espacios. Cada palabra se encuentra en una línea separada y con una línea en blanco entre oraciones. La primera columna contiene la palabra actual, la segunda su etiqueta POS procedente del etiquetador Brill y la tercera su etiqueta de fragmento procedente del corpus WSJ. Las etiquetas de fragmento contienen el nombre del tipo de fragmento, por ejemplo I-NP para palabras de sintagmas nominales y I-VP para palabras de sintagmas verbales. Muchos tipos de fragmentos tienen dos tipos de etiquetas, B-fragmento para la primera palabra del fragmento y I-fragmento para cualquier otra. La etiqueta de fragmento O es usada para tokens que no son partes de ningún fragmento.

```
He          PRP   B-NP
reckons     VBZ   B-VP
the         DT    B-NP
current     JJ    I-NP
account     NN    I-NP
deficit     NN    I-NP
will        MD    B-VP
narrow      VB    I-VP
to          TO    B-PP
only        RB    B-NP
#           #     I-NP
1.8         CD    I-NP
billion     CD    I-NP
in          IN    B-PP
September   NNP   B-NP
.           .     O
```

Ilustración 11: Formato CoNLL-2000

- *CoNLL-2001 (Toulouse, Francia)* (30): Este año se incluyó un tema especial en el que se centraría el encuentro: Interacción y automatización en fuentes de aprendizaje del lenguaje. También este año se propuso una tarea compartida consistente en la segmentación de un texto en cláusulas (secuencias de palabras que contienen un sujeto y un predicado). El objetivo de esta tarea es identificar las cláusulas en textos y se compone de 3 partes: identificar la posición de comienzo de la cláusula, reconocer posiciones del final de la cláusula y construir cláusulas competas. Finalmente se presentaron con métodos de aprendizaje máquina que después de una fase de entrenamiento pudieran reconocer la segmentación de cláusulas en los datos de prueba tan bien como fuera posible.
- *CoNLL-2002 (Taipei, Taiwan)* (31): La tarea compartida de CoNLL-2002 trató sobre el reconocimiento de entidades independientes concentrándose en cuatro tipos de entidades (personas, localizaciones, organizaciones y nombres de diversas entidades que no pertenecen a los tres grupos anteriores). Estaban especialmente interesados en métodos que pudieran usar datos adicionales no anotados para mejorar el rendimiento. Los datos consistían en dos columnas separadas por un espacio simple; cada palabra se encontraba en una línea separada y con una línea vacía después de cada sentencia; el primer elemento de cada línea es una palabra y el segundo la etiqueta de la entidad; las etiquetas tienen el mismo formato que en la tarea de fragmentación (una B denota el primer ítem de un sintagma y una

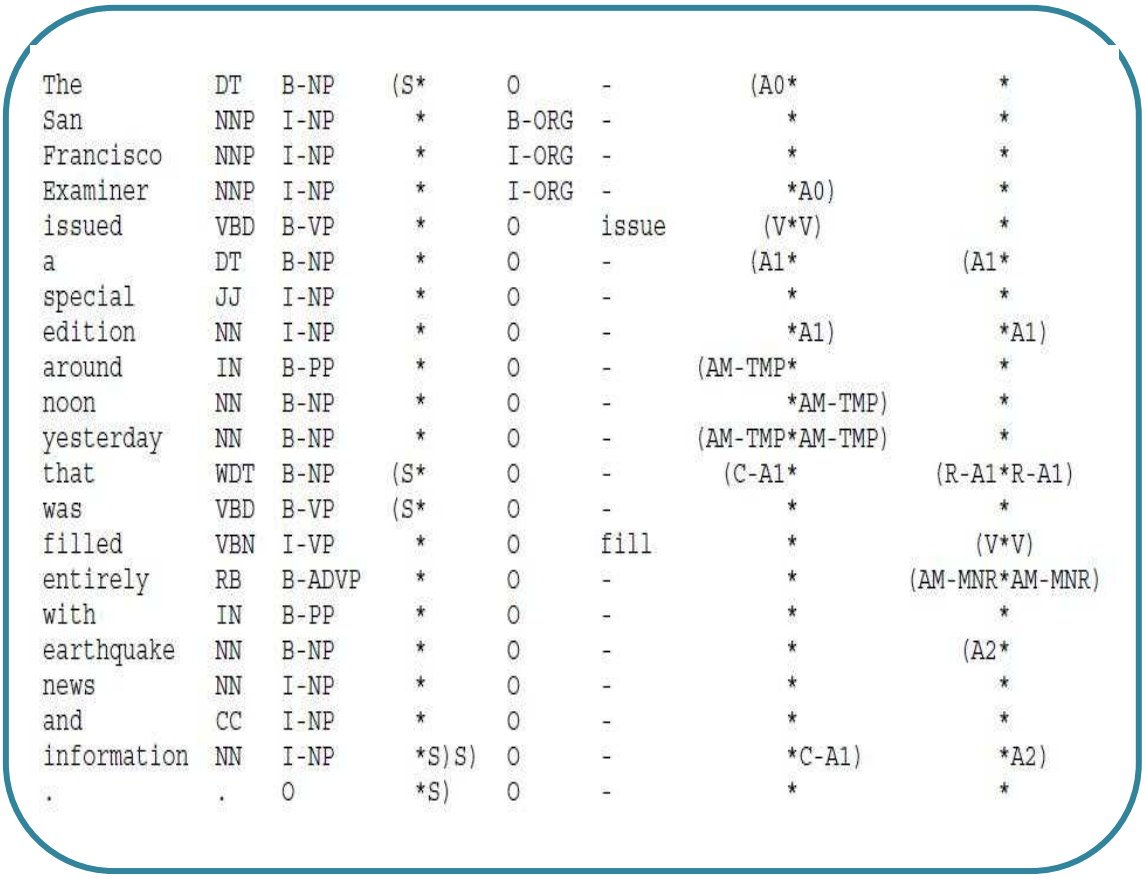
I cualquier palabra no inicial); hay cuatro tipos de sintagmas: nombres de personas (PER), organizaciones (ORG), localizaciones (LOC) y nombres varios (MISC).

- *CoNLL-2003 (Edmonton, Canadá)* (32): como en años anteriores, además de los temas generales, se fomentó el tema especial "*Aprendizaje semi-supervisado/no supervisado y técnicas de selección de muestras para aprendizaje del lenguaje*". También se propuso la siguiente tarea compartida: "enfoques de aprendizaje máquina para reconocimiento de entidades" dando especial atención al uso de múltiples fuentes de conocimiento. Este año el formato de los ficheros de datos contenía cuatro columnas separadas por un espacio simple; cada palabra se encontraba en una línea separada y con una línea vacía después de cada sentencia; el primer ítem de cada línea es una palabra, el segundo una etiqueta POS, el tercero una etiqueta de fragmento sintáctico y el cuarto la etiqueta de la entidad, tal y como se muestra en el ejemplo

```
U.N.          NNP  I-NP  I-ORG
official      NN   I-NP  O
Ekeus         NNP  I-NP  I-PER
heads         VBZ  I-VP  O
for           IN   I-PP  O
Baghdad       NNP  I-NP  I-LOC
.             .    O     O
```

Ilustración 12: Formato CoNLL-2003

- *CoNLL-2004 (Boston, USA)* (33): la conferencia del 2004 propuso una *nueva tarea compartida sobre enfoques de aprendizaje máquina para etiquetación automática de roles semánticos. El formato de este año sigue estando compuesto por varias columnas separadas por espacios. Las columnas de este formato contienen la palabra, etiqueta POS, fragmentos en formato IOB2, cláusulas en formato Start-End, entidades en formato IOB2, indicativos de verbos y una columna con los argumentos de la proposición en formato Start-End.



The	DT	B-NP	(S*	O	-	(A0*	*	
San	NNP	I-NP	*	B-ORG	-	*	*	
Francisco	NNP	I-NP	*	I-ORG	-	*	*	
Examiner	NNP	I-NP	*	I-ORG	-	*A0)	*	
issued	VBD	B-VP	*	O	issue	(V*V)	*	
a	DT	B-NP	*	O	-	(A1*	(A1*	
special	JJ	I-NP	*	O	-	*	*	
edition	NN	I-NP	*	O	-	*A1)	*A1)	
around	IN	B-PP	*	O	-	(AM-TMP*	*	
noon	NN	B-NP	*	O	-	*AM-TMP)	*	
yesterday	NN	B-NP	*	O	-	(AM-TMP*AM-TMP)	*	
that	WDT	B-NP	(S*	O	-	(C-A1*	(R-A1*R-A1)	
was	VBD	B-VP	(S*	O	-	*	*	
filled	VBN	I-VP	*	O	fill	*	(V*V)	
entirely	RB	B-ADVP	*	O	-	*	(AM-MNR*AM-MNR)	
with	IN	B-PP	*	O	-	*	*	
earthquake	NN	B-NP	*	O	-	*	(A2*	
news	NN	I-NP	*	O	-	*	*	
and	CC	I-NP	*	O	-	*	*	
information	NN	I-NP	*S)S)	O	-	*C-A1)	*A2)	
.	.	O	*S)	O	-	*	*	

Ilustración 13: Formato CoNLL-2004

- *CoNLL-2005 (Ann Arbor, USA)* (34)*:* este año se quiso promover el tema de modelado de fenómenos lingüísticos más profundo que los que han sido cubiertos en el pasado. La tarea compartida de CoNLL-2005, llamada Etiquetación de rol semántico, trató sobre el reconocimiento de roles semánticos para la lengua inglesa, con el principal objetivo de incrementa la cantidad de información semántica y sintáctica para aumentar el rendimiento de los sistemas de aprendizaje automático para la tarea. El objetivo de la tarea fue desarrollar un sistema de aprendizaje máquina para reconocer a los participantes de las proposiciones gobernadas por los verbos. El formato estaba compuesto por varias columnas separadas por espacios, con una línea para cada token y una línea en blanco después del último token. Las columnas representan diferentes anotaciones de la sentencia con etiquetas para cada token. Para anotaciones estructuradas se utilizó el formato Start-End. Las distintas anotaciones de una sentencia fueron agrupadas en los siguientes bloques: WORDS (las palabras de la sentencia), NE (Named Entities), POS (etiqueta Part-Of-Speech), PARTIAL_SYNT (sintaxis parcial, fragmentos (primera columna) y cláusulas (segunda columna)), FULL_SYNT (árbol

sintáctico completo), VS (VerbNet de los verbos), TARGETS (forma infinitiva de los verbos) y PROPS (para cada verbo, una columna que representa los argumentos del verbo).

- *CoNLL-X (New York City, USA)* (35): además de los temas tratados anualmente, este año se fomentaron los temas relacionados con teorías de aprendizaje, arquitecturas, algoritmos, métodos, o técnicas para mejorar la robustez de los sistemas NLP basados en aprendizaje. La tarea compartida de este año trató sobre análisis de dependencias multilingüe, consistiendo en asignar estructuras de dependencias etiquetadas para un rango de lenguajes por medio de un analizador de dependencias totalmente automático. Los ficheros de datos contienen oraciones separadas por una línea en blanco, con la información de cada token en líneas distintas siguiente el siguiente formato de columnas: contador de tokens empezando en 1 en cada nueva sentencia (ID), palabra (FORM), raíz de la palabra (LEMMA), etiquetas POS (CPOSTAG y POSTAG), conjunto desordenado de características sintácticas y/o morfológicas (FEATS), ID de la cabeza del token actual (HEAD), relación de dependencia con la cabeza (DEPREL), cabeza proyectiva del token actual (PHEAD) y relación de dependencia con el PHEAD (PDEPREL)

```
1   Cathy       Cathy       N     N     eigen|ev|neut                  2   su      _ _
2   zag         zie         V     V     trans|ovt|1of2of3|ev           0   ROOT    _ _
3   hen         hen         Pron  Pron  per|3|mv|datofacc              2   obj1    _ _
4   wild        wild        Adj   Adj   attr|stell|onverv              5   mod     _ _
5   zwaaien     zwaai       N     N     soort|mv|neut                  2   vc      _ _
6   .           .           Punc  Punc  punt                           5   punct   _ _
```

Ilustración 14: Formato CoNLL-2006

- *EMNLP-CoNLL 2007 (Praga, República Checa)* (36): La tarea compartida del 2007 consistió en el análisis de dependencias por segundo año con un seguimiento multilingüe como en 2006 y un nuevo seguimiento de adaptación para los diferentes dominios del Inglés. El formato de datos fue el mismo que el año anterior.
- *CoNLL-2008 (Manchester, UK)* (37): el tema especial de este año trató sobre modelos que explicaran los fenómenos naturales relacionados con el lenguaje humano. Y la tarea compartida fue el "Análisis conjunto de dependencias sintácticas y semánticas". El formato de los datos estuvo muy influenciado en los usados en

2005, 2006 y 2007. En la siguiente tabla se muestra la descripción de las columnas de dicho formato:

Nº	Nombre	Estado	Descripción
1	ID	Input	Contador de token, empieza a 1 en cada sentencia
2	FORM	Input	Palabra
3	LEMMA	Input	Raíz de la palabra
4	GPOS	Input	Etiqueta *Gold part-of-speech (POS)*
5	PPOS	Input	Etiqueta *Predicted POS*
6	SPLIT_FORM	Input	Tokens divididos por guiones y barras
7	SPLIT_LEMMA	Input	Raíz de SPLIT_FORM
8	PPOSS	Input	Etiqueta *Predicted POS* de los tokens divididos
9	HEAD	Output	Cabeza sintáctica del token actual
10	DEPREL	Output	Relación de dependencia con HEAD
11	PRED	Output	Conjunto de roles de los predicados semánticos
12+	ARG	Output	Columnas con las etiquetas de argumento para cada uno de los predicados semánticos siguiendo un orden textual

Tabla 2: Formato de datos de CoNLL-2008

- *CoNLL-2009 (Boulder, USA)* (38)*:* en el año 2009 el tema especial consistió en el estudio de métodos supervisado, semi-supervisados y mínimamente supervisados en el aprendizaje del lenguaje natural, así como métodos de aprendizaje incremental. La tarea de este año se centró en las dependencias semánticas y sintácticas en múltiples lenguajes (inglés más checo, chino, español, catalán, japonés y alemán). Los datos y el formato de este año fueron similares a los del 2008. Las columnas ID, FROM, LEMMA, POS, FEAT, HEAD y DEPREL son las mismas que en 2006 y 2007; FEAT es un conjunto de características morfológicas definidas para un lenguaje particular. PLEMA, PPOS, PFEAT, PHEAD y PDEPREL son variantes automáticamente predichas de las columnas LEMMA, POS, FEAT, HEAD y DEPREL. PRED es la misma que en los datos del 2008. Las columnas APRED se corresponden con las columnas ARG de 2008. Y FILLPRED contiene Y para las líneas en las que PRED es fijo.

- *CoNLL-2010 (Uppsala, Suecia)* (39)*:* en CoNLL-2010 el tema de especial interés fue la inducción gramatical y la tarea compartida de este año consistió en el aprendizaje para detectar coberturas y

su alcance en textos de lenguaje natural con los objetivos de aprender a detectar señales de cobertura en textos en lenguaje natural y aprender a resolver el alcance dentro de la sentencia.

- *CoNLL-2011 (Portland, Oregón, USA)* (40)*:* la tarea compartida del 2011 ha sido el Modelado de correferencia sin restricciones en OntoNotes, es decir, identificar todas las menciones de entidades y eventos en el texto y agruparlos en clases equivalentes. El formato CoNLL de este año contiene los datos en una estructura tubular similar a las usadas en tareas anteriores. En la siguiente tabla se muestra la descripción de cada una de las columnas del formato:

Columna	Tipo	Descripción
1	Document ID	Variación en el nombre del documento
2	Part Number	Algunos ficheros son divididos en múltiples partes
3	Word Number	
4	Word Itself	
5	Part-of-speech	
6	Parse bit	estructura del primer paréntesis abierto en el análisis, y la palabra/POS sustituido por un *
7	Predicate lemma	El lema predicado es mencionado por las filas de los que tenemos información de las funciones semánticas.
8	Predicate Frameset	ID de PropBank de la columna 7
9	Word sense	Sentido de la palabra de la columna 3
10	Speaker/Author	Nombre del hablante o autor
11	Named entities	identifica los tramos que representan a diversas entidades con nombre
12:N	Predicate Arguments	Una columna por cada argumento para el predicado mencionado en la columna 7
N	Coreference	información de la cadena de correferencia codificada en una estructura de paréntesis

Tabla 3: Formato CoNLL-2011

Para la realización del presente proyecto nos decidimos por la utilización del formato descrito en la edición 2006 (35), ya que en sus columnas recoge información, además de sintáctica y semántica, sobre dependencias entre tokens.

2.7 TRABAJOS ANTERIORES EXTRACCIÓN DE INTERACCIONES FARMACOLÓGICAS DE TEXTOS BIOMEDICOS

La primera aproximación desarrollada orientada a la extracción de interacciones farmacológicas fue el sistema descrito en (1), en el que se estudiaron varias técnicas de extracción de la información. Se realizó un detallado estudio sobre diversas técnicas de extracción de la información aplicadas al dominio farmacológico. Basándose en dicho estudio se propusieron dos aproximaciones distintas para la extracción de interacciones. La primera aproximación proponía un enfoque híbrido combinando el análisis sintáctico superficial y la aplicación de patrones léxicos definidos por un farmacéutico. La segunda aproximación se basó en el aprendizaje supervisado, concretamente en el uso de métodos kernels. Para la evaluación de ambas propuestas se desarrolló y anotó un corpus con interacciones farmacológicas, el corpus DrugDDI, que fue el primer corpus en el dominio biomédico anotado con este tipo de información. Los experimentos realizados demostraron que el enfoque basado en kernels consiguió mejores resultados que los obtenidos por el enfoque basado en información sintáctica y patrones léxicos.

Método	Precisión	Recall	Medida-F
Enfoque híbrido	0,487	0,257	0.336
Método Kernel (training-test)	0,50	0,71	0,58

Tabla 4: Resultados obtenidos por las dos aproximaciones presentadas en (1)

Posteriormente Víctor Manuel Méndez González, dirigido por Isabel Segura, desarrollo el sistema Druida (13) basado en la implementación SMO del algoritmo SVM (Support Vector Machine) integrado en Weka. Sus principales aportaciones fueron el uso de las tecnologías Castor y XSD (*XML Schema Definition*) y la extracción de características (atributos). Gracias a las tecnologías Castor y XSD se facilitó la extracción de características del corpus DrugDDI, ya que el uso de Castor permite un tratamiento de formatos XML más eficaz y directo, y mediante los XSD se pueden generar automáticamente las clases Java para acceder a datos del XML en cuestión. Respecto a la extracción de atributos cabe mencionar que cada instancia (par de fármacos en la misma oración) es representado como un vector de características formado por los códigos de las familias de los fármacos, un valor booleano para indicar la existencia en la oración de un verbo con significado de interacción, un valor booleano para indicar la existencia

de un verbo modal y un booleano indicando si existe una partícula de negación entre los dos fármacos. Además, para cada instancia es necesario si el par representa una interacción (ejemplo positivo) o no (ejemplo negativo). Todos estos datos son almacenados en un fichero con formato CSV. La representación basada en características del conjunto de todas las instancias extraídas del corpus DrugDDI fue utilizada para entrenar un clasificador SVM. Los resultados obtenidos en (13) fueron los siguientes

Experimento	Precisión	Cobertura	Medida-F
SVM	0,447	0,127	0,198

Tabla 5: Resultados obtenidos por el sistema DRUIDA

En julio de 2011 Beatriz Nombela Escobar defendió su proyecto fin de carrera (14) partiendo del estudio realizado por Víctor Manuel al que se le añaden características nuevas: nombre del fármaco, PoS (part-of-speech), *head word* (indica si una palabra es la principal dentro de una frase), el PoS del *head word*, ventanas de tamaño 1, 2, 3, 4 y 5 del fármaco (tokens anteriores y posteriores al fármaco en cuestión) e información de contexto en el que se encuentran los dos fármacos (caminos de palabras, PoS, tipos sintácticos, longitud del camino, número de tipos sintácticos en el camino, N-gramas, etc.). En función de los atributos que se escojan y cómo de relevantes sean, los algoritmos utilizados serán más o menos capaces de construir un buen modelo a partir de los datos. Por este motivo una de las aportaciones de este trabajo fue el estudio de algoritmos para la selección de características como los algoritmos *Gain Ratio* y CFS. Una vez finalizada la selección de atributos, se somete a los datos a distintos algoritmos de clasificación obteniendo los mejores resultados con Naive Bayes, IBk y LogitBoost. En la siguiente tabla se muestran los resultados obtenidos por los algoritmos citados anteriormente, con los mejores resultados de precisión, recall y medida-F marcados en negrita:

Experimento	Precisión	Recall	Medida-F
CFS - NaiveBayes	0,755	0,193	0,307
CFS - IBk	0,612	0,302	0,404
CFS - LogitBoost	**0,8**	0,013	0,025
GainRatio - NaiveBayes	0,71	0,326	0,447
GainRatio - Ibk	0,678	**0,342**	**0,455**
GainRatio - LogitBoost	0,692	0,058	0,107

Tabla 6: Resultados obtenidos en (14)

En el siguiente apartado describiremos los 10 trabajos presentados en el workshop DDIExctraction 2011

2.7.1 DDIEXTRACTION 2011

La tarea DDIExtraction 2011, celebrada el 7 de Septiembre del 2011 en Huelva y conjuntamente ubicada con la 27ª Conferencia de la Sociedad Española para el Procesamiento del Lenguaje Natural (SEPLN 2011), se centra en la extracción de interacciones fármaco-fármaco a partir de textos biomédicos y con el objetivo de promover el desarrollo de la minería de textos y los sistemas de extracción de información aplicada al ámbito farmacológico y la posterior comparación de técnicas avanzadas.

La tarea DDIExtraction 2011 se constituye por 10 trabajos de investigación sobre la extracción de DDI desarrollados desde distintas perspectivas, las cuales se pueden dividir en trabajos basados en características y basados en kernels.

Trabajos basados en características

En el trabajo "*Drug-drug Interaction Extraction from Biomedical Texts with SVM and RLS classifiers*" (41) se describe el sistema en cuestión, basado en *Turku Event Extraction System* que extrae información en dos pasos principales: detección de palabras de activación (nodos) que denotan entidades en el texto, y detección de sus relaciones. Cada par de entidades farmacológicas de una sentencia es un candidato a interacción farmacológica. Para la representación de características de uso un componente derivado del detector de eventos del sistema Turku, consiguiendo así características como construcciones de caminos de tokens y de dependencias, n-gramas y número de palabras de la oración. También se le añadieron las siguientes características: 1) característica para indicar si un par candidato está presente en DrugBank y si los está como un par de interacción conocido; 2) CUI del fármaco obtenido con la herramienta MetaMap (42); 3) nombres completos y abreviado del fármaco; 4) tipo semántico de la entidad farmacológica obtenido por MetaMap. Después de la extracción de características se utilizaron dos métodos de clasificación, SVM y RLS generando los siguientes resultados.

Características	Clasificador	Precisión	Recall	Medida-F
Corpus	SVM	0.6705	0.4636	0.5482
Corpus+DrugBank	SVM	0.6513	0.4874	0.5576
Corpus+DrugBank	RLS	0.5804	**0.6887**	**0.6299**
Corpus+DrugBank+MetaMap	SVM	**0.6740**	0.4901	0.5675

Tabla 7: Resultados obtenidos en (41)

"*Feature Selection for drug-drug interaction detection using machine learning based approaches*" (43) es un enfoque basado en clasificadores SVM, concretamente LIBSVM, trabajando también en la selección de características con el objetivo de obtener las más relevantes. Para cada par de entidades del corpus de desarrollo se buscó si aparecía alguna vez como interacción o no.

Las características utilizadas en este trabajo fueron las siguientes:

- **Coordinación:** antes de la extracción de características se procesaron las oraciones para borrar entidades (etiquetadas como fármacos) en estructuras coordinativas con uno de los dos fármacos candidatos añadiendo así 3 características: 1) número *de entidades borradas*; 2) *palabras de coordinación* (*or, comma, and,…*); 3) *indicador de que la sentencia ha sido reducida*.
- **Superficiales:** toman en cuenta la posición de los fármacos en la sentencia: 1) *distancia* entre los dos fármacos; 2) *presencia de otros fármacos* entre las dos entidades.
- **Léxicas:** compuestas por las palabras del contexto de las dos entidades, incluyendo verbos y preposiciones que expresan interacción: 1) *palabras y raíces que constituyen la entidad*; 2) *raíces de las tres palabras a izquierda y derecha* de la entidad candidata; 3) *raíces de las palabras entre los conceptos*; 4) *raíces de los verbos entre las 3 palabras a izquierda y derecha*; 5) *preposiciones* entre los candidatos.
- **Morfosintácticas:** tienen en cuenta la información sintáctica para la expresión de relaciones: 1) *etiquetas morfosintácticas* de las tres palabras a izquierda y derecha de las entidades; 2) *presencia de una preposición* entre dos entidades; 3) *presencia de signos de puntuación* entre los candidato; 4) *tamaño del camino del árbol constituyente* entre las dos entidades; 5) *ascendiente común más bajo* de las dos entidades en el árbol.

- **Semánticas:** 1) *tipo semántico de UMLS* de las dos entidades; 2) *clases VerbNet* de los verbos entre las tres palabras a izquierda y derecha de los conceptos y entre ellas; 3) *relación entre los dos fármacos en el UMLS*.
- **Específicas del corpus:** 1) característica que indica *si uno de los dos fármacos es el fármaco más frecuente del fichero*; 2) característica que indica *si una de las dos entidades es referida con el término "drug"*.

Los resultados obtenidos con las distintas combinaciones de los datos fueron los siguientes:

	Precisión	Recall	Medida-F
LIBSVM	0.5487	0.6119	0.5786
LIBSVM + selección de características	0.5498	**0.6503**	0.5959
LIBSVM + selección de características + conocimiento	**0.5518**	0.6490	**0.5965**

Tabla 8: Resultados obtenidos por (43)

En el trabajo "*Automatic drug-drug interaction detection: a machine learning approach with maximal frequent sequence extraction*" (44) se describe un sistema basado en aprendizaje automático (RandomForest (45)), definiendo un conjunto de características para estimular el modelo y evaluando el rendimiento del sistema con y sin *Maximal Frequent Sequences* (MFS, algoritmo de máxima secuencia). Antes de la obtención de características, se procesó cada oración del corpus tokenizándola con normas de tokenización del Inglés y particularidades como el reemplazo de los números por *num*, obtención de *raíces*, sustitución de tokens por "*#drug#*", etc. Las características utilizadas fueron las siguientes:

- **Bolsa de palabras:** se descartaron aquellas con una frecuencia menor de 3 y las *stopwords* (palabras sin significado como artículos, pronombres, preposiciones, etc. que son filtradas antes o después del procesamiento de datos en lenguaje natural). Con el conjunto resultante se genero un conjunto de datos en el que cada muestra era una posible interacción y cada característica la presencia o no de cada palabra entre los dos fármacos.
- **Categoría de palabras:** se definieron distintas categorías de palabras como subordinadas, marcadores de independencia, aposiciones, coordinadores, absolutos, cuantificadores y de negación. Para cada categoría se definieron dos características, una indicando cuántas veces aparece la palabra en la oración y

la otra indicando cuántas veces aparece la palabra entre los dos fármacos de la posible interacción.

- **Secuencia de máxima frecuencia:** es similar a la bolsa de palabras, usando secuencias de palabras como características. Se utilizó *Maximal Fequent Sequences* (MFS). Una secuencia es máxima si no es una subsecuencia de ninguna otra.
- **Características a nivel de token y carácter:** tales como número de tokens, número de comas, número de caracteres, número de veces que aparece el carácter '*%*'…
- **Características a nivel de fármaco:** se ha obtenido información sobre el fármaco principal, el fármaco más frecuente, si el fármaco es "*drug", "alcohol", "ethanol"*…

"*A machine learning approach to extract drug-drug interactions in an unbalanced dataset*" (46) nos describe un enfoque basado en aprendizaje automático usando técnicas de procesamiento del lenguaje natural (NLP) para analizar documentos y extraer las características que los representan. Se experimentó con varios algoritmos (SVM, Naive Bayes, Decisión Trees, AdaBoost) en combinación con la técnica de selección de características "*Chi-squared feature selection"* (47) para poder reducir la dimensionabilidad del problema. Las características escogidas para construir el conjunto de datos se describen a continuación. Primeramente se extrajo el ID de los fármacos indicando la sentencia y sintagma al que pertenece; a continuación se escogió un subconjunto de características compuesto por una lista de palabras clave. Cada atributo es representado como un valor binario que indica la presencia o ausencia de esta palabra clave entre los dos fármacos, considerando una ventana de 3 tokens antes del primer fármaco y 3 tokens después del segundo. También se añadieron al conjunto de características la distancia entre los fármacos en número de palabras y sintagmas, dos características que representan el tipo semántico de cada fármaco y un valor binario indicando si hay interacción farmacológica entre el par de fármacos. Los 10 mejores resultados obtenidos gracias a este enfoque son mostrados en la siguiente tabla donde CST indica si el modelo ha sido construido usando un entrenamiento sensible al coste, FS muestra cuando ha sido utilizado el método de selección de características "*Chi-squared*" (47) y KW Lem muestra que se ha utilizado un proceso de lematización.

Run	Algoritmo	Combinación	Precisión	Recall	Medida-F
1	RandomForest (I = 50)	CST, FS, Sampling	0.5000	0.4437	**0.4702**
2	RandomForest (I = 50)	CST, FS, Sampling, KW Lem.	0.4662	0.4291	0.4669
4	RandomForest (I = 10)	CST, Sampling, KW Lem.	0.4004	**0.4874**	0.4397
7	RandomForest (I = 50)	CST, KW Lem.	0.6087	0.3152	0.4154
8	MultiBoosting	FS, Sampling.	**0.6433**	0.2556	0.3659

Tabla 9: Resultados obtenidos en (46)

En "*Drug-drug interactions discovery based on CRFs, SVMs and Rule-Based Methods*" (48) se estudian tres enfoques diferentes basados en la combinación de los algoritmos CRFs, SVMs y un enfoque basado en reglas. El primero de ellos es un enfoque híbrido que combina una técnica de aprendizaje supervisado basado en CRFs con un método basado en reglas; el segundo y el tercero son muy similares, basándose en métodos de aprendizaje supervisado CRFs y SVM respectivamente. Aunque ambos métodos se centran en los pares farmacológicos y predicen directamente la presencia o ausencia de interacción, el primero considera un token por vez siendo la predicción de categoría semántica token por token. Por lo tanto, para este primer enfoque, fue necesario un paso de procesamiento para crear los pares de interacción, a partir de cada entidad utilizando las siguientes normas: 1) si una oración contiene menos de dos tokens etiquetados como *DrugInteracting* entonces no se genera ningún par de interacción; 2) un par de interacción debe contener dos tokens etiquetados como *DrugInteracting*; 3) uno y solo uno de los tokens involucrados en el par de interacción debe ser el fármaco referencial o uno de sus nombres comerciales. El conjunto de características utilizado para los tres enfoques se compone de los siguientes tipos:

- **Características ortográficas:** identificación de los tokens.
- **Características POS:** etiquetas POS de los tokens.
- **Características de puntuación:** contienen alguna puntuación especial de las oraciones. Por ejemplo después de inspeccionar el corpus observamos que dada una medicación en concreto, el signo ':' es normalmente precedido por un fármaco que

interactúa y seguido de la explicación de los efectos de la interacción.

- **Características semánticas:** 1) *característica raíz* que toma en cuenta la raíz asociada a cada palabra; 2) *característica UMLS* basada en el Metatesauro UMLS con el correspondiente grupo semántico para cada palabra; 3) *característica de nombre comercial* que reconoce los correspondientes nombres comerciales de entre los incluidos en una lista; 4) *característica de nombre farmacológico estándar* que prueba para cada token si coincide con el nombre del fármaco estándar; 5) *característica de entidad farmacológica* que permite al modelo reconocer las entidades farmacológicas anotadas por la herramienta MetaMap (42).
- **Características de contexto:** se extendieron todas estas clases de características descritas arriba para una ventana de tamaño 3.

Enfoque	Precisión	Recall	Medid-F
Híbrido	0.4037	**0.4887**	**0.4422**
CRFs	**0.6405**	0.2596	0.3695
SVMs	0.4101	0.4199	0.4149

Tabla 10: Resultados obtenidos en (48)

"An experimental exploration of drug-drug interaction extraction from biomedical texts" (49) se basa en el uso de un aprendizaje supervisado basado en características para la extracción de DDIs. Como en todos los trabajos basados en características, modela la extracción de relaciones farmacológicas como un problema de clasificación en el que cada ejemplo es generado a partir de textos y formado como un vector de características para la clasificación utilizando el software LIBSVM-2.9 (2001). Para la generación de ejemplos se tuvieron en cuenta las siguientes dos suposiciones:

- **Suposición 1:** en el paso de entrenamiento, si hay al menos unas DDI anotada en la sentencia asignamos la etiqueta DDI de esta sentencia a 1. En el paso de prueba si una sentencia ha sido clasificada como una sentencia DDI entonces todos los pares de fármacos dentro de esa sentencia son clasificados como DDIs.
- **Suposición 2:** en el paso de entrenamiento, si el número de DDIs es mayor o igual que el número de no DDIs en la sentencia, etiquetamos dicha sentencia como una sentencia DDI. En el paso de prueba, si una sentencia es clasificada como una sentencia

DDI, todos los pares de fármacos dentro de la sentencia son clasificados como DDIs también.

Para la composición del vector de características tenemos las siguientes 6 características:

- **F1: Token entre el par farmacológico.** Incluye los tokens entre dos entidades farmacológicas.
- **F2: Raíz de las entidades.** Consiste en la raíz de las entidades anotadas en la sentencia dada.
- **F3: Tipos semánticos UMLS de las entidades.**
- **F4: Información de otros fármacos.** Indica si hay otros fármacos entre el actual par farmacológico y el número que hay.
- **F5: Posición relativa entre verbos y entidades farmacológicas.** Característica que guarda si hay un verbo antes, entro o después del par farmacológico.
- **F6: Posición de verbos y entidades farmacológicas.** Se diferencia de la característica F5 (que solo guarda información de posición de verbos y entidades farmacológicas) en que indica la posición relativa de los 3 verbos más cercanos antes y después de la entidad farmacológica.

En este trabajo se configuran 5 diferentes sistemas de clasificación con diferentes estrategias de generación de ejemplos y diferentes tipos de características con los siguientes resultados:

Sistema	Descripción	Precisión	Recall	Medida-F
1	Suposición 1, todas las palabras del texto	0.1437	**0.7682**	0.2421
2	Suposición 2, todas las palabras del texto	0.3963	0.1695	0.2375
3	F1	0.3149	0.6848	**0.4314**
	F1, F2	0.2808	0.4291	0.3394
	F1, F2, F3	0.3270	0.3192	0.3231
	F1, F2, F3, F4	0.3796	0.3113	0.3421
	F1, F2, F3, F4, F5	**0.4171**	0.3563	0.3843
4	F6	0.3270	0.2728	0.2975
5	Majority voting	0.2957	0.4649	0.3615

Tabla 11: Resultados obtenidos en (49)

Trabajos basados en funciones kernel

En esta sección pasaremos a hacer un resumen de los trabajos basados en funciones kernel que participaron en la tarea DDI Extraction 2011.

En "*Drug-drug Interaction Extraction using Composite Kernels*" (50) se nos describen enfoques kernel individuales y composiciones de los mismos. Como enfoques individuales se nos presentan: 1) un método kernel lingüístico poco profundo basado en características como el propuesto por Giulano (51); 2) el método kernel denominado *midly extended dependency tree kernel* (MEDT) (52) que está basado en el análisis de árboles sintácticos; 3) varias combinaciones de los kernels MEDT (52), PST (*phrase structure* tree) (53), LC (local context) (51), GC (*global context*) (51), y SL (*shallow linguistic*) (51). En Este estudio se observo que los mejores resultados en términos de medida-F (63,7%) lo obtuvo la combinación de kernels MEDT, PST y GC denominada en el estudio como K_{GMP}.

En "*Extraction of drug-drug interactions using all paths graph kernel*" (54) se combina la información del corpus DrugDDI con información de resúmenes de MEDLINE, usando posteriormente el analizador Stanford para generar grafos de dependencias. Una vez hecho esto se aplicó el método kernel *all graph paths* (55) obteniendo los resultados más bajos de la competición (medida-F del 16,82%).

En el trabajo realizado en "*Relation Extraction for drug-drug interactions using ensemble learning*" (56) se realizan tanto un enfoque kernel enfocado en 3 métodos (AGP, kBSPS y SL) como un enfoque basado en casos y la combinación de ambos enfoques. En el enfoque basado en casos se utilizó una versión personalizada de Moara (sistema que participó en *BioNLP'09 Event Extraction Challenge* (57)) en el que cada par de fármacos es representado por el contexto local, como los tokens entre un par de fármacos y como características se consideraron las etiquetas POS, el rol y la raíz de la palabra. Los resultados obtenidos por los distintos enfoques se muestran en la siguiente tabla.

Método	Precisión	Recall	Medida-F
APG	0.55	0.752	0.634
SL	0.496	**0.762**	0.601
Moara	0.468	0.423	0.444
APG/Moara/SL	0.605	0.719	**0.657**
APG/kBSPS/SL	**0.614**	0.701	0.655

Tabla 12: Resultados obtenidos en (56)

En "*Two different machine learning techniques for drug-drug interaction extraction*" (58) se desarrolló un sistema combinado de dos métodos diferentes de aprendizaje automático para la extracción de DDIs. El primero (FBM) usa un clasificador SVM con un conjunto de características morfosintácticas y semánticas extraídas de textos. El segundo método (KBM) usa un enfoque kernel consistente en una combinación de un kernel *midly extended dependency tree kernel* (MEDT), un *phrase structure tree* (PST) y una *shallow linguistic kernel* (SL). Para el primer enfoque se escogió el siguiente conjunto de características para describir a cada candidato a DDI:

- Características de palabras: incluye las palabras de D1, de D2, palabras entre D1 y D2 y el número de palabras, las 3 palabras precedentes a D1, las 3 palabras posteriores a D2 y las raíces de todas estas palabras.
- Características morfosintácticas: incluye las etiquetas POS de las palabras de cada fármaco (D1 y D2), POS de las 3 palabras previas y de las 3 posteriores.
- Otras características: incluye, entre otros, verbos entre D1 y D2 y el número de verbos que aparecen, el primer verbo antes de D1 y el primer verbo después de D2.
- Palabras de activación: indica si una determinada palabra de activación aparece en la sentencia (*induce, inhibit,…*).
- Negación: indica si una partícula de negación es detectada a una determinada distancia de caracteres antes, entre y después de los dos fármacos en cuestión.

Para el segundo enfoque el sistema utiliza un kernel compuesto K_{SMP} que combina múltiples kernels basados en árboles y características (SL, MEDT y PST). Los resultados de ambos enfoques como de su combinación son detallados en la siguiente tabla.

	FBM	KBM	Unión
Verdaderos positivos	319	513	532
Falsos positivos	133	344	376
Falsos negativos	436	242	223
Verdaderos Negativos	6138	5927	5895
Precisión	**0.7058**	0.5986	0.5859
Recall	0.4225	0.6795	**0.7046**
Medida-F	0.5286	0.6365	**0.6398**

Tabla 13: Resultados obtenidos en (58)

Discusión de los resultados

Según las descripciones anteriores de los 10 trabajos expuestos en el workshop DDIExtraction 2011 observamos unos mejores resultados en los métodos basados en kernels, destacando entre ellos los resultados obtenidos en "*Relation Extraction for drug-drug interactions using ensemble learning*" (56) dado que es el sistema con mayor valor de medida-F (65,7%) seguido de *Two different machine learning techniques for drug-drug interaction extraction*" (58) con una medida-F del 63,98%.

En el caso de los enfoques basados en características se observan resultados en general bastante menores, a excepción de los valores obtenidos en "*Drug-drug Interaction Extraction from Biomedical Texts with SVM and RLS classifiers*" (41) con medida-F del 62,99% (precisión del 58,04% y recall del 68,87%).

En general, los enfoques basados en métodos kernels obtienen mejores resultados que los métodos basados en características. La mayoría de los sistemas han utilizado información principalmente sintáctica, sin embargo la información semántica ha sido poco utilizada.

3 DESCRIPCIÓN DEL SISTEMA

3.1 PRINCIPALES TECONOLOGÍAS APLICADAS

- *Java:* lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90 (59). Este lenguaje tiene mucha similitud con la sintaxis de C y C++ pero simplifica el modelo de objetos y elimina herramientas de bajo nivel por la tendencia a producir errores en su manipulación. Cabe destacar que corre sobre una máquina virtual de manera que es independiente de la plataforma.
- *XML:* del inglés, *Extensible Markup Language* (60), es un metalenguaje extensible de etiquetas desarrollado por el *World Wide Web Consortium* (W3C). Nació como una simplificación y adaptación del *Standard Generalized Markup Language* (SGML) y permite definir lenguajes específicos. Es un estándar de facto para el intercambio de información estructurada entre diferentes plataformas.
- *DOM:* del inglés *Document Object Model* (61), es una interfaz de programación de aplicaciones que proporciona un conjunto estándar de objetos para representar documentos *Hypertext Markup Language* (HTML) y XML, de esta forma es posible acceder y manipular los datos representados. Se puede acceder y modificar tanto el contenido como la estructura.
- *Log4j:* (62) se corresponde con una biblioteca open source desarrollada en Java por la fundación Apache Software y permite elegir la salida y el nivel de granularidad de los mensajes en tiempo de ejecución. Esta configuración se realiza en tiempo de ejecución ya sea mediante código o mediante ficheros de configuración.
- *XSD:* (16) *XML Schema* es un lenguaje desarrollado por el W3C que se usa para definir la estructura y las restricciones en cuanto a contenido y tipo en documentos XML de forma muy precisa, más allá de las normas sintácticas impuestas por el propio lenguaje XML. De esta forma se puede tener una abstracción de alto nivel correspondiente al documento.
- *CSV:* (63) es un formato abierto y sencillo utilizado para representar datos en forma de tabla en la que las columnas se separan por comas y las filas por saltos de línea. Los campos que contengan un coma, salto de línea o una comilla doble deben delimitarse mediante comillas dobles. No lleva asociado ni juego

de caracteres ni el formato para saltos de línea, de manera que todo se debe indicar al abrir el fichero.

- *Castor:* (15) es un *framework open source* desarrollado en Java para enlazar datos entre objetos Java, documentos XML y bases de datos relacionales. Castor posibilita tanto el enlace de Java con XML, persistencia de Java con *Structured Language Query* (SQL) y otros más.
- *Snowball:* (64) es un lenguaje diseñado para procesar cadenas de caracteres y crear algoritmos de stemming, es decir, implementar analizadores de palabras para extraer sus raíces. Cubre múltiples idiomas y existen implementaciones en distintos lenguajes de programación.
- *Stanford:* Este paquete es una implementación en Java de análisis probabilístico del lenguaje natural. La versión original de este analizador fue escrito principalmente por Dan Klein (2), con el código de apoyo y desarrollo de la gramática lingüística por Christopher Manning. Además de proporcionar un analizador de inglés, el analizador puede ser y ha sido adaptado para trabajar con otros idiomas. Proporciona salida de las dependencias de Stanford así como árboles de estructura de frase. El paquete incluye componentes para la invocación de línea de comandos, una interfaz gráfica de análisis de Java, y una API de Java. En el apartado 4 comentaremos más detenidamente las características de este analizador.
- *CoNLL (Conferece of Natural Language Learning):* Para nuestro proyecto utilizaremos el formato desarrollado en la edición del año 2006 de CoNLL.

3.2 EL CORPUS DRUGDDI

3.2.1 CORPUS DDI

En el dominio biomédico existen distintos corpus anotados para la realización de experimentos pero la mayoría están dedicados a la interacción de proteínas y/o genes; esto supone una gran carencia en el ámbito de las interacciones farmacológicas.

Como base para este proyecto usaremos el corpus DrugDDI que es el único extraído mediante técnicas de ingeniería lingüística para este tipo de relaciones. Este corpus, que pretende asentarse como un estándar de facto para la evaluación y aplicación de técnicas de extracción de interacciones farmacológicas, además de estar anotado

dispone de documentación que nos facilita el trabajo y es libre para uso académico.

La base de datos DrugBank[1] fue la utilizada por Isabel Segura (1) para recolectar un conjunto de textos que describían DDIs (Drug-Drug Interactions). El corpus DrugDDI surge a partir de la extracción de estos textos. A continuación se integra en el sistema el uso de la herramienta UMLS Metamap Transfer (MMTx), encargada de dividir en oraciones los textos, tokenizar, realizar el POST, el análisis sintáctico y enlazar las frases con conceptos UMLS. Finalmente se desarrolla y utiliza la herramienta visual DDIAnnotate tool para anotar las interacciones, herramienta capaz de resaltar los elementos correspondientes a fármacos y añadir la interacción correspondiente entre cada par de fármacos.

DrugBank (65), tal y como hemos comentado anteriormente, es una base de datos que representa y enlaza gran cantidad de información sobre fármacos (unas 4800 sustancias farmacológicas y más de 3000 fármacos experimentales) combinando datos químicos, farmacológicos y farmacéuticos. La información aportada por DrugBank contiene datos como la nomenclatura, propiedades físicas, estructura, fórmulas químicas y dolencias para la que están recomendadas. Además, DrugBank tiene conexiones con otras fuentes de conocimiento como *PubMed*[2] (servicio americano que ofrece información sobre revistas científicas de medicina) o *pathway*[3] (describe los procesos relacionados con el fármaco en distintos organismos).

En el campo *Drug Interactions* aparecen de forma estructurada interacciones con otros fármacos. En el campo *Interactions* de las fichas de los fármacos se pueden ver los textos extraídos de diferentes fuentes médicas. Estos datos son verificados por farmacéuticos acreditados y añadidos manualmente a la base de conocimiento. Precisamente la información contenida en el campo *Interactions* es la tomada como fuente de datos a procesar en el corpus DrugDDI al ser considerada de confianza y representativa como forma de expresar comúnmente las

[1] http://www.drugbank.ca

[2] http://www.ncbi.nlm.nih.gov/pubmed/

[3] http://pathman.smpdb.ca/

IFF; dichos textos son semejantes a los orientados al ámbito farmacéutico que nos entramos dentro de los resúmenes de Medline.

Debido a lo costoso que resulta el trabajo de anotación (DrugBank contiene 13242 DDIs descritas para cada uno de los fármacos contenidos en ella, junto con enlaces a documentos) se selecciona una porción de estos textos para realizar el procesado y la posterior anotación. Como primera aproximación para la obtención del corpus solo fueron anotados 579 documentos de los 930 que se obtuvieron al descargar, gracias a la herramienta RobotMaker, los documentos que contienen las interacciones para un grupo de 1000 fármacos escogido aleatoriamente.

Potassium-depleting diuretics are a major contributing factor to digitalis toxicity. Calcium, particularly if administered rapidly by the intravenous route, may produce serious arrhythmias in digitalized patients. Quinidine, verapamil, amiodarone, propafenone, indomethacin, itraconazole, alprazolam, and spironolactone raise the serum digoxin concentration due to a reduction in clearance and/or in volume of distribution of the drug, with the implication that digitalis intoxication may result. Erythromycin and clarithromycin (and possibly other macrolide antibiotics) and tetracycline may increase digoxin absorption in patients who inactivate digoxin by bacterial metabolism in the lower intestine, so that digitalis intoxication may result [...]

Ilustración 15: Ejemplo de texto de DrugBank sobre interacciones para la Digoxina

3.2.2 PROCESAMIENTO Y ANOTACIÓN DEL CORPUS

Para el procesamiento del corpus se utilizó *Unified Medical Language* (UMLS) (66) (67) cuyo principal objetivo es ayudar en el desarrollo de tecnologías de procesamiento de lenguaje natural en textos biomédicos. UMLS se compone de las siguientes fuentes de conocimiento:

- **El Metatesauro:** núcleo de UMLS, ontología de dominio biomédico que integra multitud de terminología. Formado por una colección de conceptos y términos extraídos de distintos

vocabularios controlados incluyendo también sus relaciones. Como ejemplo de dicha terminología que integra tenemos *Medical Subject Heading* (MeSH) (68), *Systematized Nomenclature of Medicine-Clinical Terms* (SNOMED CT) (69) entre muchas otras.

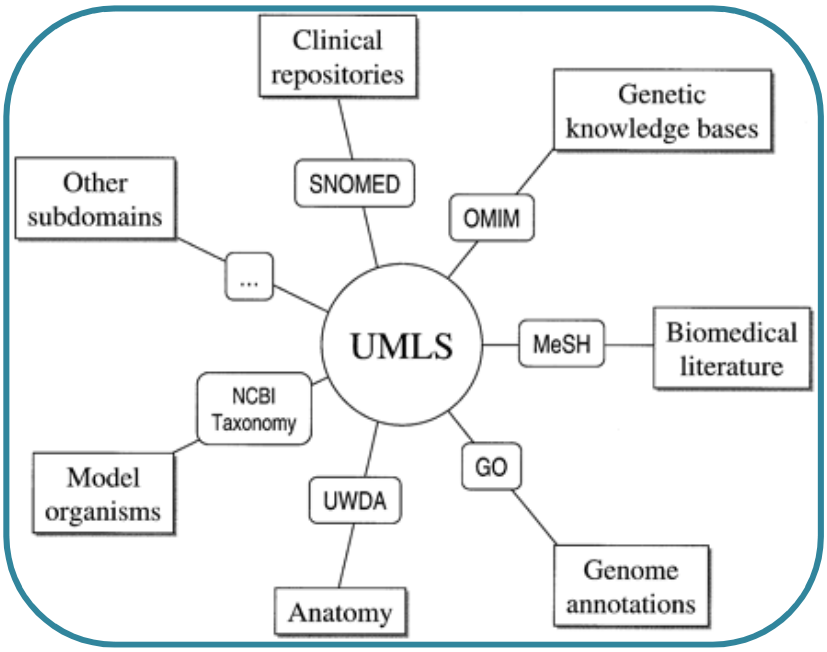


Ilustración 16: Subdominios integrados en UMLS

- **La Red Semántica:** conjunto de categorías y relaciones con el fin de poder clasificar y relacionar las entradas del Metatesauro. Está formada por 135 tipos semánticos tales como *Pharmaceutical substance* (phsu), *Amino Acid, Peptide or Protein* (aapp), *Desease or Syndrome* (dsyn) o *Gene or Genome* (gngm).
- **El Lexicón Especializado:** base de datos con información sintáctica, morfológica y ortográfica para el uso de Procesamiento de Lenguaje Natural (PLN).

Para el análisis sintáctico y semántico de los documentos del corpus se utiliza la herramienta MMTx, programa que mapea texto a conceptos del metasauro de UMLS, permitiendo procesar oraciones, dividir en tokens, PoS-tagging, parseo sintáctico profundo y enlace de frases con conceptos UMLS. De esta forma se etiquetan todos los datos necesarios y se almacena toda la información transformando la salida de MMTx a XML. Dicha salida en formato XML contiene los tipos de frases identificados, tales como *noun phrase* (NP), *prepositional phrase* (PP), *verbal phrase* (VP), *adjectival phrase* (ADJ), *adverbial phrase* (ADV), *conjunctions* (CONJ) o *unknown* (UNK).Una vez realizado el análisis sintáctico, MMTx busca las expresiones en el metatesauro de UMLS generando una serie de variantes con el uso del lexicón y técnicas

lingüísticas. Finalmente se asigna un identificador único a cada concepto, *Concept Unique Identifier* (CUI), su nombre y sus categorías semánticas.

```
<?xml version="1.0" encoding="UTF-8"?><!DOCTYPE SENTENCES SYSTEM "DTD.dtd">
<SENTENCES>
    [...]
    <SENTENCE ID="s3" TEXT="Based on adult data, lower doses of caffeine may
be needed following coadministration of drugs which are reported to decrease
caffeine elimination (e.g., cimetidine and ketoconazole) and higher caffeine
doses may be needed following coadministration of drugs that increase caffeine
elimination (e.g., phenobarbital and phenytoin).">
        <PHRASES>
            [...]
            <PHRASE ID="s3.p47" NUMTOKENS="2" TEXT="caffeine elimination"
TYPE="NP">
                <MAPPINGS>
                    <MULTIMAPCUI>
                        <MAP CUI="C0006644" NAME="Caffeine"
NAME_SHORT="Caffeine" PROB="694" SEMTYPES="orch,phsu" USAN="NO"/>
                        <MAP CUI="C0518891" NAME="Elimination outcomes"
NAME_SHORT="Elimination" PROB="861" SEMTYPES="inpr"/>
                    </MULTIMAPCUI>
                    <MULTIMAPCUI>
                        <MAP CUI="C0006644" NAME="Caffeine"
NAME_SHORT="Caffeine" PROB="694" SEMTYPES="orch,phsu" USAN="NO"/>
                        <MAP CUI="C0221102" NAME="Excretory function"
NAME_SHORT="Elimination" PROB="861" SEMTYPES="phsf"/>
                    </MULTIMAPCUI>
                </MAPPINGS>
                <TOKENS>
                    <TOKEN ISHEAD="false" ORD="0" POS="noun" WORD="caffeine"/>
                    <TOKEN ISHEAD="true" ORD="1" POS="noun"
WORD="elimination"/>
                </TOKENS>
            </PHRASE>
            [...]
        </PHRASES>
        <DDIS>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p35" DRUG_2="s3.p50" ID="1"
NAME_DRUG_1="of caffeine" NAME_DRUG_2="cimetidine" SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p35" DRUG_2="s3.p52" ID="2"
NAME_DRUG_1="of caffeine" NAME_DRUG_2="ketoconazole" SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p55" DRUG_2="s3.p67" ID="3"
NAME_DRUG_1="higher caffeine doses" NAME_DRUG_2="phenobarbital"
SEVERITY="undefined"/>
            <DDI CERTAINTY="undefined" DRUG_1="s3.p55" DRUG_2="s3.p69" ID="4"
NAME_DRUG_1="higher caffeine doses" NAME_DRUG_2="phenytoin"
SEVERITY="undefined"/>
        </DDIS>
```

Ilustración 17: Ejemplo de XML para la cafeína (Corpus DrugDDI)

Tras una revisión de la Red Semántica para obtener los diferentes tipos semánticos que representan los fármacos, se consideraron los siguientes:

- *Clinical drug*
- *Pharmacological substance*
- *Antibiotic*

- *Biologically Active Substance*
- *Chemical Viewed Structurally*
- *Amino Acid, Peptide or Protein*

En el Corpus DDI solo se anotaron las interacciones a nivel de oración, dejando así sin anotar las interacciones que abarcaban varias oraciones. En la siguiente tabla se muestra información sobre el total de documentos, oraciones, frases y tokens que componen el corpus DrugDDI.

	Conjunto de datos
Documentos	579
Oraciones	5806
Frases	66021
Tokens	127653

Tabla 14: Estadísticas de los elementos del corpus DrugDDI

En la Tabla 19 se observa el número de elementos que contienen interacciones farmacológicas y los que no las contienen.

	Total	No DDI	DDI
Ficheros	579	164	415
Oraciones	5806	3762	2044
Total de DDIs anotados		3160	

Tabla 15: Estadísticas de DDIs en el conjunto de datos

Y por último mostraremos la media de oraciones con DDIs por documento:

	Media por documento
Nº de oraciones	10,03
Nº de oraciones con al menos una DDI	3,53
Nº de oraciones que no contienen DDI	6,5
Nº de DDIs	5,46 (0,54 por documento)

Tabla 16: Media de oraciones con DDI por documento

3.3 ARQUITECTURA DEL SISTEMA

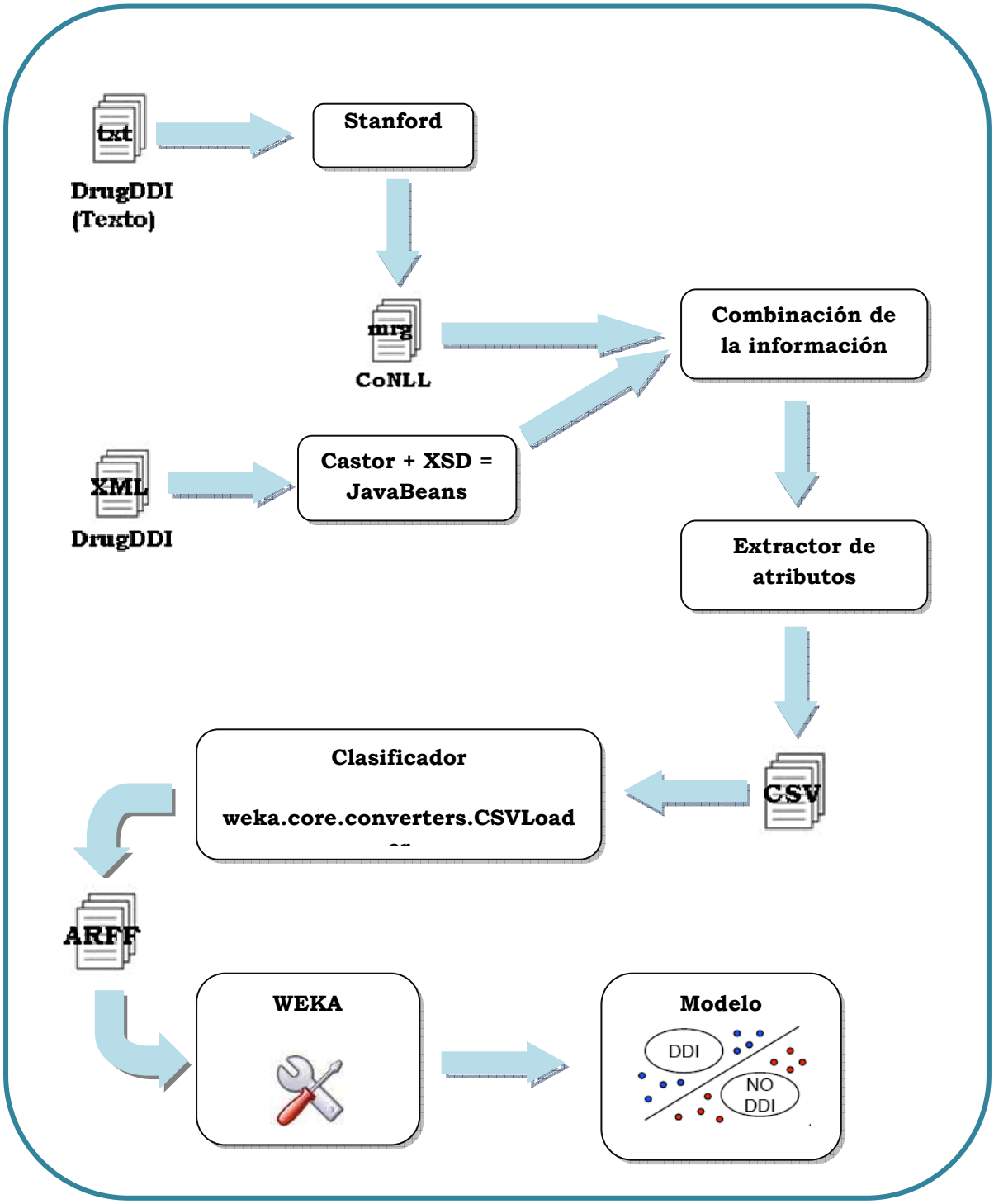


Ilustración 18: Estructura del sistema

Nuestro sistema comienza recibiendo como entrada un conjunto de ficheros en formato de texto (del cual se obtuvo el corpus DrugDDI en formato XML) que son analizados por el analizador sintáctico Stanford. Gracias a este analizador convertimos la información de dichos ficheros de texto en distintos formatos (formato *penn* de Stanford, *dependencias* de Stanford y formato *CoNLL*) basándonos en una modificación de un módulo de transformación encontrado en (70).

Posteriormente combinamos la información de los ficheros en formato CoNLL obtenidos previamente con la del corpus DrugDDI (XML). El procesador lee los ficheros XML del corpus y mediante la tecnología Castor y los *XML Schema Definition* (XSD) mapea el contenido de los ficheros a instancias Java, lo cual nos ayudará en la tarea de extracción de información. Leyendo dichos objetos Java detectamos los tokens que son fármacos mediante la comprobación de sus tipos semánticos (considerando como fármacos aquellos cuyo tipo sea "clnd"[4], "phsu"[5], "antb"[6], "bacs"[7], "chvs"[8] o "aapp"[9]) y se anota en un nuevo formato que combina la información, junto con los identificadores de oración, de sintagma, de token, el token en sí, la raíz del token, PoS, identificador del token del que dependen, relación de dependencia e identificador del sintagma del fármaco con el que interactúa (en el caso de haber interacción). En este nuevo formato, también del estilo de CoNLL, la información sintáctica anotada ha sido la obtenida gracias a Stanford (ya que dicho analizador es lo que distingue este proyecto de sus antecesores) en vez de la ya anotada por la herramienta MMTx en los XML, e integrada con la información semántica proporcionada por MMTx (reconocimiento de los nombre de fármacos por medio de los tipos semánticos descritos anteriormente). Una vez recogida toda la información se procede a cruzar todos los fármacos obtenidos y a anotar si interaccionan entre sí o no. Dicho formato intermedio tiene la siguiente estructura:

---

[4] Clinical drug

[5] Pharmacological substance

[6] Antibiotic

[7] Biologically Active Substance

[8] Chemical Viewed Structurally

[9] Amino Acid, Peptide or Protein

s0	s0.p0	1	Azlocillin	Azlocillin	NNP	5	nsubjpass	B-DRUG	s0.p6,s0.p8,s0.p11
s0	s0.p1	2	should	should	MD	5	aux	O-DRUG	_
s0	s0.p2	3	not	not	RB	5	neg	O-DRUG	_
s0	s0.p3	4	be	be	VB	5	auxpass	O-DRUG	_
s0	s0.p4	5	administered	administ	VBN	0	ROOT	O-DRUG	_
s0	s0.p5	6	concomitantly	concomit	RB	5	advmod	O-DRUG	_
s0	s0.p6	7	with	with	IN	5	prep	O-DRUG	s0.p0
s0	s0.p6	8	amikacin	amikacin	NN	7	pobj	B-DRUG	s0.p0
s0	s0.p6	9	,	,	,	8	PUNCT	O-DRUG	s0.p0
s0	s0.p7	10	ciprofloxacin	ciprofloxacin	NN	17	nn	B-DRUG	_
s0	s0.p7	11	,	,	,	10	PUNCT	O-DRUG	_
s0	s0.p8	12	gentamicin	gentamicin	NN	17	conj	B-DRUG	s0.p0
s0	s0.p8	13	,	,	,	12	PUNCT	O-DRUG	s0.p0
s0	s0.p9	14	netilmicin	netilmicin	NN	17	conj	B-DRUG	_
s0	s0.p9	15	,	,	,	14	PUNCT	O-DRUG	_
s0	s0.p10	16	or	or	CC	17	cc	O-DRUG	_
s0	s0.p11	17	tobramycin	tobramycin	NN	8	appos	B-DRUG	s0.p0
s0	s0.p11	18	.	.	.	17	PUNCT	O-DRUG	s0.p0

Ilustración 19: Ejemplo de formato que combina información de Stanford y del corpus DrugDDI

- Columna 1: identificador de sentencia con forma "s[número de sentencia]" comenzando por el identificador *s0*. Ej.: *s0.*
- Columna 2: identificador de sintagma con forma [*identificador_de_sentencia*].[identificador de sintagma]. El identificador de sintagma es de la forma "p[numero de sintagma]" comenzando con el sintagma número 0 para cada nueva oración. Ej.: *s0.p5.*
- Columna 3: identificador de token de la oración. Cada nueva oración comienza con el identificador de token 1. Ej.: 6.
- Columna 4: token en cuestión. Ej.: *concomitanly.*
- Columna 5: raíz del token. Ej.: *concomit.*
- Columna 6: etiqueta PoS del token obtenida a partir de la información semántica anotada en el corpus. Ej.: RB.
- Columna 7: número de token con el que existe una relación de dependencia, siendo un 0 cuando el token es la raíz de la oración. Esta información se obtiene de la obtenida por el análisis con Stanford. Ej.: *5.*
- Columna 8: tipo de dependencia existente entre el token actual y el indicado en la columna anterior e igualmente obtenida gracias al analizador Stanford. Cuando el token en cuestión es la raíz de la oración se indica con "*ROOT*". Ej.: *advmod.*
- Columna 9: indicador de fármaco existiendo tres posibilidades. Los tokens marcados como *"B-DRUG"* son aquellos reconocidos como token inicial de nombre de fármaco. Los marcados como "*I-DRUG*" son aquellos

identificados como tokens intermedios de nombre de fármaco y "*O-DRUG*" indica que el token en cuestión no pertenece a ningún nombre farmacológico. Ej.: *O-DRUG*.

- Columna 10: secuencia de identificadores de sintagmas que contienen una entidad farmacológica que interactúa con el sintagma al que pertenece el token actual (si no existe interacción se marca con "_". Ej.: *"_"*.

Para finalizar procedemos a la extracción de características, detalladas a continuación en la sección 3.4. Dichas características son volcadas a dos ficheros (uno de entrenamiento y otro de test) con formato CSV escribiendo las cabeceras con los nombres de los atributos y a continuación una fila por cada instancia (par de fármacos) con todas las características. Posteriormente este fichero se convierte al formato de Weka (.arff) mediante la utilidad de Weka *CSVLoader*.

3.4 CONJUNTO DE CARACTERÍSTICAS UTILIZADO PARA DESRIBIR LAS INSTANCIAS

Las características descritas a continuación en este apartado están basadas en trabajos anteriores como los realizados por Razvan Bunescu y Raymond J. Mooney en (71), por C. Giuliano, A. Lavelli y L. Romano en (51) y por M. Zhang, J. Zhang y J. Su en (72).

A partir del corpus DrugDDI y la información en formato CoNLL obtenida por el analizador Stanford (2), anteriormente descritos, proporcionaremos las instancias a los clasificadores. Combinamos dicha información sintáctica y semántica de cada oración para posteriormente generar una seria de atributos que nos servirán para entrenar los algoritmos que utilicemos.

Además de la mucha información sintáctica relacionada con los fármacos, en este proyecto se han utilizado otros atributos que nos indican si el verbo que relaciona los fármacos es negativo, modal o de interacción, que ya fueron utilizados en (13), tal y como hizo Beatriz en su PFC (14).

En primer lugar se extrae información básica de cada fármaco; continuamos extrayendo los atributos con información sobre las dependencias sintácticas obtenidas gracias al analizador Stanford y posteriormente los atributos sobre la ventana y el contexto de cada fármaco. Para finalizar extraeremos tres características heredadas del Sistema Druida (13) con valores booleanos (0 ó 1):

- Existencia de un verbo de interacción (acceler, decrease,…)
- Existencia de negación en la interacción (no, not, neither,…)
- Existencia de verbo modal en la interacción (may, might,…)

Para la explicación de los atributos tomaremos la sentencia "*Azlocillin should not be administered concomitantly with amikacin, ciprofloxacin, gentamicin, netilmicin, or tobramycin.*" como ejemplo para la relación entre Azlocillin y amikacin.

3.4.1 INFORMACIÓN LEXICA Y MORFOSINTATICA DE CADA FÁRMACO

Estas características han sido obtenidas utilizando la herramienta Standford. Uno de los objetivos del proyecto es estudiar el efecto de un analizador sintáctico independiente del dominio (como Standford) y compararlo con los resultados obtenidos utilizando el analizador MetaMap utilizado en trabajos previos (1) (13) (14). Las características son los siguientes:

- **words_drug1**: atributo con los tokens que componen el primer fármaco de la relación. Ej.: *Azlocillin*
- **word_pos_drug1**: atributo POS de los tokens del primer fármaco de la relación. Ej.: *NNP*
- **words_drug2***:* atributo con los tokens que componen el segundo fármaco de la relación. Ej.: *amikacin*
- **word_pos_drug2**: atributo POS de los tokens del segundo fármaco de la relación. Ej.: *NN*
- **path_tokens**: camino de tokens en la sentencia desde el primer fármaco hasta el segundo. El carácter $ indica tanto el principio como el final del atributo y los tokens son separados por guiones bajos. Ej.: *$should_not_be_administered_concomitantly_with$*
- **path_pos**: POS de los tokens del camino entre ambos fármacos. Tanto el comienzo como el final es indicado mediante el carácter $ y los tokens son separados por guiones bajos. Ej.: *$MD_RB_VB_VBN_RB_IN$*
- **path_length**: número de tokens entre ambos fármacos. Ej.: *6*

Contexto y ventanas de cada fármaco

Para la representación del contexto entre ambos fármacos hemos hecho uso de N-gramas, grupos de n tokens, de las palabras y de los POS entre cada par de fármacos, en concreto de bigramas y trigramas.

Cada uno de dichos N-gramas se ha separado por el símbolo $. Para la oración de ejemplo que está siendo analizada en esta memoria los atributos en cuestión serían los siguientes:

- **bigrams**:*$should_not$not_be$be_administered$administered_concomitantly$concomitantly_with$*
- **trigrams**:*$should_not_be$not_be_administered$be_administered_concomitantly$administered_concomitantly_with$*
- **pos_bigrams**:*MD_RBRB_VBVB_VBNVBN_RBRB_IN*
- **pos_trigrams**:*MD_RB_VBRB_VB_VBNVB_VBN_RBVBN_RB_IN$*

La ventana de un fármaco se entiende como un determinado número de tokens que rodea a dicho fármaco. Pueden ser de distintos tamaños, obteniendo el número de tokens anteriores y posteriores al fármaco indicado. En este proyecto se han obtenido ventanas, tanto de palabras como de PoS, de tamaños desde 1 hasta 5 para cada uno de los dos fármacos de la instancia.

Los nombres de los atributos se han formado siguiendo el patrón window{tamaño}_[word|pos]_drug{nºfármaco}, siendo así por ejemplo el atributo *window4_word_drug2* el atributo de ventana 4 de los tokens del segundo fármaco. El nombre del fármaco cuya ventana se obtiene se representa con la cadena *drugwindow*, haciendo que el calor completo del atributo sea más común y poder generar un modelo más robusto. Para conocer cómo el patrón de la interacción de una forma más genérica también se ha considerado sustituir el nombre de cualquier fármaco por su clasificación como fármaco, registrado en el formato del estilo de CoNLL que mezcla información de DrugDDI con la de Stanford (estas etiquetas son B-DRUG, I-DRUG Y O-DRUG que significan token principal del fármaco, token intermedio o que no es un fármaco).

Para la oración de ejemplo que estamos utilizando tendríamos los siguientes atributos, basados en los atributos de tamaño de ventana utilizados en (14):

- **window1_word_drug1**: *drugwindow_should*
- **window1_pos_drug1**: *drugwindow_MD*
- **window2_word_drug1**: *drugwindow_should_not*
- **window2_pos_drug1**: *drugwindow_MD_RB*
- **window3_word_drug1**: *drugwindow_should_not_be*
- **window3_pos_drug1**: *drugwindow_MD_RB_VB*

- **window4_word_drug1**: *drugwindow_should_not_be_administered*
- **window4_pos_drug1**: *drugwindow_MD_RB_VB_VBN*
- **window5_word_drug1**: *drugwindow_should_not_be_administered_concomitantly*
- **window5_pos_drug1**: *drugwindow_MD_RB_VB_VBN_RB*
- **window1_word_drug2**: *with_drugwindow_COMMA*
- **window1_pos_drug2**: *IN_drugwindow_COMMA*
- **window2_word_drug2**: *concomitantly_with_drugwindow_COMMA_ciprofloxacin*
- **window2_pos_drug2**: *RB_IN_drugwindow_COMMA_NN*
- **window3_word_drug2**: *administered_concomitantly_with_drugwindow_COMMA_ciprofloxacin_COMMA*
- **window3_pos_drug2**: *VBN_RB_IN_drugwindow_COMMA_NN_COMMA*
- **window4_word_drug2**: *be_administered_concomitantly_with_drugwindow_COMMA_ciprofloxacin_COMMA_gentamicin*
- **window4_pos_drug2**: *VB_VBN_RB_IN_drugwindow_COMMA_NN_COMMA_NN*
- **window5_word_drug2**: *not_be_administered_concomitantly_with_drugwindow_COMMA_ciprofloxacin_COMMA_gentamicin_COMMA*
- **window5_pos_drug2**: *RB_VB_VBN_RB_IN_drugwindow_COMMA_NN_COMMA_NN_COMMA*

Estas características son heredadas tanto del sistema DRUIDA (13) como del desarrollado por Beatriz (14).

- **Neg.Part**: valor booleano que indica si en la interacción aparece un elemento negativo o no (not, absent,…) Ej.: *1*
- **Modal:** valor booleano que indica si en la interacción hay un verbo modal (can, may, …). Ej.: *1*
- **verb.Interact:** valor booleano que indica si hay un verbo de interacción (increase, acceler, …). Ej.: *0*
- **DDI:** valor booleano que indica si hay interacción entre los dos fármacos. Ej.: *1*

Modales	can, could, may, might, should, must, have, has, had
Negación	but not, not, have no effect, has no effect, no detected, unable to, neither, lack of, no, fail to
Interacción	elevate, enhance, exacerbate, extend, increase, intensify, potentiate, promote, prolong, raise, rise, stimulate, augment, go up, accelerate, antagonize, alter, change, induce, influence, inhibit, interfere

Tabla 17: Valores considerados como modal, negación e interacción

3.4.2 DEPENDENCIAS SINTÁCTICAS

Este tipo de atributos tampoco precisa ningún procesamiento en especial ya que la información nos la proporciona Stanford. Sólo hay que ir siguiendo los índices de los tokens de los que depende el que nos interesa hasta llegar a la raíz de la oración (generalmente un verbo):

- **path_words_to_root1**: camino de tokens dependientes desde el fármaco 1 hasta la raíz. Ej.: *$Azlocillin_administered$*
- **path_syntactic_type_to_root1**: tipos de dependencias de los tokens del camino de dependencias entre el fármaco 1 y la raíz. Ej.: *$nsubjpass_ROOT$*
- **path_words_to_root2**: camino de tokens dependientes desde el fármaco 2 hasta la raíz. Ej.: *$amikacin_with_administered $*
- **path_syntactic_type_to_root2**: tipos de dependencias de los tokens del camino de dependencias entre el fármaco 1 y la raíz. Ej.: *$pobj_prep_ROOT$*

3.5 EXPERIMENTOS

Previamente a iniciar esta fase, se han obtenido ficheros arff con distinto número de instancias; dado que el corpus DrugDDI es un corpus completamente desbalanceado, con tan solo un 10% de instancias positivas, se obtuvieron ficheros de entrenamiento con distintos porcentajes de instancias negativas (25%, 50%, 75% y 100%) y la totalidad de instancias positivas del conjunto de entrenamiento. El motivo de esta diferenciación de porcentajes es estudiar cómo afectan distintas proporciones de instancias negativas en el proceso de aprendizaje. En esta fase hemos realizado experimentos tanto utilizando *cross-validation* como con el conjunto test, con el objeto de poder compararnos con los resultados obtenidos en los trabajos presentados en la tarea DDIExtraction2011. Durante el proceso de experimentación nos han surgido problemas de memoria con Weka. Estos problemas son

frecuentes si existen grandes cantidades de datos. Para intentar solucionarlos personalizamos uno de los parámetros de la Máquina Virtual de Java para ejecutar Weka con más memoria que la asignada por defecto por el sistema operativo. A los parámetros de ejecución de Weka le añadimos –Xmx, que asigna el máximo tamaño de la pila de Java estableciendo el tamaño máximo de la memoria que la JVM va a utilizar para ejecutar dicha herramienta. Con este parámetro conseguimos eliminar prácticamente todos los problemas de memoria que nos surgieron, a excepción de los surgidos con los algoritmos MultilayerPerceptron y SMO.

3.5.1 CROSS-VALIDATION

Con la técnica *cross-validation* el corpus original se subdivide en un número de conjuntos denominados *folds*. Se entrena el modelo para cada uno de los *folds* utilizando todos los datos excepto los incluidos en el subconjunto y a continuación se procede a probar el modelo con los datos anteriormente excluidos en el entrenamiento. La elección del valor *k* depende del tamaño y características de los datos y dado que el valor más común es 10 (se ha comprobado a través de infinidad de pruebas con diferentes técnicas de aprendizaje que con la división en 10 *folds* se obtiene la mejor estimación de error) en nuestro experimento hemos utilizado dicho valor.

Como puede observarse en la siguiente tabla, en la que se han marcado en negrita las mejores medidas, se aprecian unos resultados bastante variados. Fijándonos en la precisión y el recall observamos una gran variabilidad en los casos de Naive Bayes y MultiBoostAB. En el caso de Naive Bayes la precisión es de la segunda más baja obtenida (30,8%) mientras que el recall es el más alto de todos los resultados (93,3%). Por el contrario, MultiBoostAB presenta una precisión elevada (71,7%) frente a un recall insignificante (2,1%). Los algoritmos HyperPipes e IBk tienen valores más similares, siendo menor la diferencia entre precisión y recall. En cuanto a los valores obtenidos de la medida-F (que combina precisión y recall) los valores no son muy elevados, siendo MultiBoostAB el algoritmo con peor resultado (0,041) e IBk el algoritmo con mejor valor (0,542).

Algoritmo	Precisión	Recall	Medida-F
Bagging	0,662	0,124	0.209
ClassificationViaClustering	0,162	0,348	0,221
HyperPipes	0,614	0,305	0,407
IBk	*0,594*	*0,498*	***0,542***
IBk (K = 2)	***0,717***	*0,387*	*0,502*
IBk (K = 3)	*0,680*	*0,449*	*0.541*
JRip	***0.717***	*0,159*	*0,260*
KStar	0,544	0,488	0,515
MultiBoostAB	***0,717***	*0,021*	*0,041*
NaiveBayes	*0,308*	***0,933***	*0,463*
OneR	0,660	0,125	0,21
Prism	***0,716***	*0,432*	*0,539*
RandomForest	0,688	0,152	0,249
RandomSubSpace	0,703	0,128	0,216
RandomTree	0,564	0,206	0,302
VFI	0,512	0,539	0,525

Tabla 18: Resultados obtenidos mediante cross-validation

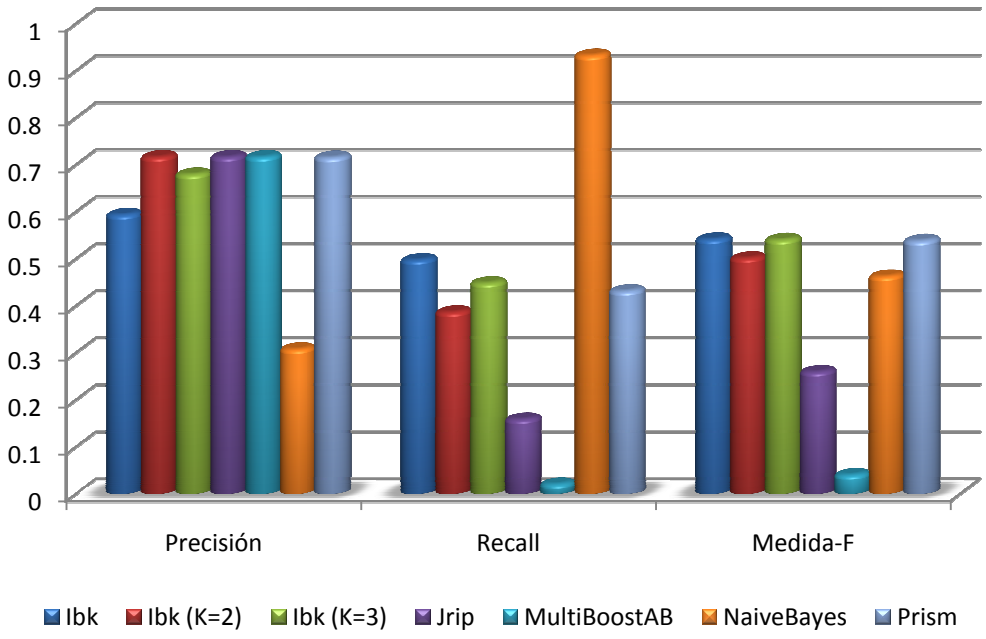


Tabla 19: Comparación resultados cross-validation

3.5.2 TRAINING-TEST

A continuación realizamos experimentos con el método de evaluación *training-test*, en el que los datos están repartidos en dos conjuntos: entrenamiento y evaluación. Los datos de entrenamiento se utilizan para el entrenamiento del modelo mientras que los de pruebas se utilizan para la validación de dicho modelo. La siguiente tabla muestra la estructura de dichos conjuntos de datos, los mismos que se utilizaron en la tarea DDIExtraction 2011.

Corpus	Oraciones	Pares positivos	Pares negativos	Pares totales
Training	4,267	2,402	21,425	23,827
Test	1,539	755	6,271	7,026

Tabla 20: Estructura del corpus DrugDDI

A continuación mostramos los resultados obtenidos por diferentes algoritmos para el mismo fichero de prueba habiendo entrenado los algoritmos con ficheros de entrenamiento con distintos porcentajes de instancias negativas (25%, 50%, 75%, 100%).

Algoritmo	Precisión	Recall	Medida-F
Bagging	0.126	0.105	0.114
ClassificationViaClustering	*0.138*	*0.238*	*0.175*
HyperPipes	0.164	0.218	0.187
IBk	*0.175*	*0.338*	*0.23*
IBk (K = 2)	*0.191*	*0.301*	*0.234*
IBk (K = 3)	*0.188*	*0.308*	*0.233*
JRip	**0.282**	0.127	0.175
KStar	0.148	0.338	0.206
MultiBoostAB	0.226	0.45	**0.301**
NaiveBayes	*0.143*	***0.747***	*0.24*
OneR	0.125	0.103	0.113
Prism	0.129	0.192	0.155
RandomForest	0.126	0.118	0.122
RandomSubSpace	0.125	0.103	0.113
RandomTree	0.129	0.111	0.12
VFI	*0.145*	*0.57*	*0.231*

Tabla 21: Resultados para algoritmos entrenados con fichero con 25% de negativas

Los resultados para el fichero de entrenamiento con el 25% de instancias negativas muestran una precisión del 28.2% con el algoritmo JRip pero su recall y medida-F son inferiores a los de otros algoritmos. El algoritmo NaiveBayes, por el contrario ofrece un porcentaje muy elevado de recall del 74.7% pero por el contrario pierde en precisión (14,3%). La mejor medida-F obtenida ha sido con el algoritmo MultiBoostAB obteniendo una precisión del 22.6%, recall del 45% y medida-F del 30.1%.

Algoritmo	Precisión	Recall	Medida-F
Bagging	0.126	0.103	0.113
ClassificationViaClustering	*0.09*	*0.041*	*0.057*
HyperPipes	0.194	0.041	0.068
IBk	*0.191*	*0.205*	*0.198*
IBk (K = 2)	*0.23*	*0.157*	*0.187*
IBk (K = 3)	***0.243***	*0.162*	*0.194*
JRip	0	0	0
KStar	0.144	0.197	0.166
MultiBoostAB	0	0	0
NaiveBayes	*0.145*	***0.806***	***0.245***
OneR	0.123	0.098	0.109
Prism	0.121	0.106	0.113
RandomForest	0.149	0.072	0.097
RandomSubSpace	0.123	0.098	0.109
RandomTree	0.131	0.179	0.151
VFI	*0.141*	*0.546*	*0.224*

Tabla 22: Resultados para algoritmos entrenados con fichero con 50% de negativas

En el caso de los datos con el fichero de entrenamiento del 50% de instancias negativas, la mejor precisión la ofrece el algoritmo IBk (K = 3) siendo del 24.3% y con un recall y medida-F del 16.2% y 19.4% respectivamente. Centrándonos en el recall, el algoritmo que mejores resultados ofrece es NaiveBayes con un 80.6%, bajando la precisión al 14.5% y subiendo la medida-F al 24.5%

Con el 75% de las instancias negativas de entrenamiento obtenemos con el algoritmo HyperPipes una precisión de 29.6% pero con un recall alarmantemente bajo (1.7%). Por el contrario, el algoritmo NaiveBayes nos ofrece un recall elevado (80.3%) frente a una precisión no tan baja (14.2%).

Algoritmo	Precisión	Recall	Medida-F
Bagging	0.125	0.098	0.11
ClassificationViaClustering	*0.162*	*0.496*	*0.224*
HyperPipes	**0.296**	0.017	0.033
IBk	*0.176*	*0.129*	*0.149*
IBk (K = 2)	*0.264*	*0.1*	*0.146*
IBk (K = 3)	*0.282*	*0.1*	*0.148*
JRip	0	0	0
KStar	0.131	0.133	0.132
MultiBoostAB	0	0	0
NaiveBayes	*0.142*	***0.803***	***0.242***
OneR	0.123	0.096	0.108
Prism	0.111	0.084	0.096
RandomForest	0.189	0.046	0.074
RandomSubSpace	0.123	0.096	0.108
RandomTree	0.123	0.096	0.108
VFI	*0.139*	*0.548*	*0.222*

Tabla 23: Resultados para algoritmos entrenados con fichero con 75% de negativas

Finalmente con el total de las instancias negativas de entrenamiento obtenemos con el algoritmo HyperPipes una precisión del 50% con un recall insignificante del 0.4%. Con el algoritmo NaiveBayes la precisión desciende hasta el 14.1% aumentando por el contrario la medida recall hasta el 79.5% y consiguiendo una medida-F del 24%. Otro algoritmo con similar medida-F es ClassificationViaClustering con un 25.5%, una precisión del 22.6% y un recall del 29.3%.

Algoritmo	Precisión	Recall	Medida-F
Bagging	0.123	0.094	0.107
ClassificationViaClustering	*0.226*	*0.293*	***0.255***
HyperPipes	**0.5**	0.004	0.009
IBk	*0.185*	*0.098*	*0.128*
IBk (K = 2)	*0.287*	*0.072*	*0.115*
IBk (K = 3)	*0.302*	*0.07*	*0.113*
JRip	0	0	0
KStar	0.128	0.098	0.111
MultiBoostAB	0	0	0
NaiveBayes	*0.141*	***0.795***	*0.24*
OneR	0.118	0.092	0.103
Prism	0.119	0.075	0.092
RandomForest	0.324	0.024	0.045
RandomSubSpace	0.118	0.092	0.103
RandomTree	0.118	0.092	0.103
VFI	*0.14*	*0.537*	*0.222*

Tabla 24: Resultados para algoritmos entrenados con fichero con 100% de negativas

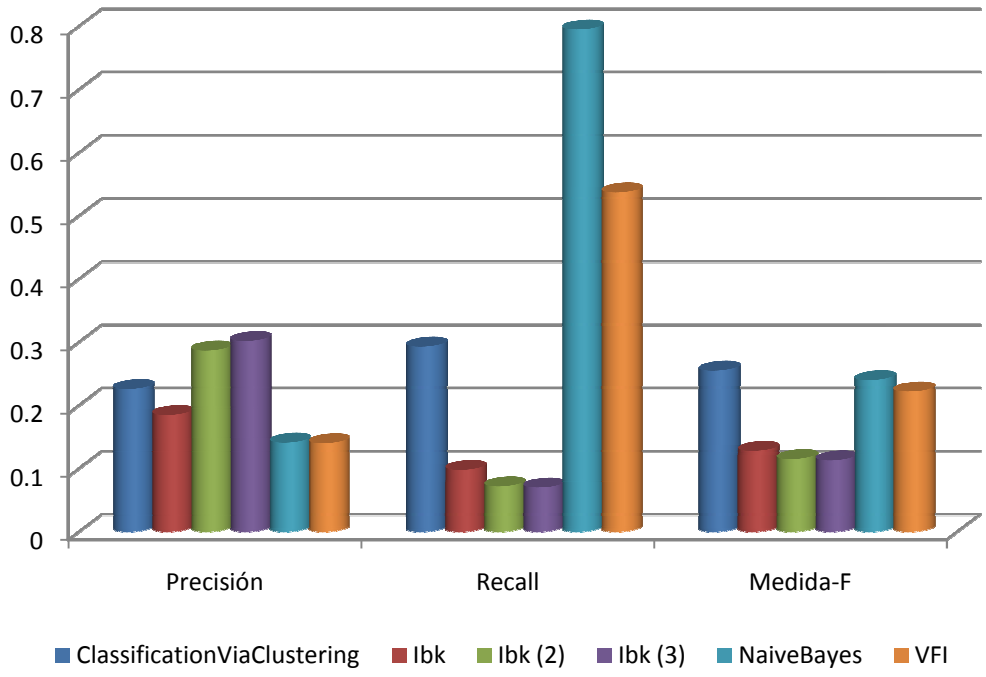


Tabla 25: Comparación de algoritmos para entrenamiento con 100% de instancias

En la tabla 25 mostramos gráficamente una comparación de los algoritmos que mejor resultados han generado con el mayor número de instancias negativas, ya que este caso de sistema no balanceado es el que más se asemeja a la realidad de este dominio.

Analizando los resultados obtenidos con distintos porcentajes de instancias negativas, observamos que la mayoría de los algoritmos tienden obtener resultados más bajos (tanto en medida-F, recall como precisión) cuanto mayor es el porcentaje de instancias negativas. En algunos algoritmos como el Naive Bayes hemos observado una ligera mejoría en los resultados entre el 25% y 50% seguida de un descenso de los resultados según va aumentando el porcentaje de instancias negativas.

3.5.3 APORTACIÓN DE LOS TIPOS DE CARACTERISTICAS

En esta sección dividimos las características entre tres los tipos descritos en la sección 3.4 (información léxica y morfosintáctica, contexto y ventanas de cada fármaco y dependencias sintácticas). Para cada grupo de características construimos tres ficheros (fichero de entrenamiento, fichero de prueba, y fichero con el total de las instancias) anotando únicamente las características del grupo en cuestión.

Para cada grupo de características, se utilizó el fichero con el total de las instancias correspondiente para entrenar el algoritmo IBk (k = 1) con *cross-validation (folds 10)*. Posteriormente, los ficheros de entrenamiento y prueba fueron utilizados para entrenar y evaluar el mismo algoritmo IBk *(training-test)*.

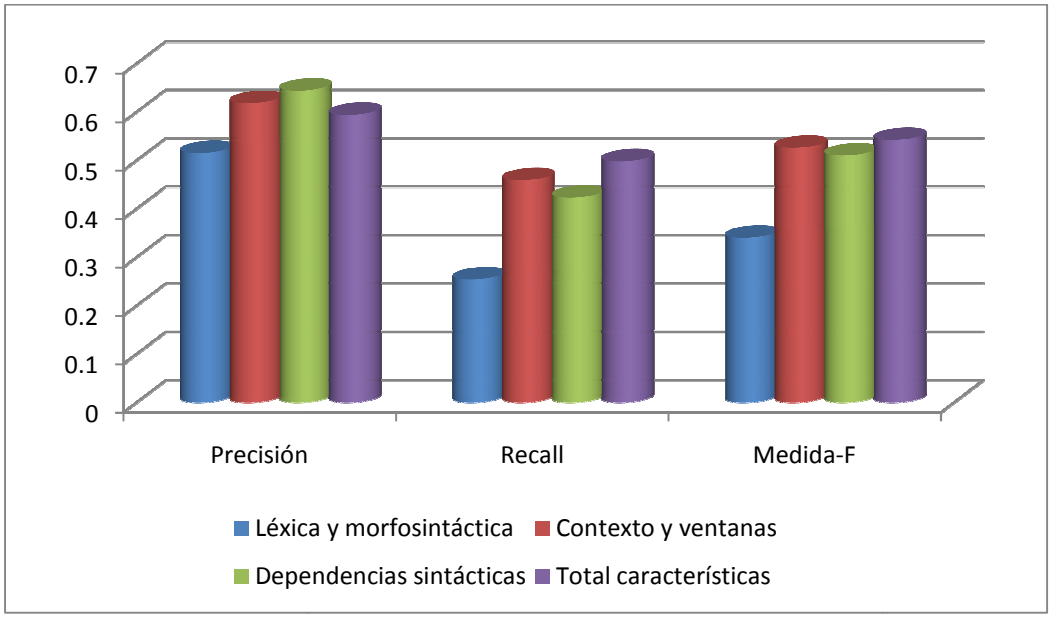
En los siguientes apartados comparamos los resultados de cada uno de los grupos tanto para el método *cross-validation* como con *training-test*.

Cross-validation

La siguiente tabla muestra los resultados obtenidos para el algoritmo IBk. Elegimos dicho algoritmo para la comparación de resultados ya que fue el que obtuvo mejores resultados para el método cross-validation evaluado sobre el conjunto completo de características (medida-F = 0,542). Los resultados obtenidos en cada uno de los tres grupos de características fueron los siguientes:

Grupo de características	Precisión	Recall	Medida-F
Léxica y morfosintáctica	0,515	0,254	0,34
Contexto y ventanas	0,618	0,459	0,526
Dependencias	0,644	0,423	0,51
Conjunto total	0,594	0,498	0,542

Tabla 26: Comparación entre grupos de características para cross validation



Como conclusión sobre los resultados anteriores podemos comentar que el análisis de los distintos grupos de características por separado no mejora los resultados del conjunto total, cuyas medida-F y recall son superiores a cualquiera de las demás. Dentro de ellas, el grupo de características que consigue unos mejores resultados son el contexto y ventanas de cada fármaco.

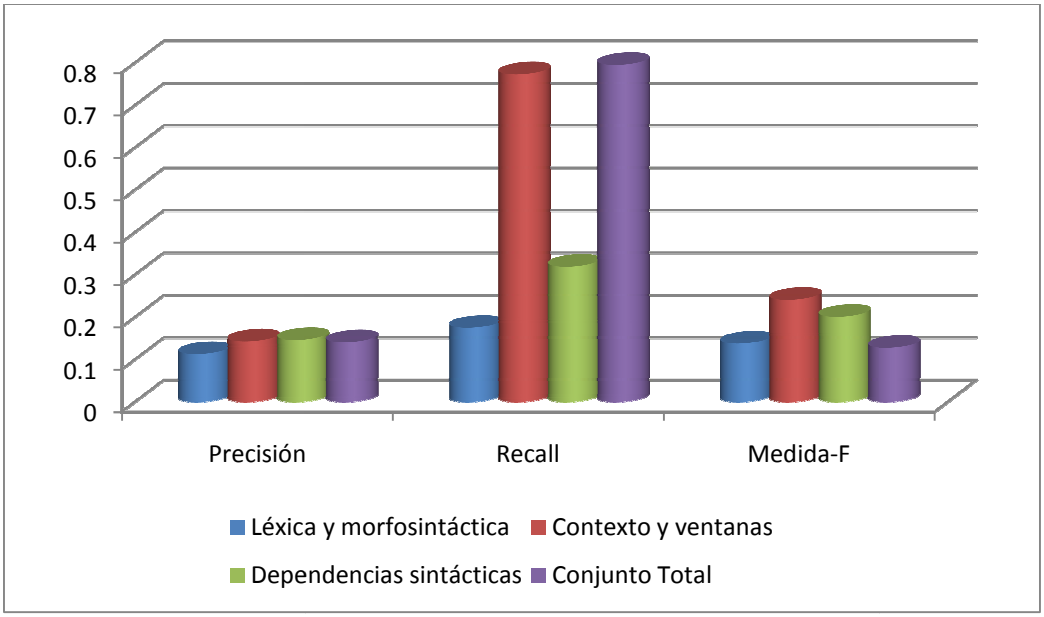
Training-Test

En esta sección evaluamos los datos con el algoritmo Naive Bayes utilizando el conjunto training para entrenar y evaluando sobre el conjunto test para cada uno de los tres grupos de características descritos anteriormente. La siguiente tabla contiene los resultados obtenidos para cada conjunto de características y el conjunto total. En este análisis, no se ha realizado un estudio sobre la influencia del conjunto de instancias negativas y positivas sobre los resultados finales, conservando el 100% de las instancias negativas. Aunque hubiera sido interesante estudiar el efecto de las características en distintas distribuciones de ejemplos negativos, hemos decidido

centrarnos en la situación real (todas las instancias negativas) para ver la contribución de cada grupo de características en un escenario real.

Grupo de características	Precisión	Recall	Medida-F
Léxica y morfosintáctica	0,113	0,175	0,138
Contexto y ventanas	0,143	0,773	0,241
Dependencias	0,146	0,319	0,201
Conjunto total	0,141	0,795	0,24

Tabla 27: Comparación entre grupos de características para training-test



Tal y como ocurrió con el método *cross-validation*, el análisis de los distintos grupos de características por separado no obtiene mejores resultados que el análisis del conjunto total tanto para el conjunto de características léxicas y morfosintácticas como para el conjunto de dependencias sintácticas. En cambio en este caso podemos observar unos resultados muy similares entre el conjunto de características de contexto y ventana de fármaco y el del total de las características. En este caso el grupo que consigue un mejor modelado de los datos es igualmente el del contexto y ventanas de cada fármaco.

Tanto estos últimos resultados como los obtenidos mediante cross-validation nos indican que el grupo de características con información más relevante para los algoritmos es el grupo de características de contexto y ventanas de cada fármaco. Estos resultados concuerdan tanto con los obtenidos en el trabajo realizado por Giuliano y Lavelli en (51) como en (71) por Bunescu y Mooney.

4 DISCUSIÓN DE LOS RESULTADOS

En este apartado vamos a proceder a comparar nuestros resultados obtenidos en el análisis del total de instancias mediante los algoritmos *ClassificationViaClustering* y *Naive Bayes*, dado que son los mejores resultados obtenidos en el entorno más parecido a la realidad, con los obtenidos tanto mediante el método kernel aplicado en (1) como en los trabajos desarrollados en (13), (14) y con los mejores resultados de los descritos en el apartado 2.7.1. En primer lugar compararemos nuestros resultados obtenidos (*ClassificationViaClustering* y *Naive Bayes*) con el método *training-test* con los obtenidos con el método kernel desarrollado en (1), evaluados también sobre el conjunto test.

Algoritmo	Precisión	Recall	Medida-F
Naive Bayes	0,141	**0,795**	0.24
ClassificationViaClustering	0,226	0,293	0,255
Método kernel aplicado en (1)	**0,50**	0,71	**0,58**

Tabla 28: Comparación resultados Naive Bayes y ClassificationViaClustering con método kernel aplicado en (1)

Observando los resultados anteriores, el método kernel da mejor resultado que los obtenidos en este proyecto con la evaluación *training-test*. Las diferencias más notables se dan tanto en la precisión como en la medida-F, obteniendo el método kernel aplicado en (1) valores superiores al doble nuestros mejores resultados. Sobre el recall cabe mencionar que el algoritmo *Naive Bayes* (79,5%) obtiene un ligero mejor resultado que el método kernel (71%).

En la siguiente tabla podemos ver la comparación entre nuestros resultados y los obtenidos en (1) pero evaluando ambos experimentos con *cross-validation*:

Algoritmo	Precisión	Recall	Medida-F
Naive Bayes	0,308	**0,933**	0,463
ClassificationViaClustering	0,162	0,348	0,221
IBk	***0,594***	*0,498*	***0,542***
Prism	***0,716***	*0,432*	*0,539*
Método kernel aplicado en (1)	0,57	0,800	**0,640**

Tabla 29: Comparación resultados obtenidos (*cross-validation*) - método kernel aplicado en (1) (*cross validation*)

Al igual que en la evaluación *training-test*, los resultados obtenidos en (1) son mejores que los obtenidos con los algoritmos Naive Bayes y ClassificationViaClustering, aunque Naive Bayes continúa obteniendo mejor recall que el método kernel. Ya que estos dos algoritmos no han sido los únicos con los que hemos analizado los datos, observamos que utilizando *cross-validation* los algoritmos IBk y Prism obtienen resultados que merecen ser comentados. El primero de ellos obtiene una mejor precisión (59,4%) que el método kernel (57%) aunque disminuye ligeramente la medida-F (54,42) y el recall se reduce casi a la mitad (49,8%). En el caso del algoritmo Prism cabe destacar una mejora en los resultados de precisión (aumentando hasta el 71,6%) con recall y medida-F parecidas a las del algoritmo IBk. La medida-F del método kernel se diferencia del resto en un 10% en el mejor de los casos. Observando los resultados podemos concluir que el método kernel aplicado en (1) es capaz de detectar en mejor medida si dos fármacos interaccionan entre sí.

En la siguiente tabla comparando nuestros mejores resultados tanto con los obtenidos mediante el algoritmo SVM aplicado en (13), como con los conseguidos en (14) y los mejores resultados obtenidos en el workshop DDIExtraction 2011 (tanto con el mejor método kernel como con el mejor método basado en características).

Algoritmo	Precisión	Recall	Medida-F
Naive Bayes	0,141	**0,795**	0.24
ClassificationViaClustering	0,226	0,293	0,255
SVM aplicado en (13)	0,447	0,127	0,198
GainRatio - Ibk aplicado en (14)	**0,678**	0,342	0,455
APG/Moara/SL aplicado en (56)	0,605	0,719	**0,657**
RLS aplicado en (41)	0,5804	0,6887	0,6299

Tabla 30: Comparación de nuestros resultados con los estudiados en el estado del arte

Como se puede apreciar a simple vista, el método kernel aplicado en (56) obtiene los mejores resultados con una medida-F del 65,7%, seguido del sistema desarrollado en (41) con un 62,99%. Además también se observa que nuestros mejores resultados (*Naive Bayes* y *ClassificationViaClustering*) obtienen resultados muy bajos, estando por debajo de ellos tan solo los obtenidos a través del método SVM aplicado en el sistema DRUIDA (13). Es en la medida recall en la única en la que uno de nuestros algoritmos obtiene mejor resultado (79,5%) que el resto (seguido por un 71,9%). En la siguiente ilustración tenemos una

representación visual de los resultados anteriores junto con los obtenidos con el método kernel aplicado en (1).

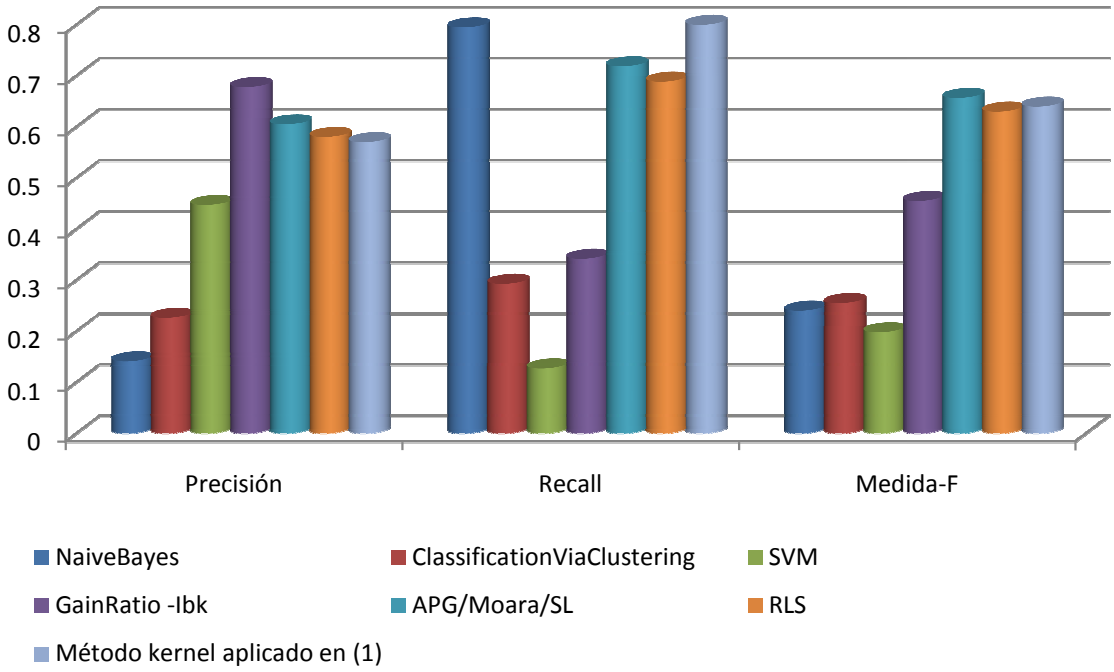


Tabla 31: Comparación gráfica de resultados

5 CONCLUSIONES Y LÍNEAS DE TRABAJO FUTURO

5.1 OBJETIVOS LOGRADOS

Revisando los objetivos listados en el apartado 1.2 de la memoria vemos que hemos conseguido cumplir todos ellos. En primero lugar se ha realizado un estudio previo sobre los sistemas desarrollados en (1), (13) y (14) ya que este proyecto se realiza a partir de dichos proyectos.

También se ha estudiado en profundidad el comportamiento del analizador Stanford (2) para el procesamiento de los textos y la extracción de información sintáctica y semántica, que junto con la ayuda de las tecnologías Castor y XSD, para el análisis del corpus DrugDDI, nos ha servido para componer un nuevo formato similar a los formatos CoNLL estudiados (en concreto similar al del año 2006 (35)).

En cuanto a la extracción de características hemos escogido aquellas que hemos considerado de mayor importancia basándonos en las características utilizadas en (14) y añadiendo algunas nuevas que se adaptan a la información contenida en nuestro nuevo formato. El valor de dichas características ha sido almacenado en primer lugar en formato CSV (por su simplicidad) para ser posteriormente transformado a formato arff (propio de Weka).

A continuación se han probado diferentes algoritmos de aprendizaje automático gracias a la herramienta Weka para saber cuáles eran los que mejores resultados generaban en nuestros experimentos. Los elegidos para las comparaciones han sido *ClassificationViaClustering*, *IBk* (con K = 1, 2 y 3), *NaiveBayes* y *VFI*.

Posteriormente se han comparado los resultados obtenidos para los algoritmos mencionados anteriormente con los obtenidos mediante el método kernel desarrollado en (1) y los obtenidos en los trabajos desarrollados en (13) y (14). También se han estudiado los trabajos presentados en el encuentro DDIExtraction 2011 y comparado alguno de los resultados con los obtenidos en este proyecto.

Tal como se puede observar en las comparaciones realizadas sobre los distintos trabajos en el apartado 4, los resultados del método kernel aplicado en (1) son mejores que los obtenidos con los métodos basados en features y utilizando los clásicos algoritmos de aprendizaje automático supervisado.

5.2 TRABAJOS FUTUROS

Como línea de trabajo futuro por la que seguir investigando para mejorar los resultados obtenidos en este proyecto, se podría empezar por profundizar en el estudio de los algoritmos de clasificación. Durante este proyecto se han realizado experimentos con distintos clasificadores y se han mostrados los mejores resultados obtenidos. Investigando profundamente sobre el funcionamiento de los algoritmos y los tipos de características que son realmente eficaces, se podrían mejorar posiblemente los resultados.

Se podrían analizar más detalladamente los resultados obtenidos. Las curvas de precisión-recall o ROC nos podrían ayudar a entender mejor los resultados. En las curvas ROC hay que tener en cuenta el ratio de falsos positivos (porcentaje de ejemplos negativos clasificados erróneamente como positivos) y el ratio de verdaderos positivos (porcentaje de ejemplos positivos etiquetados correctamente). Las curvas ROC representan el rendimiento de un clasificador sin tener en cuenta la distribución de las clases ni de los costes de errores, realizándose la representación de manera intuitiva y robusta. En las curvas precisión-recall (alternativa a la curva ROC cuando hay una desviación grande en la distribución de clases) se representa en el eje horizontal el recall, y la precisión en el eje vertical. Al tener unos datos muy desbalanceados, estas curvas podrían suponer una mejor representación de los resultados.

Otra posible línea de trabajo podría ser la aplicación de pruebas de significación estadística de McNemar que permiten comparar diferentes clasificadores determinando estadísticamente qué algoritmo es mejor.

La detección de información adicional de las interacciones (tales como certeza, dosis de los fármacos, particularidades de los pacientes, etc.) podría ser una importante línea de trabajo futuro. Y aunque nos hemos centrado en las interacciones farmacológicas como eventos peligrosos, no todas lo son, haciendo interesante la detección de interacciones entre fármacos cuyos resultados sea la potenciación del tratamiento.

Finalmente, debido a los no tan buenos resultados obtenidos en este proyecto, se podría realizar un análisis de errores para averiguar qué partes de nuestro sistema se podrían mejorar.

6 ANEXO

6.1 PRESUPUESTO

6.1.1 DESCRIPCIÓN DEL PROYECTO

Autor: Mª del Mar Castro Peláez

Departamento: Informática

Título: Estudio de la explotación de información sintáctica para la extracción de interacciones farmacológicas en textos biomédicos

Duración: inicio 15 octubre de 2010 y finalización el 30 septiembre de 2011. En total se han invertido 415 horas en la realización del presente proyecto.

6.1.2 CÁLCULO DE COSTES

Costes de Personal

Proyecto realizado por una persona, la cual se ha encargado del desarrollo de todas las funcionalidades llevadas a cabo en el proyecto. En la siguiente tabla se muestra el desglose de personal:

Fase	Coste/hora	Total horas	Total Coste
Análisis	€ 25,00	45	€ 1125,00
Diseño	€ 35,00	35	€ 1225,00
Codificación	€ 45,00	120	€ 5400,00
Pruebas	€ 30,00	90	€ 2700,00
Experimentación	€ 30,00	55	€ 1650,00
Documentación	€ 45,00	70	€ 3150,00
TOTAL		**415**	**€ 15250,00**

Tabla 32: Costes de personal

Costes de Equipos

Para la realización de este proyecto se ha necesitado un equipo durante la duración total del mismo. Además se ha utilizado otro equipo para la realización de experimentos, disponible durante un total de 3 meses.

Concepto	Unidades	Total Coste
Equipo general	1	€ 400,00
Equipo experimentación	1	€ 100,00
TOTAL		**€ 500,00**

Tabla 33: Costes de Equipos

Costes de Software

El sistema operativo del equipo generas viene incluido en la tabla anterior, pues al estar incluido en el equipo sus costes se contabilizan en los costes de equipo. El sistema operativo del equipo de experimentación es una distribución Linux, que al ser gratuita no conlleva ningún coste. El software utilizado en las pruebas y experimentación es Weka, que al ser una herramienta de código abierto tampoco conlleva ningún coste. Para el desarrollo de la documentación se ha utilizado Microsoft Office 2007, coste que está incluido dentro de los costes del equipo general al venir ya instalado.

Concepto	Unidades	Total Coste
S.O. Equipo experimentación	1	€ 0,00
Weka	1	€ 0,00
TOTAL		**€ 0,00**

Tabla 34: Costes de software

Costes de Material Fungible

Los costes asociados al material fungible de papel, cartuchos de tinta de impresora y otros gastos no contemplados en los conceptos anteriores son:

Concepto	Total coste
Papel	€ 26,00
Tinta impresora	€ 16,00
Otros gastos	€ 52,00
TOTAL	**€ 94,00**

Tabla 35: Costes de material fungible

6.1.3 PRESUPUESTO

A continuación, se muestra el coste total del presupuesto junto con una lista de todos los conceptos definidos anteriormente:

Concepto	Total coste
Coste de personal	€ 15250,00
Coste de equipos	€ 500,00
Costes de software	€ 0,00
Costes de material fungible	€ 94,00
TOTAL	**€ 15844,00**

Tabla 36: Presupuesto total

Por tanto el coste total del presupuesto asciende a la cifra de **QUINCE MIL OCHOCIENTOS CUARENTA Y CUATRO EUROS** (15.844,00 €), sin incluir el Impuesto sobre el Valor Añadido (I.V.A.).

REFERENCIAS

1. *Application of Information Extraction techniques to pharmacological domain: Extracting drug-drug interactions.* **Segura Bedmar, Isabel.** Universidad Carlos III de Madrid : s.n., 2010.

2. The Stanford Natural Language Processing Group. *The Stanford Parser: A statical parser.* [En línea] http://nlp.stanford.edu/software/lex-parser.shtml.

3. **LT Khon, JM Corrigan and MS Donaldson.** *To err is human: building a safer health system.* Institute of Medicine, Nationa Academy of Sciences. Washington DC : s.n., 1999.

4. **JU Rosholm, L. Bjerrum, J. Hallas, J. Worm and LF Gram.** *Polypharmacy and the risk of drug-drug interactions among Danish elderly. A prescription database study.* 1998. Danish medical bulletin.

5. **WHO.** *The importance of pharmacovigilance: Safety monitoring of medical products.* World Health Organization. 2002.

6. **plus, BOT.** Base de datos del conocimiento sanitario. 2008.

7. IH Stockley. *Stockleys Drug Interaction. Pharmaceutical Press.* 2007.

8. **Tatro, D.S.** Drug interaction facts. Facts and Comparisons. ST. Louis : s.n., 2003.

9. *Extraction of gene-disease relations fromd Medline using domain dictionaries and machine learning.* **H.W. Chun, Y.Tsuruoka, J. D. Kim, R. Shiba, N. Nagata, T. Hishiki and J. Tsujii.** 2006. Pacific Symposium on Biocomputing. Vol. 11, págs. 4-15.

10. *Classifying semantic relations in bioscience texts.* **Hearst, B. Rosario and M.** Barcelona : s.n., 2004. Proceedings of the 42nd Annual Meeting of the Association for Computacional Linguistics.

11. *A statical approach to scanning the biomedical literature for pharmacogenetics knowledge.* **D. L. Rubin, C. F. Thorn, T. E. Klein and R. B. Atman.** 2005, Journal of the American Medical Informatics Association.

12. *Information Extraction.* **Sarawagi, S.** 2007, Foundations and Trends in Databases, págs. 261-377.

13. *Un sistema para la extracción de interacciones farmacológicas basado en Support Vector Machines (SVM).* **Méndez González, Víctor Manuel.** Universidad Carlos III de Madrid : s.n., 2010.

14. *Aplicación de técnicas de aprendizaje automático para la extracción de información en textos farmacológicos.* **Nombela Escobar, Beatriz.** Universidad Carlos III de Madrid : s.n., 2011.

15. Castor. [En línea] http://www.castor.org/.

16. W3C XML Schema. [En línea] http://www.w3.org/XML/Schema.

17. **Mitchell, T.** Machine Learning. s.l. : McGraw Hill, 1997.

18. Weka 3 - Data Mining with open source machine learning software in Java. [En línea] http://www.cs.waikato.ac.nz/ml/weka/.

19. **Witten, I. y Frank, E.** *Data Mining: Practical machine learning tools and techniques.* s.l. : Morgan Kaufmann Pub, 2005.

20. **Witten, I., et al.** *Weka machine: Practical machine learning tools and techniques with Java implementations.* Hamilton, New Zealand: University of Wakaito, Department of Computer Science : s.n., 1999.

21. **Sararwagi, S.** Information Extraction. Foundations and Trends in Databases. 2008, Vol. 1, págs. 261-377.

22. **Klein, Dan y Manning, Christopher.** *Fast Exact Inference with a Factored Model for Natural Language Parsing.* 2003.

23. *Accurate Unlexicalized Parsing .* 2003.

24. **Manning, Marie-Catherine de Marneffe and Christopher.** *Stanford Dependencies manual.* 2008.

25. CoNLL: the conference of SIGNLL. [En línea] http://ifarm.nl/signll/conll/.

26. Proceedings of CoNLL-97. [En línea] http://www.clips.ua.ac.be/conll97/proceedings.html.

27. Proceedings of CoNLL-98. [En línea] http://www.clips.ua.ac.be/conll98/proceedings.html.

28. Conference on Computational Natural Language Learning (CoNLL-99). [En línea] http://www.clips.ua.ac.be/conll99/.

29. Conference on Computational Natural Language Learning (CoNLL-2000). [En línea] http://www.clips.ua.ac.be/conll2000/.

30. Conference on Computational Natural Language Learning (CoNLL-2001). [En línea] http://www.clips.ua.ac.be/conll2001/.

31. Conference on Computational Natural Language Learning (CoNLL-2002). [En línea] http://www.clips.ua.ac.be/conll2002/.

32. Conference on Computational Natural Language Learning (CoNLL-2003). [En línea] http://www.clips.ua.ac.be/conll2003/.

33. Conference on Computational Natural Language Learning (CoNLL-2004). [En línea] http://www.clips.ua.ac.be/conll2004/.

34. CoNLL-2005. [En línea] http://www.clips.ua.ac.be/conll2005/.

35. CoNLL-X. [En línea] http://www.clips.ua.ac.be/conll2006/.

36. EMNLP-CoNLL 2007. [En línea] http://www.cs.jhu.edu/EMNLP-CoNLL-2007/.

37. CoNLL-2008. [En línea] http://www.clips.ua.ac.be/conll2008/.

38. CoNLL-2009. [En línea] http://www.clips.ua.ac.be/conll2009/.

39. CoNLL-2010. [En línea] http://www.clips.ua.ac.be/conll2010/.

40. CoNLL-2011. [En línea] http://www.clips.ua.ac.be/conll2011/.

41. *Drug-drug Interaction Extraction from Biomedical Texts with SVM and RLS classifiers.* **Jari Björne, Antti Airol, Tapio Pahikkala and Tapio Salakoski.** DDI Exctraction 2001.

42. MetaMap Portal. [En línea] http://metamap.nlm.nih.gov/.

43. *Feature Selection for drug-drig interaction detection using machine learning based approaches.* **Anne-Lyse Minard, Lamia Makour,Anne-Laure Ligozat and Brigitte Grau.** DDI Extraction 2011.

44. *Automatic drug-drug interaction detection: a machine learning approach with maximal frequent sequence extraction.* **Sandra Garcia-**

Blasco, Santiago M. Mola-Velasco, Roxana Danger, and Paolo Rosso. DDI Extraction 2011.

45. Random forest. [En línea] http://stat-www.berkeley.edu/users/breiman/RandomForests/cc_home.htm.

46. *A machine learning approach to extract drug-drug interactions in an unbalanced dataset.* **Jacinto Mata, Ramón Santano, Daniel Blanco, Marcos Lucero, Manuel J. Maña.** DDI Extraction 2011.

47. Chi Square Feature Selection. [En línea] http://nlp.stanford.edu/IR-book/html/htmledition/chi-square-feature-selection-1.html.

48. *Drug-drug interactions discovery based on CRFs, SVMs and Rule-Based Methods.* **Stefania Rubrichi, Matteo Gabetta, Riccardo Bellazzi, Cristiana Larizza and Silvana Quaglini.** DDI Extraction 2011.

49. *An experimental exploration of drug-drug interaction extraction from biomedical texts.* **Man Lan, Jiang Zhao, Kezun Zhang, Honglei Shi and Jingli Cai.** DDI Extraction 2011.

50. *Drug-drug Interaction Extraction using composite Kernels.* **Lavelli, Md. Faisal Mahbub Chowdhury and Alberto.** DDI Extraction 2011.

51. *Exploiting shallow linguistic information for relation extraction from biomedical literature.* **Giuliano, C., Lavelli, A., Romano, L.** Trento, Italia : s.n., 2006. Proceedings of the 11th Conference of European Chapter of the Association for Computacional Linguistics (EACL'2006). págs. 401-408.

52. *A study on dependency tree kernels for automatic extraction of protein-protein interaction.* **Chowdhury, M.F.M., Lavelli, A., Moschitti, A.** Potland, Oregón, USA. : s.n., 2011. Proceedings of BioNLP 2011 Workshop. págs. 124-133.

53. *A study on convolucion kernels for shallow semantic parsing.* **Moschitti, A.** Barcelona : s.n., 2004. Proceedings of the 42nd Annual Meeting of the Association for Computacional Linguistics /ACL'04).

54. *Extraction of drug-drug interactions using all paths graph kernels.* **Shreyas Kamik, Abhinita Subhadarshini, Zhiping Wang, Luis M. Rocha and Lang Li.** DDI Extraction 2011.

55. *All-paths graph kernel for protein-protein interaction extraction with evaluation of cross-corpus learning.* **Antti Airola, Sampo Pyysalo, Jari Björne, Tapio Pahikkala, Filip Ginter and Tapio Salakoski.** 2008. BMC Bioinformatics.

56. *Relation Extraction for drug-drug interactions using ensemble learning.* **Philippe Thomas, Mariana Neves, Ill'es Solt, Domonkos Tikk and Ulf Leser.** DDI Extraction 2011.

57. BioNLP'09 Shared Task on Event Extraction. [En línea] http://www-tsujii.is.s.u-tokyo.ac.jp/GENIA/SharedTask/.

58. *Two different machine learning techniques for drug-drug interaction extraction.* **Md. Faisal Mahbub Chowdhury, Asma Ben Abacha, Alberto Lavelli and Pierre Zweigenbaum.** DDI Extraction 2011.

59. Lenguaje de programación Java. [En línea] http://www.oracle.com/technetwork/java/index.html.

60. Extensible Markup Language (XML). [En línea] http://www.spanish-translator-services.com/espanol/t/infoset.htm.

61. Documento Object Model (DOM). *Wikipedia (The Free Encyclopedia).* [En línea] http://html.conclase.net/w3c/dom1-es/cover.html.

62. Log4j. [En línea] http://logging.apache.org/log4j/.

63. Comma-separated values (CSV). [En línea] http://www.creativyst.com/Doc/Articles/CSV/CSV01.htm.

64. Snowball. [En línea] http://snowball.tartarus.org/index.php.

65. *DrugBank: a knowledgebase for drugs, drugs actions and drug targets.* **Wishart, D., et al.** 2007. Nucleic acids research.

66. *The Unified Medical Language System.* **Lindberg, D., Humphreys, B. y McGray, A.** 1993, Methods of information in Medicine, Vol. 32, pág. 281.

67. *The unfied medical language system (UMLS): integrating biomedical terminology.* **Bodonreider, O.** Nucleid Acids Research. Vol. 32.

68. *Medical subject headings (MeSH).* **Lipscomb, C.** 2000, Bulletin of Medical Language System, Vol. 88, pág. 265.

69. *SNOMED RT: a reference terminology for health care.* **Spackman, K., Campbell, K y CÃ, R.** 1997. Proceedings of the AMIA Annual Fall Symposium. pág. 640.

70. [En línea] http://stp.ling.uu.se/~nivre/exp/PennToStanford.java.

71. *A shortest Path Dependency Kernel for Relation Extraction.* **Bunescu, Razvan y Mooney, Raymond J.** Vancouver : s.n., 2005. Proceedings of the Joint Conference on Human Language Technology / Empirical Methods in Natural Language Processing (HLT/EMNLP).

72. *Exploring syntactic features for relation extraction using a convolution tree kernel.* **M. Zhang, J. Zhang y J. Su.** 2006. Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computacional Linguistics. págs. 288-295.

73. CoNLL: The conference of SIGNLL. [En línea] http://ifarm.nl/signll/conll/.

74. *Exctracting interactions between proteins and from the literature.* **Zhou, D. y He, Y.** Journal of biomedical informatics, Vol. 41, págs. 393-407.