eman ta zabal zazu

ESCUELA UNIVERSITARIA DE INGENIERÍA TÉCNICA INDUSTRIAL DE BILBAO

GRADO EN INGENIERÍA INFORMÁTICA DE GESTIÓN Y SISTEMAS DE INFORMACIÓN

TRABAJO FIN DE GRADO

2013 / 2014

SISTEMA DE AYUDA AL DIAGNÓSTICO CLÍNICO: CLASIFICACIÓN DE DIAGNÓSTICOS CLÍNICOS

MEMORIA TFG

DATOS DE LAS ALUMNAS	DATOS DE LAS DIRECTORAS
NEREA AGUIRRE GARCÍA	ALICIA PÉREZ RAMÍREZ DEPARTAMENTO: LENGUAJES Y SISTEMAS INFORMÁTICOS
ESTIBALIZ AMILLANO SOLANO	ARANTZA CASILLAS RUBIO DEPARTAMENTO: ELECTRICIDAD Y ELECTRÓNICA
FDO.: FECHA:	FDO.: FECHA:

Nerea Aguirre y Estibaliz Amillano: *Sistema de Ayuda al Diagnóstico Clínico: Clasificación de Diagnósticos Clínicos,* 06/07/2013.

Trabajo Fin de Grado presentado dentro del Grado en Ingeniería Informática de Gestión y Sistemas de Información en la Escuela Universitaria en Ingeniería Técnica Industrial de Bilbao (UPV-EHU).

Este documento está basado en los estilos tipográficos "`classicthesis`" y "`arsclassica`", ambos disponibles para LaTeX vía CTAN. Estos estilos han sido diseñados siguiendo los criterios indicados en *The elements of typographic style* por el tipógrafo Robert Bringhurst.

Para Pedro, Jorge,
nuestros amigos y nuestras familias.

RESUMEN

El trabajo realizado en este proyecto se enmarca dentro del área de Procesamiento del Lenguaje Natural aplicado al ámbito de la medicina. Para este fin se han utilizado técnicas de minería de datos y aprendizaje automático. El objetivo principal de este proyecto es el desarrollo de un modelo capaz de automatizar la clasificación de textos clínicos según el estándar ICD-9-CM (codificación estándar utilizada por la red hospitalaria europea). Aunque existe una herramienta web (https://eciemaps.mspsi.es/ecieMaps/browser/index_9_mc.html) que facilita la clasificación, este trabajo, hoy en día es realizado manualmente. Básicamente se trata de un diccionario *on-line* de los términos del estándar.

Basándonos en trabajos previos relacionados, se ha obtenido un *baseline* a partir del cual se ha construido el proyecto. En primer lugar, como en cualquier trabajo relacionado con los Sistemas de Apoyo a la Decisión (DSS) se ha estructurado el trabajo en dos módulos principales: el preproceso y la clasificación. En el módulo dedicado al preproceso, se tratan los datos para hacerlos comprensibles a los algoritmos de clasificación. En este primer módulo también se realiza una fase de adición de atributos que aporten información útil a la hora de la clasificación y una posterior selección de los mismos, por si alguno fuera redundante o irrelevante. En el segundo módulo dedicado a la clasificación, seleccionamos aquellos algoritmos que consideramos mejores, basándonos para ello, en otros trabajos previos que abordan un problema similar. Una vez seleccionados los algoritmos, se procede a realizar barridos de parámetros que optimicen su rendimiento.

Finalmente, se ha realizado la experimentación con distintas técnicas de preprocesamiento de los datos y con los distintos algoritmos de clasificación automática. Esta última de experimentación tiene como objetivo, encontrar la combinación de métodos que optimice el rendimiento de ambos módulos, y por tanto de todo el sistema.

ÍNDICE GENERAL

I Capítulos 1

1 INTRODUCCIÓN 3
- 1.1 Estructura del documento 3
- 1.2 Problema abordado 3
- 1.3 Motivación 4
- 1.4 Propósito 4
- 1.5 Contribución 5
- 1.6 Ámbito 5
- 1.7 Marco Teórico: Introducción a la minería de datos 5
  - 1.7.1 Preproceso 6
  - 1.7.2 Clasificación 7

2 PLANTEAMIENTO INICIAL 9
- 2.1 Objetivos principales 9
- 2.2 Arquitectura 9
- 2.3 Alcance 9
- 2.4 Planificación temporal 12
- 2.5 Herramientas 16
- 2.6 Gestión de riesgos 16
- 2.7 Evaluación económica 18

3 ANÁLISIS DE ANTECEDENTES 21
- 3.1 MD para el precesamiento del lenguaje natural 21
- 3.2 Estado del arte 22
- 3.3 Discusión de antecedentes en relación a nuestro trabajo 24

4 CAPTURA DE REQUISITOS 27
- 4.1 Casos de uso 27
- 4.2 Modelo de dominio 33

5 ANÁLISIS Y DISEÑO 35
- 5.1 Diagrama de clases 35
- 5.2 Diagramas de secuencia 37

6 DESARROLLO: DATOS 47
- 6.1 ICD-9-CM 47
- 6.2 Descripción de las muestras 48
- 6.3 Estructura 49
- 6.4 Volumen del corpus 50

7 DESARROLLO: PREPROCESO 53
- 7.1 Marco teórico 53
  - 7.1.1 Preparación de los datos 53
  - 7.1.2 Normalización del texto 55
  - 7.1.3 Representación de los datos 56
  - 7.1.4 Nuevos atributos 59
  - 7.1.5 Reducción dimensional 64

7.1.6 Volumen del corpus preprocesado 76
7.2 Marco experimental 77
7.2.1 Normalización 77
7.2.2 Representación del texto 77
7.2.3 Nuevos atributos 78
7.2.4 Reducción dimensional 79
7.2.5 Discusión de resultados 83

8 DESARROLLO: CLASIFICACIÓN 85
8.1 Marco teórico 85
8.1.1 *Naive Bayes* 85
8.1.2 Decision Trees - Árboles de decisión 86
8.1.3 Random Forest - Bosques aleatorios 88
8.1.4 Support Vector Machine o Máquina de Soporte Vectorial 89
8.2 Marco experimental 90
8.2.1 Desarrollo del modelo de clasificación 90
8.2.2 Elección de las formas de evaluación del modelo 90
8.2.3 Elección de los conjuntos de datos 90
8.2.4 Resultados de los barridos de parámetros 91
8.2.5 Resultados de la evaluación por resustitución 92
8.2.6 Resultados de la evaluación honesta 93
8.2.7 Resultados de la evaluación por *k-foldCrossValidation* 94
8.2.8 Conclusiones sobre los clasificadores 94

9 SOFTWARE 95
9.1 Uso del software 95

10 CONCLUSIONES Y TRABAJO FUTURO 101
10.1 Consecución de objetivos y análisis de resultados 101
10.2 Conclusiones y contribuciones científicas 102
10.3 Comentarios e impresiones finales respecto de la gestión 102
10.4 Trabajo futuro 104

II Apéndices 107

A TABLAS DE RESULTADOS 109
A.1 Experimentos Preproceso 109
A.1.1 Normalización 109
A.1.2 Representación del texto 110
A.1.3 Nuevos atributos 111
A.1.4 Reducción dimensional 112
A.2 Experimentos Clasificación 115
A.2.1 Resultados de los conjuntos de datos, representados de forma de *BagOfWords* 116
A.2.2 Resultados de los conjuntos de datos, representados de forma nominal 118

B GLOSARIO 119

C LISTA DE ACRÓNIMOS 121

D FIGURAS 123

BIBLIOGRAFÍA 126

ÍNDICE DE FIGURAS

Figura 1 Estructura de un conjunto de datos adecuado. 6
Figura 2 Matriz de confusión. 7
Figura 3 Representación de las diferentes fases. 9
Figura 4 Estructura de Descomposición del Trabajo. 10
Figura 5 Planificación temporal. 15
Figura 6 Casos de uso. 27
Figura 7 Modelo de dominio. 33
Figura 8 Diagrama de clases. 36
Figura 9 Diagrama de secuencia Preparar Datos. 38
Figura 10 Diagrama de secuencia Añadir Features. 39
Figura 11 Diagrama de secuencia Modo Representación. 40
Figura 12 Diagrama de secuencia Seleccionar Features. 41
Figura 13 Diagrama de secuencia Clasificación. 42
Figura 14 Extracto de la clasificación ICD-9-CM 47
Figura 15 Extracto de los documentos de partida. 48
Figura 16 Extracto del estándar ICD-9-CM. 48
Figura 17 Declaración de atributos. 49
Figura 18 Representación de una instancia. 50
Figura 19 Estructura de un documento `.arff`. 50
Figura 20 Estructura original de los datos. 53
Figura 21 Cabecera de un fichero `.arff`. 54
Figura 22 Cuerpo de un fichero `.arff`. 54
Figura 23 Cabecera del fichero1.arff. 54
Figura 24 Cabecera del fichero2.arff. 55
Figura 25 Unificación de cabeceras. 55
Figura 26 Normalización de los datos. 55
Figura 27 Atributos de una instancia. 56
Figura 28 Atributos en formato original. 58
Figura 29 Atributos tras aplicar BOW. 58
Figura 30 Diferencia entre los tipos *String* y nominal. 58
Figura 31 Atributo término ICD-9-CM. 59
Figura 32 Lista de sinónimos. 59
Figura 33 Atributo sinónimos. 60
Figura 34 Atributo distancia de Levenshtein. 60
Figura 35 Atributo *N-gramms* compartidos. 61
Figura 36 Rango de variación de *N*. 61
Figura 37 División en *Unigramms*. 61
Figura 38 División en *Bigramms*. 61
Figura 39 División en *Trigramms*. 62
Figura 40 División en 4*gramms*. 62
Figura 41 Rango de variación de *N*. 62
Figura 42 División en *Unigramms*. 62
Figura 43 División en *Bigramms*. 62
Figura 44 División en *Trigramms*. 63
Figura 45 División en 4*gramms*. 63
Figura 46 Atributo palabras compartidas. 63
Figura 47 Atributo palabras compartidas en posición. 64
Figura 48 *Forward Selection*. 67

Figura 49 *Backward Selection*. 68
Figura 50 Figura obtenida de [Ian H. Witten(2011)]. 68
Figura 51 Árbol generado por el algoritmo *Decission Tree*. 69
Figura 52 Diferencia entre la Selección de atributos y la Extracción de atributos. 70
Figura 53 Comparación entre métodos de Selección de atributos. 73
Figura 54 Figura obtenida de [Alpaydin(2010)]. 74
Figura 55 Figura obtenida de [Mitchel(1997)] éste es un árbol de decisión para saber si jugar o no al Tennis. Este árbol clasifica los sábados por la mañana según sean o no apropiados para jugar un partido de Tennis. 87
Figura 56 Figura que representa un árbol similar a los generados por Weka [Ian H. Witten(2011)] a partir de los datos utilizados para este proyecto. 88
Figura 57 Figura obtenida de [Alpaydin(2010)] Vemos una función que marca el margen de una de las dos clases presentes en el problema, puntos o pluses. Las instancias redondeadas son los soportes vectoriales. 89
Figura 58 Esquema del software. 95
Figura 59 PreparарDatos.jar. 96
Figura 60 AñadirFeatures.jar. 96
Figura 61 ModoRepresentacion.jar. 97
Figura 62 SeleccionarFeatures.jar. 98
Figura 63 Clasificacion.jar. 99
Figura 64 Comparación entre el tiempo estimado y el tiempo real invertido por Estibaliz Amillano. 103
Figura 65 Comparación entre el tiempo estimado y el tiempo real invertido por Nerea Aguirre. 104
Figura 66 Diagrama que ilustra el funcionamiento del trabajo futuro propuesto. 105

ÍNDICE DE CUADROS

Tabla 1 Comparativa entre algunos diagnósticos escritos por el médico y su correspondencia con el estándar ICD-9-CM. 4
Tabla 2 Caso de uso PrepararDatos. 28
Tabla 3 Caso de uso AñadirFeatures. 29
Tabla 4 Caso de uso ModoRepresentación. 30
Tabla 5 Caso de uso SeleccionarFeatures. 31
Tabla 6 Caso de uso Clasificación. 32
Tabla 7 Datos del corpus disponible. 51
Tabla 8 Datos del corpus disponible. 76
Tabla 9 Tabla de resultados - Sin normalización. 77
Tabla 10 Tabla de resultados - Con normalización. 77
Tabla 11 Tabla de resultados - Representación BOW 78
Tabla 12 Tabla de resultados - Representación nominal. 78
Tabla 13 Tabla de resultados - Sin atributos añadidos. 78
Tabla 14 Tabla de resultados - Con atributos añadidos. 79
Tabla 15 Tabla de resultados - Con atributo sinónimos añadido. 79
Tabla 16 Tabla de resultados - Sin ninguna técnica de reducción dimensional. 79
Tabla 17 Tabla de resultados - InfoGain.Ranking. 80
Tabla 18 Tabla de resultados - TF-IDF. 80
Tabla 19 Tabla de resultados - Cfs.Best first search. 80
Tabla 20 Tabla de resultados - Classifier.Best first search. 80
Tabla 21 Tabla de resultados - Wrapper.Best first search. 80
Tabla 22 Tabla de resultados - Remove. 81
Tabla 23 Tabla de resultados - Sin ninguna técnica de reducción dimensional. 82
Tabla 24 Tabla de resultados - InfoGain.Ranking. 82
Tabla 25 Tabla de resultados - TF-IDF. 82
Tabla 26 Tabla de resultados - Cfs.Best first search. 82
Tabla 27 Tabla de resultados - Classifier.Best first search. 82
Tabla 28 Tabla de resultados - Remove. 82
Tabla 29 Representación en forma de *StringToWordVector* sin nuevos atributos. Evaluación por *Holdout*. 91
Tabla 30 Representación en forma de *StringToWordVector* con sinónimos. Evaluación por *Holdout*. 91
Tabla 31 Representación en forma nominal, datos normalizados, sin nuevos atributos. 92
Tabla 32 Representación en forma nominal, datos normalizados, incluyendo el nuevo atributo. 92
Tabla 33 Representación en forma nominal, datos normalizados, sin nuevos atributos. 93
Tabla 34 Representación en forma nominal, datos normalizados, incluyendo el nuevo atributo. 93
Tabla 35 Representación en forma nominal, datos normalizados, sin nuevos atributos. Evaluación por *10-foldCrossValidation*. 94

Tabla 36 Representación en forma nominal, datos normalizados, con el nuevo atributo. Evaluación por *10-foldCrossValidation*. 94
Tabla 37 Tabla de resultados Completa - Sin normalización 109
Tabla 38 Tabla de resultados Completa - Con normalización 109
Tabla 39 Tabla de resultados no honestos completa - Sin normalización 110
Tabla 40 Tabla de resultados no honestos completa - Con normalización 110
Tabla 41 Tabla de resultados completa - Representación BOW 110
Tabla 42 Tabla de resultados completa - Representación nominal 110
Tabla 43 Tabla de resultados no honestos completa - Representación BOW 110
Tabla 44 Tabla de resultados no honestos completa - Representación nominal 110
Tabla 45 Tabla de resultados completa - Sin atributos añadidos 111
Tabla 46 Tabla de resultados completa - Con atributos añadidos 111
Tabla 47 Tabla de resultados completa - Con atributo sinónimos añadido 111
Tabla 48 Tabla de resultados no honestos completa - Sin atributos añadidos 111
Tabla 49 Tabla de resultados no honestos completa - Con atributos añadidos 111
Tabla 50 Tabla de resultados no honestos completa - Con atributo sinónimos añadido 112
Tabla 51 Tabla de resultados completa - Sin ninguna técnica de reducción dimensional 112
Tabla 52 Tabla de resultados completa - InfoGain.Ranking 112
Tabla 53 Tabla de resultados completa - TF-IDF 112
Tabla 54 Tabla de resultados completa - Cfs.Best first search 113
Tabla 55 Tabla de resultados completa - Classifier.Best first search 113
Tabla 56 Tabla de resultados completa - Wrapper.Best first search 113
Tabla 57 Tabla de resultados completa - Remove 113
Tabla 58 Tabla de resultados completa - Sin ninguna técnica de reducción dimensional 113
Tabla 59 Tabla de resultados completa - InfoGain.Ranking 114
Tabla 60 Tabla de resultados completa - TF-IDF 114
Tabla 61 Tabla de resultados completa - Cfs.Best first search 114
Tabla 62 Tabla de resultados completa - Classifier.Best first search 114
Tabla 63 Tabla de resultados completa - Remove 114

Parte I

Capítulos

1 INTRODUCCIÓN

En este capítulo se realiza una introducción al Trabajo de Fin de Grado presentado. Se explicará la estructura del documento, el problema abordado, la motivación y el propósito que nos han llevado a desarrollarlo, así como la contribución realizada, el ámbito en el que se trabaja y una breve introducción a la Minería de Datos.

1.1 ESTRUCTURA DEL DOCUMENTO

Este documento está compuesto por 10 capítulos, siendo el primero de ellos una introducción general. En el segundo se recoge el planteamiento inicial de este proyecto, los objetivos propuestos, planificaciones, evaluaciones, etc. A continuación se expone el trabajo previo relacionado, el cual ha servido de base y guía. El cuarto y quinto capítulo tratan aspectos generales sobre el proyecto, como la captura de requisitos, el análisis y el diseño. Los tres siguientes capítulos contienen el desarrollo del proyecto, una presentación detallada de los datos utilizados y el trabajo realizado dividido en las fases de "Preproceso" y "Clasificación", tanto teóricamente como a nivel de experimentación. En el capítulo nueve, se explica el software diseñado para abordar esta tarea, y finalmente en el último capítulo, se exponen el análisis de los resultados y objetivos conseguidos, las conclusiones alcanzadas y el trabajo futuro. Además se incluyen como apéndices, un glosario que recopila ciertos términos técnicos utilizados a lo largo de este documento, un conjunto de tablas de resultados de experimentos realizados que, por espacio y claridad, no se han incluido en la memoria. También se incluyen en los apéndices, una lista de acrónimos y un enlace a una carpeta compartida, que contiene todas las figuras que aparecen en este documento. Finalmente se encuentra la bibliografía.

1.2 PROBLEMA ABORDADO

El problema abordado en este proyecto es la automatización del proceso de clasificación de diagnósticos clínicos, asignándo a cada término diagnóstico, el código correspondiente del estándar. En la tabla 1 se muestra un ejemplo del término escrito por el médico junto con el código que le corresponde y su forma estándar. El sistema de codificación o estándar con el que trabajaremos es "Clasificación Internacional de Enfermedades, 9ª Edición, Modificación Clínica" (CIE-9-MC). A partir de ahora, nos referiremos a él como ICD-9-CM, *International Statistical Classification of Diseases, Ninth Revision, Clinical Modification*, para más información ir al capítulo número 6, dedicado a los datos 6.

La razón por la cual se quiere automatizar la clasificacón de estos diagnósticos, es que están escritos en lenguaje natural (con faltas de ortografía, abreviaturas acrónimos etc), y actualmente para estandarizarlos es necesario realizar este proceso manualmente. En la tabla 1 podemos ver una compa-

CÓDIGO	DIAGNÓSTICO DEL MÉDICO	ESTÁNDAR
327.3	D/ SAOS	Trastorno del ritmo circadiano del sueño
782.0	Parestesias a estudio	Trastorno de la sensibilidad
728.6	Dupuytren	Contractura de fascia palmar
401.9	HTA	Hipertensión No especificada

Tabla 1: Comparativa entre algunos diagnósticos escritos por el médico y su correspondencia con el estándar ICD-9-CM.

rativa entre algunos diagnósticos escritos por el médico en lenguaje espontáneo y su correspondencia con el ICD-9-CM.

El hecho de que no sigan ningún estándar, es debido a que el médico escribe el diagnóstico del paciente en el momento de la consulta, por lo tanto cada médico lo escribe de una forma diferente y sin seguir ninguna codificación. Ésta es una de las dos dificultades principales a las que nos enfrentamos.

Por otro lado, tenemos que el caso más báscio de clasificación mediante aprendizaje automático, es aquél que afronta una disyunción binaria, es decir, que discrimina entre dos opciones o clases. Por ejemplo, un sistema que decide si un paciente está o no enfermo, asignaría a una situación o instancia descrita por ciertas características o atributos (edad, peso, síntomas etc), la clase "Si" o la clase "No". Sin embargo, en este caso queremos saber el código correspondiente al diagnóstico redactado por el médico. Que conozcamos, en el campo de la clasificación de diagnósticos clínicos, el mayor número de opciones o clases que se han abordado ha sido de 45 [Zhang & Patrick(2008)]. Por lo tanto pensamos que ésta es otra de las dificultades del proyecto ya que, se pretende conseguir una herramienta capaz de discriminar entre las más de 14000 clases posibles. Las 14000 clases entre las que debe decidir el sistema son las correspondientes al estándar.

Concluimos que las dos principales dificultades son: la forma en que los diagnósticos están redactados, utilizando el lenguaje espontáneo y personal de cada médico, y la gran cantidad de clases diferentes existentes para realizar la clasificación.

1.3 MOTIVACIÓN

Los diagnósticos clínicos redactados por médicos contienen mucha información de gran importancia. Sin embargo, no están redactados de una manera estándar. La red de hospitales que configuran el sistema español de sanidad utiliza ICD-9-CM (ver acrónimo en C) para codificar partes de alta hospitalaria. Hoy en día, este trabajo lo realizan a mano los médicos y teniendo en cuenta la dificultad de esta tarea ha surgido la idea de este proyecto, cuyo principal objetivo es automatizar la clasificación de diagnósticos clínicos en lenguaje natural siguiendo el estándar del ICD-9-CM.

1.4 PROPÓSITO

El propósito de este proyecto consiste en la experimentación con varios sistemas de aprendizaje automático para solventar el problema abordado y así lograr automatizar la tarea de la clasificación de diagnósticos clínicos. El principal objetivo es obtener un método efectivo para clasificar con la mayor

precisión posible, los diagnósticos clínicos redactados en lenguaje natural en códigos clínicos predefinidos (ICD-9-CM). Para ello, se divide el problema en dos bloques principales, que se optimizarán por separado. El primer bloque, que llamaremos "preproceso", se ocupará del tratamiento de los datos para representarlos de la forma que más información aporten. El segundo bloque, "clasificación", se ocupará de todo lo relacionado con la clasificación del diagnóstico. Ambos bloques se explican más en profundidad en los capítulos 7 y 8 respectivamente.

1.5 CONTRIBUCIÓN

La principal contribución de este proyecto se basa en la implementación de un método capaz de realizar esta tarea combinando los métodos de preproceso y clasificación más adecuados, con el objetivo de conseguir un alto rendimiento. Tal y como se expone en el capítulo sobre análisis de antecedentes 3, anteriormente se han llevado a cabo otros proyectos con objetivos similares. Sin embargo, han trabajado con un máximo de 45 clases diferentes, frente a las 14000 a las que nos enfrentamos en este caso. Además, hasta ahora, no se ha intentado nunca para diagnósticos clínicos en castellano.

1.6 ÁMBITO

Este proyecto se ha llevado a cabo utilizando el entorno de programación Eclipse. Se ha desarrollado utilizando el lenguaje Java e incluyendo librerías de Weka, las cuales se explican en la sección 2.5. Consideramos muy importante el acceso a las máquinas de la universidad vía `ssh` posibilitándonos lanzar los experimentos desde casa ya que nuestros portátiles no disponen de la capacidad de cálculo suficiente. Para la realización de este proyecto se establece una realción estrecha entre las integrantes del grupo debido al carácter grupal del mismo y por tanto, la necesidad de una coordinación minuciosa.

1.7 MARCO TEÓRICO: INTRODUCCIÓN A LA MINERÍA DE DATOS

En los próximos capítulos se alude a menudo a conceptos relacionados con la Minería de Datos. Por ello, en esta sección, se recoge un resumen de las bases de la Minería de Datos, el aprendizaje automático y los Sistemas de Apoyo a la Decisión. Comenzaremos por definir la Minería de Datos, tal y como se define en [Ian H. Witten(2011)].

"La Minería de Datos es la extracción de información potencialmente útil, desconocida hasta el momento, de un conjunto de datos."

Por otro lado, el Aprendizaje Automático nos provee de las bases técnicas de la Minería de Datos. La idea es construir programas informáticos capaces de encontrar patrones. Estos patrones, sirven para generalizar y conseguir predicciones acertadas a partir de datos futuros. El proceso se reduce a: tomar los conjuntos de datos tal y como están, e inferir un modelo. Este mode-

lo puede utilizarse para realizar predicciones, cuando los datos disponibles describan situaciones ocurridas en el pasado. Sin embargo, en este trabajo nos centraremos en las aplicaciones del Aprendizaje Automático en las que el resultado del "aprendizaje", es un modelo capaz de clasificar ejemplos. Estos ejemplos vienen descritos por una serie de características que a partir de ahora llamaremos **atributos**. Los atributos son la entrada del modelo de clasificación y la **clase** será la salida del modelo.

Un Sistema de Aprendizaje Automático se divide en dos módulos principales: el **preproceso** de los datos y la **clasificación** de los mismos. Además los datos no siempre están representados de la forma que más información aporta. Por lo tanto es necesario aplicar una serie de técnicas de preproceso de datos, con el fin de optimizar el rendimiento del sistema de aprendizaje.

1.7.1 Preproceso

Los datos no están estructurados tal y como exigen las herramientas de clasificación, en nuestro caso, la herramienta utilizada es Weka.
[Ian H. Witten(2011)]. Por lo tanto, es necesario tratar los datos para que se adapten a un formato compatible con las librerías de Weka. Es necesario que cada ejemplo o **instancia** a clasificar, esté descrito por una serie de características o **atributos** diferenciados. Uno de éstos atributos será lo que se denomina **clase**, que es lo que se quiere que el modelo aprenda a clasificar. Además el conjunto de **instancias**, deberá incluir una cabecera detallando los **atributos** que contiene cada **instancia**, así como el tipo de los mismos (numérico, nominal etc). En la figura 1 se puede observar un ejemplo de la estructura que debería tener el conjunto de datos a la entrada del sistema de aprendizaje.

```
@RELATION titulo

@ATRIBUTE edad int
@ATRIBUTE peso real
@ATRIBUTE estatura real
@ATRIBUTE class {femenino, masculino}

@DATA
27,57,1.70,femenino
25,78,1.85,masculino
...
```

Figura 1: Estructura de un conjunto de datos adecuado.

Es necesario elegir qué atributos recibirá el clasificador como entrada. Estos serán aquellos que sean mas representativos y marquen la diferencia entre las clases, es decir, sean más discriminantes entre las clases. De esta manera, quedarán descartados aquellos que sean irrelevantes y no aporten información de utilidad para el problema abordado. También se puede recurrir a otras técnicas para encontrar nuevos atributos que aporten información relevante. Este problema se aborda en la sección de preproceso de los datos 7.

		Clase Estimada		
		Positivo	**Negativo**	**Total**
Clase Real	**Positivo**	TP (true positive): Correcto Positivo	FN (false negative): Falso Negativo	p= TP + FN
	Negativo	FP (false positive): Falso Positivo	TN (true negative): Correcto Negativo	N= FP + TN
	Total	p'=TP + FP	n'= FN + TN	**N=p+n=p'+n'**

Figura 2: Matriz de confusión.

1.7.2 Clasificación

Los clasificadores se pueden dividir en dos categorías. Por un lado, la clasificación no supervisada, caracterizada por el desconocimiento de las clases de los datos y muy útil a la hora de descubrir familiaridades entre las diferentes instancias del conjunto de datos. La segunda categoría es la clasificación supervisada, ésta es la categoría en la que se enmarca este proyecto.

En los problemas de clasificación supervisada, disponemos de un conjunto de N muestras etiquetadas con su clase asociada y cada muestra se caracteriza por un conjunto de n atributos. El objetivo es inferir un modelo M que dé cuenta de las muestras y sea capaz de clasificar una nueva muestra descrita por sus n caraterísticas o atributos. El caso más simple de un problema de clasificación supervisada es aquel en el que se dispone únicamente de dos clases: positiva y negativa. En este caso el modelo deberá analizar y estudiar las caraterísticas comunes a todas aquellas muestras que pertenezcan a la clase postiva, para así ser capaz de clasificar una nueva muestra de clase desconocida, comparando los atributos de ésta con la descripción que ha aprendido.

Al conjunto de muestras clasificadas que utiliza el modelo para aprender se le llama *training set* o conjunto de entrenamiento y al conjunto de muestras no clasificadas que se utiliza para realizar una evaluación de la calidad del modelo, se le llama *test set* o conjunto de evaluación. Para estimar la calidad de un modelo se utilizará un conjunto de datos intermedio denominado *development set* o conjunto de desarrollo, este conjunto cuenta con, aproximadamente el mismo número de instancias que el conjunto de test. A partir de la evaluación del modelo obtenemos una matriz de confusión como podemos ver en la figura 2 con los valores:

- *True Positive*: Número de instancias positivas que el modelo ha clasificado como positivas.
- *True Negative*: Número de instancias negativas que el modelo ha clasificado como negativas.
- *False Positive*: Número de instancias negativas que el modelo ha clasificado como positivas.
- *False Negative*: Número de instancias positivas que el modelo ha clasificado como negativas.

La suma de estos cuatro valores se corresponde con el valor del número total de instancias, es decir, como se observa en la figura [2]: $n + p = n' + p' = N$. Estos valores sirven para obtener un conjunto de figuras de mérito,

que serán los indicadores de la calidad del clasificador. Para su cálculo se utilizan los valores normalizados de la matriz de confusión. Las ecuaciones de las figuras de mérito más típicas son:

$$Precision = \frac{tp}{tp + fp} \tag{1}$$

$$Recall = \frac{tp}{tp + fn} = TruePositiveRate(TPR) \tag{2}$$

$$Accuracy = \frac{tp + tn}{tp + fp + tn + fn} \tag{3}$$

$$F\text{-}measure = F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{4}$$

Ahora bien, esto es así cuando el problema sólo consta de dos clases, como "positivo" y "negativo". Sin embargo, existe gran variedad de problemas, como es nuestro caso, en los que el clasificador deberá discernir entre n clases diferentes. Lo que se hace en estos casos para conseguir las medidas de *Precision*, *Recall*, *Accuracy* y *F-measure*, es tomar una de las clases como positiva cada vez. Con la matriz de confusión resultante, de $n * n$ dimensiones, se calculan los promedios de las figuras de mérito.

Existen varias técnicas de evaluación de un modelo de aprendizaje automático. A continuación recopilamos las que se han utilizado en este trabajo.

- *Resustitución*, ya que, se le pasa el mismo conjunto para el entrenamiento y para el test. De forma que todas las instancias que tiene que clasificar, ya las ha visto antes. Aunque evidentemente no es fiable para dar una cifra de aciertos realista debido a que el modelo habrá visto con anterioridad todas las instancias del conjunto de test. Sin embargo, este tipo de evaluación es muy útil para saber la cota superior de aciertos del modelo.

- *Holdout* La evaluación por *holdout* es básicamente una partición del conjunto de datos en dos partes, una parte se utiliza para el entrenamiento y la otra se utiliza para el test. En el caso de este proyecto, los conjuntos de entrenamiento y test, vienen definidos previamente.

- *k-foldCrossValidation* Esta técnica simplemente es hacer el *Holdout* k veces. Es decir se parte el conjunto y se divide en k subconjuntos, entonces se toma uno para test y el resto para entrenamiento. Este proceso se repite k veces, y finalmente se calcula una media de los resultados obtenidos tras la evaluación de cada parte k_i del conjunto total. Precisamente debido a que aporta una media de k resultados en vez de uno sólo, esta técnica de evaluación es más realista de cómo funcionará el sistema ante una partición no-vista con anterioridad. Sin embargo, tiene el problema del alto coste temporal, ya que si el *Holdout* tiene un coste temporal n, el *k-foldCrossValidation* tendrá un coste del orden de $n * k$.

2 PLANTEAMIENTO INICIAL

En este capítulo se exponen distintos aspectos relacionados con la gestión del proyecto. También se presentan las distintas cuestiones a decidir y planificar antes de comenzar. Se expondrán los objetivos perseguidos en este trabajo, la arquitectura implementada, el alcance del proyecto mediante la definición del ciclo de vida y de un diagrama de Estructura de Descomposición del Trabajo (EDT). También se presentará la planificación temporal expresada mediante un diagrama Gantt, las herramientas necesarias para el desarrollo del proyecto, el análisis y gestión de riesgos y la evaluación económica.

2.1 OBJETIVOS PRINCIPALES

El principal objetivo de este proyecto es desarrollar un software capaz de alcanzar la mayor precisión y cobertura posibles, en la clasificación de diagnósticos clínicos.

Hay que tener en cuenta, que tenemos una cota superior que no se superará en ningún caso. Esto es debido a que el sistema alcanza su máximo rendimiento cuando tiene como entrada para entrenar y para testear, el mismo conjunto de datos. De manera que todas las instancias que se le piden que clasifique, ya las habría visto previamente en la fase de entrenamiento.

2.2 ARQUITECTURA

La arquitectura implementada en este proyecto es de tipo *pipeline*. Se parte de unos datos de entrada a los que se les aplica una serie de operaciones o filtros de manera secuencial. La salida de cada una de estas operaciones es la entrada de la siguiente. Cada fase se ejecuta de manera independiente a las demás. En la figura 3 se muestra este estilo de arquitectura de manera gráfica. En el caso concreto de nuestro proyecto la secuencia de operaciones aplicadas sería la siguiente: Preparación de los datos → Extracción de *features* → Modo representación → Selección de features → Clasificación

Figura 3: Representación de las diferentes fases.

2.3 ALCANCE

Ciclo de vida. Como punto de partida para la planificación, elegiremos el ciclo de vida adecuado a nuestro proyecto. Hemos elegido un ciclo de vida incremental. Dividimos la duración total del proyecto en diferentes etapas

o *sprints* de 15 días cada uno marcados por las reuniones de seguimiento con nuestras tutoras. En estas reuniones después de cada *sprint* se revisa el trabajo realizado durante ese periodo de tiempo, y se decide el trabajo a desarrollar durante el siguiente *sprint*. Trás cada etapa vamos consiguiendo un prototipo del proyecto final con más requisitos satisfechos. Se sigue una metodología similar al modelo *"Scrum"* pero con la diferencia de que en este caso, no están definidas todas las tareas desde un principio, ya que se trata de un proyecto de investigación, en el que se han ido añadiendo y modificando tareas según se iban viendo los resultados obtenidos y las necesidades surgidas.

Estructura de Descomposición del Trabajo (EDT). A continuación realizaremos un EDT. Ésta es una buena herramienta para la gestión de nuestro proyecto. Vamos a realizar una estructura jerarquizada, completa y descendente en la que se van a reflejar las distintas acciones y sus responsables durante el desarrollo de nuestro proyecto. El diagrama de Estructura de Descomposición del Trabajo, recogido en la lista de acrónimos presente al final de esta memoria C, es un elemento organizativo completo y jerarquizado que permite identificar en cada momento cuál es la situación del proyecto y quién es el responsable de la acción a realizar. El diagrama EDT de nuestro proyecto es el mostrado en la figura 4.

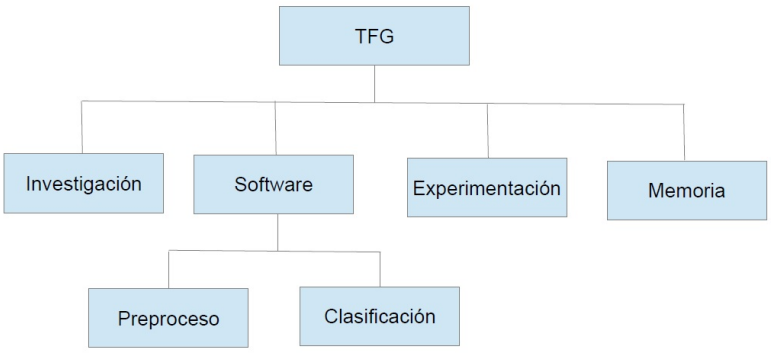


Figura 4: Estructura de Descomposición del Trabajo.

1. **Investigación.**
   Responsables: Nerea Aguirre y Estibaliz Amillano.
   Esfuerzo: 58 horas/persona.
   Duración: 29 horas.
   Descripción: Realizar el trabajo de documentación e investigación. Informarse sobre trabajo previo realizado relacionado con nuestro proyecto y posibles técnicas a aplicar adecuadas a las características de este trabajo.
   Entradas: Libros, artículos, tesis y documentos especializados en la temática abarcada en este trabajo.
   Salidas/Entregables: Documentos con resúmenes y datos importantes que se utilizarán para la ayuda a la elaboración de nuestro proyecto.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores y un editor de textos LaTeX.
   Precedencias: Ninguna.

2. **Software.**
   Responsables: Nerea Aguirre y Estibaliz Amillano.
   Esfuerzo: 474 horas/persona.
   Duración: 237 horas.
   Descripción: Diseño y programación del software de este trabajo.
   Entradas: corpus disponible para la realización de este proyecto, documentación especializada sobre las técnicas aplicables a este proyecto así como los documentos y notas redactados en la tarea de investigación, y documentación especializada sobre las librerías de Weka [Ian H. Witten(2011)], explicada en la sección dedicada a las herramientas 2.5 .
   Salidas/Entregables: Software completo.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores y las herramientas Eclipse y Weka.
   Precedencias: Tarea de investigación empezada.

   a) **Software. Preproceso**
   Responsables: Estibaliz Amillano.
   Esfuerzo: 237 horas/persona.
   Duración: 118.5 horas.
   Descripción: Diseño y programación de la parte del preproceso del software.
   Entradas: corpus disponible para la realización de este proyecto, documentación especializada sobre las técnicas aplicables a este proyecto así como los documentos y notas redactados en la tarea de investigación, y documentación especializada sobre la utilización de las librerías de Weka.
   Salidas/Entregables: La parte del preproceso del software elaborado en este trabajo.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores y las herramientas Eclipse y Weka.
   Precedencias: Tarea de investigación empezada.

   b) **Software. Clasificación.**
   Responsables: Nerea Aguirre.
   Esfuerzo: 237 horas/persona.
   Duración: 118.5 horas.
   Descripción: Diseño y programación de la parte de clasificación del software.
   Entradas: corpus disponible para la realización de este proyecto, documentación especializada sobre las técnicas aplicables a este proyecto así como los documentos y notas redactados en la tarea de investigación, y documentación especializada sobre la utilización de las librerías de Weka.
   Salidas/Entregables: La parte de la clasificación del software elaborado en este trabajo.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores y las herramientas Eclipse y Weka.

Precedencias: Tarea de investigación empezada.

3. **Experimentación.**
   Responsables: Nerea Aguirre y Estibaliz Amillano.
   Esfuerzo: 20 horas/persona.
   Duración: 10 horas.
   Descripción: Realización de los experimentos pertinentes para obtener la información necesaria que nos permita tomar decisiones sobre las técnicas más adecuadas a utilizar.
   Entradas: corpus disponible para la realización de este proyecto y el software desarrollado en este trabajo .
   Salidas/Entregables: Tablas de resultados y conclusiones alcanzadas.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores.
   Precedencias: Tareas de investigación y software en un punto de desarrollo avanzado .

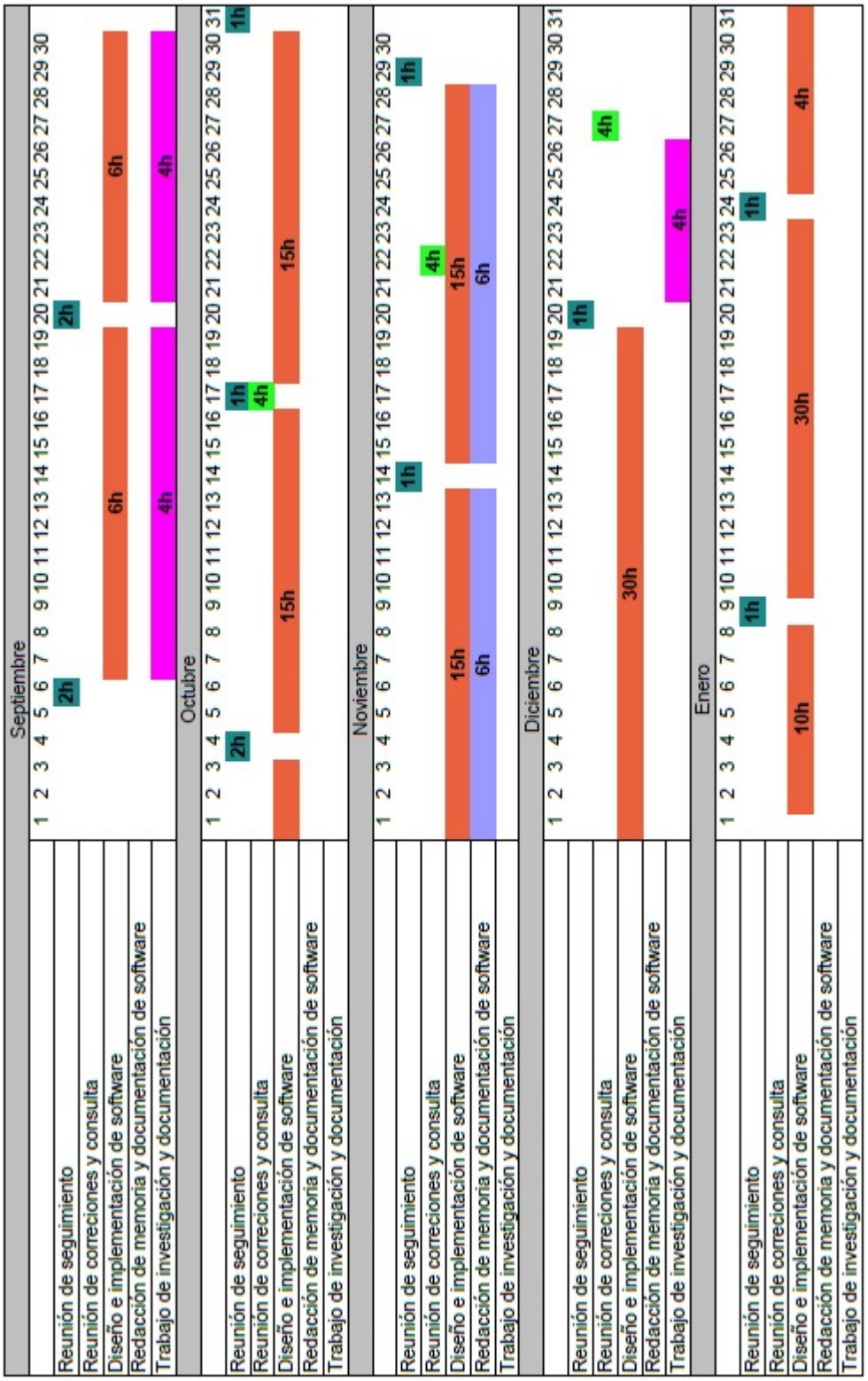
4. **Memoria.**
   Responsables: Nerea Aguirre y Estibaliz Amillano.
   Esfuerzo: 222 horas/persona.
   Duración: 111 horas.
   Descripción: Redacción de la memoria del trabajo realizado y documentación del software desarrollado.
   Entradas: Libros, artículos, tesis y demás documentos especializados en la temática abarcada en este trabajo.
   Salidas/Entregables: Memoria del proyecto.
   Recursos necesarios: Los dos miembros implicados con sus correspondientes ordenadores y un editor de texto LaTeX.
   Precedencias: Tareas de investigación, software y experimentación.

2.4 PLANIFICACIÓN TEMPORAL

En la figura 5 se detalla la planificación temporal del proyecto, representada mediante un diagrama de Gantt. La planificación de horas es la estimada al comienzo del proyecto, para uno de los integrantes del equipo, 424 horas. Estimamos el mismo tiempo para ambos integrantes, aunque como se puede ver en las figuras 64 y 65 en el capítulo 10 donde se muestra una comparación entre las horas estimadas y las horas reales invertidas, ésto no ha sido así exactamente. Por tanto para conocer el total de horas invertidas estimadas, habría que duplicar el tiempo reflejado en el diagrama, es decir, en este trabajo estaba previsto invertir un total de 848 horas (sin contar las horas invertidas por las dos personas encargadas de la tutorización del mismo).

DIAGRAMA GANTT

Tareas:
Reunión de seguimiento
Reunión de correciones y consulta
Diseño de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación


Septiembre
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Reunión de seguimiento
Reunión de correciones y consulta
Diseño e implementación de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación
2h
2h
6h
4h
6h
4h
Octubre
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Reunión de seguimiento
Reunión de correciones y consulta
Diseño e implementación de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación
2h
1h
4h
15h
15h
Noviembre
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30
Reunión de seguimiento
Reunión de correciones y consulta
Diseño e implementación de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación
1h
1h
4h
15h
6h
6h
Diciembre
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Reunión de seguimiento
Reunión de correciones y consulta
Diseño e implementación de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación
1h
4h
30h
4h
Enero
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Reunión de seguimiento
Reunión de correciones y consulta
Diseño e implementación de software
Redacción de memoria y documentación de software
Trabajo de investigación y documentación
1h
1h
10h
30h
4h

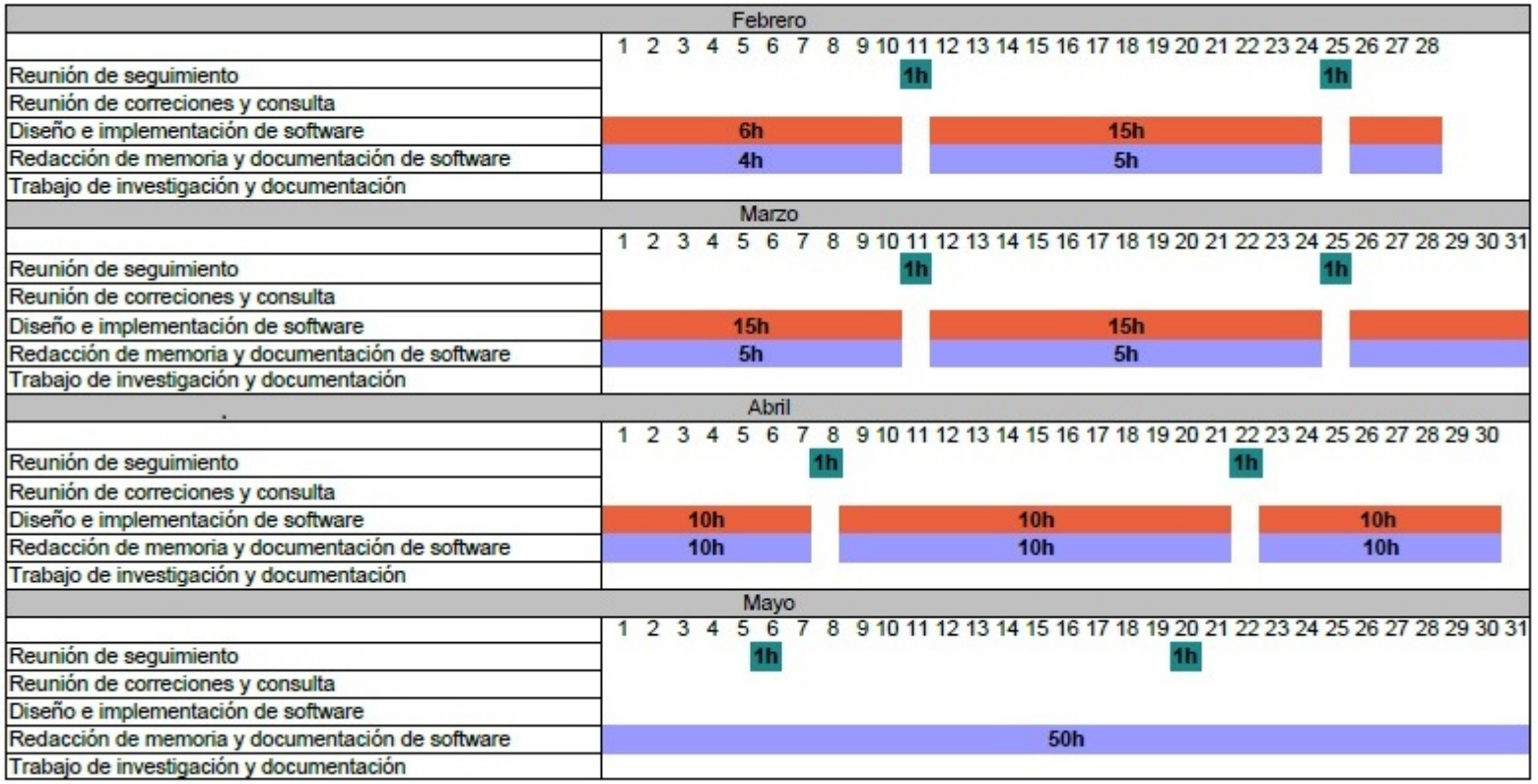


Figura 5: Planificación temporal.

2.5 HERRAMIENTAS

En esta sección describiremos las herramientas de las que nos serviremos para desarrollar el proyecto:

- Eclipse: Entorno de programación para desarrollar el software en Java, para más información sobre Eclipse y Java consultar en la bibliografía; [Java(2014)] y [Eclipse(2014)].
- Weka: Software libre para la ayuda en el desarrollo de proyectos de minería de datos y aprendizaje automático. Contiene herramientas de preproceso, clasificación, regresión, etc. Se trata de un software desarrollado por la universidad de Waikato. Para más información consultar en la bibliografía [Ian H. Witten(2011)].
- Dropbox: Servicio disponible para el almacenamiento de archivos en la nube. Se permite compartir estos archivos con uno o más usuarios en línea [Dropbox(2014)].
- Documentación especializada: Textos clínicos y notas médicas proporcionados por el Hospital de Galdakao.
- Documentación especializada sobre LaTeX . Necesaria para el aprendizaje de una forma de editar textos desconocida hasta el momento [LaTeX(2014)].
- Procesadores de textos LaTeX (Kile, TeXworks, WinEdt) [LaTeX(2014)].
- Cacoo: Herramienta web utilizada para la creación de diagramas y figuras. Elegida debido a que ofrece la posibilidad de editar diagramas por múltiples usuarios, en tiempo real. [Cacoo(2014)]
- Cliente VPN. Software que permite la conexión remota vía `ssh`. En el caso de este proyecto lo utilizamos para ejecutar experimentos en las máquinas de la universidad.

2.6 GESTIÓN DE RIESGOS

1. **Pérdida o corrupción del corpus.**
   - **Descripción:** En caso de perder o corromper el corpus completo o parte de el, nos encontraríamos ante una grave situación debido al carácter confidencial de los datos.
   - **Prevención:** Se almacenarán copias duplicadas del corpus en dos memorias flash diferentes cuidándoloas con especial atención y responsabilidad. No se compartirá ningún extracto del corpus vía Internet.
   - **Plan de contingencia:** Se restaurarán los datos obteniéndolos de la otra memoria.
   - **Probabilidad:** Poco probable.
   - **Impacto:** En caso de sufrir esta pérdida el impacto será importante.
2. **Mala planificación.**

- **Descripción:** Debido a que se trata de un proyecto de una duración en el tiempo considerable, hay que tener en cuenta que es difícil realizar un planificación temporal correcta desde el principio puesto que es muy probable que surjan complicaciones en las tareas planteadas y que lleven más tiempo del planificado, o que por el contrario ciertas partes del proyecto resulten más sencillas de lo esperado en un primer momento.
- **Prevención:** Para evitar que esto ocurra se realizará un planificación temporal analizando atentamente el tiempo que debería llevar cada tarea y basada en experiencias previas con proyectos similares.
- **Plan de contingencia:** En caso de detectarse un error en la planificación, habrá que modificarla ajustando los plazos desde el momento en que se ha producido dicho error hasta el final del proyecto.
- **Probabilidad:** Muy probable
- **Impacto:** El impacto de este riesgo en caso de producirse, variará dependiendo de la magnitud del error cometido.

3. **Pérdida de datos.**
   - **Descripción:** Pérdida de los datos del proyecto en un ordenador causada por un virus u otras razones.
   - **Prevención:** Para evitarlo se realizarán copias de seguridad de los avances que se vayan haciendo.
   - **Plan de contingencia:** Hacer una recuperación de las copias de seguridad realizadas.
   - **Probabilidad:** Poco probable.
   - **Impacto:** El impacto de este riesgo en caso de producirse, variará dependiendo de la cantidad de datos perdidos.
4. **Pérdida de un equipo.**
   - **Descripción:** Hay que considerar la pérdida de un ordenador por diferentes causas, ya que puede estropearse o romperse así como perderse o ser robado.
   - **Prevención:** Para evitarlo se tendrá especial cuidado con cualquier equipo que se utilice para la realización del proyecto. Se utilizarán fundas especiales o resistentes para reducir el impacto de golpes y se pondrá especial atención en no dejarlos descuidados ni olvidados.
   - **Plan de contingencia:** Recuperar los datos del dropbox y conseguir un equipo nuevo.
   - **Probabilidad:** Probable.
   - **Impacto:** El impacto de este riesgo en caso de producirse, variará dependiendo de si es una pérdida temporal o definitiva y de la cantidad de datos perdidos en caso de perderse alguno.
5. **Pérdida de algún miembro del equipo.**
   - **Descripción:** Este proyecto se esta realizando en colaboración dentro de un equipo de trabajo y por lo tanto hay que considerar el riesgo de sufrir una baja en el grupo, temporal o definitiva, por enfermedad, abandono u otros motivos.

- **Prevención:** Se realizarán reuniones periódicas para estar al tanto de la situación de todos los componentes del equipo y del trabajo que estan llevando a cabo.
- **Plan de contingencia:** En caso de que se produjera una baja temporal se le mantendrá informado de los avances durante su ausencia para minimizar el impacto. En caso de ser una baja definitiva se reasignarán las tareas que debía realizar y afecten al trabajo de otros miembros del equipo.
- **Probabilidad:** Poco probable.
- **Impacto:** En caso de baja temporal el impacto dependerá del tiempo que se ausente. En caso de baja definitiva el impacto puede ser importante.

6. **Pérdida del servicio de Dropbox.**
   - **Descripción:** En este proyecto se utiliza Dropbox como herramienta de almacenamiento de la información y por tanto se debe considerar el riesgo de la pérdida de servicio de este.
   - **Prevención:** Se descargará la aplicación de Dropbox en el equipo para que los datos esten almacenados en la memoria y no solo en la red.
   - **Plan de contingencia:** En caso de perder Dropbox se abrirá una cuenta en otra aplicación de almacenamiento web y se cargarán los datos en la misma.
   - **Probabilidad:** Poco probable.
   - **Impacto:** En caso de sufrir esta pérdida el impacto será mínimo.

2.7 EVALUACIÓN ECONÓMICA

Antes de entrar a valorar el contexto económico de este proyecto es necesario mencionar que no existe ninguna pretensión económica, ya que se trata de un Trabajo Fin de Grado desarrollado dentro del grupo de investigación IXA (http://ixa.si.ehu.es). La finalidad de este trabajo no es de carácter comercial, puesto que está englobado en un proyecto de investigación mucho más amplio. No obstante vamos a presentar los costes asociados a la realización del trabajo y los costes que supondría si el trabajo hubiese sido remunerado.

Costes en equipos informáticos:

Elemento	C. adquisición	Duración	Unidades	Tiempo uso
Portátiles	800 €	60 meses	2	**12 meses**
Mesa	1600 €	72 meses	4	**4 meses**

Costes salariales:

Personal	Salario/hora	Total horas	Salario total
Investigador 1	15 €	424 horas	6360 €
Investigador 2	15 €	424 horas	6360 €

A tenor de los datos tabulados, se obtienen los datos necesarios para estimar el posible coste económico derivado de los equipos y el personal ne-

cesario para el desarrollo. En las siguientes ecuaciones (5, 6, 7, 8, 9, 10) se detallan los cálculos necesarios para obtener el coste total.

$$\text{Amortización equipos} = = \frac{\text{Coste total}}{\text{Duración estimada}} * \text{Tiempo de uso} * \text{Unidades} \tag{5}$$

$$\text{Amortización ordenadores portátiles} = \frac{800}{60} * 12 * 2 = 320{,}00\ € \tag{6}$$

$$\text{Amortización máquinas laboratorio} = \frac{1600}{72} * 4 * 4 = 355{,}56\ € \tag{7}$$

$$\text{Costes salariales} = \text{Coste por hora} * \text{n.º horas} * \text{n.º de investigadores} \tag{8}$$

$$\text{Costes salariales} = 15 * 424 * 2 = 12720{,}00\ € \tag{9}$$

$$\text{Total costes} : 320 + 355{,}56 + 12720 = 13395{,}56\ € \tag{10}$$

Por tanto el coste total del proyecto ascendería a un total de 13395,56 €, no obstante habría que mencionar que Estibaliz recibió una beca de colaboración en investigación del Gobierno Vasco por lo que en este caso los costes totales se reducirían en 2800,00 €, que es la cuantía de la beca mencionada. Finalmente concluimos que este trabajo podría valorarse en 10595,56 €.

3 ANÁLISIS DE ANTECEDENTES

En este capítulo se realiza un análisis de los antecedentes de nuestro proyecto. En la sección 3.1, se explica el estado en que se encuentra la minería de datos en el área de la lingüística computacional para procesamiento del lenguaje natural, área en que se desarrolla nuestro proyecto . A continuación, en la sección titulada "Estado del arte" 3.2, explicamos las similitudes y diferencias con trabajos previos relacionados, y aspectos en los que nos han sido de ayuda.

3.1 MINERÍA DE DATOS EN EL ÁREA DE LA LINGÜÍSTICA COMPUTACIONAL PARA PROCESAMIENTO DEL LENGUAJE NATURAL

Las técnicas de procesamiento de lenguaje natural permiten extraer información útil y conocimiento de textos. Gracias a éstas técnicas, tenemos sistemas que son capaces de llevar a cabo tareas tales como la traducción automática o el diálogo con el usuario. Tal y como se explica en el artículo [Gelbukh(2002)], el esquema general que siguen la mayoría de los sistemas que implementan procesamiento del lenguaje es el siguiente: en primer lugar, se realiza un preproceso de los datos, aplicándoles las técnicas que más convengan, con el fin de obtener una representación del texto que preserva sus características más relevantes. Después, el programa principal manipula esta representación, transformándola según la tarea, buscando en ella las subestructuras necesarias, etc. Finalmente, si es necesario, los cambios realizados sobre la representación obtenida en el preproceso, por el programa principal, se transforman en lenguaje natural.

El procesamiento del lenguaje natural no es una tarea sencilla. A la hora de implementar estos sistemas hay que enfrentarse a diversos problemas. El primero de ellos es la ambigüedad del lenguaje. Una misma expresión o palabra puede interpretarse de diferentes maneras puesto que dependerá del contexto concreto de cada caso. La desambiguación, es algo que el cerebro humano hace automáticamente sin darse cuenta. Sin embargo, para una máquina, elegir la opción del significado correcto puede entrañar una gran complejidad. Se hace necesario por tanto, aportar a la máquina el conocimiento necesario para entender el sentido y el contexto de cada texto. Para dotar a la máquina de tal conocimiento, sería necesaria una base de datos de gran tamaño para almacenar tal cantidad de información. El aprendizaje automático tiene como fin el descubrimiento totalmente automático de las regularidades y las relaciones en los datos y así solucionar los problemas anteriormente mencionados.

En nuestro caso, el problema de la ambigüedad es uno de las mayores dificultades a las que nos enfrentamos, puesto que, como ya hemos mencionado anteriormente, cada médico escribe el diagnóstico de una forma diferente. Además hay una considerable cantidad de diagnósticos que tienen el mismo código ICD-9-CM para ambas partes del cuerpo, es decir, una afec-

ción del brazo derecho tiene el mismo código que una afección en el brazo izquierdo. Para ayudar al sistema a decidir qué términos son equivalentes, se ha utilizado una forma de representación basada en clases de equivalencia o "etiquetas" de sinónimos, ésto se explica en el capítulo dedicado al preproceso, en la sección 7.2.3.

3.2 ESTADO DEL ARTE

Desde hace años se han llevado a cabo diversos estudios con objetivos similares al que perseguimos nosotras en este trabajo dentro del proyecto SENDAUR, automatizar la clasificación de diagnósticos clínicos escritos en lenguaje natural siguiendo el estándar del ICD-9-CM. Sin embargo, no conocemos ningún estudio que se haya realizado para textos escritos en castellano.

Aronson en [Aronson & et al.(2007)] realizó un estudio en el que se adaptó el *Medical Text Indexer* o Indexador de Textos Médicos (MTI) de la Librería Nacional para convertir textos clínicos en su correspondiente código ICD-9-CM. El *MTI* se comportaba ligeramente parecido a un clasificador *SVM* (el cual se detalla en el capítulo dedicado a la clasificación 8.1.4). Sólo contaba con las características obtenidas después de aplicar el filtro *Bag of Words* 7.1.3.1. Uniendo el *MTI*, el clasificador *SVM*, el clasificador *KNN* C y un simple reconocedor de patrones, la nota final obtenida en el test oficial fue de 85.

El estudio de Aronson resulta muy útil, ya que al igual que nosotras, pretende clasificar los diagnósticos según el estándar ICD-9-CM. Por tanto, también probaremos el filtro *Bag of Words* en nuestro trabajo.

Yitao Zhang [Zhang(2006)], utilizó un corpus compuesto por el conjunto de datos oficial para el Concurso de Medicina Computacional de 2007. El reto consistía en clasificar 1954 informes radiológicos del *Cincinnati Children's Hospital Medical Center* divididos en 978 instancias para el conjunto de entrenamiento y 976 instancias para el conjunto de test. Este corpus presenta un total de 45 códigos diferentes pertenecientes al ICD-9-CM. Para realizar los experimentos, en este estudio se utiliza el clasificador *SVM* y se genera uno por cada instancia del corpus que coincide con algún código ICD-9-CM. Cada clasificador realiza una decisión binaria "Si" o "No" haciendo referencia a si debería asignarse o no el código a la instancia. En cuanto al preproceso, todos los términos negados o que no tienen relación con conceptos clínicos, son eliminados del corpus. Con las técnicas explicadas previamente, consiguieron una F-measure de 0,85.

En nuestro caso, a diferencia de Zhang, no se conoce el informe completo, únicamente se dispone del término diagnóstico. También existen un par de diferencias con respecto al tamaño del conjunto de entrenamiento y al número de clases a predecir. Mientras que en el concurso de 2007 se trataba con un total de 45 códigos, en nuestro caso tratamos con el estándar ICD-9-CM completo, esto es un total de 14000 códigos posibles. En cuanto al tamaño del conjunto de entrenamiento, contamos con un conjunto de entrenamiento de 23130 instancias y un conjunto de test de 2850 instancias frente a las 978 y 976 instancias con las que contaba Zhang para el conjunto de entrenamiento y el conjunto de test respectivamente. En nuestro caso generaremos un solo clasificador que decida qué código corresponde a cada instancia, a diferencia del estudio de Zhang, en el que se generó uno por cada instancia que coincidía con algún código ICD-9-CM.

Julia Medori y Cédrick Fairon en [J.Medori(2010)] llevaron a cabo un estudio sobre la selección de atributos necesaria para optimizar los resultados, en un proyecto que realizaron con el objetivo de ayudar a la codificación de informes de alta de pacientes. Los informes estaban escritos en Francés. Al igual que en nuestro proyecto, se enfretaban al estándar completo del ICD-9-CM. El proceso se automatizó en el hospital de Bruselas. El sistema elaborado en dicho proyecto estaba estructurado en dos módulos, un módulo de extracción en el que se obtiene la información considerada relevante e importante para la codificación y un módulo de codificación en el que se asignan los códigos. Julia Medori y Cédrick Fairon, construyeron un clasificador para cada código o clase presente en el corpus. Llevaron a cabo 4 experimentos. En el Experimento 1, seleccionaron los atributos identificados como diagnósticos y se eliminaron aquellos que aparecían negados. También normalizaron el corpus, es decir, eliminaron tildes y *stop words* o palabras vacías B, y pasaron a minúsculas todas las palabras. De esta forma, el sistema aceptará como iguales "afección" y "afeccion". En el Experimento 2, se normalizó el corpus y se utilizó una técnica de *stemming* B mejorando los resultados. En el Experimento 3 se utilizaron todas las palabras que aparecían, menos las *stop words*, y se normalizó el texto. Esto empeoró los resultados demostrando la importancia del proceso de selección de atributos. En el Experimento 4, en lugar de utilizar cada código como una clase se establecieron categorías de códigos agrupándolos según los 3 primeros dígitos, consiguiendo así una considerable mejora de los resultados.

El problema abordado por Julia Medori y Cédrick Fairon [J.Medori(2010)], tiene varias similitudes con el nuestro. En primer lugar, el estándar de codificación es el mismo, lo cual implica que el número de opciones a discriminar por el sistema de aprendizaje automático es el mismo (del orden de 14000). Por otra parte, ellos también se enfrentaron al problema en un idioma diferente del inglés, que es el idioma con más recursos en este campo. Por lo tanto, hemos aprovechado algunas de sus ideas, como la normalización del texto, la eliminación de las tildes y el paso en nuestro caso a mayúsculas de todas las palabras. Puesto que en este estudio quedó demostrada la importancia de las técnicas de selección de atributos, decidimos incorporarlas también en nuestro proyecto. Sin embargo, existe una gran diferencia: ellos disponían del informe completo, mientras que en nuestro caso, se dispone unicamente del término diagnóstico. Por esta razón no se han implementado aquellas técnicas que aportan información de textos largos con signos de puntuación, negación de términos etc.

Jon Patrick [Zhang & Patrick(2008)], utiliza como corpus una colección de textos patológicos obtenidos del *"South Western Area Pathology Service"* (SWAPS). Todas estas notas clínicas están redactadas por los médicos en forma de lenguaje natural. El experimento realizado en este estudio está estructurado en 5 apartados; preproceso, representación del texto en un vector (Indexado), reducir dimensiones filtrando características, aprendizaje automático y código SNOMED. En este proyecto se utilizan distintas técnicas de preproceso. Se aplican técnicas de *stemming*, *N-gramm tokenization* y *Stop Words Exclusion*, además de la representación del texto en un vector. Antes de realizar la clasificación, Jon Patrick utiliza técnicas de reducción dimensional, tales como las basadas en la frecuencia de los términos (*TF-IDF*) y las basadas en Ganancia de Información (*Info Gain*). Ambas técnicas que también hemos implementado nosotras y que se detallan en el capítulo 7, en la sección 7.1.5.2. En este estudio Jon Patrick utiliza tres algoritmos de aprendizaje automático; *Maximum Entropy* o Máxima Entropía, *Decision Tree*

o Árbol de Decisión, y *Support Vector Machine (SVM)* o Máquinas de Soporte Vectorial. Estos dos últimos algoritmos también los hemos implementado en este proyecto y se explican en la sección 8.1.

3.3 DISCUSIÓN DE ANTECEDENTES EN RELACIÓN A NUESTRO TRABAJO

Teniendo estos proyectos un objetivo final similar al nuestro es de esperar que se encuentren muchas similitudes en las líneas de investigación y experimentación. De hecho, informarnos y estudiar las conclusiones obtenidas por ellos nos ha servido de base y guía para elegir la metodología seguida.

Al igual que en el estudio de Aronson [Aronson & et al.(2007)], en la primera ronda de experimentos que realizamos el único preproceso aplicado fue la técnica del *Bag of Words* 7.1.3.1. Además, una vez establecido el *Baseline* o línea base del proyecto, y lanzado con clasificadores con poco coste temporal, se implementó el clasificador *SMO* como variante del *SVM*, que explicamos en la sección 8.1.4.

También nos hemos basado en el trabajo de Yitao Zhang [Zhang(2006)], incluyendo algunas técnicas aplicadas en su proyecto, como el *Bag of Words*, detallado en la sección 7.1.3.1, o el *SVM*. Aunque el corpus que nosotras hemos utilizado es considerablemente más extenso, con un total de 23130 instancias en el conjunto de entrenamiento y 2850 en el conjunto de test. Además, los datos no son un informe médico completo sino un diagnóstico clínico, que no recoge los síntomas o la falta de ellos, por tanto no hay negaciones ni conceptos que no pertenezcan al ámbito de la medicina. Por esta razón otras técnicas aplicadas como la eliminación de negaciones o conceptos no relacionados con la médicina, también utilizada en el trabajo de Julia Medori y Cédrick Fairon, en nuestro proyecto no tienen sentido.

Como Julia Medori y Cédrick Fairon [J.Medori(2010)], se podría decir que nuestro trabajo también presenta dos módulos, el preproceso y la clasificación, los cuales realizan tareas similares a los de extracción y codificación presentes en el suyo. Además ambos trabajos coinciden en ciertas técnicas como la normalización del texto y difieren en otras como la eliminación de negaciones anteriormente mencionada.

El trabajo de Jon Patrick [Zhang & Patrick(2008)] quizá sea el más parecido al nuestro habiéndolo tomado como primera guía y base para empezar la elaboración de nuestro proyecto. Al igual que los datos de los que disponemos nosotras para nuestro proyecto, todas las notas clínicas disponibles están redactadas por los médicos en lenguaje natural. Jon Patrick trata de automatizar la clasificación a códigos SNOMED y en nuestro caso lo hacemos a códigos ICD-9-CM. La estructura que nosotras hemos seguido es bastante similar a la de su trabajo habiendo llevado a cabo también una fase de preproceso. En esta fase incluimos la representación del texto en forma de vector y la reducción dimensional. Posteriormente se procede a una fase de clasificación mediante aprendizaje automático. Al igual que Jon Patrick, nosotras también hemos utilizado la técnica de *N-gramm tokenization*, que definimos en el glosario B. Sin embargo, no hemos utilizado la técnica de *stemming*, por no ser una técnica adecuada para nuestros datos debido a que el corpus utilizado contiene sólo diagnósticos. Por tanto, no hay muchas variaciones en las formas gramaticales de las palabras. Otra similitud entre el proyecto de Jon Patrick y el nuestro es la representación del texto en

forma de vector, como detallamos en la sección 7.1.3.1. Aunque él lo utiliza como única técnica de representación y nosotras como una de las opciónes posibles, ya que hemos experimentado también con otras representaciones de texto, como puede verse en la sección 7.1.3. También hemos utilizado las técnicas de reducción dimensional que emplea Jon Patrick, como *TFIDF* e *Info Gain* entre otras, detalladas en la sección 7.1.5.2. En cuanto a la parte de clasificación también encontramos similitudes entre ambos trabajos ya que en nuestro proyecto se aplica el *Decision Tree* o Árbol de Decisión, concretamente el algoritmo que implementan las librerías de Weka, *J48*, y el *Support Vector Machine* o Máquina de Soporte Vectorial entre otros que se enumeran y explican en la sección 8.1.

Finalmente, como hemos mencionado anteriormente, tanto en el estudio de Aronson como en el de Julia Medori y Cédrick Fairon, se genera un clasificador binario para cada código o clase presente en el corpus. En nuestro proyecto se ha valorado y comenzado a implementar este método de clasificación, aunque en el momento de entrega de esta memoria sigue en elaboración y desarrollo.

4 CAPTURA DE REQUISITOS

Este trabajo se enmarca dentro del proyecto SENDAUR financiado por el Gobierno Vasco desarrollado en colaboración entre el grupo de investigación IXA (http://ixa.si.ehu.es) y el Hospital de Galdakao. Por tanto, se nos pide satisfacer las necesidades y rquisitos indicados por el grupo directamente, debido a que estamos desarrollando una parte de un proyecto más extenso y por el Hospital indirectamente.

Se debe conseguir un sistema capaz de clasificar automáticamente diagnósticos clínicos según la clasificación ICD-9-CM y se sugiere explorar técnicas de minería de datos.

4.1 CASOS DE USO

En la figura 6 se muestran los casos de uso correspondientes al software desarrollado en este trabajo. Se exponen de izquierda a derecha y de arriba abajo indicando el orden de desarrollo lógico (PrepararDatos, AñadirFeatures, ModoRepresentación, SeleccionarFeatures y Clasificación). Sin embargo, son independientes entre sí y el orden puede ser variable. El actor es el mismo para todos ellos.

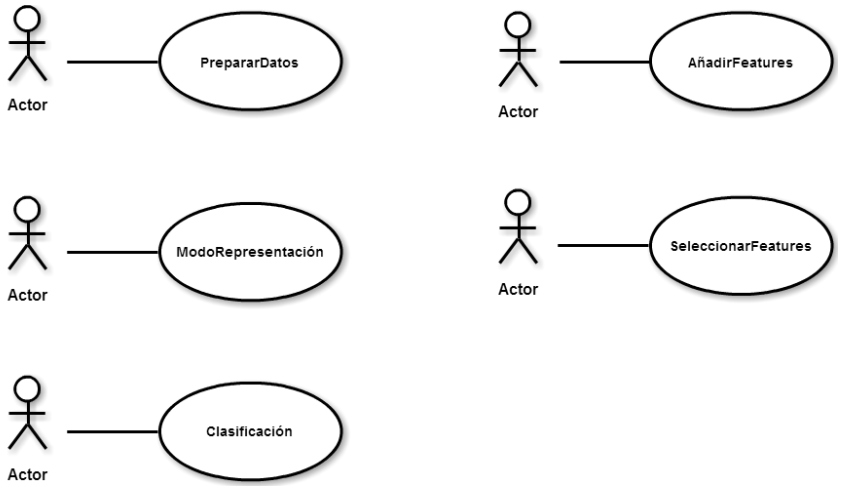


Figura 6: Casos de uso.

A continuación, en las tablas 2, 3, 4, 5 y 6, se muestra cada uno de los casos de uso extendidos.

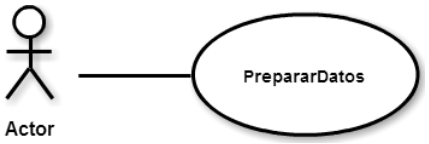

Nombre. Preparar Datos
Descripción. A partir de un documento en formato `.txt` se genera un nuevo documento en formato `.arff`.
Actores. Usuario
Precondiciones. Los datos de entrada deben estar en formato `.txt` y con la estructura {atributo}:{clase}. Se requieren dos argumentos de entrada, el conjunto de datos y la opción correspondiente a normalización o a no normalización.
Postcondiciones. Si el usuario elige normalizar se generan dos ficheros `.arff`; uno sin normalizar y otro normalizado. Si no se elige la opcion de normalizar, se genera un archivo `.arff` sin normalizar.
Flujo de eventos. 1. El usuario lanza `PrepararDatos.jar` pasándole como primer argumento la ruta del fichero `.txt` que se quiere covertir a `.arff` y como segundo argumento un 0 si no quiere normalizar el texto o un 1 si quiere normalizarlo. 2. Al finalizar la ejecución se habrá generado el fichero `.arff` deseado en la misma ruta que el fichero de entrada.

Tabla 2: Caso de uso PrepararDatos.


Nombre. Añadir features
Descripción. A un conjunto de instancias pasadas como entrada se le añaden nuevas *features* o atributos.
Actores. Usuario
Precondiciones. Los datos de entrada deben estar en formato `.arff`. Las instancias deben seguir la representación original de los datos, es decir, un atributo de tipo *String* y el atributo clase de tipo nominal. Se requieren cuatro argumentos de entrada, el conjunto de instancias, el documento de los estándares del ICD-9-CM, el documento con la lista de sinónimos de los términos, y la opción correspondiente a que atributos se desea anãdir.
Postcondiciones. Se genera un fichero `.arff` que contiene el conjunto de instancias de entrada con los nuevos atributos añadidos.
Flujo de eventos. 1. El usuario lanza `AñadirFeatures.jar` pasándole como argumentos la ruta del conjunto de instancias, la ruta del ICD-9-CM, la ruta de la lista de sinónimos y un 0 si se quieren añadir todas las features o un 1 si solo se desea añadir el atributo “sinónimos”. 2. Al finalizar la ejecución se habrá generado el fichero `.arff` deseado en la misma ruta que el fichero de entrada.

Tabla 3: Caso de uso AñadirFeatures.

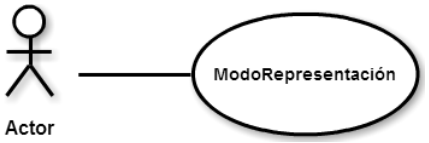

Nombre. Modo representación
Descripción. Un conjunto de instancias pasadas como entrada se representan en forma de vector o en forma nominal.
Actores. Usuario
Precondiciones. Los datos de entrada deben estar en formato `.arff`. Se requieren dos argumentos de entrada, el conjunto de instancias, y la opción correspondiente a la forma de representación deseada.
Postcondiciones. Se genera un fichero `.arff` que contiene el conjunto de instancias de entrada representadas de la manera escogida.
Flujo de eventos. 1. El usuario lanza `ModoRepresentacion.jar` pasándole como argumentos la ruta del conjunto de instancias, y un `0` si se quieren representar en forma de vector o un `1` si se quieren representar de manera nominal. 2. Al finalizar la ejecución se habrá generado el fichero `.arff` deseado en la misma ruta que el fichero de entrada.

Tabla 4: Caso de uso ModoRepresentación.


Nombre. Seleccionar features
Descripción. Se seleccionan los atributos mas informativos y relevantes de un conjunto de instancias pasadas como entrada.
Actores. Usuario
Precondiciones. Los datos de entrada deben estar en formato `.arff`. Se requieren tres argumentos de entrada, el conjunto de instancias de entrenamiento, el conjunto de instancias de evaluación y la opción correspondiente al método de selección deseado.
Postcondiciones. Se generan dos ficheros `.arff` que contienen los conjuntos de instancias de entrada tras eliminar los atributos considerados no informativos o irrelevantes por el método escogido.
Flujo de eventos. 1. El usuario lanza `SeleccionarFeatures.jar` pasándole como argumentos las rutas de los conjuntos de instancias de *train* y *test*, y el método de selección deseado (0=InfoGain, 1=TFIDF, 2=Remove, 3=subsetSelectionClassifier, 4=SubsetSelectionCFS, 5=SubsetSelectionWrapper, 6=PCA). 2. Al finalizar la ejecución se habrá generado el fichero `.arff` deseado en la misma ruta que el primer fichero de entrada.

Tabla 5: Caso de uso SeleccionarFeatures.

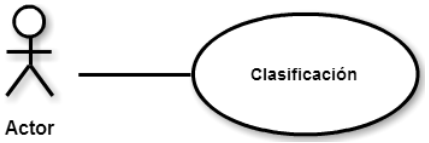

Nombre. Clasificación
Descripción. Se entrena el algoritmo de clasificación escogido por el usuario con un conjunto de instancias para dicho fin, y a continuación el modelo realiza la clasificación de otro conjunto de instancias pasadas como entrada. Comparando las clases predichas con las reales se estima la precisión con la que actúa el modelo.
Actores. Usuario
Precondiciones. Los datos de entrada deben estar en formato `.arff`. Se requieren seis argumentos de entrada, las instancias de entrenamiento, las instancias de evaluación, la opción correspondiente al algoritmo de clasificación deseado, la opción correspondiente al barrido o no barrido de parámetros, la opción correspondiente a evaluación por *holdout* o evaluación cruzada y el número de *folds* para hacer la evaluación cruzada.
Postcondiciones. Se genera un fichero `.arff` con las predicciones generadas. Además se obtiene como salida los resultados obtenidos según distintas medidas de calidad de los clasificadores.
Flujo de eventos. 1. El usuario lanza `Clasificacion.jar` pasándole como argumentos las rutas de los conjuntos de instancias de *train* y *test*, el algoritmo de clasificación deseado (0=NaiveBayes, 1=J48, 2=RandomForest, 3=SMO, 4=BayesNet), un 0 si se desea barrer parámetros o un 1 si no, un 0 si se desea evaluación por *holdout* o un 1 si se prefiere evaluación cruzada, y el número de *folds* para realizar *k-fold cross validation* en caso de haberse elegido la evaluación cruzada. 2. Al finalizar la ejecución se habrá generado el fichero `.arff` con las predicciones en la misma ruta que el primer fichero de entrada.

Tabla 6: Caso de uso Clasificación.

4.2 MODELO DE DOMINIO

En la figura 7 se muestra el modelo de dominio correspondiente al software desarrollado en este trabajo. En él se indican las relaciones y dependencias existentes entre las clases del software.

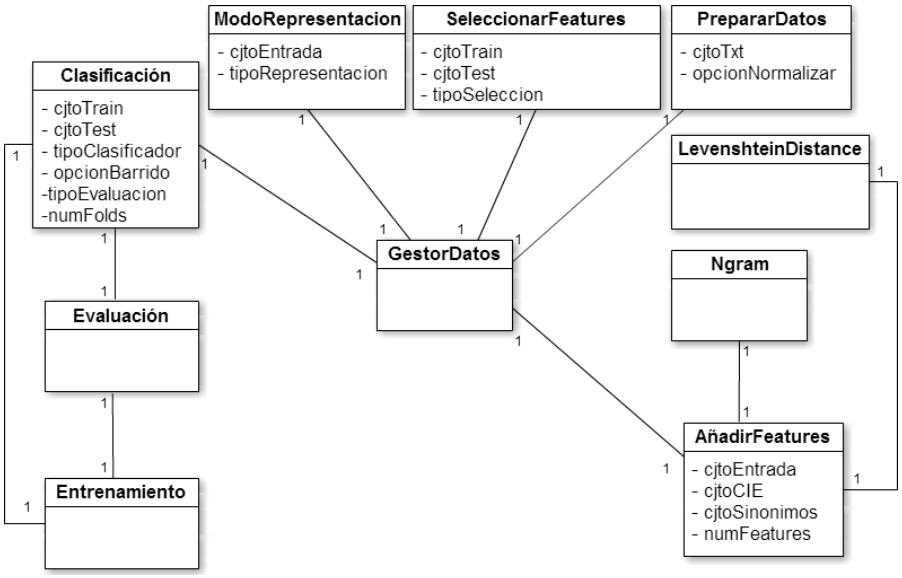


Figura 7: Modelo de dominio.

Se observa que casi todas las clases del diagrama están unidas por una relación con la clase `GestorDatos`. Ésta es una clase de servicio que contiene métodos de carga y lectura de datos y creación de ficheros. Las relaciones que unen la clase `GestorDatos` con las clases `Clasificación`, `Modo Representación`, `SeleccionarFeatures`, `PreparарDatos` y `AñadirFeatures` se corresponden con la utilización de los métodos mencionados por parte de estas clases. Por otra parte, las relaciones que unen la clase `AñadirFeatures` con las clases `Ngram` y `LevenshteinDistance` son debidas a que la primera necesita de estas dos clases de servicio para generar dos de los atributos que añade a las instancias. Por último, las tres relaciones existentes entre `Clasificación`, `Evaluación` y `Entrenamiento` indican la manera en que se desarrolla la clasificación. Se necesitan las 3 fases para realizar una clasificación completa.

5 ANÁLISIS Y DISEÑO

5.1 DIAGRAMA DE CLASES

En la figura 8 se muestra el diagrama de clases correspondiente al software desarrollado en este proyecto. Como se puede ver, está compuesto por 10 clases: `PrepararDatos`, `AnñadirFeatures`, `ModoRepresentación`, `SeleccionarFeatures`, `Clasificación`, `Entrenamiento`, `Evaluación`, `LevenshteinDistance`, `Ngram` y `GestorDatos`. Cada una de ellas está compuesta por un conjunto de atributos que le caracteriza y un conjunto de métodos, menos `LevenshteinDistance`, `Ngram` y `GestorDatos` que son clases de servicio y únicamente contienen métodos que pueden utilizar las demás.

Las uniones entre clases indican utilización de una sobre la otra. Es decir, la clase `AñadirFeatures` utiliza métodos de las clases de servicio `LevenshteinDistance` y `Ngram`, la clase `Clasificación` utiliza métodos de las clases `Entrenamiento` y `Evaluación`, y las clases `PrepararDatos`, `AñadirFeatures`, `ModoRepresentación`, `SeleccionarFeatures` y `Clasificación` utilizan métodos de la clase de servicio `GestorDatos`.

Debido a su tamaño, las figuras presentes en éste capítulo no se visualizan con total claridad. Por tanto, se encuentran todas ellas disponibles online en http://goo.gl/DciEPz dentro de la carpeta "Capítulo 5".

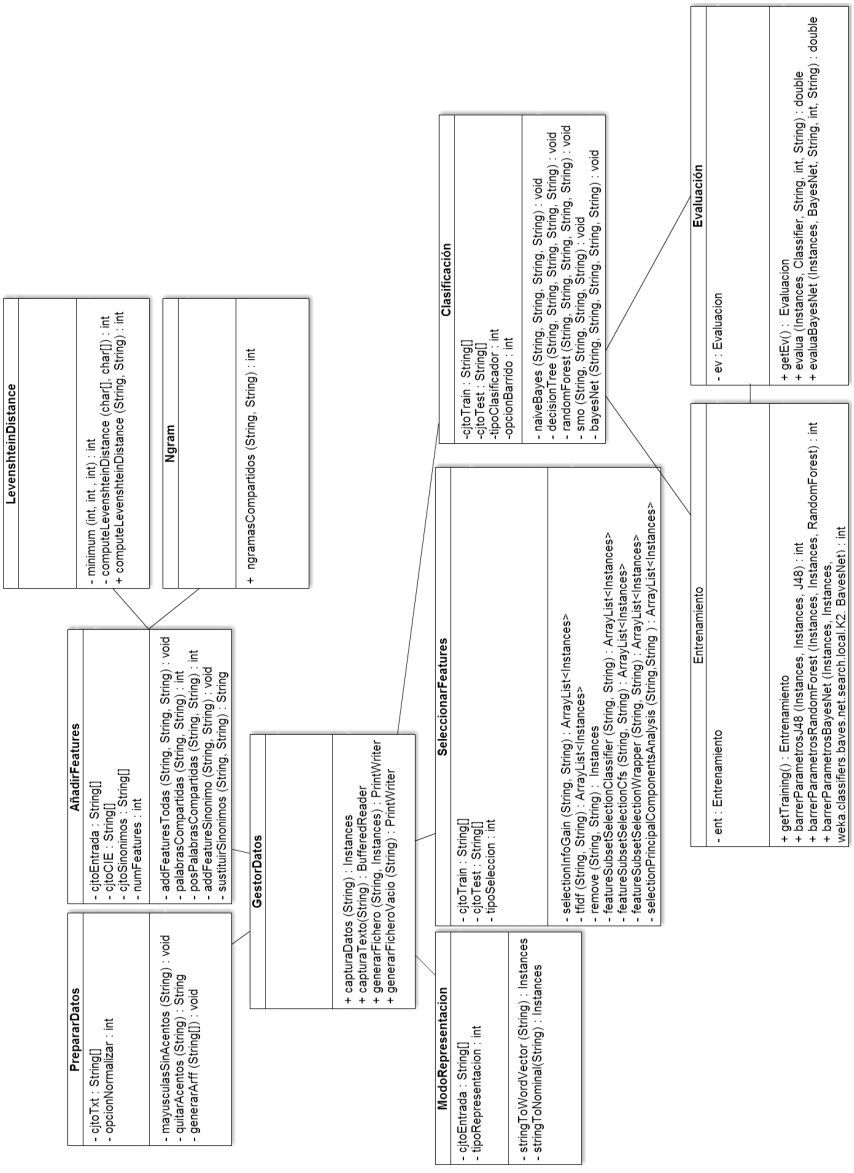


Figura 8: Diagrama de clases.

5.2 DIAGRAMAS DE SECUENCIA

En esta sección presentaremos los diagramas de secuencia correspondientes a cada uno de los cinco módulos de software desarrollados. Cada uno de estos módulos, Preparar Datos, Añadir Features, Modo Representación, Seleccionar Features y Clasificación, se corresponde con el caso de uso del mismo nombre presentado en el capítulo 4.

En las figuras 9, 10, 11, 12 y 13 se muestran los diagramas correspondientes seguidos de una explicación.

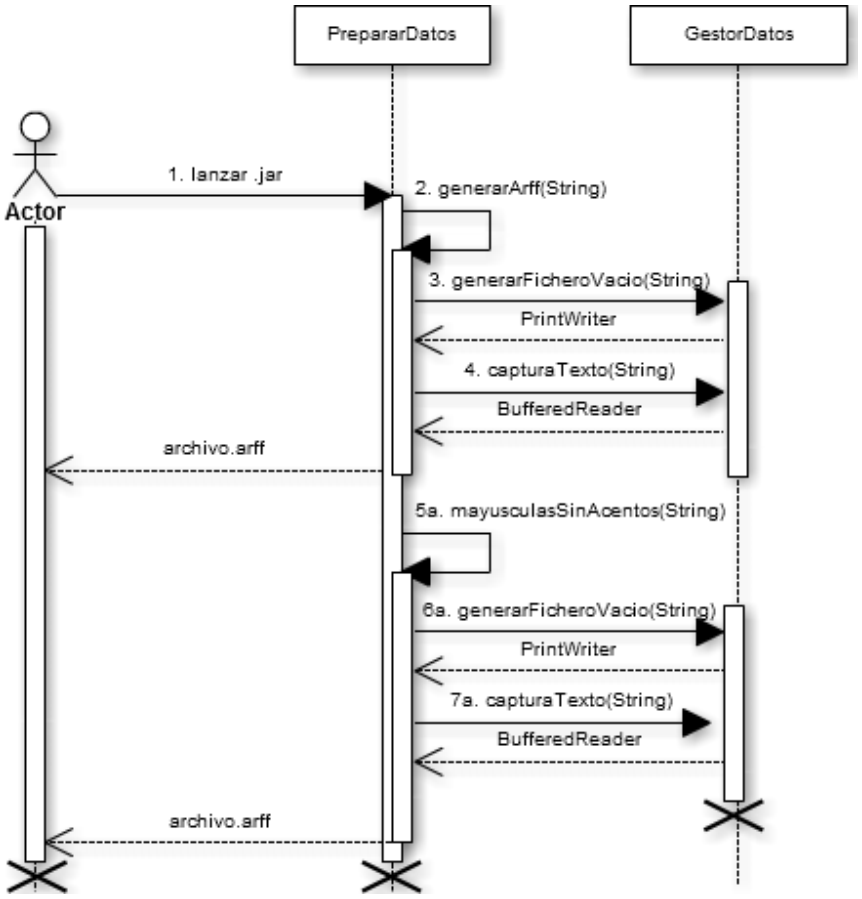


Figura 9: Diagrama de secuencia Preparar Datos.

Este módulo genera un fichero en formato `.arff`, a partir de uno en formato `.txt`. Si se quiere además devuelve otro `.arff` normalizado, es decir, con todo el texto en mayúsculas y sin acentos. Para ello, se lanza un ejecutable `PrepararDatos.jar` en el que se realizan llamadas a métodos de la clase `PrepararDatos`; `GenerarArff` y `mayusculasSinAcentos`. El primero genera el fichero `.arff` sin normalizar y el segundo lo normaliza y genera otro fichero. Estos dos métodos internamente realizan llamadas a los métodos `generarFicheroVacio` y `capturaTexto` de la clase de servicio `GestorDatos`. El método `generarFicheroVacio`, genera un fichero en el que el método llamante escribirá el `.arff` generado, y el método `capturaTexto` proporciona una variable de tipo `BufferedReader` que permite al método llamante leer un fichero, el `.txt` en el caso de `GenerarArff` y el `.arff` generado por éste, en el caso de `mayusculasSinAcentos`. Para obtener más información sobre la generación del fichero `.arff` y sobre la normalización se pueden consultar las secciones 7.1.1 y 7.1.2 respectivamente, en el capítulo 7 sobre el Preproceso.

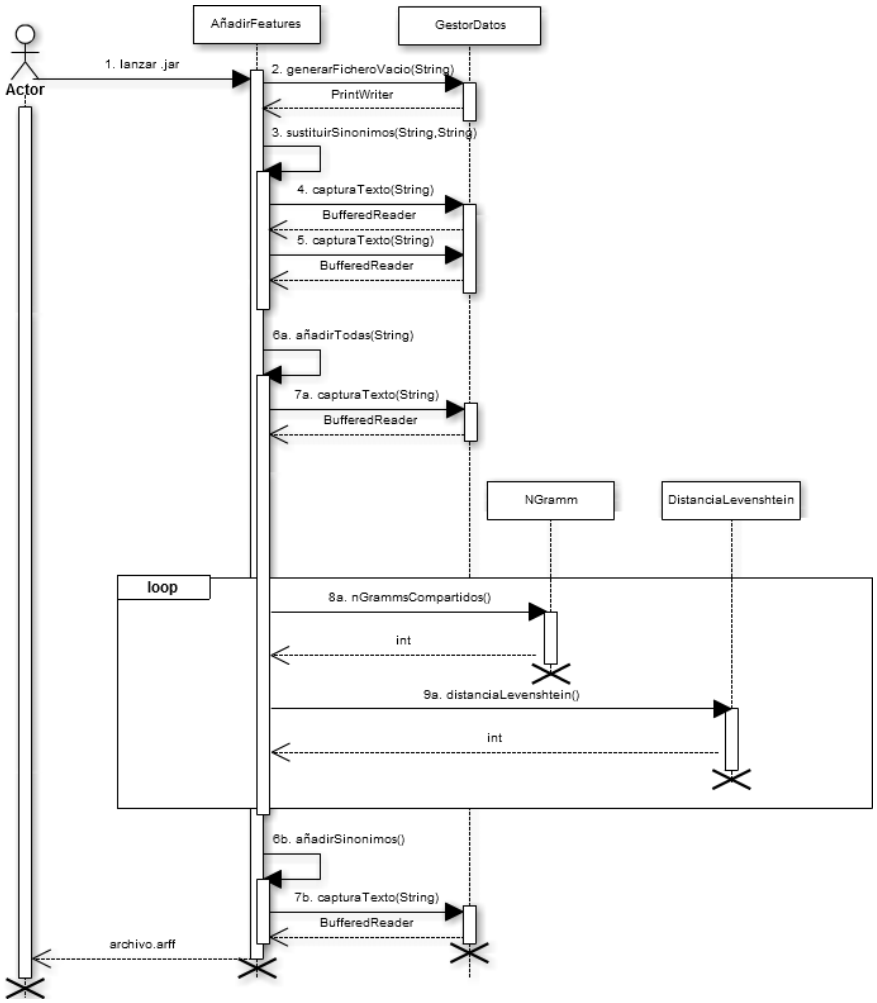


Figura 10: Diagrama de secuencia Añadir Features.

Este módulo genera un fichero en formato `.arff` a partir de otro en este mismo formato, con nuevos atributos. Se puede añadir sólo la *feature* "sinónimos" o todas las disponibles en el módulo. Para ello, se lanza un ejecutable `AñadirFeatures.jar` en el que se realizan llamadas a métodos de la clase `AñadirFeatures`. `sustituirSinónimos` para generar el atributo "sinónimos", y `añadirTodas` o `añadirSinonimos` para añadir las *features* seleccionadas. Además, estos métodos llaman internamente a otros métodos de la clase `GestorDatos`; `generarFicheroVacio` y `capturaTexto`, y a dos métodos de las clases `NGramm` y `DistanciaLevenshtein`; `nGrammsCompartidos` y `distanciaLevenshtein` respectivamente. Con la llamada a estos métodos se consiguen las *features* "*Ngramms* compartidos" y "distancia de Levenshtein". Para obtener más información sobre los atributos se puede consultar la sección 7.1.4.

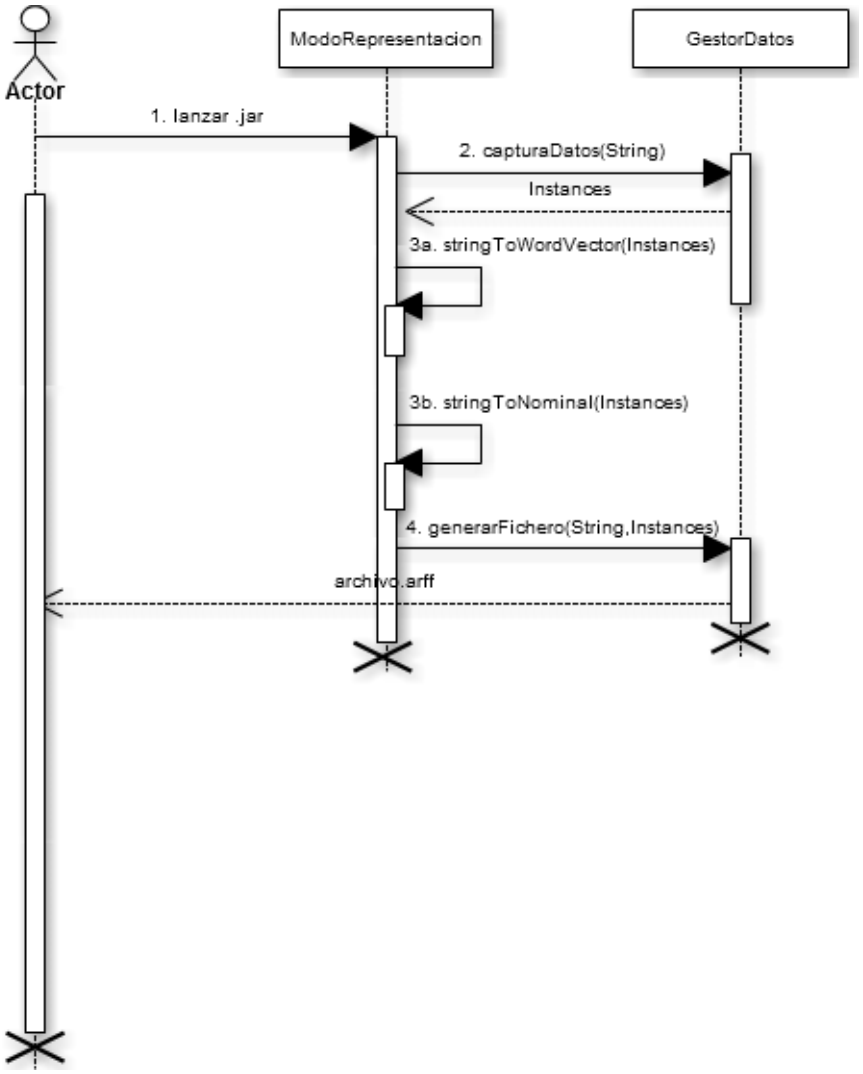


Figura 11: Diagrama de secuencia Modo Representación.

Este módulo genera un fichero en formato `.arff`, a partir de otro en este mismo formato, con los atributos representados en forma de vector o en forma nominal. Para ello, se lanza un ejecutable `ModoRepresentacion.jar` en el que se realiza una llamada a un método de la clase `ModoRepresentacion`; `stringToWordVector` o `stringToNominal` según que representación se haya elegido. Además, antes de llamar a cualquiera de los dos métodos mencionados, se realiza una llamada al método `capturaDatos` de la clase `GestorDatos` para cargar las instancias que se desea representar en una variable de tipo `Instances`. Antes de finalizar la ejecución, se hace una llamada al método `generarFichero` de la clase `GestorDatos`. Éste genera un fichero escribiendo las instancias tal y como las ha representado el método escogido, `stringToWordVector` o `stringToNominal`. Para obtener más información sobre los modos de representación se puede consultar la sección 7.1.3.

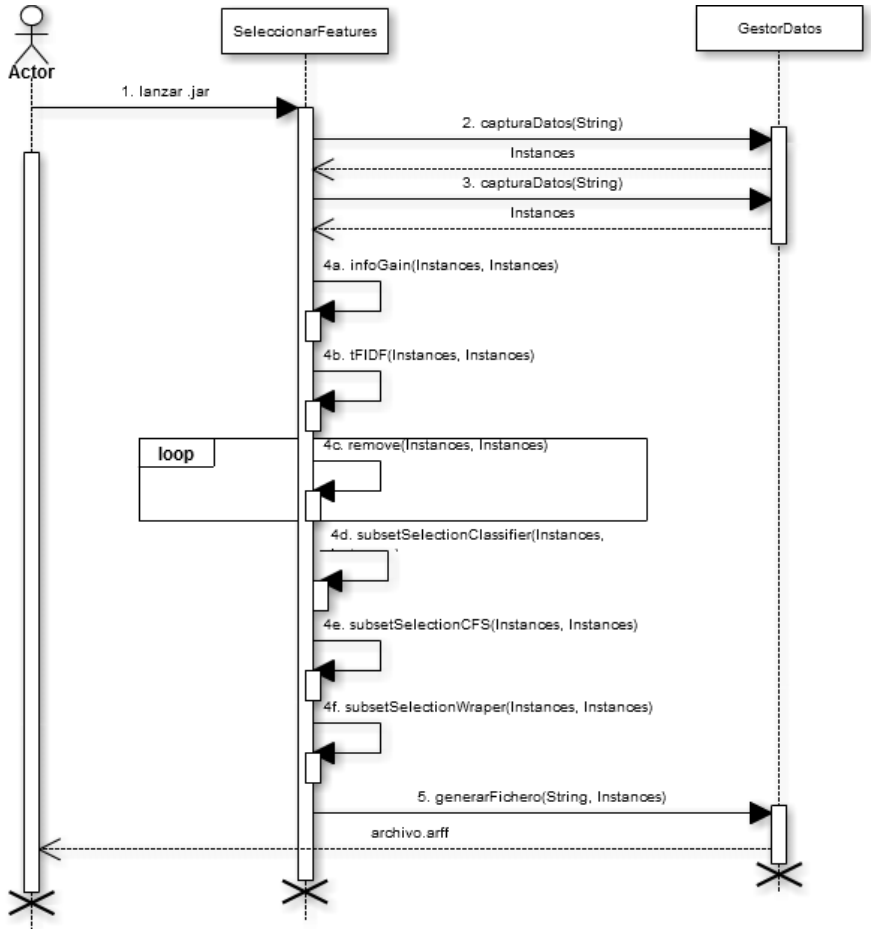


Figura 12: Diagrama de secuencia Seleccionar Features.

Este módulo genera dos ficheros en formato `.arff`, a partir de otros dos en este mismo formato (el correspondiente a entrenamiento y el correspondiente a evaluación), con los atributos resultantes tras aplicar el filtro de selección elegido por el usuario. Para ello, se lanza el ejecutable `SeleccionarFeatures.jar` en el que se realizan en primer lugar, dos llamadas al método `capturaDatos` de la clase `GestorDatos` para poder leer los ficheros `.arff`. A continuación, se realiza una llamada al método de selección de atributos escogido, de la clase `SeleccionarFeatures`. Finalmente, se realizan dos llamadas al método `generarFichero` de la clase `GestorDatos`, para generar los ficheros `.arff` con los atributos seleccionados. Para obtener más información sobre estos métodos de selección de atributos se puede consultar la sección 7.1.5.

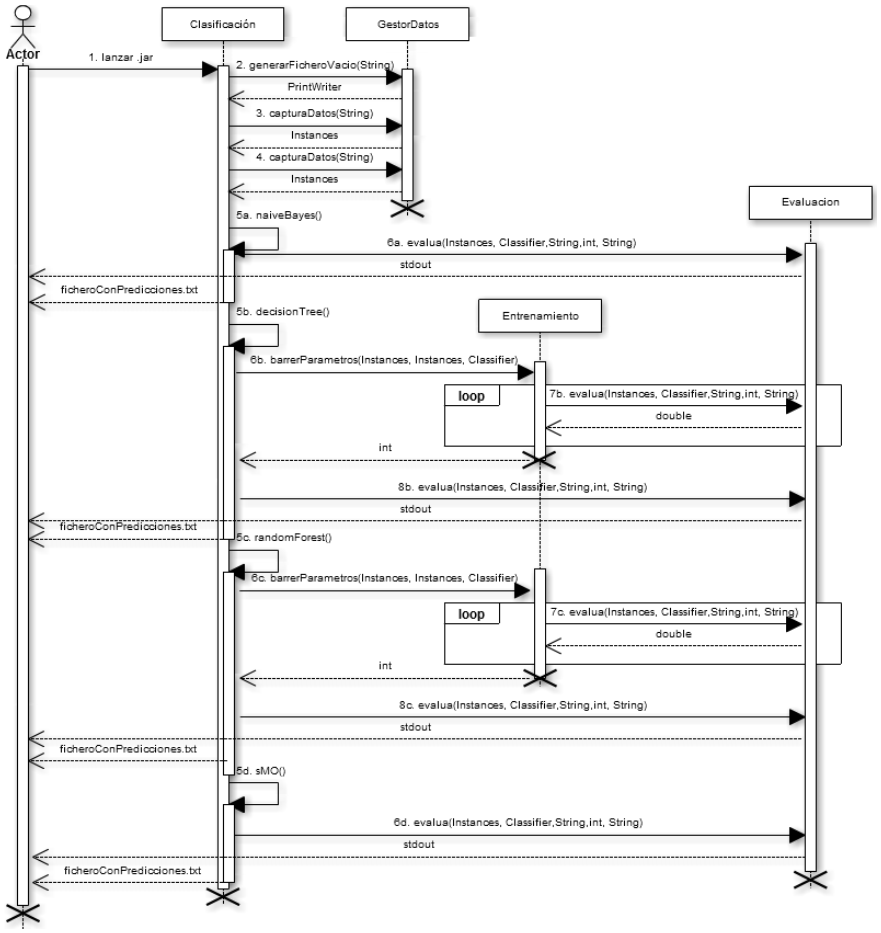


Figura 13: Diagrama de secuencia Clasificación.

Este módulo, tiene como entrada dos archivos `.arff`, el correspondiente a entrenamiento y el correspondiente a evaluación. Genera un fichero en formato `.txt` con las clases reales, comparadas con las propuestas por el clasificador seleccionado por el usuario. Además genera una salida por consola con las figuras de mérito resultantes. Para ello, se realizan llamadas a los métodos `capturaDatos` y `generarFichero` de la clase `GestorDatos`, además de la llamada al método del clasificador escogido por el usuario, de la clase `Clasificación`. Los métodos correspondientes a los clasificadores realizan internamente llamadas a los métodos `barrerParametros` y `evalua` de las clases `Entrenamiento` y `Evaluación` respectivamente. Como su nombre indica, estos métodos realizan el barrido de parámetros para encontrar el óptimo y la evaluación por *Holdout* o evaluación cruzada según haya elegido el usuario. Para obtener más información sobre los clasificadores así como sobre las figuras de mérito, la evaluación o el barrido de parametros, se puede consultar el capítulo 8 dedicado a la clasificación.

Módulo de *PreparарDatos*, figura 9.

- 1. El usuario lanza el archivo ejecutable.
- 2. Llamada al método `generarArff()` que devolverá un archivo en formato `.arff`.
- 3. `generarArff()` llama a `generarFicheroVacio(String)`.
- 4. `generarArff()` llama a `capturaTexto(String)`.
- Si $args[1] = 1$:
  - 5a. Llamada a `mayusculasSinAcentos()`.
  - 6a. Llamada a `generarFichero()`.
  - 7a. Llamada a `capturaTexto()`.

Módulo de *AñadirFeatures*, figura 10.

- 1. El usuario lanza el archivo ejecutable.
- 2. El main llama a `generarFicheroVacio(String)`.
- 3. El main llama al método `sustituirSinonimos(String, String)`.
- 4. El método `sustituirSinonimos(String, String)` llama a `capturaTexto(String)`.
- 5. El método `sustituirSinonimos(String, String)` llama a `capturaTexto(String)`.
- Si $args[3] = 0$:
  - 6a. Llamada a `añadirTodas(String)`.
  - 7a. Llamada a `capturaTexto(String)`.
  - Por cada instancia:
  - 8a. Llamada a `nGrammsCompartidos()`.
  - 9a. Llamada a `distanciaLevenshtein()`.
- Si $args[3] = 1$:
  - 6b. Llamada a `añadirSinonimos()`.
  - 7b. El main llama a `capturaTexto(String)`.

Módulo de *Modo Representación*, figura 11.

- 1. El usuario lanza el archivo ejecutable.
- 2. El main llama a `capturaDatos()`.
- Si $args[1] = 0$:
  - 3a. Llamada a `stringToWordVector(Instances)`.
- Si $args[1] = 1$:
  - 3b. Llamada a `stringToNominal(Instances)`.
- 4 . El main llama a *generarFichero(String, Instances)*.

Módulo de *Seleccionar Features*, figura 12.

- 1. El usuario lanza el archivo ejecutable.
- 2. El main llama a `capturaDatos()`.

- 3. El main llama a `capturaDatos()`.
- Si $args[2] = 0$:
  - 4a. Llamada a `infoGain(Instances, Instances)`.
- Si $args[2] = 1$:
  - 4b. Llamada a `tfidf(Instances, Instances)`.
- Si $args[2] = 2$:
  - Por cada atributo:
  - 4c. Llamada a `remove(Instances, Instances)`.
- Si $args[2] = 3$:
  - 4d. Llamada a `subsetSelectionClassifier(Instances, Instances)`.
- Si $args[2] = 4$:
  - 4e. Llamada a `subsetSelectionCFS(Instances, Instances)`.
- Si $args[2] = 5$:
  - 4f. Llamada a `subsetSelectionWraper(Instances, Instances)`.
- 5. El main llama a `generarFichero(String, Instances)`.
- 6. El main llama a `generarFichero(String, Instances)`.

Módulo de *Clasificación,* figura 13.

- 1. El usuario lanza el archivo ejecutable.
- 2. El main llama a `generarFicheroVacio(String)`.
- 3. El main llama a `capturaDatos(String)`.
- 4. El main llama a `capturaDatos(String)`.
- Si $args[2] = 0$:
  - 5a. Llamada a `naiveBayes()`.
  - 6a. Llamada a `evalua(Instances, Classifier, String, Int, String)`.
- Si $args[2] = 1$:
  - 5b. Llamada a `decisionTree()`.
  - 6b. Llamada a `barrerParametros(Instances, Instances, Classifier)`.
  - Para cada parámetro del barrido:
    - 7b. Llamada a `evalua(Instances, Classifier, String, Int, String)`.
  - 8b. Llamada a `evalua(Instances, Classifier, String, Int, String)`.
- Si $args[2] = 2$:
  - 5c. Llamada a `randomForest()`.
  - 6c. Llamada a `barrerParametros(Instances, Instances, Classifier)`.
  - Para cada parámetro del barrido:

    - 7c. Llamada a `evalua(Instances, Classifier, String, Int, String)`.
  - 8c. Llamada a `evalua(Instances, Classifier, String, Int, String)`.
- Si $args[3] = 3$:
  - 5d. Llamada a `sMO()`.
  - 6d. Llamada a `evalua(Instances, Classifier, String, Int, String)`.

6 DESARROLLO: DATOS

En este capítulo se analizan los datos disponibles para la experimentación. En primer lugar se presenta el sistema de clasificación internacional ICD-9-CM, a continuación se analizan las muestras disponibles y su estructura, para terminar con una descripción del volumen de los conjuntos de datos.

6.1 ICD-9-CM

La Clasificación Internacional de Enfermedades y Problemas Relacionados con la Salud (ICD) proporciona códigos médicos para clasificar enfermedades, así como una gran variedad de signos, síntomas, descubrimientos anormales, quejas, circunstancias y causas externas de éstas.

Existe una página web del ministerio (http://eciemaps.mspsi.es/) en la que está disponible la clasificación ICD-9-CM completa, en la figura 14 se muestra un extracto de ésta correspondiente al código 401. Cada código está asociado a una descripción. Además se observa que cada uno consta de dos partes separadas por un punto, la clase principal y el modificador no esencial. Es decir, el término general al que hace referencia el código y el término indicado con un mayor grado de especificación.

401 Hipertensión esencial
401.0 Maligna
401.1 Benigna
401.9 No especificada

Figura 14: Extracto de la clasificación ICD-9-CM

Las distintas ediciones del ICD son publicadas por la Organización Mundial de la Salud (OMS). Aunque la última versión es ICD-10, en muchos países se sigue utilizando la 9 para diversos usos, como es por ejemplo el caso de la red hospitalaria asociada al Ministerio de Sanidad, Servicios Sociales e Igualdad español, o de este proyecto en el que se utiliza el ICD-9-CM para clasificar diagnósticos clínicos.

El ICD no sólo se utiliza para clasificar diagnósticos clínicos como se ha mencionado, también se clasifican otro tipo de documentos sanitarios como pueden ser los certificados de defunción. Esta clasificación se realiza con el objetivo de facilitar el almacenamiento y posterior recuperación de datos sanitarios con fines epidemiológicos o de calidad clínica. Además el almacenamiento de estos registros proporciona la base para la realización de estadísticas, como la tasa de mortalidad, entre los miembros de la Organización Mundial de la Salud. La información proporcionada por dichas estadísticas, ayuda a los países en la toma de decisiones sobre la asignación de recursos. [WHO(2014)] En algunos países estos datos también los utilizan las aseguradoras para decidir si pagar o no un tratamiento médico. Junto

con la factura expedida por el hospital o el médico que trata al paciente, las aseguradoras exijen el diagnóstico expresado según el ICD-9-CM para determinar si el paciente necesita el tratamiento indicado en la factura.

6.2 DESCRIPCIÓN DE LAS MUESTRAS

El conjunto de datos utilizado para la realización de este proyecto fué proporcionado por el Hospital de Galdakao. Se trata de datos sobre pacientes reales y por tanto se eliminó con anterioridad toda la información de caracter personal debido a temas de confidencialidad. Se anonimizaron totalmente los textos en el Hospital de Galdakao antes de entregárnoslos. Se parte de varios documentos en formato texto ASCII cada uno de ellos con una lista de enfermedades o patologías y su clase asociada. En la figura 15 se muestra un extracto de uno de los documentos como muestra del formato en que se presentan, y en la figura 16 se muestra el estándar para los mismos códigos. Un extracto del documento del que disponemos con la clasificación estándar.

```
{Hipertensión arterial}:{4019}
{nevus clínicamente atípico}:{2169}
{GLAUCOMA NORMOTENSIVO}:{3659}
{GLAUCOMA CRONICO ANGULO ABIERTO}:{36511}
{Sospecha de Ca. de próstata.}:{185}
{E.R.G.E.}:{53081}
```

Figura 15: Extracto de los documentos de partida.

```
{Hipertensión no especificado de otra manera}:{401.9}
{Neoplasia benigna piel no especificado de otra manera}:{216.9}
{glaucoma no especificado de otra manera}:{365.9}
{glaucoma de angulo abierto primario}:{365.11}
{Neoplasia maligna de la próstata'}:{185}
{Reflujo esofágico}:{530.81}
```

Figura 16: Extracto del estándar ICD-9-CM.

Como se puede observar las descripciones asociadas a los códigos en los documentos aportados por el Hospital de Galdakao, no siempre coinciden con las descripciones del estándar ICD-9-CM. Además, no siguen ninguna regla de normalización de la escritura, es decir, algunos elementos se presentan en minúsculas, otros en mayúsculas y otros combinando ambas. En ocasiones también se encuentran abreviaturas, acrónimos e incluso faltas de ortografía. Ésto se debe a que son textos escritos por médicos mientras pasan consulta utilizando un lenguaje natural en un tiempo limitado.

También se observa que en las muestras proporcionadas por el hospital, los códigos están representados sin puntos. En el ICD-9-CM los códigos están compuestos por un grupo de dígitos correpondientes a un conjunto

general (clase principal), y otro grupo correspondientes a un subconjunto específico (modificador no esencial), separados por un punto.

La solución para abordar estas diferencias se trata en el capítulo 7 dedicado al preproceso de los datos.

6.3 ESTRUCTURA

Para el desarrollo de nuestro proyecto, debido a las librerías utilizadas, necesitamos convertir los textos en un formato compatible con las librerías de Weka (`.arff`) siguiendo la estructura propia de este tipo de datos. [Ian H. Witten(2011)]

Un documento `.arff` correctamente generado, sigue la siguiente estructura: en la primera línea encontramos la relación del documento escrita de la siguiente manera: @relation *relaciónArff*. La relación es como un identificador para el documento, en ocasiones se trata del nombre del mismo o del nombre seguido de los distintos filtros que se le han ido aplicando. A continuación se indican los atributos que componen las instancias del `.arff` seguidos de su correspondiente tipo de datos (*String* o *Nominal* en el caso de nuestros datos). En caso de ser un atributo nominal, no se indica el tipo en sí, se indican entre llaves los posibles valores que puede tomar dicho atributo.

Instancia. *"Ejemplo presente en los datos utilizados para entrenar y evaluar un sistema de aprendizaje automático. Una instancia está caracterizada por los valores de los atributos que indican distintos aspectos de ella"*.
[Ian H. Witten(2011)]

Atributos. *" Conjunto de elementos predefinido cuyos valores caracterizan una instancia. Se le llama atributo, característica o feature"*.
[Ian H. Witten(2011)]

String. *"Tipo de datos consistente en una lista de caracteres de longitud arbitraria. Los String se almacenan internamente en una tabla y se identifican por su dirección en dicha tabla. Por ello, dos String que contengan los mismos caracteres tendrán el mismo valor"*.
[Ian H. Witten(2011)]

Nominal.*" Tipo de datos consistente en una lista de valores finitos fijados en la declaración del atributo. También se les llama categóricos o enumerados"*.
[Ian H. Witten(2011)]

En la figura 17 se muestra la declaración de atributos en un fichero `.arff` para un atributo de tipo *String* y para un atributo de tipo *Nominal*.

@attribute *nombreAtributo1* String
@attribute *nombreAtributo2* {*valor1*, *valor2*, *valor3*,...}

Figura 17: Declaración de atributos.

A continuación se indica mediante la palabra reservada @data, que lo que aparece en líneas posteriores son las instancias. A partir de la siguiente, se

indica cada instancia en una línea siguiendo la estructura mostrada en la figura 18.

@data
Atributo1, Atributo2, Atributo3...,Clase

Figura 18: Representación de una instancia.

Teniendo en cuenta que los atributos de tipo *String* aparecerán entre comillas simples, y siguiendo con el ejemplo mostrado en la figura 15, veamos en la figura 19 la estructura que siguen los documentos que hemos utilizado una vez convertidos a `.arff`. Cada instancia de la figura está formada por un atributo de tipo *String* y la clase asociada, por ejemplo, "Hipertensión arterial" y "4019" respectivamente en el caso de la primera instancia.

@relation corpus

@attribute text String
@attribute clase *4019, 2169, 3659, 36511, 185, 53081, ...*

@data
'Hipertensión arterial',4019
'nevus clínicamente atípico',2169
'GLAUCOMA NORMOTENSIVO',3659
'GLAUCOMA CRONICO ANGULO ABIERTO',36511
'Sospecha de Ca. prostata.',185
'E.R.G.E.',53081
...

Figura 19: Estructura de un documento `.arff`.

6.4 VOLUMEN DEL CORPUS

El corpus disponible para el desarrollo de este proyecto está formado por un conjunto de muestras reales escritas por médicos y un conjunto de muestras estándar del ICD-9-CM.

Las muestras reales se encuentran divididas en 3 conjuntos de datos. Un conjunto de entrenamiento (*train*), un conjunto de desarrollo (*dev*) para realizar los primeros testeos y un conjunto de test. Cada uno está compuesto por un número distinto de muestras siendo más extenso el de entrenamiento con 23130 disponibles, mientras que los de testeo (*test y dev*) contienen 2850 muestras cada uno. En la tabla 7 se muestran algunos datos sobre éstas muestras.

Conjunto de datos	Num. instancias	Num. instancias repetidas	Diagnósticos distintos	Clases distintas
Train	23130	14973	7932	1579
Dev	2850	949	1873	678
Test	2850	920	1897	702

Tabla 7: Datos del corpus disponible.

En el capítulo 7 dedicado al preproceso, en la página 76, se muestra la tabla 8 en la que se exponen los mismos datos sobre el corpus una vez preprocesado.

Las muestras estándar del ICD-9-CM se encuentran procesadas, de manera que disponemos de un documento con enfermedades y patologías asociadas a los códigos estandar del ICD-9-CM, siguiendo el mismo formato que los documentos de las muestras reales.

7 DESARROLLO: PREPROCESO

Es de gran importancia en un proyecto de estas características, tener en cuenta la necesidad de un buen preproceso. Esta fase engloba todo el tratamiento que se realiza sobre los datos de entrada antes de realizar la clasificación. Un correcto tratamiento y preparación de los datos utilizando las técnicas adecuadas, será de gran relevancia en la obtención de unos buenos resultados finales. En el preproceso de este proyecto en primer lugar se transforman los ficheros de entrada al formato adecuado para los clasificadores, se convierten los ficheros `.txt` en ficheros `.arff`. A continuación, se normalizan los datos de manera que los clasificadores consideren iguales una misma palabra escrita en mayúsculas o minúsculas, con tildes o sin tildes. Este paso facilita enormemente la tarea de clasificación suponiendo una gran mejora en los resultados. Además se experimenta con distintas técnicas de representación de textos buscando el más adecuado para los datos disponibles y los clasificadores utilizados. Se generan nuevos atributos que aporten más información para facilitar la clasificación, y posteriormente, se aplican distintas técnicas de selección de atributos para detectar los más informativos y relevantes. Sin esta fase de preproceso no sería viable abordar la tarea de la clasificación. Utilizar los datos originales supondría unos resultados muy por debajo de los obtenidos con unos datos preprocesados.

7.1 MARCO TEÓRICO

7.1.1 Preparación de los datos

La primera tarea que se debe abordar dentro del preproceso es la preparación y adecuación de los datos para el desarrollo de este trabajo. Como se explica en el capítulo 6 sobre los datos, disponemos de un conjunto de ficheros en formato `.txt` que necesitamos transformar a formato `.arff`. Para realizar esta transformación correctamente, lo primero que se debe saber es de qué atributos se dispone y de qué tipo son éstos. En la figura 20 se muestra la estructura que presentan los datos originales.

```
{Hipertensión arterial}:{4019}
{nevus clínicamente atípico}:{2169}
{GLAUCOMA NORMOTENSIVO}:{3659}
```

Figura 20: Estructura original de los datos.

Se pueden observar claramente los dos atributos presentes, por una parte el diagnóstico y por otra parte el código asociado. Este último será la clase utilizada para realizar la clasificación, siendo la clase un atributo especial que se indica de tipo nominal. Para declarar un atributo de tipo nominal, a continuación del nombre se indican los valores que puede tomar. El atributo correspondiente al diagnóstico es una cadena de caracteres y por tanto lo

indicaremos de tipo String. Sabiendo ésto ya se puede generar la cabecera necesaria para un fichero de tipo `.arff`, ésta se muestra en la figura 21.

```
@relation corpus
@attribute text String
@attribute clase {4019, 2169, 3659, 78861...}
```

Figura 21: Cabecera de un fichero `.arff`.

A continuación, se deben indicar las instancias que formarán el fichero. Las instancias son los datos en sí, es decir, cada línea del fichero original es una instancia. Cada instancia estará compuesta por los atributos indicados en la cabecera separados por una coma. En la figura 22 se muestra un extracto de esta parte de un fichero `.arff`.

```
@data
'Hipertensión arterial',4019
'nevus clínicamente atípico',2169
'GLAUCOMA NORMOTENSIVO',3659
```

Figura 22: Cuerpo de un fichero `.arff`.

Además, para completar esta fase de preparación de los datos, se debe tener en cuenta que los algoritmos de clasificación automática empleados en este trabajo, explicados en el capítulo 8 dedicado a la clasificación, requieren que el conjunto de datos utilizado para entrenar y el conjunto de datos utilizado para evaluar tengan cabeceras comunes. Los posibles valores indicados para la clase deben englobar tanto los que aparecen en el conjunto de *train* como los que aparecen en el conjunto de test. Para conseguirlo, se lleva a cabo un proceso de unificación de cabeceras. En dicho proceso se generan dos ficheros `.arff` iguales a los que recibe como entrada, exceptuando sus cabeceras. Ésta se forma a partir de la unión de las cabeceras de los dos ficheros de entrada. Se seleccionan todas las clases de cada una de ellas sin repetir las comunes. En la figura 25 se muestra como quedaría la cabecera de un fichero tras la unión de las cabeceras mostradas en las figuras 23 y 24. Para este proceso de unificación, se ha utilizado un software ya desarrollado aportado por nuestra compañera [Santiso(2014)] quien realizó su TFG en el mismo ámbito que el nuestro y colaborando con el mismo grupo de investigación.

Cabecera 1.

```
@relation corpus
@attribute text String
@attribute clase {4019, 2169, 3659, 78861}
```

Figura 23: Cabecera del fichero1.arff.

Cabecera 2.
@relation corpus
@attribute text String
@attribute clase {4020, 365, 2169, 3658, 78861}

Figura 24: Cabecera del fichero2.arff.

Cabecera común.
@relation corpus
@attribute text String
@attribute clase {4019, 2169, 3659, 78861, 4020, 365, 3658}

Figura 25: Unificación de cabeceras.

7.1.2 Normalización del texto

Desde un primer momento se observó que el corpus disponible para la realización de este proyecto no estaba escrito de una manera estandarizada. Se trata de extractos de textos clínicos redactados por distintos médicos mientras pasan consulta. Se decidió abordar este aspecto como un primer paso en el tratamiento de los datos aplicando una serie de normalizaciones al corpus. Se eliminaron las tildes, se pasaron todas las palabras a mayúsculas y se decidió ignorar los signos de puntuación. Además, se igualaron los códigos de las clases presentes en los textos de entrenamiento y evaluación con las presentes en el documento disponible del estándar ICD-9-CM. Los códigos estándar están divididos por un punto intermedio separando la clase principal y el modificador no esencial, mientras que los médicos lo escriben seguido prescindiendo de los puntos. Para igualarlos eliminamos los puntos presentes en los estándar. Por ejemplo, un código que en el ICD-9-CM aparece como 293.82 , siendo 293 el código correspondiente a la clase principal “Delirios transitorios debidos a enfermedades clasificadas en otro lugar", 293.8 “Otros trastornos mentales transitorios especificados debidos a enfermedades clasificadas en otro lugar” y 293.82 “ Trastorno psicótico con alucinaciones en enfermedades clasificadas en otro lugar”, en los textos escritos por médicos aparecería como 29382. Por este motivo se decidió normalizar también los códigos correspondientes a la clase eliminando los puntos en los códigos estándar. En la figura 26 se muestra algún ejemplo de la normalización de los textos mediante las mencionadas técnicas.

Eliminación de tildes y paso a mayúsculas.
‘Hipertensión arterial’,4019
‘HIPERTENSION ARTERIAL’,4019
Eliminación del punto divisor de la clase estándar
‘Cólera debida a vibrio cholerae’,001.0
‘COLERA DEBIDA A VIBRIO CHOLERAE’,0010

Figura 26: Normalización de los datos.

Como se expone más adelante en la sección "Marco Experimental" de este mismo capítulo 7.2.1, esta fase de normalización del texto nos ofreció una gran mejora en los resultados obtenidos.

7.1.3 Representación de los datos

Al comienzo del desarrollo de un proyecto se dispone de un conjunto de datos para la realización del mismo que no se encuentra en el formato o estructura más adecuado para abordar la tarea y para la utilización de las herramientas necesarias. Es en este momento cuando se debe escoger la manera de representación que mejor se prevea que se vaya a adaptar a las necesidades del trabajo, pensando en los problemas y dificultades que puedan surgir. Pueden aparecer problemas de incompatibilidad de formato o de incremento en el coste computacional entre otros. En este proyecto se ha optado por experimentar con dos representaciones distintas. Por un lado se ha probado una técnica de representación de textos en forma de vector, y por otro lado se ha probado una representación del texto transformando sus atributos de tipo *String* a nominal.

7.1.3.1 *Representación de textos en vector*

El corpus disponible para la realización de este proyecto se compone de un conjunto de diagnósticos con sus clases asociadas en formato de texto plano. Tal y como se muestra en la figura 27, las instancias están formadas por dos atributos, uno de tipo *String* y el atributo clase de tipo nominal.

'HIPERTENSION ARTERIAL',4019

'HIPERTENSION ARTERIAL' - Atributo tipo String
4019 - Atributo clase tipo nominal

Figura 27: Atributos de una instancia.

Existen diversas técnicas de representación de textos para convertirlos a forma vectorial. En nuestro proyecto hemos decidido utilizar la técnica **Bag Of Words** (BOW). El *Bag Of Words* es una técnica de representación numérica de textos que solo tiene en cuenta la presencia o ausencia de palabras ignorando el orden de éstas dentro del texto. Como su propio nombre indica, "Bolsa de palabras", se trata de juntar todas las palabras que componen el texto e indicar con una variable la presencia o ausencia de cada una. Por ejemplo, si tenemos el texto "DOLOR TORACICO A ESTUDIO" lo primero que hace esta técnica es juntar todas ellas en una "bolsa" para a continuación representarlas numéricamente ignorando el orden inicial. Asigna a cada una un número de atributo y la representa de la siguiente manera: {0 1,2 1,3 1,1 1}, representando el primer número de cada par, el número de atributo al que hace referencia, y el segundo la presencia de cada uno en el texto, indicando el número de apariciones.

Como se indica en el capítulo 3 dedicado a los antecedentes, *Bag Of Words* se ha utilizado antes en otros trabajos similares al nuestro, como por ejemplo en el de [Zhang & Patrick(2008)] o en el de [Zhang(2006)].

Tras aplicar esta técnica ya no se dispone del texto con una estructura sintáctica, se tiene una lista de palabras que aparecen en él pero no se conoce el

orden. Imaginemos que tenemos la instancia "GLAUCOMA CRONICO ANGULO ABIERTO", al aplicarle *Bag Of Words* las introducimos en una bolsa y al sacarlas perderemos el orden en que éstas aparecen. Serán equivalentes "GLAUCOMA CRONICO ANGULO ABIERTO", "CRONICO ABIERTO GLAUCOMA ANGULO" y cualquier otra combinación de estas cuatro palabras. La información que se mantiene tras la aplicación es la presencia de dichas palabras en la instancia. En nuestro caso disponemos de un conjunto de instancias compuestas por un atributo de tipo *String* al que se ha llamado "text", y la clase asociada, y queremos aplicar la técnica *Bag Of Words* al atributo de tipo *String*.

Estando el software programado en Java y utilizando librerías de Weka [Ian H. Witten(2011)], para utilizar esta técnica se aplica el filtro `weka.filters.unsupervised.attribute.StringToWordVector`. A continuación se describen parámetros relevantes que sirven para especificar el comportamiento del filtro:

`wordsToKeep`, restringe las palabras utilizadas del texto, quedando así restringidas también las dimensiones del vector. Los sufijos, como por ejemplo femeninos y masculinos o plurales, serían eliminados para evitar duplicidades y ciertas palabras como conjunciones o preposiciones que no aportan información relevante no se utilizan.

El parámetro `outputWordCounts` determina el contenido del vector. En caso de indicarlo a `True`, el contenido de la posición j en la instancia i es un valor entero que indica el número de veces que aparece la palabra j en la instancia i, y en caso de indicarlo a `False`, el contenido de la posición j en la instancia i es un valor `booleano` que indica si la palabra j está presente en la instancia i o no con un 1 o con un 0 respectivamente.

El parámetro `lowerCaseTokens` permite considerar como iguales las palabras escritas en mayúsculas o en minúsculas (sin reescribirlas). En nuestro caso este parámetro no es de gran utilidad debido a la regla de normalización aplicada, mencionada en el apartado 7.1.2, de diseño e implementación propia. Sin embargo, hemos fijado este parámetro a `True` por que así lo hemos creído coherente.

En este proyecto la asignación de parámetros que hemos considerado más conveniente ha sido la siguiente:
El parámetro `wordsToKeep` lo hemos establecido a 5000, un número suficientemente grande para que utilice todas las palabras. En nuestro caso no queríamos que eliminase ninguna palabra puesto que al tratarse de diagnósticos y no de textos completos, consideramos que prescindir de algunas supondría una pérdida de información importante.
El parámetro `outputWordCounts` lo hemos establecido a `True`.
El parámetro `lowerCaseTokens` lo hemos establecido a `True`.

Tras aplicar esta técnica al corpus del proyecto la primera diferencia que se observa entre el `.arff` original y el `.arff` generado es la lista de atributos. El atributo "text" de tipo *String* sobre el que hemos aplicado el filtro ya no existe, ha sido sustituido por una lista de N (número de palabras distintas presentes en el texto) atributos nominales. Se muestra en las figuras 28 y 29.

La otra diferencia, obviamente, es la forma en la que se indican las instancias. Una instancia antes representada como 'HIPERTENSION ARTERIAL',4019 tras aplicar el BOW se indica como {0 1,2 1,2494 4019}.

Para interpretar la correspondencia entre ambas hay que fijarse en que indica cada par de números. El 0 1 significa que el atributo 0 aparece 1 vez en esta instancia y el 2 1 significa que el atributo 2 aparece una vez en esta instancia. Observando la lista de atributos mostrada en la figura 29, se

```
@attribute text string
@attribute clase {0,4019,2169,3659,78861,V5849,60000,185,38910,59651...
1629,36511,6089,5409,78869,28983,1469,V5871,V7189,74330,V6759,4019...}
@data
'HIPERTENSION ARTERIAL',4019
'NEVUS CLINICAMENTE ATIPICO',2169
'GLAUCOMA NORMOTENSIVO',3659
....
'CONTROL T. VESICAL.',V6759
'HTA',4019
```

Figura 28: Atributos en formato original.

```
@attribute arterial
@attribute control
@attribute hipertension
@attribute hta
...
```

Figura 29: Atributos tras aplicar BOW.

puede comprobar que el atributo 0 es arterial y el atributo 2 es hipertensión. Con este ejemplo se puede ver también que como hemos mencionado anteriormente tras aplicar el BOW se pierde el orden de las palabras dentro del texto. El último par de números, 2494 4019, es el correspondiente a la clase. A este atributo no se le ha aplicado *Bag Of Words* pero el filtro de Weka utilizado, les añade por defecto el número de atributo correspondiente, el último en este caso. [Alpaydin(2010)]

7.1.3.2 *Representación de textos con atributos nominales*

La principal diferencia entre un atributo de tipo *String* y un atributo de tipo nominal es el conocimiento o desconocimiento de los valores que puede tomar dicho atributo. Como se muestra en la figura 30, si es de tipo nominal en la declaración del atributo se indica el nombre del mismo y un conjunto finito de valores, mientras que si es de tipo *String* se indica unicamente el nombre del atributo y el tipo.

Atributo de tipo *String*
@attribute text String

Atributo de tipo nominal
@attribute text {'HIPERTENSION ARTERIAL', 'NEVUS CLINICAMENTE ATIPICO', 'GLAUCOMA NORMOTENSIVO', 'MICCION A INTERVALOS'...}

Figura 30: Diferencia entre los tipos *String* y nominal.

Para realizar la conversión se utiliza el filtro de Weka `weka.filters.unsupervised.attribute.StringToNominal` [Ian H. Witten(2011)], al que hay que indicarle los atributos a convertir. En este caso la clase ya está en formato nominal y es el atributo "text" el que queremos convertir. Sabiendo que en nuestros `.arff` indicamos siempre la clase al final de cada instancia, el parámetro `AttributeRange` se establecerá de la siguiente manera, `setAttributeRange (''first-last-1'')`, dejando únicamente el último sin convertir, la clase en este caso. De esta manera, el atributo text expresado inicialmente como @attribute text String, trás la aplicación del filtro `StringToNominal` quedará expresado de la manera indicada en la figura 30.

7.1.4 Nuevos atributos

Teniendo en cuenta que disponemos de un documento con los códigos y descripciones estándar del ICD-9-CM y de un documento con unas listas de sinónimos correspondientes a los términos clínicos presentes en los datos, surge la idea de añadir nuevos atributos generados a partir de la información que éstos nos aportan. Es fácil imaginar que proporcionando más información al clasificador sobre cada instancia éste será capaz de ofrecer mejores resultados. A continuación se muestra una lista detallada de los atributos añadidos:

1. **Término ICD-9-CM.** Término estandarizado recogido en el ICD-9-CM correspondiente al atributo "text" de cada instancia. Para cada diagnóstico se busca en el estándar aquel que tiene la misma clase asociada y será este término el que se añada como atributo. Se muestra un ejemplo en la figura 31 .

Instancia original
'OBSTRUCCION VENA CENTRAL DE LA RETINA', 35235
Instancia estándar
'OCLUSION DE VENA RETINIANA CENTRAL', 35235
Nueva instancia
'OBSTRUCCION VENA CENTRAL DE LA RETINA', 'OCLUSION DE VENA RETINIANA CENTRAL', 35235

Figura 31: Atributo término ICD-9-CM.

2. **Sinónimos.** Se le añade a cada instancia una forma alternativa más general del diagnóstico. Los términos alternativos están recogidos en una lista de sinónimos en los que para cada término genérico propuesto se indican las palabras equivalentes. En la figura 32 se muestra la estructura de ésta.

ABDOMENALT [{ABDOMEN} | {ABDOMINAL}];
ALCOHOLICOALT [{ALCOHOLICO} | {ENOLICO} | {ETILICO}];
AMIGDALAALT [{AMIGDALA} | {AMIGDALAR}];

Figura 32: Lista de sinónimos.

En el extracto de la lista de sinónimos mostrado en la figura 32, se indica que 'ABODOMENALT' es el término alternativo para 'ABDOMEN' y para 'ABDOMINAL'. 'ALCOHOLICOALT' es el término alternativo para 'ALCOHOLICO', 'ENOLICO' y 'ETILICO'. 'AMIGDALAALT' es el término alternativo para 'AMIGDALA' y 'AMIGDALAR'.

Para cada instancia se buscan los términos que componen el diagnóstico en la lista de sinónimos y aquellos encontrados se sustituyen por su forma alternativa. En la figura 33 se muestra un ejemplo.

Instancia original
'OBSTRUCCION VENA CENTRAL DE LA RETINA', 35235
Lista de sinónimos
855ALT [{OBSTRUCCION} | {OCLUSION}];
VENAALT [{VENA} | {VENOSO }];
RETINAALT [{RETINA} | {RETINIANO}];
Nueva instancia
'OBSTRUCCION VENA CENTRAL DE LA RETINA', '855ALT VENAALT CENTRAL DE LA RETINAALT', 35235

Figura 33: Atributo sinónimos.

3. **Distancia de Levenshtein.** Se le añade a cada instancia la distancia de Levenshtein existente entre ésta y su correspondiente del ICD-9-CM .

   "*Se llama distancia de Levenshtein o distancia de edición al menor número de operaciones de edición (inserciones, eliminaciones o sustituciones) necesarias para transformar una cadena de caracteres en otra. El algoritmo utilizado para calcular esta distancia entre dos cadenas tiene un coste* O(*mxn) siendo m y n las longitudes de las cadenas*". [Black & Pieterse(2013)].

   Para calcular la distancia de Levensthein hemos utilizado un software disponible en la web [Contributors(2014)] al que pasándole dos *String* devuelve la distancia de edición entre ellos. En la figura 34 se muestra un ejemplo del proceso.

Instancia original
'PROBABLE TIROIDITIS SUBAGUDA', 2451
Instancia estándar
'TIROIDITIS SUBAGUDA', 2451
Operaciones de edición
Añadir: 1.' P', 2. 'R', 3. 'O', 4. 'B', 5. 'A', 6. 'B', 7. 'L', 8. 'E', 9. ' '
Nueva instancia
'PROBABLE TIROIDITIS SUBAGUDA', 'TIROIDITIS SUBAGUDA, 9, 2451

Figura 34: Atributo distancia de Levenshtein.

4. **Número de *N-gramms* compartidos.** Se le añade a cada instancia el número de *N-gramms* que comparte con su correspondiente del ICD-9-CM. Se divide el diagnóstico (atributo "text") de cada instancia en

N-gramms o grupos de palabras, siendo N el número de palabras en cada conjunto de estas. N va variando desde 1 hasta el número total de palabras que contiene el diagnóstico, es decir, los *N-gramms* más básicos serán cada una de las palabras por separado, y el más complejo será el diagnóstico completo. En la figura 35 se muestra un ejemplo.

Instancia original
'DIABETES MELLITUS TIPO 2', 25000
Instancia estándar
'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 25000
***Ngramms* compartidos**
1. 'DIABETES', 2. 'MELLITUS', 3. 'TIPO', 4.'DIABETES MELLITUS'
Nueva instancia
'DIABETES MELLITUS TIPO 2', 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 9, 25000

Figura 35: Atributo *N-gramms* compartidos.

A continuación se expone el proceso seguido para su obtención. En primer lugar, como se muestra en la figura 36, se debe establecer los valores que puede tomar N. Como se ha mencionado anteriormente, su rango de variación estará comprendido entre 1 y el número total de palabras que componen la cadena.

Instancia original : 'DIABETES MELLITUS TIPO 2', 25000
Número de palabras : 4

$$1 \leqslant N \leqslant 4$$

Figura 36: Rango de variación de *N*.

Una vez se ha estabecido el rango de N empezamos a descomponerlo en *Ngramms*. En las figuras 37, 38, 39 y 40, se muestra esta división para los distintos valores de N.

N **= 1 (*Unigramms*)**
Cadena original : 'DIABETES MELLITUS TIPO 2'
Unigramms : 'DIABETES', 'MELLITUS', 'TIPO', '2'

Figura 37: División en *Unigramms*.

N = 2 (***Bigramms***)
Cadena original : 'DIABETES MELLITUS TIPO 2'
Bigramms : 'DIABETES MELLITUS', 'MELLITUS TIPO', 'TIPO 2'

Figura 38: División en *Bigramms*.

N = 3 (***Trigramms***)
Cadena original : 'DIABETES MELLITUS TIPO 2'
Trigramms : 'DIABETES MELLITUS TIPO', 'MELLITUS TIPO 2'

Figura 39: División en *Trigramms*.

N = 4
Cadena original : 'DIABETES MELLITUS TIPO 2'
4gramms : 'DIABETES MELLITUS TIPO 2'

Figura 40: División en *4gramms*.

De la misma manera se repetiría el proceso para la instancia estándar. En las figuras 41, 42, 43, 44 y 45 se muestra un extracto de éste.

Instancia original : 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 25000
Número de palabras : 10

$$1 \leqslant N \leqslant 10$$

Figura 41: Rango de variación de *N*.

Una vez se ha estabecido el rango de N empezamos a descomponerlo en *Ngramms*.

N = 1 (***Unigramms***)
Cadena original : 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)'
Unigramms : 'DIABETES', 'MELLITUS', 'NO' , 'COMPLICADA', ..., TIPO, ...

Figura 42: División en *Unigramms*.

N = 2 (***Bigramms***)
Cadena original : 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)'
Bigramms : 'DIABETES MELLITUS', 'MELLITUS NO', 'NO COMPLICADA', ...

Figura 43: División en *Bigramms*.

N = 3 (***Trigramms***) **Cadena original** : 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)'
Trigramms : 'DIABETES MELLITUS NO', 'MELLITUS NO COMPLICADA', ...

Figura 44: División en *Trigramms*.

N = 4
Cadena original : 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)'
4gramms : 'DIABETES MELLITUS NO COMPLICADA', 'MELLITUS NO COMPLICADA NI', ...

Figura 45: División en *4gramms*.

Y así, se seguría descomponiendo en *N-gramms* hasta llegar a N = 10. Comparando ambas divisiones podemos ver que se comparten 4 *N-gramms*, 'DIABETES', 'MELLITUS', 'TIPO' (figuras 37 y 42) y 'DIABETES MELLITUS' (figuras 38 y 43).

5. **Número de palabras compartidas.** Se le añade a cada instancia el número de palabras que comparte con su correspondiente estándar del ICD-9-CM. En la figura 46 se muestra un ejemplo.

Instancia original
'DIABETES MELLITUS TIPO 2', 25000
Instancia estándar
'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 25000
Palabras compartidas
1. 'DIABETES', 2. 'MELLITUS', 3. 'TIPO'
Nueva instancia
'DIABETES MELLITUS TIPO 2', 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 3, 25000

Figura 46: Atributo palabras compartidas.

6. **Número de palabras compartidas en la misma posición.** Se le añade a cada instancia el número de palabras que comparte con su correspondiente del ICD-9-CM teniendo como condición además, que estas palabras ocupen la misma posición en ambas. En la figura 47 se muestra un ejemplo.

Instancia original
'DIABETES MELLITUS TIPO 2', 25000
Instancia estándar
'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 25000
Palabras compartidas en posición
1. 'DIABETES', 2. 'MELLITUS'
Nueva instancia
'DIABETES MELLITUS TIPO 2', 'DIABETES MELLITUS NO COMPLICADA NI DESCOMPENSADA TIPO II (NO INSULINO-DEPENDIENTE)', 2, 25000

Figura 47: Atributo palabras compartidas en posición.

Se puede observar que 5 de los nuevos atributos mencionados (Término del ICD-9-CM, distancia de Levenshtein, número de *N-gramms* compartidos, número de palabras compartidas y número de palabras compartidas en la misma posición) surgen a partir de la información proporcionada por el documento que recoge el estandar del ICD-9-CM. Para localizar el estándar correspondiente a cada instancia utilizamos la clase asociada. Ésto hace que los resultados obtenidos al añadir estos atributos no sean resultados realistas, son optimistas.

En fase de entrenamiento y desarrollo del proyecto disponemos de documentos para entrenar y evaluar el sistema en los que aparece la clase asociada a cada instancia, permitiéndonos compararla con la del estándar ICD-9-CM. Sin embargo, en una situación real en la que se tenga que clasificar un diagnóstico desconocido, no conoceremos la clase y por tanto no podremos generar estos atributos. Es por ésto que no conseguiremos unos resultados realistas utilizando dichos atributos.

Por tanto, tal y como se indica en el "Marco Experimental" de este mismo capítulo 7.2 se hará una distinción entre los experimentos realizados tras añadir todos los nuevos atributos generados y los realizados tras añadir únicamente el atributo sinónimos, el único del que obtendremos resultados realistas.

Se debe tener en cuenta también otra desventaja en la utilización de los 5 atributos mencionados. Durante el desarrollo de este trabajo nos hemos dado cuenta de que el documento del ICD-9-CM del que disponemos no está completo. Concretamente 348 instancias presentes en el corpus no tienen su correspondiente estándar en dicho documento (se dispone de un fichero en el que se indican éstas instancias por si alguien lo solicitase, sin embargo, por temas de confidencialidad, no lo incluímos en la memoria). Ésto provoca que al añadir los nuevos atributos se pierdan algunas instancias, ya que al no encontrar su correspondiente estándar no se pueden generar los atributos y dada la imposibilidad de que una instancia tenga menos atributos que las demás, ésta es eliminada. Una posible solución sería completar el ICD-9-CM. Sin embargo, tras analizar la situación se decide no hacerlo,

ya que como se ha indicado anteriormente, éstos atributos no aportan resultados realistas. Los experimentos que se realicen con ellos nos servirán únicamente para confirmar el correcto funcionamiento de los métodos de selección de atributos y en esto no influirá la pérdida de algunas instancias.

7.1.5 Reducción dimensional

Tal y como dice en el libro Machine Learning [Alpaydin(2010)], la complejidad de un clasificador depende del número de atributos que compongan una instancia. Existen diversos métodos gracias a los cuales podemos hacer una selección del conjunto de atributos quedándonos con los más informativos y relevantes entre todos los disponibles. De esta manera estamos disminuyendo el número de atributos con los que llegarán las instancias a los clasificadores, disminuyendo así la complejidad tanto en tiempo como en espacio.

En la mayoría de clasificadores la complejidad depende del número de dimensiones "d" de entrada (el número de atributos) y del tamaño de la muestra. Es por esto que si reducimos la dimensionalidad en una fase de preproceso, disminuimos la complejidad, reduciendo así el coste del proceso de clasificación.

Por otra parte, cuando los datos se pueden representar utilizando menos atributos sin pérdida de información se hace más fácil su representación y análisis.

7.1.5.1 *Evaluadores de atributos*

Como dice [Ian H. Witten(2011)], se distinguen dos grandes familias de Evaluadores de atributos para ayudar en la selección del conjunto óptimo. *Ranking* (*Single-Attribute Evaluator*) y Selección (*Attribute Subset Evaluator*).

- **Ranking** *"Evalúa de forma individual la calidad de cada uno de los atributos según la métrica de relevancia establecida (Chi Square, Information Gain, etc.). La métrica cuantifica la capacidad predictiva del atributo respecto de la clase. En base a esa capacidad, como resultado se obtienen los atributos ordenados por prorrateo, es decir, los primeros son los más relevantes según esa métrica".*

  Hay que tener en cuenta que estos métodos únicamente nos proporcionan un *ranking* de los atributos pero no realizan ninguna selección. Es de gran utilidad para conocer los atributos y detectar los más relevantes pero para reducir la dimensión como tal, se aplican técnicas de selección. Existe el riesgo de que se detecten como necesarios dos o más atributos muy informativos pero que sean redundantes. Por ejemplo se daría en nuestro caso si además de añadir los atributos "sinónimos" y "distancia de Levenshtein" mencionados en la sección 7.1.4 hubiesemos añadido un atributo que exprese el estándar del CIE en sinónimos y otro que calcule la distancia de Levenshtein entre éste y el diagnóstico original expresado en sinónimos. Es probable que en el *ranking* las dos distancias quedasen seguidas, y en caso de ser detectadas como necesarias y relevantes las seleccionaríamos para el subconjunto óptimo. Sin embargo, estos dos atributos son redundantes, ambos expresan la distancia entre el original y el estándar.

- **Selección**

  " Busca subconjuntos de atributos suficientes para determinar la clase. No trabaja con atributos individualmente sino con subconjuntos de atributos a fin de descartar aquellos que son redundantes. Requiere seleccionar un criterio para buscar los subconjuntos, un método de búsqueda".

  Los algoritmos que simplemente establecen un *ranking* entre los atributos suelen ser más rápidos que los que tratan las interdependencias y seleccionan un conjunto de atributos informativo y no redundante. Mientras que en el *ranking* hay que establecer el número de atributos a conservar, en la selección el propio algoritmo prescribe el número.

  En la siguiente sección 7.1.5.2 se explican algunos métodos de reducción dimensional para llevar a cabo la selección de atributos.

7.1.5.2 *Métodos de reducción dimensional*

Como dice [Alpaydin(2010)], existen dos métodos principales para reducir la dimensión. Selección de atributos (*Feature Selection*) y Extracción de atributos.

- **Selección de atributos. *Feature Subset Selection***

  "Técnicas con el objetivo de encontrar las n *dimensiones (atributos) entre las* d *dimensiones originales que aportan más información y descartar las demás."*

  El proceso de selección de atributos es de gran importancia. Si se dejan atributos irrelevantes, provocará confusión en el algoritmo de clasificación, ya que tratará estos atributos de la misma manera que a los que aportan información relevante e importante. Esto hará que se desvíen los resultados disminuyendo el rendimiento. Las técnicas de selección de atributos, tratan de encontrar el mejor subconjunto de atributos entre los disponibles. Este subconjunto estará compuesto por el menor número de dimensiones posible que más contribuyan en la optimización del *accuracy*. Existen 2^d posibles subconjuntos de d atributos pero debido al coste computacional y temporal que esto supondría no se pueden probar todos a menos que d sea muy pequeño, por tanto empleamos heurísticos para conseguir una solución razonable en un tiempo razonable, sabiendo que cabe la posibilidad de que no sea la óptima.

  Para llevar a cabo esta tarea se requiere determinar una estrategia de búsqueda. Los **métodos de búsqueda** recorren el espacio de atributos para encontrar un buen subconjunto. Se puede hacer una división entre ellos según su manera de recorrer el espacio, con un proceso de *"Forward Selection"* (selección hacia delante) o *"Backward Selection"* (selección hacia atrás).

  Forward selection. Se empieza sin atributos y se van añadiendo uno a uno el que más disminuya el error hasta que ninguno lo haga o no en la medida en que se considera razonable. Éste es un procedimiento de búsqueda local que no garantiza conseguir el subconjunto óptimo. Ésto se debe a que dos atributos pueden no ser buenos independientemente pero que juntos disminuyan el error considerablemente, y este algoritmo no lo detectará. Es posible generalizarlo y añadir más de un atributo cada vez pero esto aumentará el coste de computación.

También es posible comprobar qué atributos añadidos previamente pueden ser eliminados tras cada nueva inclusión, aumentando así el espacio de búsqueda, pero también la complejidad. En la figura 48 se muestra el proceso seguido por éste método.

F = conjunto de atributos de dimensiones x_i, $i = 1, ..., d$
$E(F)$ = error cometido en la evaluación utilizando F

begin
$F = \varnothing$
Mientras $E(F \cup x_j) < E(F)$
Para $i = 1$ hasta $i = d$
Calcular $E(F \cup x_i)$

$$j = \arg\min_i E(F \cup x_i) \tag{11}$$

fin para
Si

$$E(F \cup x_j) < E(F) \tag{12}$$

Anñadir x_j a F
fin si
fin mientras
end

Figura 48: *Forward Selection*.

Cada atributo añadido supone el coste correspondiente a la observación de dicho atributo e incrementa la complejidad del clasificador. Este proceso puede ser costoso ya que para reducir las dimensiones de d a k se entrena y evalúa el sistema $d + (d-1) + (d-2) + ... + (d-k)$ veces lo que significa un coste de $O(d^2)$.

Backward selection. Se empieza con todas las variables y se van quitando una a una, la que más disminuya el error o la que menos lo aumente, siempre dentro de un rango de error permitido predefinido, hasta que cualquier eliminación lo aumente en una cantidad fuera de lo aceptable. Las variantes y generalizaciones mencionadas para *Forward selection* también se permiten para *Backward selection*. Empezamos con el conjunto F igual al conjunto completo de atributos y como se muestra en la figura 49, se realiza un proceso similar al indicado para *Forward selection* pero eliminado cada vez un atributo en lugar de añadirlo.

F = conjunto de atributos de dimensiones x_i, $i = 1, ..., d$
$E(F)$ = error cometido en la evaluación utilizando F

begin
F= Conjunto completo de atributos
Mientras $E(F - x_j) < E(F)$
Para $i = 1$ hasta $i = d$
Calcular $E(F - x_i)$

$$j = \arg\min_i E(F - x_i) \tag{13}$$

fin para
Si

$$E(F - x_j) < E(F) \tag{14}$$

Añadir x_j a F
fin si
fin mientras
end

Figura 49: *Backward Selection*.

Generalmente la complejidad de ambos procesos es similar, sin embargo, entrenar un sistema con muchos atributos es más costoso que entrenar uno con pocos y por tanto, sobre todo si se esperan muchos atributos no útiles, es mejor utilizar *Forward selection*. La figura 50 muestra de manera ilustrativa el proceso seguido por ambas técnicas. En ella, los atributos están representados en forma de "bolitas" y los recuadros que las encierran son los distintos subconjuntos de atributos que van generando ambos procesos. Se observa que la diferencia entre las dos técnicas reside en la dirección de búsqueda.

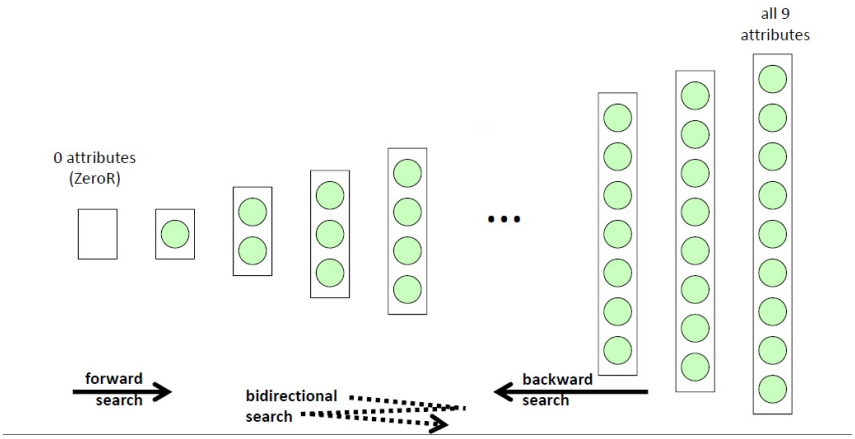


Figura 50: Figura obtenida de [Ian H. Witten(2011)].

Por otro lado, según la manera en que proceden las técnicas de selección de atributos se puede hacer una división entre *Método Filter* (método de filtrado) o *Método Wrapper* (método de envoltura), tal y como explica [Ian H. Witten(2011)]

Para utilizar estos métodos, Weka ofrece en sus librerías para Java las clases `FilteredSubsetEval` y `WrapperSubsetEval` respectivamente. [Ian H. Witten(2011)]

Método filter. Se hace una evaluación independiente basada en las características generales de los datos. Se filtra el conjunto de atributos para generar el subconjunto que nos proporcione mejores expectativas.

Método wrapper. Se hace una evaluación del subconjunto de atributos, utilizando el algoritmo de clasificación que se utilizará finalmente. La mayor dificultad de estos procesos reside en que no hay un criterio universal establecido que determine si un atributo es relevante o no, sin embargo, hay diferentes propuestas.

Una de las técnicas de selección más simple consiste en la división del espacio de manera que todas las instancias de entrenamiento queden separadas, es decir, no habrá dos instancias con los mismos valores en sus atributos. Lo más intuitivo es seleccionar el subconjunto de atributos más pequeño que distinga todas las instancias como únicas. Esto sería fácil mediante una búsqueda exhaustiva, pero sin embargo, supone un coste computacional considerable.

Algunos algoritmos de clasificación se pueden utilizar para la selección de atributos. Por ejemplo, se podría utilizar un algoritmo *Decision Tree*, explicado en el capítulo 8 dedicado a la clasificación, y generar el subconjunto de atributos con aquellos que haya usado. Si el algoritmo de clasificación que se va a utilizar finalmente es este, no tiene sentido utilizar esta técnica. Será efectiva si se va a utilizar otro algoritmo de clasificación. En la figura 51 se muestra un ejemplo de cómo se podría aplicar ésta técnica. Imaginemos una tarea sencilla a realizar con los datos que disponemos. Supongamos que las instancias están compuestas por todos los atributos disponibles, el diagnóstico original y los seis nuevos generados explicados en la sección 7.1.4. Se pide determinar si el diagnóstico original y el diagnóstico indicado en el estándar ICD-9-CM pueden considerarse iguales (una variación de 5 operaciones de edición permitidas). Se realiza la clasificación mediante el algoritmo *Decision Tree* y genera el árbol de la figura 51. Éste ha utilizado los atributos "Término ICD-9-CM" y "Distancia de Levenshtein" y por tanto éste será el subconjunto de atributos que escojamos.

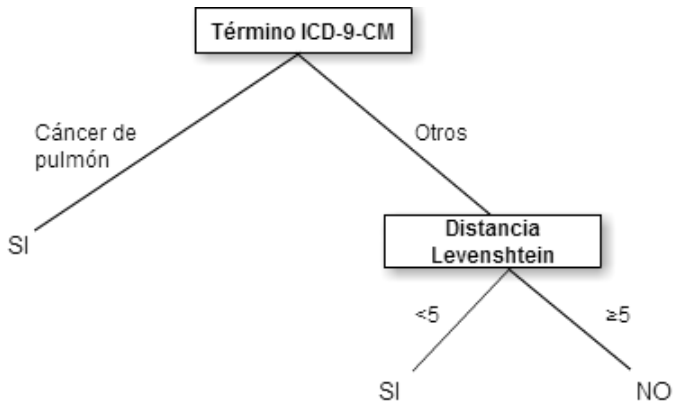


Figura 51: Árbol generado por el algoritmo *Decission Tree*.

- **Extracción de atributos**

  "Técnicas con el objetivo de determinar un nuevo conjunto de n dimensiones (atributos) que son combinaciones de las d dimensiones originales." [Alpaydin(2010)]

  [Alpaydin(2010)] afirma que en ciertas ocasiones estos métodos son más eficaces que los de selección.

  En la figura 52 se muestra la principal diferencia entre estas técnicas y las de Selección. Mientras que los métodos de selección eligen un subconjunto de los atributos existentes sin ninguna trasformación, los de extracción transforman los atributos en un espacio dimensional menor.

Selección de atributos
$[x_1, x_2, ..., x_n] = [x_{i1}, x_{i2}, ..., x_{im}]$
Extracción de atributos
$[x_1, x_2, ..., x_n] = [y_1, y_2, ..., y_m] = f([x_1, x_2, ..., x_n])$

Figura 52: Diferencia entre la Selección de atributos y la Extracción de atributos.

  Según [Alpaydin(2010)], los métodos de extracción de atributos más utilizados son *Principal Component Analysis (PCA)*, método utilizado en este proyecto y explicado en la sección 7.1.5.3, *Linear Discriminant Analysis (LDA)* y *Factor Analysis (FA)* entre otros.

7.1.5.3 *Técnicas utilizadas*

En este proyecto hemos experimentado con distintas técnicas tratando de abordar al menos un método de cada familia o división planteado anteriormente.

- **Ranking - Ganancia en Información**

  Hemos utilizado el evaluador de atributos por *Ranking* explicado en la sección 7.1.5.1, aplicando el criterio de Ganancia en Información como métrica de relevancia.

  Para ello hemos utilizado el evaluador de atributos de las librerías de Weka `weka.attributeSelection.InfoGainAttributeEval`, estableciendo el *Ranking* (`weka.attributeSelection.Ranker`) como método de búsqueda. [Ian H. Witten(2011)] Al método `Ranker` se le pueden ajustar dos parámetros:

  `numToSelect` - número de atributos que se desea mantener.

  `threshold` - el umbral inferior del valor que tiene que presentar un atributo en la métrica adoptada para hacer la selección a fin de no ser descartado.

  En este caso, en el primer parámetro hemos indicado que mantenga todos los atributos ya que queríamos obtener el *ranking* del conjunto completo de atributos para así poder contrastarlo con los métodos de selección, comprobando que estos últimos escogen los atributos situados en las primeras posiciones del *ranking*. En el segundo parámetro hemos dejado el valor por defecto, -1.7976931348623157E308 (el mínimo *Long integer* en Java). El criterio de Ganancia en Información se basa en la teoría de la Entropía, determina el decremento esperado de

entropía al conocer el resultado de un suceso (el valor de un atributo), aplicándolo a este caso concreto, mide la cantidad de información que aporta la aparición de una palabra en un texto.

$$\begin{aligned} IG(w) &= -\sum_{j=1}^{K} P(c_j)\log P(c_j) + P(w)\sum_{j=1}^{K} P(c_j|w)\log P(c_j|w) + \\ &+ P(\bar{w})\sum_{j=1}^{K} P(c_j|\bar{w})\log P(c_j|\bar{w}) \end{aligned} \tag{15}$$

donde se llama c_j a la posible clase, $P(c_j)$ al conjunto de documentos que pertenecen a la clase c_j, y $P(w)$ es la estimación del conjunto de documentos en el que aparece el término w. $P(c_j \mid w)$ es el conjunto de documentos de la clase c_j en los que aparece al menos una vez el término w y $P(c_j|\bar{w})$ el conjunto de documentos de la clase c_j que no contienen el término w.

- **Selección de atributos - Métodos de búsqueda**

  Para escoger el método de búsqueda a aplicar en las técnicas de selección de atributos explicadas en la sección 7.1.5.2, hemos considerado entre los disponibles en las librerías de Weka [Ian H. Witten(2011)] cual sería el más adecuado. En un principio consideramos utilizar el método de búsqueda ***Exhaustive Search***, el cual realiza una búsqueda exhaustiva del espacio de subconjuntos de atributos empezando por el conjunto vacío. El método *Exhaustive Search* busca el mejor subconjunto haciendo múltiples pruebas. Sin embargo, esta idea se descartó enseguida debido a que supone un coste muy elevado, especialmente cuando aumenta el número de atributos.

  El método de búsqueda escogido finalmente para este proyecto es el ***BestFirst Search***, este método busca en el espacio de subconjuntos de atributos aumentándolo gradualmente con posibilidad de retroceso. Se puede determinar el número permitido de nodos, atributos añadidos al subconjunto, consecutivos que no mejoran el resultado antes de que el sistema retroceda. Este método de búsqueda puede empezar con el conjunto vacío de atributos y seguir una estrategia de *Forward selection* o empezar con el conjunto de atributos completo y seguir una estrategia de *Backward selection*. También es posible empezar en un punto cualquiera y buscar en las dos direcciones teniendo en cuenta todas las posibilidades de incorporación o eliminación atributo por atributo. Los subconjuntos que han sido evaluados se guardan en memoria *cache* por temas de eficiencia. En este caso concreto hemos decidido que se realice una búsqueda siguiendo *Forward selection* y determinando el número de nodos máximo sin aportar mejora a 5, el parámetro establecido por defecto.

  Hemos decidido utilizar tres evaluadores de la familia Selección, explicada en la sección 7.1.5.1, que aplican técnicas de selección de atributos, *Subset Selection* concretamente, y siguen la técnica *Wrapper*, explicada en la sección 7.1.5.2. Se trata de los evaluadores de Weka `ClassifierSubsetEval`, `WrapperSubsetEval` y `CfsSubsetEval` [Ian H. Witten(2011)].

Selección de atributos - Classifier

Para evaluar los atributos mediante el *Classifier* hemos aplicado el evaluador de Weka `weka.attributeSelection.ClassifierSubsetEval` [Ian H. Witten(2011)].

`ClassifierSubsetEval` evalúa subconjuntos de atributos sobre el *train* o sobre otro conjunto de test si éste le es proporcionado. Utiliza un clasificador para estimar el rendimiento de cada subconjunto de atributos. En este proyecto hemos decidido utilizar el clasificador *RandomForest*, explicado en el capítulo 8, para evaluar el conjunto *train* con los distintos subconjuntos de atributos, debido a que hemos observado que dicho clasificador ofrece buenos resultados sin un coste computacional y temporal demasiado elevado.

Selección de atributos - Wrapper

Para evaluar los atributos mediante el *Wrapper* hemos aplicado el evaluador de Weka `weka.attributeSelection.WrapperSubsetEval` [Ian H. Witten(2011)].

Al utilizar este método, se hace una evaluación del subconjunto de atributos utilizando el algoritmo de clasificación que se utilizará finalmente, el *RandomForest* en este caso por la razón expuesta anteriormente. La diferencia entre este evaluador y el `ClassifierSubsetEval`, es que el `WrapperSubsetEval` hace una validación cruzada interna y el `ClassifierSubsetEval` no.

Selección de atributos - Correlation based Feature Selection

Para evaluar los atributos mediante *Correlation based Feature Selection* hemos aplicado el evaluador de Weka `weka.attributeSelection.CfsSubsetEval` [Ian H. Witten(2011)].

Este método de selección evalúa el valor de cada subconjunto de atributos considerando la capacidad predictiva de cada uno de ellos individualmente y teniendo en cuenta el grado de redundancia entre ellos. El subconjunto de atributos escogido será aquel cuyos atributos estén altamente correlacionados con la clase y poco correlacionados entre ellos. De esta manera, se ignoran tanto los atributos irrelevantes como los redundantes ya que los primeros tendrán una baja correlación con la clase y los segundos una correlación alta entre ellos.

Por ejemplo, supongamos que en este proyecto además de los atributos disponibles, el original y los generados explicados en la sección 7.1.4, disponemos de otro atributo que contiene el término estándar del ICD-9-CM expresado en sinónimos. Utilizando la técnica *Correlation based Feature Selection*, el término del ICD-9-CM se seleccionará para el subconjunto óptimo de atributos por tener una alta correlación con la clase, pero el término del ICD-9-CM expresado en sinónimos no se seleccionará por tener una correlación alta con el atributo término del ICD-9-CM.

La ecuación según la cual actúa el evaluador *Cfs* se expone a continuación

$$M_s = \frac{k\overline{T_{cf}}}{\sqrt{k + k(k-1)\overline{r_{ff}}}} \tag{16}$$

Donde M_s es el valor de un atributo del subconjunto S que contiene k atributos, $\overline{T_{cf}}$ es la media de la correlación atributo - clase, y $\overline{r_{ff}}$ es la media de la correlación atributo - atributo.

El numerador podría entenderse como un indicador de cómo de predictivo de la clase es un subconjunto de atributos, y el denominador un indicador de cuanta redundancia hay entre los atributos de dicho subconjunto. [Hall(1999)].

Tanto `ClassifierSubsetEval` como `WrapperSubsetEval` son métodos de selección *Scheme-dependent* mientras que `CfsSubsetEval` se trata de un método *Scheme-independent*.

Los primeros son simples y directos pero lentos. Utilizan un evaluador con *ranking* que actúa atributo por atributo para eliminar los irrelevantes y combina un evaluador de subconjuntos de atributos con un método de búsqueda para eliminar los redundantes. Los segundos, como se ha mencionado al explicar el método `CfsSubsetEval` eligen un subconjunto de atributos fijándose en ciertas reglas de correlación. Los métodos *Scheme-dependent* suelen ser más precisos pero suponen un coste computacional mucho más elevado.

Basándonos en la experiencia de los experimentos que hemos realizado y mostramos en la sección "Marco experimental" 7.2 podemos afirmar que esto es así ya que en experimentos realizados con los mismos datos obtenemos tiempos muy diferentes para los 3 métodos: 77.79 min utilizando la técnica *Wrapper*, 10.77 min utilizando la técnica *Classifier* y 5.6 seg utilizando la técnica *Correlation based Feature Selection*. Además a continuación se muestra la figura 53 con una comparación entre los 3 en la que se concluye que `CfsSubsetEval` es casi tan bueno como `WrapperSubsetEval` pero mucho más rápido. Se observa que `CfsSubsetEval` ha realizado la misma selección que `ClassifierSubsetEval` pero tardando menos tiempo. Además, vemos que la selección que ha realizado `WrapperSubsetEval` difiere únicamente en haber seleccionado un sólo atributo de los dos seleccionados por `CfsSubsetEval` y `ClassifierSubsetEval`, necesitando para ello un tiempo mucho mayor. Por ésto, concluimos que `CfsSubsetEval` es casi tan bueno como `WrapperSubsetEval` pero mucho más rápido.

Técnica de selección	Tiempo	Atributos seleccionados
`CfsSubsetEval`	5.6 seg	Término del ICD-9-CM y Sinónimos
`ClassifierSubsetEval`	10.77 min	Término del ICD-9-CM y Sinónimos
`WrapperSubsetEval`	77.79 min	Término del ICD-9-CM

Figura 53: Comparación entre métodos de Selección de atributos.

- **Extracción de atributos - Principal Component Analysis (PCA)**

  Como método de extracción de atributos, sección 7.1.5.2, hemos decidido utilizar *Principal Component Analysis*, un evaluador de la familia *Ranking*, explicado en la sección 7.1.5.1. Este método crea un nuevo conjunto de dimensiones que son combinaciones lineales de las originales.

  Como explica [Alpaydin(2010)], el criterio de maximización utilizado es la varianza. Se pretende proyectar la muestra en una dirección en que se haga más evidente la diferencia entre los puntos de la misma. Primero se centra la muestra y posteriormente se rotan los ejes hasta

alinearlos en la dirección de la varianza más alta, tal y como se muestra en la figura 54. Si en uno de los ejes la varianza es muy pequeña éste se puede ignorar, reduciendo la dimensionalidad.

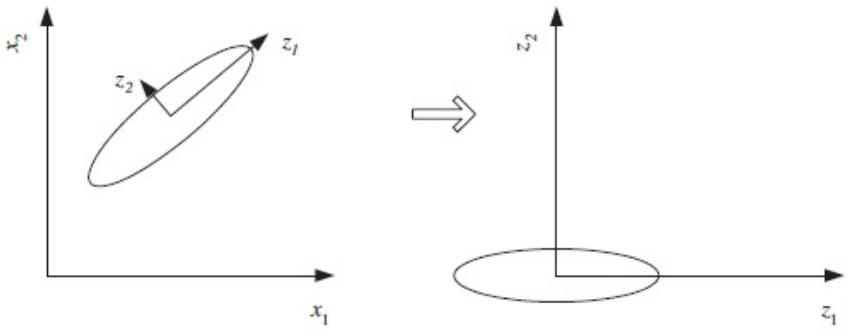


Figura 54: Figura obtenida de [Alpaydin(2010)].

Este método es muy sensible a los *outliers* (datos que no cumplen con el comportamiento general del conjunto de datos al que pertenecen. [Han & Kamber(2006)]), unos pocos puntos alejados del centro suponen un gran efecto en las varianzas. Para evitarlo, se utilizan los métodos de Estimación Robusta que permiten el cálculo de parámetros en presencia de *outliers*. Uno de los más simples es el cálculo de la distancia de Mahalanobis de los puntos, descartando los que están alejados.

La distancia de Mahalanobis entre dos variables aleatorias con la misma distribución de probabilidad $\vec{x}$ y $\vec{y}$ con matriz de covarianza Σ se define según la ecuación 17.

$$d_m(\vec{x},\vec{y}) = \sqrt{(\vec{x}-\vec{y})^T \Sigma^{-1} (\vec{x}-\vec{y})} \tag{17}$$

Esta técnica, al igual que la selección de atributos basada en Ganancia de Información explicada anteriormente, se aplica conjuntamente con el método de búsqueda *Ranker*. Utilizando las librerías de Weka [Ian H. Witten(2011)], hemos aplicado esta técnica en nuestro proyecto utilizando el filtro `weka.attributeSelection.PrincipalComponents`.

Además de los mencionados métodos de *Ranking* y Selección utilizados, se han explorado otras técnicas con el mismo fin de reducir la dimensionalidad. Se han probado técnicas *Term frequency - Inverse document frequency* (Técnicas de Frecuencia de los términos en el documento) y el filtro de atributos *Remove*. Estas técnicas se explican a continuación.

- **Técnicas de Frecuencia de los términos en el documento**

  Las técnicas basadas en la frecuencia de aparición en el documento son técnicas de selección de atributos cuyo objetivo es eliminar los términos que aparezcan en el conjunto de entrenamiento menos de un número de veces predeterminado. De esta manera, son ignorados aquellos términos raros o especiales que no son informativos ni relevantes para la clasificación o que no tienen influencia sobre la actuación global del clasificador.

  Se define la frecuencia relativa del término w_i en el documento d_j, *term frequency (TF)*, según la ecuación

$$TF(w_i, d_j) = \frac{f(w_i, d_j)}{\sum_{w_i \in \mathbb{V}} f(w_i, d_j)} \tag{18}$$

donde f(w_i, d_j) representa el número de veces que aparece el término w_i en el documento d_j, $\sum_{w_i \in \mathbb{V}} f(w_i, d_j)$ representa el número total de términos que aparecen en el documento d_j (representa el número total de palabras del documento, no el número de palabras distintas), V es el conjunto de términos o vocabulario de la aplicación, es decir, $V = \{w1, ..., w_i, ..\} \subseteq W$ donde $w_i \neq w_j \ \forall i \neq j$ (conjunto de términos distintos en el conjunto de documentos).

Puede parecer que cuanto mayor sea TF(w_i,d_j), más característico o relevante es el término w_i para describir el documento d_j . Sin embargo, los términos como determinantes o artículos son muy frecuentes en todos los documentos, y por tanto no son buenos atributos predictores. Para atenuar la relevancia que se le asocia al término w_i se define la frecuencia relativa de los documentos que contienen el término w_i, *document frequency (DF)*, según la expresión

$$DF(w_i) = \frac{\sum_{d_j \in \mathbb{D}} \delta(w_i, d_j)}{|D|} \tag{19}$$

donde $\delta(w_i, d_j)$ representa la delta de Kronecker sobre la pertenencia del término w_i en el documento d_j según se indica a continuación

$$\delta(w_i, d_j) = \begin{cases} 1 & \text{si } w_i \text{ está presente en el documento } d_j \\ 0 & \text{si } w_i \text{ no está presente en el documento } d_j \end{cases} \tag{20}$$

$\sum_{d_j \in \mathbb{D}} \delta(w_i, d_j)$ representa el número de documentos que contienen el término w_i, |D| representa el número total de documentos.

Cuanto menor sea DF(w_i; d_j) (o cuanto mayor sea el inverso), más ayudará el término w_i a discriminar entre los distintos documentos. A fin de determinar cuantitativamente el grado de relevancia del término w_i en el conjunto de documentos se define *TF-IDF (term frequency-inverse document frequency)* según la expresión

$$TF - IDF(w_i, d_j) = TF(w_i, d_j).\log \frac{1}{DF(w_i)} \tag{21}$$

En resumen, *TF* es una medida para cuantificar la relevancia de un término dentro de un documento. *IDF* es una medida para cuantificar la relevancia de un término en un conjunto de documentos. *TF-IDF* es una medida que combina *TF* e *IDF*, de modo que cuantifica la relevancia de un término dentro de un documento considerando los demás documentos. En definitiva, cuanto mayor sea la frecuencia del término w_i en el documento d_j y cuanto menor sea la frecuencia del término

w_i en el conjunto de documentos, tanto mayor será *TF-IDF*(w_i,d_j). De este modo, para los determinantes, artículos y otros términos frecuentes en muchos documentos, el *TF-IDF* toma un valor cercano a 0. *TF-IDF* es una métrica utilizada para cuantificar lo discriminativo que resulta el término w_i. [C. Manning & Schutze(2008)]

- **Filtro Remove**

  Otra técnica que hemos probado para realizar la selección de atributos es la aplicación del filtro de Weka `weka.filters.unsupervised.attribute.Remove` sobre los datos, [Ian H. Witten(2011)]. Este filtro se utiliza para eliminar un rango de atributos del conjunto de datos sobre el que se aplica. Permite ajustar dos parámetros:

  `attributeIndices` - Rango de atributos sobre los que se desea aplicar el filtro.

  `invertSelection` - Determina la acción a realizar sobre el rango de atributos especificado. En caso de indicarlo a `True` dicho rango de atributos será mantenido, mientras que si se indica a `False` será eliminado. Por defecto este parámetro está a `False` pero en nuestro software lo hemos indicado a `True`.

  En cuanto al parámetro `attributeIndices` hemos decidido experimentar con distintos rangos para encontrar el óptimo en cada caso concreto, empezando por el conjunto completo y eliminando un atributo cada vez hasta llegar a eliminar todos menos uno, realizando así una búsqueda de los atributos óptimos siguiendo una técnica de *Backward Selection*.

7.1.6 Volumen del corpus preprocesado

En el capítulo 6 sobre los datos, se muestra la tabla 7 en la página 51, en la que se muestran distintos datos sobre el volumen del corpus original. Tras aplicar las distintas técnicas de normalización indicadas en el apartado 7.1.2 es posible que estos datos hayan cambiado, ya que estandarizando las reglas de escritura, algunas instancias que originalmente eran distintas siendo normalizadas se igualan. Además, al aplicar la técnica de representación *Bag Of Words*, como se ha explicado anteriormente en la sección 7.1.3, se pierde el orden de las palabras dentro de cada instancia. Por tanto, habrá instancias que se igualen representando los textos de ésta manera.

En la tabla 8 se muestran los datos sobre el volumen del corpus tras aplicar las técnicas de normalización, y tras representar un texto normalizado utilizando la técnica *Bag Of Words*. No se indican los datos sobre el corpus trás aplicar las demás técnicas de preproceso explicadas en este capítulo, debido a que no influyen sobre éstos.

Técnica aplicada	Conjunto de datos	Num. instancias	Num. instancias repetidas	Diagnósticos distintos	Clases distintas
Normalización	Train	23130	16503	6415	1579
	Dev	2850	1207	1611	678
	Test	2850	1179	1636	702
Normalización + BOW	Train	23130	17647	5260	1579
	Dev	2850	1404	1409	678
	Test	2850	1314	1496	702

Tabla 8: Datos del corpus disponible.

7.2 MARCO EXPERIMENTAL

Con el objetivo de probar el rendimiento y eficiencia de las técnicas explicadas anteriormente en la sección 7.1 aplicadas a este proyecto concreto, se ha realizado un conjunto de experimentos englobando todas ellas. Los resultados mostrados a continuación son los correspondientes a dichos experimentos utilizando el clasificador *RandomForest* debido a que como se explica en el capítulo 8 dedicado a la Clasificación ofrece buenos resultados sin un coste computacional excesivo. Además, en el Apéndice A se muestran las tablas de resultados completos. Los mismos experimentos utilizando el resto de clasificadores así como los resultados correspondientes a una experimentación "No honesta", es decir, utilizando los mismos datos para entrenar y para evaluar.

7.2.1 Normalización

El primer paso en el preproceso de los datos fue la aplicación de unas técnicas básicas de normalización de textos, explicadas en la sección 7.1.2. Se eliminaron las tildes, se pasaron todas las palabras a mayúsculas y se decidió ignorar los signos de puntuación.

A continuación se muestran dos tablas con los resultados correspondientes a dos experimentos realizados con los mismos datos antes y después de normalizarlos, las tablas 9 y 10 respectivamente. En ambos experimentos, los datos de entrada son el conjunto *train* y el conjunto *dev*.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.766	0.008	0.85	0.766	0.782	0.966

Tabla 9: Tabla de resultados - Sin normalización.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.816	0.006	0.87	0.816	**0.822**	0.976

Tabla 10: Tabla de resultados - Con normalización.

Se puede observar la gran mejora que aportan las técnicas de normalización. Considerando también que son técnicas simples que no suponen un coste computacional ni temporal elevado, **se decide normalizar** todos los textos disponibles para el desarrollo de este proyecto y realizar los distintos experimentos con ellos a partir de este momento. (Se ofrecen las tablas de resultados exhaustivos 37 y 38 en la página 109).

7.2.2 Representación del texto

Como se expone en la sección anterior "Marco teórico", se han utilizado dos maneras de representar los textos, mediante *Bag Of Words* (BOW), explicado en la sección 7.1.3.1, y mediante una transformación de los atributos a nominales, explicado en la sección 7.1.3.2. En el capítulo sobre el trabajo previo relacionado, en la sección sobre el estado del arte 3.2 mencionamos que debido a las conclusiones obtenidas en el estudio de Aronson [Aronson & et al.(2007)], esperamos que la técnica *Bag Of Words* sea de utilidad en nuestro trabajo. A continuación vemos si estábamos en lo cierto.

En las tablas 11 y 12, se muestran los resultados correspondientes a dos experimentos realizados con los mismos datos (conjunto *train* y conjunto *dev* normalizados) pero representados de las dos maneras mencionadas.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.881	0	0.883	0.881	**0.876**	0.959

Tabla 11: Tabla de resultados - Representación BOW

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.816	0.006	0.87	0.816	0.822	0.976

Tabla 12: Tabla de resultados - Representación nominal.

Observando la figura de mérito *F-measure* en las tablas 11 y 12, se puede ver que los resultados obtenidos de los experimentos realizados con la representación BOW son ligeramente mejores, sin embargo, esta mejora se puede considerar insignificante si se tiene en cuenta el gran incremento que supone en el coste computacional y temporal realizar la clasificación de textos con esta representación.

Por tanto considerando ambos factores, resultados ofrecidos y coste computacional, **consideramos más adecuada para este proyecto la representación de los textos mediante atributos nominales** (Se ofrecen las tablas de resultados exhaustivos 41, 42 en la página 110).

7.2.3 Nuevos atributos

Añadiendo nuevos atributos a cada instancia del corpus disponible, aportamos más información a los clasificadores facilitándoles la tarea. Como se indica en el "Marco teórico" de este mismo capítulo hemos generado 6 nuevos atributos, explicados en la sección 7.1.4. Sin embargo, los resultados obtenidos añadiendo todos ellos no pueden considerarse realistas por las razones expuestas anteriormente. Por tanto, los experimentos realizados con estos atributos nos servirán por una parte como cota optimista y por otra como demostración del buen funcionamiento de los métodos de reducción dimensional utilizados, esperando que elijan dichos atributos.

A continuación se muestran tres tablas (13, 14 y 15) con los resultados correspondientes a tres experimentos. En el primero de ellos se han utilizado como datos de entrada textos normalizados en formato nominal sin atributos añadidos, en el segundo los mismos textos pero con los nuevos atributos (Término del ICD-9-CM, sinónimos, distancia de Levenshtein, número de N-gramms compartidos, número de palabras compartidas y número de palabras compartidas en la misma posición) y en el último se repite el experimento anterior pero solo se añade el atributo sinónimos (el único no mencionado anteriormente como "no realista").

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.816	0.006	0.87	0.816	0.822	0.976

Tabla 13: Tabla de resultados - Sin atributos añadidos.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.975	0.001	0.961	0.975	**0.966**	0.989

Tabla 14: Tabla de resultados - Con atributos añadidos.

A la vista de los resultados obtenidos confirmamos que tal y como creíamos **la aportación de más información a las instancias ha supuesto una gran mejora**. Como hemos dicho se trata de una cota optimista, veamos a continuación, en la tabla 15, si el atributo sinónimos aporta suficiente información como para ofrecer una mejora razonable en unos resultados realistas.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
RandomForest	0.826	0.006	0.872	0.826	0.83	0.977

Tabla 15: Tabla de resultados - Con atributo sinónimos añadido.

Observamos que aunque en menor medida que añadiendo todos, **incluir solamente el atributo sinónimos también mejora los resultados obtenidos** frente a los mostrados en la tabla 13 (experimentación sin atributos añadidos). Suponemos por tanto que éste es un buen atributo (Se ofrecen las tablas de resultados exhaustivos 45, 46, 47 en las páginas 111 y 111). Mediante el siguiente bloque de experimentos en el que se aplican técnicas de reducción dimensional, explicadas en la sección 7.2.4, comprobamos si estamos en lo cierto.

7.2.4 Reducción dimensional

Partiendo de los textos a los que les hemos añadido los nuevos atributos hemos realizado experimentos para probar las distintas técnicas de reducción dimensional expuestas en el apartado "Marco teórico". La técnica de reducción PCA explicada en el apartado 7.1.5, es una buena técnica según la investigación y documentación realizada para el desarrollo de este proyecto. Ha ofrecido buenos resultados en trabajos similares y por tanto supusimos que sería adecuada para el nuestro. Sin embargo, en este proyecto ha sido imposible su aplicación debido a problemas de memoria de los equipos disponibles ya que supone un coste computacional muy elevado. A continuación se muestran las tablas de resultados en las que se puede ver qué atributos selecciona cada técnica, además de los resultados que nos ofrecen. Debido al carácter no realista de los resultados ofrecidos por algunos de los nuevos atributos mencionado en el apartado 7.2.3, se hace una distinción entre los experimentos realizados con todos los nuevos atributos añadidos (Tablas de resultados 16, 17, 18, 19, 20, 21, 22) y los experimentos con solo el atributo realista añadido. (Tablas de resultados 23, 24, 25, 26, 27, 28). En todos ellos se utiliza como datos de entrada, el conjunto *train* y el conjunto (dev) normalizados representados de manera nominal.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.975	0.001	0.961	0.975	0.966	0.989

Tabla 16: Tabla de resultados - Sin ninguna técnica de reducción dimensional.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.978	0.001	0.964	0.978	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
1. Término ICD-9-CM 2. Text 3. Sinónimos 4. Dist. Levenshtein 5. N-gramms comp. 6. Palabras comp. 7. Palabras en posición comp.	7	-

Tabla 17: Tabla de resultados - InfoGain.Ranking.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.976	0.001	0.959	0.976	0.966	0.989

Tabla 18: Tabla de resultados - TF-IDF.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	2	Término del ICD-9-CM Sinónimos

Tabla 19: Tabla de resultados - Cfs.Best first search.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	2	Término del ICD-9-CM Sinónimos

Tabla 20: Tabla de resultados - Classifier.Best first search.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Término del ICD-9-CM

Tabla 21: Tabla de resultados - Wrapper.Best first search.

Podemos observar que en los experimentos anteriores realizados utilizando todo el conjunto de atributos generado, **el atributo elegido como más informativo y relevante tanto por los métodos de *Ranking* como por los métodos de selección es el Término del ICD-9-CM**. Estos resultados demuestran el buen funcionamiento de las técnicas de reducción dimensional utilizadas, ya que analizando con lógica los atributos añadidos es fácil pensar que éste será el mejor. El razonamiento utilizado para llegar a esta conclusión es simple, los atributos distancia de Levenshtein, N-gramms compartidos, palabras compartidas y palabras en posición compartidas están generados a partir del término del ICD-9-CM y por tanto sólo aportan parte de la información de éste. Por otra parte quedarían los atributos text y sinónimos, tratándose los dos de descripciones del diagnóstico, la descrip-

ATR. SELEC.	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Text Térm. ICD-9-CM Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.975	0.001	0.961	0.975	0.966	0.989
Térm. ICD-9-CM Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.979	0.001	0.964	0.979	**0.97**	0.989
Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.851	0.002	0.825	0.851	0.824	0.982
Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.395	0.008	0.286	0.395	0.309	0.935
N-gramms comp. Palabras comp. Pal. Pos. comp.	0.1	0.031	0.042	0.1	0.044	0.725
Palabras comp. Pal. Pos. comp.	0.094	0.032	0.023	0.094	0.032	0.721
Pal. Pos. comp.	0.069	0.034	0.008	0.069	0.014	0.675

Tabla 22: Tabla de resultados - Remove.

ción original y la descripción expresada con sinónimos, se evidencia que la descripción estándar (Término del ICD-9-CM) facilitará más la clasificación.

Se observa que la selección realizada por las técnicas *Cfs, Classifier y Wrapper* no solo coincide entre sí, también coincide con el *ranking* realizado por la técnica *InfoGain*. *Wrapper* ha seleccionado el atributo término del ICD-9-CM, y *Cfs* y *Classifier* han seleccionado éste y el atributo sinónimos. En el *ranking* generado por *InfoGain* podemos observar que el atributo término del ICD-9-CM es el primero y el atributo sinónimos el tercero. Podemos concluir que el ***Wrapper* ha realizado una mejor selección** ya que habiendo seleccionado un solo atributo ofrece un resultado igual de bueno al de *Cfs* y *Classifier*, siendo este atributo además el primer posicionado en el *Ranking* generado por *InfoGain*.

Analizando además los resultados generados de los experimentos realizados aplicando el filtro *Remove*, podemos confirmar la importancia del atributo Término del ICD-9-CM y la irrelevancia del resto. Podemos ver que cuando se elimina el primer atributo ("text") los resultados mejoran. Ésto nos indica que dicho atributo es distractor, es decir, confunde al clasificador y su presencia no sólo no aporta ninguna mejora, sino que empeora los resultados.

Tras eliminar el atributo término del ICD-9-CM vemos que los resultados empiezan a empeorar (De *F-measure* = 0.97 a *F-measure* = 0.824) y eliminando además el atributo sinónimos podemos ver el gran decremento en los resultados que esto supone (De *F-measure* = 0.824 a *F-measure* = 0.309). Estos resultados nos confirman las conclusiones obtenidas previamente. Podemos decir que **el término del ICD-9-CM es el atributo más informativo seguido de sinónimos aunque la selección idónea es quedarse únicamente con el término del ICD-9-CM**.

Tras comprobar que el atributo sinónimos es considerado relevante y necesario por la mayoría de los métodos, vemos a continuación en las tablas 23, 24, 25, 26, 27, 28, como se comportan utilizando tan solo los atributos originales y dicho atributo.

Como se ha explicado anteriormente en la sección "Marco teórico" 7.1, la técnica *Wrapper* es una buena técnica pero conlleva un coste computacional elevado. Es por ésto que en este conjunto de experimentos no hemos conseguido aplicarla debido a problemas de memoria de los equipos.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

Tabla 23: Tabla de resultados - Sin ninguna técnica de reducción dimensional.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
1. Text 2. Sinónimos	2	-

Tabla 24: Tabla de resultados - InfoGain.Ranking.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

Tabla 25: Tabla de resultados - TF-IDF.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.824	0.006	0.871	0.824	0.828	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Sinónimos

Tabla 26: Tabla de resultados - Cfs.Best first search.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.824	0.006	0.871	0.824	0.828	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Sinónimos

Tabla 27: Tabla de resultados - Classifier.Best first search.

ATR. SELEC.	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Text Sinónimos	0.826	0.006	0.872	0.826	**0.83**	0.977
Sinónimos	0.824	0.006	0.871	0.824	0.828	0.977

Tabla 28: Tabla de resultados - Remove.

A la vista de los resultados podemos deducir que el atributo sinónimos es un buen atributo de carácter informativo y relevante ya que los métodos de selección de atributos eligen éste y eliminan el atributo text. Sin embargo también se aprecia la importancia del atributo original (text). Éste no sólo ocupa la primera posición en el *ranking* generado por el método *InfoGain* sino que además observamos un decremento en los resultados cuando se elimina dicho atributo. Se trata únicamente de un pequeño empeoramiento considerado insignificante por los métodos de selección utilizados, la decisión que se debe tomar en este punto es si para la tarea abordada se puede prescindir del atributo text a pesar de perder cierto nivel de calidad en el algoritmo de clasificación y hacer caso por tanto a los métodos de selección de atributos o si por el contrario no eliminamos ninguno. **En el proyecto que estamos desarrollando la precisión en la clasificación es un aspecto de gran importancia por encima del ahorro computacional y temporal que supondría eliminar un atributo y por tanto decidimos utilizar ambos, el atributo text y el atributo sinónimos**.

7.2.5 Discusión de resultados

Trás los experimentos realizados en esta sección, obtenemos algunas conclusiones. En primer lugar, se observa la importancia de incluir una fase de normalización de los datos 7.2.1, debido a la gran mejora que aporta a los resultados sin un coste computacional elevado. En cuanto al modo de representación de los textos 7.2.2, decidimos escoger la representación nominal. Aunque los resultados ofrecidos con esta técnica sean ligeramente peores que utilizando *Bag Of Words*, el gran incremento en el coste computacional que supone realizar la clasificación con el texto representado mediante la técnica *Bag Of Words*, hace que la diferencia en los resultados sea considerada insignificante.

Observando los resultados del apartado "Nuevos atributos" 7.2.3, concluimos que el atributo elegido como más informativo y relevante tanto por los métodos de *Ranking* como por los métodos de selección es el Término del ICD-9-CM seguido del atributo Sinónimos. Sin embargo, como se ha mencionado anteriormente, añadir el atributo Término del ICD-9-CM supone obtener unos resultados no realistas y por tanto se decide añadir únicamente el atributo sinónimos.

Observando y analizando los resultados del apartado "Reducción dimensional" 7.2.4, podemos sacar las siguientes conclusiones. Confirmamos que realizando una selección de atributos cuando se dispone de un número significante de ellos mejoran los resultados, aunque cuando se dispone de pocos puede que no se llegue a apreciar la diferencia. En el primer bloque de experimentos en los que se han utilizado todos los atributos disponibles (Tablas de resultados 16, 17, 18, 19, 20, 21, 22) vemos que las figuras de mérito obtenidas en el primer experimento (sin ninguna técnica de selección) son más bajas que las obtenidas en los demás experimentos. Sin embargo, en el segundo bloque de experimentos (Tablas de resultados 23, 24, 25, 26, 27, 28) las figuras de mérito obtenidas en el primer experimento (sin ninguna técnica de selección) son iguales que las más altas obtenidas en el resto e incluso más altas que en aquellos en los que se ha hecho una selección en sí eliminando uno de los atributos.

Comparando las distintas técnicas aplicadas en el segundo bloque de experimentos no se observan grandes diferencias pero en el primero vemos que la que peor resultados nos ofrece es *TF-IDF*. Ésto se debe a que a pesar

de ser una buena técnica no es la más adecuada para nuestro proyecto. La técnica *TF-IDF*, como se ha explicado en el marco teórico 7.1.5.3, se basa en las frecuencias de aparición de los términos eliminando los menos frecuentes. De esta manera, se consigue eliminar los términos no relevantes como artículos o determinantes, pero también se eliminan ciertos términos especiales que aunque poco frecuentes son importantes, además de provocar pérdida de información.

En el caso concreto de este proyecto son mayores las pérdidas ocasionadas que los beneficios obtenidos ya que en los textos utilizados no aparecen muchas palabras prescindibles como son los artículos. Sin embargo sí aparecen términos especiales como puede ser una enfermedad poco habitual.

8 DESARROLLO: CLASIFICACIÓN

En este capítulo se tratará la clasificación de los datos mediante diversos algoritmos de aprendizaje automático. En primer lugar se detalla el marco teórico, en el que se explican los modelos de aprendizaje automático utilizados en la realización del proyecto. En segundo lugar, se detalla el proceso seguido para la implementación del *software* de clasificación, se presentan y se comentan los resultados de los experimentos.

8.1 MARCO TEÓRICO

A continuación vamos a profundizar un poco en los clasificadores que han servido para realizar este proyecto. Se ha implementado un total de cinco modelos de clasificación. Estos modelos han sido seleccionados en base al trabajo previo existente con datos de entrada similares, analizados en el capítulo 3, dedicado a los antecedentes. Los modelos implementados son; *Naive Bayes, Bayes Net, Decision Tree, Random Forest y Support Vector Machine (SVM).*

8.1.1 *Naive Bayes*

Tal y como se explica en [Ian H. Witten(2011)] este clasificador recibe el nombre de "naive" porque "ingenuamente" asume independencia condicional de unos atributos respecto de otros. A pesar de su nombre, este método es muy efectivo aplicado a conjuntos de datos reales, particularmemte combinado con técnicas de selección de atributos, las cuales se detallan en el capítulo anterior dedicado al preproceso 7, que eliminen los atributos reduntantes. Como explica Tom Mitchell [Mitchel(1997)], el clasificador *Naive Bayes* ha demostrado aportar resultados comparables a los de las redes neuronales o los árboles de decisión, siendo estos últimos mucho más costosos computacionalmente. En las tareas de aprendizaje, cada instancia x está formada por un conjunto de atributos y la función objetivo $f(x)$ sólo puede tomar un valor de un conjunto finito V. La aproximación Bayesiana para clasificar una nueva instancia descrita por una tupla con los valores de los atributos $\{a_1, a_2 ... a_n\}$, es asignar el valor más probable de la función objetivo v_{MAP}.

$$v_{MAP} = \underset{v_j \in V}{\arg\max} P(v_j | a_1, a_2 ... a_n) \tag{22}$$

Podemos utilizar el Teorema de Bayes para escribir la expresión de la siguiente forma:

$$\begin{aligned} v_{MAP} &= \underset{v_j \in V}{\arg\max} \frac{P(a_1, a_2 ... a_n | v_j) P(v_j)}{P(a_1, a_2 ... a_n)} \\ &= \underset{v_j \in V}{\arg\max} P(a_1, a_2 ... a_n | v_j) P(v_j) \end{aligned} \tag{23}$$

Ahora podemos intentar estimar los dos términos de la ecuación 23 basándonos en el conjunto de entrenamiento. La frecuencia de $P(v_j)$ se estima fácilmente contando la frecuencia con la que aparece cada término con el valor vj dentro del conjunto de entrenamiento. Sin embargo, no es factible estimar $P(a_1, a_2...a_n|v_j)$ a no ser que se disponga de un conjunto de entrenamiento muy extenso. El problema es que el número de estos términos es igual al número de posibles instancias que coincidan y tomen valor positivo. Por lo tanto, necesitamos ver cada instancia en el espacio de instancias muchas veces para obtener resultados de confianza.

Como ya hemos explicado, el clasificador *Naive Bayes* está basado en la asunción de que los valores de los atributos son condicionalmente independientes. En otras palabras, la asunción es que dado el valor de la clase de la instancia, la probabilidad de observar el conjunto $a_1, a_2...a_n$ es simplemente el producto de las probabilidades de cada atributo: $P(a_1, a_2...a_n|v_j) = \prod_i P(a_i|v_j)$. Sustituyendo esto en la ecuación 23, obtenemos la aproximación utilizada por el clasificador *Naive Bayes*.

$$v_{NB} = \underset{v_j \in V}{\arg\max} \quad P(v_j) \prod_i P(a_i|v_j) \tag{24}$$

Donde v_{NB} denota el resultado obtenido por el clasificador *Naive Bayes*. Es importante mencionar que en este clasificador el número de los distintos términos $P(a_i|v_j)$ que han de ser estimados del conjunto de entrenamiento es el número de valores diferentes de los atributos que coinciden con los valores diferentes que puede tomar la clase. Esto es un número mucho menor que si tuviésemos que estimar los términos $P(a_1, a_2...a_n|v_j)$ como se planteaba en un principio.

Una característica interesante del *Naive Bayes* es que a diferencia de otros clasificadores, no hace una búsqueda a través del espacio de posibles hipótesis, sino que la hipótesis se forma sin realizar ninguna búsqueda, simplemente contando las frecuencias de aparición de las diferentes combinaciones de datos en el conjunto de entrenamiento.

Este método parece apropiado desde un principio para nuestro conjunto de datos, ya que su uso está muy extendido en el campo de la clasificación documental, debido a su rapidez y su considerable exactitud. En la clasificación de documentos, éstos son las instancias, y la clase a estimar es el tema sobre el que trata el texto.

8.1.2 Decision Trees - Árboles de decisión

"El aprendizaje mediante árboles de decisión es un método de aproximación de una función objetivo, de valores discretos en el cual la función objetivo es representada mediante un árbol de decisión". definición tomada de [Mitchel(1997)]

Los árboles de decisión clasifican las instancias ordenándolas en forma de árbol desde la raíz hasta un nodo hoja que es el que proporciona la clasificación de la instancia. Cada nodo del árbol es un test para alguno de los atributos de la instancia a clasificar. Cada rama descendente de cada nodo se corresponde con uno de los posibles valores que puede tomar el atributo con el que se corresponde ese nodo, como se puede ver en la figura 55.

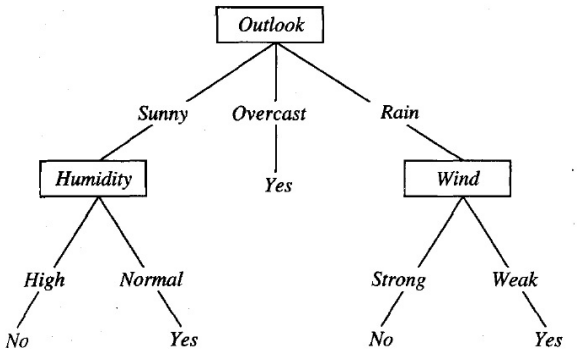


Figura 55: Figura obtenida de [Mitchel(1997)] éste es un árbol de decisión para saber si jugar o no al Tennis. Este árbol clasifica los sábados por la mañana según sean o no apropiados para jugar un partido de Tennis.

Para clasificar una instancia se empieza desde la raíz del árbol, testeando el atributo especificado por el nodo correspondiente y yendo hacia abajo por la rama que coincida con el valor del atributo testeado. Se repite el proceso hasta terminar en un nodo hoja. La figura 55 ilustra un típico árbol de decisión, en el que se decide si jugar o no un partido de Tennis, dependiendo de la meteorología prevista. Como ejemplo supongamos la siguiente instancia:

$$(\text{Outlook} = \text{Sunny}, \text{Temperature} = \text{Hot}, \text{Humidity} = \text{High}, \text{Wind} = \text{Strong})$$

En este caso, puesto que *Outlook=Sunny*, el árbol sería recorrido por la rama de la izquierda, hasta llegar al nodo correspondiente al atributo *Humidity*. En el ejemplo *Humidity=High*, por tanto se decantaría una vez más por la rama izquierda y la instancia sería clasificada como negativa, es decir, no jugar al Tennis.

En general, los árboles de decisión representan una disyunción de los valores de los atributos de las instancias. Cada camino desde la raíz del árbol hasta la hoja corresponde a un conjunto de tests de los atributos y el árbol en sí mismo, una disyunción de estas conjunciones. Por ejemplo para que la decisión del árbol representado en la figura 55 sea que sí se juegue al Tennis, tenemos la siguiente expresión.

$$\begin{aligned}&(\text{Outlook} = \text{Sunny} \wedge \text{Humidity} = \text{Normal})\\ &\vee (\text{Outlook} = \text{Overcast})\\ &\vee (\text{Outlook} = \text{Rain} \wedge \text{Wind} = \text{Weak})\end{aligned}$$

Esta expresión agrupa las tres combinaciones de atributos posibles, para que el árbol clasifique una instancia como positiva:

1. Si el pronóstico del tiempo es que estará nublado(Outlook = Overcast), siempre se recomendará jugar.

2. Si el pronóstico es soleado y además la humedad es normal.

3. Si el pronóstico es lluvia y además el viento es débil.

Fijémonos ahora en un ejemplo más propio del campo de la medicina, en el que se enmarca el proyecto. En la figura 56 observamos un árbol similar

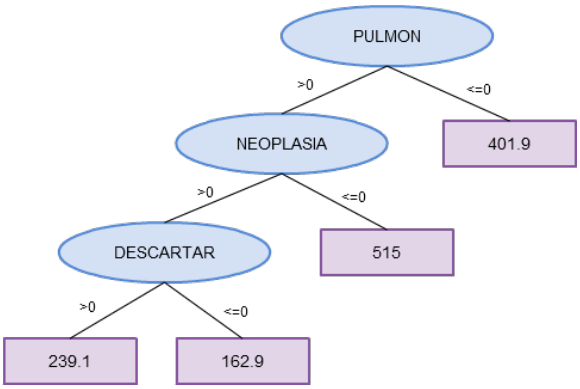


Figura 56: Figura que representa un árbol similar a los generados por Weka [Ian H. Witten(2011)] a partir de los datos utilizados para este proyecto.

a los generados por Weka a partir de los datos utilizados para este proyecto. Estos datos se encuentran representados en forma de vector de palabras, de manera que cada atributo tiene asociada una frecuencia de aparición. Por ejemplo; supongamos que nos llega la instancia "NEOPLASIA DE PULMON". Como contiene la palabra "PULMON", o lo que es lo mismo, la frecuencia de aparición de ésta palabra, es mayor que 0, el árbol de la figura 56, continuará la ejecución por la rama izquierda. A continuación llegará a un nodo que preguntará por la frecuencia de la palabra "NEOPLASIA", y como ésta es mayor que 0, continuará por la rama izquierda. LLegamos a un nodo que pregunta por la frecuencia de la palabra "DESCARTAR", y como en este caso la frecuencia de aparición es 0, nos decantamos por la rama derecha, llegando así a un nodo hoja, que asignará a nuestro ejemplo la clase 162.9 del código CIE-9-CM.

Las librerías de Weka ofrecen la implementación del algoritmo J48, que genera árboles de decisión. Para este trabajo utilizaremos este algoritmo y optimizaremos su funcionamiento para nuestros conjuntos de datos.

8.1.3 Random Forest - Bosques aleatorios

Los Bosques aleatorios, son unos clasificadores que utilizan n árboles de decisión, caracterizados por la utilización del método *Bagging* que se detalla a continuación.

> El Bagging es un método de votación formado por varios modelos-base que se construyen entrenando con conjuntos de entrenamiento ligeramente diferentes. Se generan L muestras de entrenamiento diferentes mediante *bootstrap*, de forma que, dado el conjunto de entrenamiento X de tamaño N, se toman N instancias aleatorias de X con reemplazamiento. Como la muestra se obtiene con reemplazamiento, es posible que algunas instancias se repitan y que otras no se incluyan en el conjunto muestral. Cuando se utiliza este método para generar L muestras $X_j, j = 1, ..., L$, estas muestras son similares porque han sido tomadas del mismo conjunto original, pero también son ligeramente diferentes gracias al azar.

El Random Forest genera muchos árboles o modelos-base siguiendo el método de *Bagging*, de forma que en vez de elegir el atributo que más información a priori aporta, elige uno aleatorio para cada árbol y al final, a

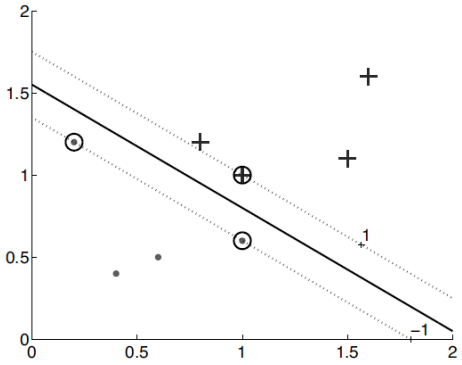


Figura 57: Figura obtenida de [Alpaydin(2010)] Vemos una función que marca el margen de una de las dos clases presentes en el problema, puntos o pluses. Las instancias redondeadas son los soportes vectoriales.

cada instancia se le asigna la clase más votada por los diferentes árboles generados.

Este método es muy útil para reducir problemas de sobreajuste, ya que, gracias al azar los conjuntos de datos serán siempre ligeramente diferentes y el modelo no se adaptará demasiado a un conjunto en concreto. En este caso, el conjunto de entrenamiento no cuenta con una instancia de cada clase, y mucho menos con varias instancias de cada clase. Por lo tanto, es muy importante que el modelo no se sobreajuste a los datos del conjunto de entrenamiento.

8.1.4 Support Vector Machine o Máquina de Soporte Vectorial

Las de soporte vectorial (SVM), inducen separadores lineales o hiperplanos en espacios de características de muy alta dimensionalidad. Suponiendo que el conjunto de ejemplos es separable, ¿cuál es el mejor hiperplano separador? La idea que hay detrás de las SVM de margen máximo consiste en seleccionar el hiperplano separador que está a la misma distancia de los ejemplos más cercanos de cada clase, como podemos observar en la figura 57. Dicho de otra manera, es el hiperplano que maximiza la distancia mínima entre los ejemplos del conjunto de datos y el hiperplano. Intuitivamente este hiperplano está situado en la posición más neutra posible con respecto a las clases representadas por el conjunto de datos, sin estar sesgado, por ejemplo hacia la clase más numerosa.

Definición formal. *"Sea* X *el dominio de todos los documentos, e* Y *un conjunto de categorías predefinidas. El objetivo es aprender una función* h *tal que dado un documento* $x \in X$*, nos devuelva el conjunto de categorías para tal documento,* $h(x) \subseteq Y$*. Para ello dispondremos de un conjunto de aprendizaje* $S = \{(x,y)\}_{i=1\ldots m}$*, formado por ejemplos* (x,y)*, donde* x *es un documento de* X *e* y *es un conjunto de categorías en* Y*. Para completar la experimentación también se dispondrá de un conjunto de evaluación o test, que se utilizará para evaluar la corrección de la función aprendida."* Definición obtenida de [José Hernandez Orallo(2004)]

El método de las SVM ha demostrado, hoy por hoy, superioridad en el problema de la clasificación de textos respecto a otros métodos, obteniendo los mejores resultados hasta la fecha en los conjuntos estándar de evaluación.

Es por ello que hemos incluido en la experimentación, una variante del SVM (el SMO) cuya implementación está incluida en las librerías de Weka.

8.2 MARCO EXPERIMENTAL

En esta sección se analizarán los resultados de los experimentos realizados para tratar de dar solución al problema que se plantea. Los experimentos se han realizado utilizando varias técnicas de evaluación, de las que se habla en el capítulo de introducción, en la sección 1.7. Se han explorado diferentes técnicas de evaluación, con el fin de obtener unos resultados fiables.

8.2.1 Desarrollo del modelo de clasificación

A continuación se detalla el proceso de desarrollo de los modelos de clasificación. Para esta tarea, dispondremos de un conjunto de entrenamiento o *train*, y un conjunto de desarrollo o *dev*, que será el que utilicemos para evaluar el modelo, y que contiene la misma cantidad de instancias que el conjunto final de test.

En primer lugar, se toma un algoritmo de aprendizaje implementado en las librerías de Weka. Se realiza un barrido de n parámetros construyendo un clasificador con el conjunto de entrenamiento para cada parámetro y evaluando el resultado. Ésta evaluación podría hacerse mediante una técnica de *k-foldCrossValidation*, cuya mayor ventaja es su fiabilidad. Sin embargo, dado su alto coste computacional y temporal y teniendo en cuenta que habría que evaluar el modelo $n * k$ veces y que el conjunto de entrenamiento puede tener del orden de 23130 instancias, esta técnica queda descartada para la fase de entrenamiento. Por lo tanto se decide utilizar la técnica de evaluación *Holdout*, que aporta resultados suficientemente fiables como para que el *software* pueda decantarse por un parámetro óptimo.

Una vez obtenido el parámetro óptimo, se utiliza para construir el clasificador y finalmente se evalúa el modelo con el conjunto de desarrollo o *dev*, y se obtiene un fichero con las predicciones y una salida por consola de las figuras de mérito.

8.2.2 Elección de las formas de evaluación del modelo

Teniendo en cuenta las ventajas y desvatajas de las tres técnicas de evaluación presentadas en el primer capítulo, en la sección 1.7.2. En primer lugar utilizamos la evaluación por resustitución, con el fin de obtener nuestra cota superior de aciertos. Para la fase de entrenamiento elegimos evaluar el modelo generado para cada parámetro mediante la técnica de evaluación *holdout*. Tal como se detalla en la sección 1.7.2, aunque los resultados que aporta el *holdout* son menos realistas que el *10-foldCrossValidation*, reducimos en gran medida el tiempo de ejecución, que en esta fase, se ve aumentado por el barrido de parámetros. Finalmente la evaluación por *k-foldCrossValidation* se ha aplicado al final de la experimentación para ser capaces de aportar unas cifras de precisión lo más realistas posibles sobre la calidad esperada del modelo entrenado.

8.2.3 Elección de los conjuntos de datos

Respecto a la elección de los atributos adicionales, recogemos la conclusión obtenida en el capítulo anterior de preproceso, en la sección 7.2 y únicamente incluiremos el atributo "sinónimo". En cuanto a la forma de representación, se ha elegido la representación nominal, ya que, según los resultados obtenidos en el preproceso, esta técnica tiene un coste temporal menor que el *StringToWordVector* y se obtienen buenos resultados. Sin embargo, con esta forma de representación, no ha sido posible probar el clasificador SMO, debido a que los recursos de memoria de las máquinas de la Escuela, no son suficientes para abordar el alto coste computacional de este clasificador.

Aunque hay que mencionar que con la representación en forma de vector de palabras, el SMO tenía muy buenos resultados. Como podemos observar en la tabla 29, sin ningún atributo adicional, obtiene un ratio de aciertos (TPR) de 0.889. Con el atributo sinónimo, el SMO mantiene su superioridad respecto del resto de clasificadores y alcanza un ratio de aciertos de 0.970 frente al 0.939 del *Random Forest*, tal y como podemos observar en la tabla 30.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.181	0.028	0.163	0.181	0.131	0.913
J48	0.851	0	0.854	0.851	0.843	0.962
RandomForest	0.881	0	0.883	0.881	0.876	0.959
SMO	**0.889**	0	0.88	0.889	0.878	0.98

Tabla 29: Representación en forma de *StringToWordVector* sin nuevos atributos. Evaluación por *Holdout*.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.394	0.02	0.328	0.394	0.312	0.959
J48	0.951	0	0.947	0.951	0.947	0.981
RandomForest	**0.939**	0	0.942	0.939	0.937	0.976
SMO	**0.97**	0	0.96	0.97	0.964	0.989

Tabla 30: Representación en forma de *StringToWordVector* con sinónimos. Evaluación por *Holdout*.

8.2.4 Resultados de los barridos de parámetros

Como se ha visto en el marco teórico, se han explorado 4 clasificadores: *Naive Bayes*, J48 (árbol de decisión), *Random Forest* y SMO (máquina de soporte vectorial). Para cada uno de ellos se han estudiado los parámetros que influyen en su rendimiento y se ha implementado el *software* necesario para realizar barridos de parámetros y encontrar el parámetro que optimice los resultados del modelo, para nuestros conjuntos de datos en particular.

- ***Naive Bayes***. En este caso no se realiza barrido de parámetros, ya que el clasificador, asignará la clase más probable a priori, asumiendo independencia estadística.
- **Árbol de decisión (J48)**. En el caso del árbol de decisión el parámetro a optimizar es el `MinNumObj`; que establece el número mínimo de instancias que habrá en cada hoja del árbol. Tras aplicar el *software* de barrido de parámetros a nuestros conjuntos de datos, se obtiene que el valor óptimo es `MinNumObj=2`. Además, tal como comentamos en el

marco teórico, es importante tener en cuenta que el modelo no debe sobreajustarse al conjunto de entrenamiento. Por tanto, establecemos el parámetro `Unpruned = false`. Así conseguimos que se pode el árbol, para evitar que el modelo se adapte perfectamente al conjunto de datos de entrenamiento.

- **Bosque aleatorio o *Random Forest***. En este caso nos interesa averiguar el valor óptimo de `NumTrees`. Este parámetro establece cuántos árboles generará el modelo escogiendo *features* aleatorias. Una vez aplicado el *software* de barrer parámetros, se obtiene que el óptimo se encuentra en 9 árboles.

- **SMO**. En este caso nos encontramos que en los primeros experimentos con resustitución, como vemos en la tabla 32, se alcanza un ratio de aciertos del 100 %. Por tanto vemos que con la opción de normalizar los datos y el `PolyKernel` se alcanzan los mejores resultados.

8.2.5 Resultados de la evaluación por resustitución

En esta subsección se presentan los resultados obtenidos de una evaluación por resustitución o no honesta, que por lo tanto tomaremos como cota superior. De esta forma podremos establecer un *baseline* y a partir de ahí decidir cómo seguir.

La tabla 31 muestra los resultados obtenidos tras preprocesar los datos, quitarles los acentos, poner todos los datos en mayúsculas y poner una contrabarra delante de las comas para que no se tomen como atributos, pero sin añadir nuevas *features* o atributos.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.489	0.019	0.571	0.489	0.497	1
J48	0.974	0	0.975	0.974	0.976	1
Random Forest	0.974	0	0.975	0.974	0.973	1

Tabla 31: Representación en forma nominal, datos normalizados, sin nuevos atributos.

En este caso analizamos los resultados obtenidos tras añadir las *features* o atributos comentados en el capítulo dedicado al preproceso de los datos 7.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Naive Bayes	0.851	0.002	0.794	0.851	0.81	0.999
J48	**1**	0	1	1	1	1
Random Forest	**1**	0	1	1	1	1

Tabla 32: Representación en forma nominal, datos normalizados, incluyendo el nuevo atributo.

Se observa que tanto en la tabla 31 como en la tabla 32 se produce un empate entre el árbol de decisión y el bosque aleatorio. Vemos en la tabla 32 que con el nuevo atributo sinónimo, ambos clasificadores alcanzan un ratio de aciertos (TPR) del 100 % y puesto que el coste temporal de estos clasificadores es mucho menor que el del SMO (del que no se han llegado a obtener resultados debido a su coste computacional), decidimos centrar la atención en estos dos clasificadores. Por otra parte, en caso de tener que elegir entre uno de los dos clasificadores, tomaríamos el *Random Forest*. Esta elección se debe a que la aleatoriedad del *Random Forest* previene problemas

de sobreajuste. Es decir, es más difícil que el modelo se ajuste demasiado al conjunto de entrenamiento y luego no sea capaz de generalizar.

Insistimos en que ésta es una cota optimista, ya que la evaluación se ha realizado con el mismo conjunto de datos que el entrenamiento. Esto supone que no hay ninguna instancia que el modelo no hubiese visto previamente. Puesto que el código CIE-9-CM tiene alrededor de 14.000 instancias y el corpus disponible cuenta con 28.830, es posible que el modelo reduzca mucho el rendimiento cuando se enfrente a instancias nuevas.

8.2.6 Resultados de la evaluación honesta

Una vez obtenido el *baseline* y los parámetros óptimos para cada uno de los modelos, se procede a evaluar los resultados mediante evaluación por *holdout*. Nos referimos a esta evaluación como "honesta", porque en la fase de entrenamiento se le pasan unos datos, y en la fase de test se le pasan otros datos que el modelo no ha visto previamente.

En la tabla 33, se observan los resultados obtenidos sin añadir nuevos atributos.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Naive Bayes	0.478	0.019	0.571	0.478	0.489	0.985
J48	**0.815**	0.006	0.869	0.815	0.821	0.975
Random Forest	**0.816**	0.006	0.87	0.816	0.822	0.976

Tabla 33: Representación en forma nominal, datos normalizados, sin nuevos atributos.

En el caso de la tabla 33, como era de esperar el rendimiento del modelo se reduce si se comparan los resultados con los de la evaluación por resustitución 31. Por ejemplo, si tomamos los resultados del clasificador *Random Forest*, observamos que en la misma situación pero en la evaluación por resustitución, se alcanza un ratio de aciertos (TPR) de 0.974 y sin embargo cuando el mismo modelo recibe un conjunto de datos para el test diferente al del entrenamiento, el ratio de aciertos disminuye hasta 0.816. También se observa que la diferencia de aciertos entre el árbol de decisión y el bosque aleatorio no es muy significativa, ya que apenas mejora en un 0.001. Como sabemos por los experimentos anteriores, el rendimiento mejora con la inclusión del atributo sinónimo. Por tanto, veamos qué sucede con esa nueva *feature* pero con una evaluación honesta o *holdout*. La tabla 34 muestra los resultados de este experimento. En el caso de la tabla 34, observamos que

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Naive Bayes	0.804	0.003	0.739	0.804	0.751	0.995
J48	**0.979**	0.001	0.964	0.979	0.97	0.989
Random Forest	**0.975**	0.001	0.961	0.975	0.966	0.989

Tabla 34: Representación en forma nominal, datos normalizados, incluyendo el nuevo atributo.

el ratio de aciertos del árbol de decisión supera por 0.004 al bosque aleatorio. Es decir, adelanta y cuadruplica la diferencia con el *Random Forest*. Esto supone que el J48 no sólo se comporta mejor que en la situación anterior donde su ratio de aciertos era de 0.815 si no que además parece resultar una opción más adecuada que el bosque aleatorio.

A la vista de los resultados obtenidos y puesto que no hay un clasificador claramente mejor que otro, ya que dependiendo de la situación puede ser

mejor el *Random Forest* o el J48, pensamos que una evaluación que reduzca al mínimo la varianza de los resultados será determinante para decantarnos por uno de los dos modelos de clasificación.

8.2.7 Resultados de la evaluación por *k-foldCrossValidation*

Procedemos a realizar una evaluación por *10-foldCrossValidation* para saber con la mayor precisión posible cuál es el mejor modelo para solucionar el problema que se nos plantea. En la tabla 35 podemos ver los resultados obtenidos sin ninguna *feature* nueva, y la comparamos con los resultados de *10-foldCrossValidation*, pasándole un conjunto de entrada con la *feature* sugerida en el capítulo de preproceso. Se observa que en ambos casos el *Random Forest* es el clasificador que mejores resultados aporta, ya que, con el atributo sinónimo, alcanza un ratio de aciertos del 0.884, superior al 0.581 que se consigue con el J48. 7.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Naive Bayes	0.432	0.021	0.541	0.432	0.446	0.955
J48	0.036	0.036	0.001	0.036	0.003	0.448
Random Forest	**0.58**	0.015	0.713	0.58	0.613	0.835

Tabla 35: Representación en forma nominal, datos normalizados, sin nuevos atributos. Evaluación por *10-foldCrossValidation*.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Naive Bayes	0.773	0.004	0.725	0.773	0.721	0.99
J48	**0.581**	0.015	0.713	0.581	0.613	0.839
Random Forest	**0.884**	0.007	0.836	0.884	0.851	0.918

Tabla 36: Representación en forma nominal, datos normalizados, con el nuevo atributo. Evaluación por *10-foldCrossValidation*.

8.2.8 Conclusiones sobre los clasificadores

Se han explorado un total de cuatro clasificadores, *Naive Bayes*, J48, *Random Forest* y SMO. El último de ellos, queda descartado debido a su alto coste computacional con la representación en forma nominal. Sin embargo, lo recomendamos para la forma de representación en vector de palabras (`StringToWordVector`).

Respecto al *Naive Bayes*, observamos que sus resultados con el nuevo atributo (0.773 de ratio de aciertos), son buenos si tenemos en cuenta la simplicidad de este clasificador, que hace la fuerte suposición de la independencia condicional entre atributos.

En cuando a la "competición" entre el J48 y *Random Forest*, vemos que en la evaluación por *holdout*, el J48(0.979) supera al bosque aleatorio(0.975). Sin embargo, en la evaluación por *10-foldCrossValidation*, el *Random Forest* es el que supera con una diferencia de 0.303 sobre el J48. Teniendo en cuenta que esta última forma de evaluación aporta los resultados más realistas, y teniendo en cuenta que el *Random Forest* tarda cerca de 6 veces menos que el J48 (11.586.648.532 nanosegundos frente a 59.897.975.013), concluímos que la mejor opción es el algoritmo de clasificación *Random Forest*.

9 SOFTWARE

9.1 USO DEL SOFTWARE

El software desarrollado en este proyecto está compuesto por 5 ejecutables `.jar` independientes entre sí. Para ejecutar cada uno de los `.jar`, se lanza un comando por consola pasándole los argumentos adecuados. En la figura 58 se muestra un esquema de las 5 partes de las que consta el software indicando el orden lógico en que deberían ejecutarse.

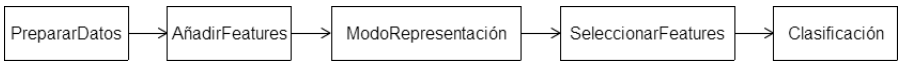


Figura 58: Esquema del software.

A continuación se explica cada uno de los ejecutables:

1. PrepararDatos.jar La funcionalidad que lleva a cabo este ejecutable es la generación de un fichero en formato `.arff` a partir de un fichero en formato `.txt`. Se puede elegir si además se desea normalizar los datos o si se prefiere dejarlos como están. Ambas funcionalidades se explican en el Marco teórico del Preproceso en las secciones 7.1.1 y 7.1.2, dedicadas a la Preparación de los datos y a la Normalización respectivamente.

   Se requieren dos argumentos:

   a) El primero de ellos la ruta del fichero `.txt` que se quiere covertir a `.arff`.

   b) El segundo la opción correspondiente a la normalización o no normalización de los datos, un 0 si no se quiere normalizar y un 1 si se quiere normalizar.

   El comando que se lanza en consola para ejecutar este `.jar` será:

   ```
   nice nohup java -jar -Xmx4000M PrepararDatos.jar
   rutaTxt.txt [0,1] > salidaInfo.txt
   ```

   Donde `nice` es el comando utilizado para aportar prioridad al proceso, `nohup` es el comando utilizado para ejecutar el proceso en el *background* y que de esta manera se ejecute de forma independiente de la sesión no interrrumpiéndose incluso en caso de salir de la terminal. `-Xmx` se utiliza para aumentar el tamaño de memoria de la máquina virtual en Java, en este caso la aumentamos a 4000M. Finalmente, salidaInfo.txt es el fichero al que redirigimos la salida estándar del proceso. En este caso contendrá, si se produjesen, errores relativos a los argumentos, número de argumentos incorrectos o argumento con un valor invalido.

   El fichero `.arff` producido se genera en la misma ruta que el fichero `.txt` pasado como argumento, con el mismo nombre cambiándole la extensión de `.txt` a `.arff`.

En la figura 59 se muestra de manera esquematizada la entrada y salida de PrepararDatos.jar.

Figura 59: PrepararDatos.jar.

2. AñadirFeatures.jar

   La funcionalidad que lleva a cabo este ejecutable es la adición de nuevas *features* o atributos a las instancias de un fichero en formato `.arff`. Esta funcionalidad se explica en el Marco teórico del Preproceso en la sección 7.1.4, dedicada a la generación de nuevos atributos.

   Se requieren cuatro argumentos:

   a) El primero de ellos la ruta del fichero `.arff` que se quiere tratar.

   b) El segundo la ruta de un fichero `.arff` que contenga el estándar del ICD-9-CM (clasificación estándar explicada en el capítulo 6 sobre los datos).

   c) El tercero la ruta de un fichero `.arff` que contenga una lista de sinónimos para los estándares del ICD-9-CM.

   d) El cuarto la opción correspondiente a cuántos atributos se desea añadir, un 0 si se quieren añadir todos los atributos disponibles y un 1 si sólo se desea añadir el atributo sinónimos.

   El comando que se lanza en consola para ejecutar este archivo será:

```
nice nohup java -jar -Xmx4000M AñadirFeatures.jar
rutaArff.arff rutaICD.arff rutaSinonimos.arff [0,1] >
salidaInfo.txt
```

   En este caso, el fichero salidaInfo.txt contendrá la misma información indicada para el ejecutable PrepararDatos.jar. Errores relativos a los argumentos, número de argumentos incorrectos o argumento con un valor invalido.

   El fichero `.arff` producido se genera en la misma ruta que el fichero `.arff` introducido como primer argumento, con el mismo nombre añadiendo al final "ConFeatures".

   En la figura 60 se muestra de manera esquematizada la entrada y salida de AñadirFeatures.jar.

Figura 60: AñadirFeatures.jar.

3. ModoRepresentacion.jar

   La funcionalidad que lleva a cabo este ejecutable es la representación de un fichero en formato `.arff` mediante la técnica escogida, *BagOfWords* o *Nominal*, ambas explicadas en el Marco teórico del Preproceso en la sección 7.1.3.

Se requieren dos argumentos:

a) El primero de ellos la ruta del fichero `.arff` que se quiere representar.

b) El segundo la opción correspondiente al tipo de representación deseado, un `0` para representación en forma de vector y un `1` para representación en forma nominal.

El comando que se lanza en consola para ejecutar este archivo será:

```
nice nohup java -jar -Xmx4000M ModoRepresentacion.jar
rutaArff.arff [0,1] > salidaInfo.txt
```

El fichero salidaInfo.txt contendrá la misma información indicada para los ejecutables anteriores.

El fichero `.arff` producido se genera en la misma ruta que el fichero `.arff` introducido como argumento, con el mismo nombre añadiendo al final "*WordVector*" o "*Nominal*" según se elija representación en forma de vector o en forma nominal respectivamente.

En la figura 61 se muestra de manera esquematizada la entrada y salida de ModoRepresentacion.jar.

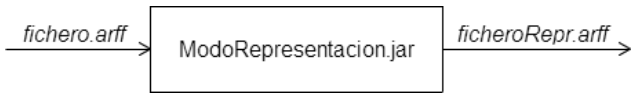


Figura 61: ModoRepresentacion.jar.

4. SeleccionarFeatures.jar

   La funcionalidad que lleva a cabo este ejecutable es la selección de *features* o atributos de las instancias de un fichero en formato `.arff`. Todas las técnicas disponibles en éste ejecutable para realizar la selección, se explican en el Marco teórico del Preproceso, en la sección 7.1.5 dedicada a la Reducción dimensional.

   Se requieren tres argumentos:

   a) La ruta del fichero `.arff` usado para entrenamiento.

   b) La ruta del fichero `.arff` usado para evaluación.

   c) La opción correspondiente al método de selección que se desea utilizar, `0` para *InfoGain*, `1` para *TFIDF*, `2` para *Remove*, `3` para *subsetSelectionClassifier*, `4` para *SubsetSelectionCFS*, `5` para *SubsetSelectionWrapper* y `6` para *PCA*.

   El comando que se lanza en consola para ejecutar este archivo será:

```
nice nohup java -jar -Xmx4000M SeleccionarFeatures.jar
rutaTrain.arff rutaTest.arff [0,1,2,3,4,5,6] >
salidaInfo.txt
```

   El fichero salidaInfo.txt contendrá la información indicada para los ejecutables anteriores relativa a los errores en argumentos, además del tiempo de ejecución transcurrido desde el inicio hasta el fin y algunos datos sobre el proceso realizado. Número de atributos seleccionados, atributos seleccionados, el rango de atributos mantenido en caso de utilizar la técnica *Remove* y el *Ranking* generado entre otros.

El fichero `.arff` producido se genera en la misma ruta que el fichero `.arff` introducido como primer argumento, con el mismo nombre añadiendo al final la técnica de selección utilizada.

En la figura 62 se muestra de manera esquematizada la entrada y salida de SeleccionarFeatures.jar.

Figura 62: SeleccionarFeatures.jar.

5. Clasificacion.jar

   La funcionalidad que lleva a cabo este ejecutable es la construcción de un modelo realizando el entrenamiento con un conjunto de instancias de un fichero en formato `.arff` y la clasificación de otro conjunto de instancias, pudiendo así evaluar el modelo construido. Los clasificadores disponibles en este ejecutable para realizar la clasificación se explican en el Marco Teórico del capítulo 8.

   Se requieren seis argumentos:

   a) La ruta del `.arff` utilizado para entrenar.

   b) La ruta del `.arff` utilizado para evaluar.

   c) La opción correspondiente al clasificador que se desea utilizar, 0 para *NaiveBayes*, 1 para *J48*, 2 para *RandomForest*, 3 para *SMO* y 4 para *BayesNet*.

   d) La opción correspondiente al barrido de parámetros, 0 para barrer y 1 para hacerlo sin barrido.

   e) La opción correspondiente a la evaluación, 0 si se desea evaluar con *Holdout* y 1 si se prefiere evaluar con evaluación cruzada *(k-fold cross validation)*.

   f) El número de *folds* con que se quiere hacer la evaluación cruzada en caso de haber elegido ésta.

   El comando que se lanza en consola para ejecutar este archivo será:

```
nice nohup java -jar -Xmx4000M Clasificacion.jar
rutaTrain.arff rutaTest.arff [0,1,2,3,4] [0,1] [0,1] n >
salidaInfo.txt
```

   El fichero salidaInfo.txt contendrá la información indicada para los ejecutables anteriores relativa a los errores en argumentos, además del tiempo de ejecución transcurrido desde el inicio hasta el fin y algunos datos sobre el proceso realizado. El número de instancias correctamente clasificadas y distintas medidas de calidad del modelo, como *f-measure*, *True Positive Rate* o *ROC Area* entre otros.

   El fichero `.arff` producido con las predicciones realizadas por el clasificador escogido, se genera en la misma ruta que el fichero `.arff` introducido como primer argumento, con el mismo nombre añadiendo al final "Predicciones" y el nombre del clasificador utilizado.

   En la figura 63 se muestra de manera esquematizada la entrada y salida de Clasificacion.jar.

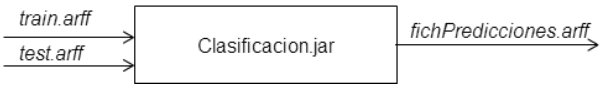


Figura 63: Clasificacion.jar.

10 CONCLUSIONES Y TRABAJO FUTURO

10.1 CONSECUCIÓN DE OBJETIVOS Y ANÁLISIS DE RESULTADOS

Habiendo finalizado el proyecto y a la vista de los resultados obtenidos, se observa que siendo nuestro umbral optimista de 1, tal y como veíamos en el capítulo 8, el mejor resultado realista obtenido, mediante la unión de las mejores técnicas de preprocesamiento con el mejor clasificador, es 0,884. Por lo tanto, y teniendo en cuenta la disminución en la precisión es de un 11,6 %, concluimos que aunque los resultados son mejorables, también se ajustan a los objetivos establecidos al principio del trabajo. Además, hay que mencionar que la precisión de los primeros experimentos lanzados rondaba el 0,20. Por lo tanto, creemos que el progreso es notable. A continuación presentamos las conlusiones obtenidas a la vista de los resultados de los experimentos.

- Preproceso:
  - La mejor forma de representar los datos es la nominal, ya que aunque se puede observar una ligera mejora en los resultados ofrecidos con los datos representados en forma de vector, esta manera de representación supone un aumento considerable en el coste computacional en el proceso de clasificación.
  - Los resultados mejoran notablemente cuando se incluye la *feature* sinónimo.
  - Aunque el desarrollo del resto de *features* ha resultado interesante y enriquecedor, no se pueden incluir en el modelo final, debido a que son *features* "tramposas" (se han extraído comparando las instancias con el ICD-9-CM). Dicho esto, han resultado muy útiles para comprobar que nuestro sistema de selección de atributos funciona correctamente, ya que las elige como las más informativas. Además estos atributos resultarán cruciales en el trabajo futuro, ya que en él se buscarán atributos tales como la distancia de Levenshtein entre las instancias o el número de Ngramas compartidos entre ellas. Para esta tarea se podrá reutilizar el *software* implementado.
- Clasificación:
  - El mejor algoritmo de clasificación es el *RandomForest*. Debido a la relación entre coste temporal y tasa de aciertos. Éste clasificador es óptimo para nuestro problema cuando genera 9 árboles, o lo que es lo mismo, el parámetro *numTrees* es igual a 9.

10.2 CONCLUSIONES Y CONTRIBUCIONES CIENTÍFICAS

La principal contribución científica de este proyecto es la publicación de un artículo en la revista del XXX Congreso de la Sociedad Española para el Procesamiento del Lenguaje Natural, con ISSN: 1135-5948 (edición empresa) y con ISSN: 1989-7553 (edición digital). El artículo se titula *The aid of machine learning to overcome the classification of real health discharge reports written in Spanish* y sus autores son; Alicia Pérez, Arantza Casillas, Koldo Gojenola, Maite Oronoz, Nerea Aguirre y Estibaliz Amillano. Se encuentra accesible en la url http://goo.gl/mhZWIf.

Además pensamos que este trabajo contribuye al campo del procesamiento del lenguaje natural en castellano, ya que, si bien es cierto que existen trabajos muy estrechamente relacionados con éste, también es cierto que, como ya se ha comentado previamente, la gran mayoría de trabajos relacionados están en inglés. Además, se ha abordado un problema de clasificación con un número elevado de clases, del orden de 14,000, a diferencia de otros trabajos previos que abordaban problemas con aproximadamente 45 clases diferentes.

10.3 COMENTARIOS E IMPRESIONES FINALES RESPECTO DE LA GESTIÓN

Gracias al desarrollo de este trabajo, hemos obtenido algunas conclusiones interesantes:

- Librerías de Weka: Se ha descubierto que algunas librebrías de Weka, en concreto el filtro *StringToWordVector*, no empieza a contar desde cero como es habitual en Java. Este hecho provocó dificultades en un principio ya que, se perdía el primer valor del atributo clase cada vez que se aplicaba, esto desembocaba en que todas las instancias con ese valor de clase se quedaban sin clasificar o se clasificaban erróneamente. Una vez descubierta la causa del problema, se solucionó introduciendo un 0 adicional como primer valor para el atributo clase.

- Coste temporal exagerado del método de clasificación *Bayes Net*: En un principio se pensó en el *Bayes Net* como un candidato válido para la tarea abordada. Sin embargo, tras la primera ronda de experimentos, se descartó debido a su elevado coste temporal.

- Utilidad e importancia de una buena planificación: Gracias a la planificación inicial hemos sido capaces de superar con éxito ciertos imprevistos. Por ejemplo, en un principio el grupo de trabajo contaba con un integrante más de la facultad de matemáticas de Leioa, cuyo proyecto iba a desarrollarse conjuntamente con el nuestro, aportando justificaciones razonadas a los experimentos desde el punto de vista matemático. Sin embargo a causa de las incompatibilidades horarias, decidió continuar individualmente el proyecto. Esta situación está contemplada en el apartado 2.6 de Análisis de Riesgos. De acuerdo con el plan de contingencia establecido para este tipo de situaciones, las actas de todas las reuniones estaban accesibles para todos los miembros del grupo de trabajo en Dropbox, por tanto durante las bajas tempora-

les, el tercer integrante estubo al tanto de todos los avances y trabajo pendiente. Una vez que la baja fué definitiva, el proyecto siguó su curso tomando las decisiones en base a los resultados obtenidos por el software y no a razonamientos matemáticos.

- Respecto a la planificación temporal, habría que comentar que dado el carácter de investigación del proyecto, la planificación del mismo, se realizó dando cierto margen a posibles complicaciones que pudieran alargar la fecha de finalización. Habiendo terminado y con una vista global de las horas invertidas, concluimos que no se ajustan del todo a lo planificado, ya que, una vez desarrollado todo el software, las horas dedicadas a la experimentación se dispararon respecto a lo que se había previsto en un primer momento, superando incluso la holgura establecida. Las horas que finalmente se han invertido superan en tres meses lo previsto en un principio, tal y como se puede observar en la figura 5, del capítulo "Planteamiento incial". Este retraso en la finalización del proyecto, se debe a que el tiempo previsto en un principio para redactar la memoria era insuficiente. Además a pesar de que estaba previsto que los integrantes del equipo trabajasen las mismas horas y prácticamente en paralelo, finalmente no ha sido así. Por tanto, debido a la necesidad de reflejar el trabajo individual de cada integrante del equipo, en las figuras 64 y 65 se muestra una comparativa del trabajo que ha desarrollado cada una junto con el previsto al comienzo del proyecto. Llama la atención que no exista tiempo estimado para la experimentación. Esto se debe a que dada la gran capacidad de cálculo de los ordenadores de hoy en día, se estimó que el tiempo dedicado a la experimentación sería despreciable y que se realizaría simultáneamente a otras tareas.

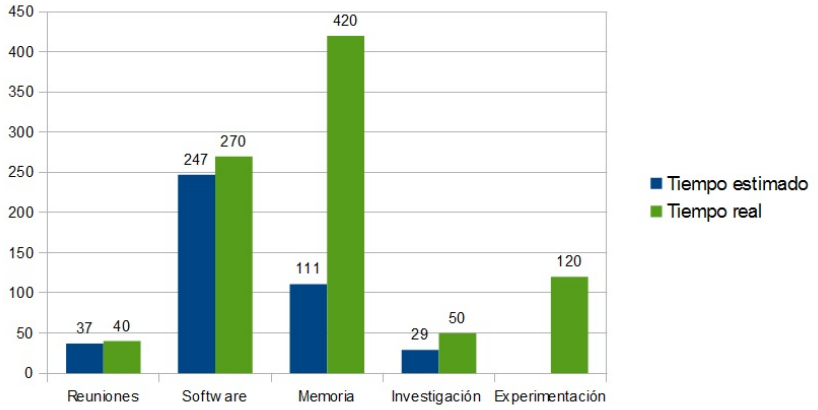


Figura 64: Comparación entre el tiempo estimado y el tiempo real invertido por Estibaliz Amillano.

- Relevancia del formato de los datos: Esta es una cuestión interesante, ya que en este tipo de estudios no se le suele dar mucha importancia al formato de los datos. Sin embargo, según nuestra experiencia, esta tarea ha ocupado más tiempo del esperado. Esto es debido a que los clasificadores no aceptan cualquier fomato de entrada, sumado a que es necesario tener muy en cuenta el formato de escritura (que tanto la lectura como la escritura se realicen en formato UTF-8).

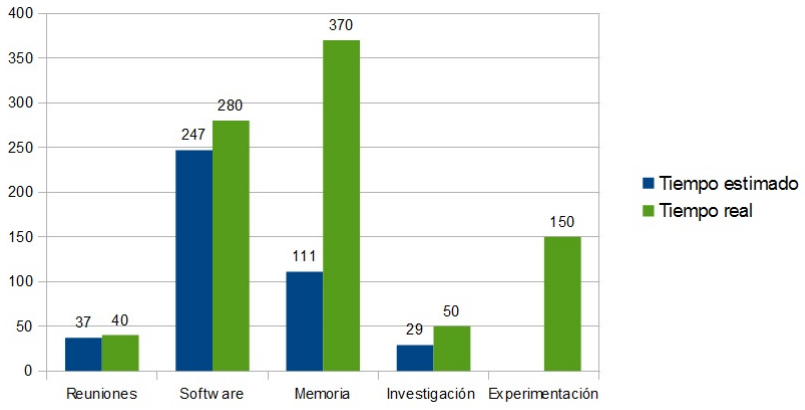


Figura 65: Comparación entre el tiempo estimado y el tiempo real invertido por Nerea Aguirre.

- El control de versiones: Este aspecto es de vital importancia dada la naturaleza grupal del trabajo, se ha hecho necesaria un control minucioso de las diferentes versiones, tanto del software como de la memoria. Esta tarea ha sido llevada a cabo vía Dropbox de forma que cada nueva versión llevaba asignada la fecha de modificación de la misma. Como ventaja asociada se tienen copias de seguridad de las sucesivas versiones, lo cual facilita volver a un punto seguro en caso de fallo o pérdida de datos.

- El trabajo en equipo: Creemos que ha sido una gran oportunidad para poner en práctica los conocimientos y metodologías de trabajo en equipo, aprendidas durante el Grado. Pensamos que el trabajo se aligera cuando hay más de un participante implicado en la realización de un proyecto, y ciertamente los picos de trabajo son más abordables, además de la rapidez que aporta el hecho de que dos personas busquen la solución a un problema inesperado en vez de una sola. Dicho esto, también hemos descubierto que esta forma de trabajar tiene sus inconvenientes; se incrementa el tiempo dedicado a coordinación y la necesidad de consensuar ciertas decisiones, las puede llegar a ralentizar. además es necesario un exhaustivo control de versiones, de otra forma se corre el riesgo de que los integrantes se "pisen" el trabajo entre sí. Por último, el tiempo invertido no siempre es multiplicable por 2, esto es debido a que algunas tareas sensibles, como las tareas de planificación y organización, requieren de ambos integrantes para su correcta realización.

10.4 TRABAJO FUTURO

Como trabajo futuro se plantea continuar con la implementación de lo que hemos llamado los *Clasificadores especializados*, que consiste en generar un clasificador por cada clase del conjunto de entrenamiento que coincida con alguna clase del ICD-9-CM, tal y como se observa en la figura 66. Se trata de clasificadores binarios que realizarán una clasificación binaria, es decir, "SI" o "NO" dependiendo de si la instancia a clasificar pertenece a la clase para la que se han construido o no. Esta implementación está empezada y muy

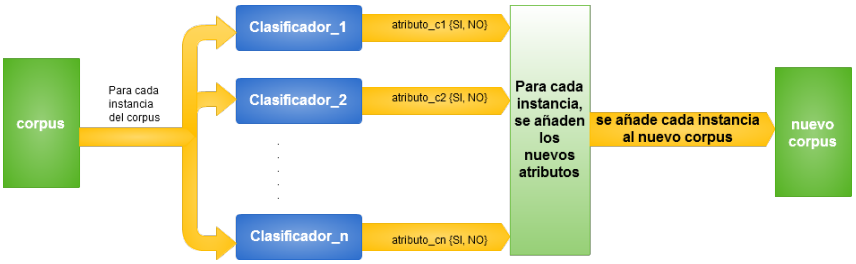


Figura 66: Diagrama que ilustra el funcionamiento del trabajo futuro propuesto.

encaminada para lo que, presumiblemente será otro Trabajo Fin de Grado. Ya que en esta misma línea se plantea la idea, de que estos clasificadores en vez de una predicción binaria, realicen una predicción probabilística, o dicho de otra manera, asignen una probabilidad de pertenencia o no a cada instancia. Esta probabilidad se incorporaría como nuevo atributo o *feature* al conjunto de datos y se terminaría el proceso con un último clasificador que asignase la clase correcta basándose en las predicciones de los *Clasificadores especializados*.

Por otra parte pensamos que se debería hacer una batería de experimentos con datos en el mismo formato pero diferentes, ya que, de esta forma quedaría probada la utilidad del software desarrollado. Respecto a los experimentos creemos que sería muy interesante repetir los más relevantes, debido a que prácticamente al final se ha incorporado un contador de tiempo. Así se podrían aportar datos exactos sobre el coste temporal de los algoritmos propuestos.

Parte II

Apéndices

A TABLAS DE RESULTADOS

En la sección Marco experimental de los capítulos 7 y 8, dedicados al Preproceso y la Clasificación respectivamente, se muestran varias tablas de resultados de experimentos realizados. Sin embargo, por temas de claridad no se muestran todos, si no sólo los realizados con el clasificador *Random Forest* debido a que como se menciona en el capítulo 8 dedicado a la Clasificación, ofrece buenos resultados sin un coste computacional excesivo. En este apéndice se exponen las tablas de resultados completas. Se muestran los resultados de los mismos experimentos utilizando el resto de clasificadores así como los resultados correspondientes a una experimentación "No honesta", es decir, utilizando los mismos datos para entrenar y para evaluar.

A.1 EXPERIMENTOS PREPROCESO

En esta sección se muestran las tablas completas de resultados correspondientes a los experimentos de las técnicas de Preproceso expuestos en la sección 7.2, dedicada al Marco Experimental del Preproceso.

A.1.1 Normalización

A continuación se muestran las tablas de resultados de los experimentos de Normalización. Se amplian los resultados expuestos en la sección 7.2.1. Se muestran los resultados correspondientes a la clasificación utilizando *NaiveBayes*, *J48* y *RandomForest*, clasificadores explicados en el capítulo 8, con los mismos datos. Sin normalizar en la tabla 37 y normalizados en la tabla 38. En las tablas 39 y 40 se muestran los resultados correspondientes a una experimentación utilizando una evaluación "No honesta". Los datos se encuentran sin normalizar y normalizados respectivamente.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.395	0.022	0.525	0.395	0.412	0.974
J48	0.766	0.008	0.85	0.766	0.782	0.965
RandomForest	0.766	0.008	0.85	0.766	0.782	0.966

Tabla 37: Tabla de resultados Completa - Sin normalización

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.478	0.019	0.571	0.478	0.489	0.985
J48	0.815	0.006	0.869	0.815	0.821	0.975
RandomForest	0.816	0.006	0.87	0.816	0.822	0.976

Tabla 38: Tabla de resultados Completa - Con normalización

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.42	0.021	0.536	0.42	0.432	1
J48	0.978	0	0.978	0.978	0.977	1

Tabla 39: Tabla de resultados no honestos completa - Sin normalización

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.489	0.019	0.571	0.489	0.497	1
J48	0.974	0	0.975	0.974	0.976	1
RandomForest	0.974	0	0.975	0.974	0.973	1

Tabla 40: Tabla de resultados no honestos completa - Con normalización

A.1.2 Representación del texto

A continuación se muestran las tablas de resultados de los experimentos de Representación de textos. Se amplian los resultados expuestos en la sección 7.2.2. Se muestran los resultados correspondientes a la clasificación utilizando *NaiveBayes*, *J48* y *RandomForest*, clasificadores explicados en el capítulo 8, con los mismos datos normalizados. Utilizando el método de representación *BagOfWords* en la tabla 41 y el Nominal en la tabla 42. En las tablas 43 y44 se muestran los resultados correspondientes a una experimentación utilizando una evaluación "No honesta". Los datos se encuentran normalizados en ambos casos y representados mediante *BagOfWords* y Nominal respectivamente.

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.181	0.028	0.163	0.181	0.131	0.913
J48	0.851	0	0.854	0.851	0.843	0.962
RandomForest	0.881	0	0.883	0.881	0.876	0.959

Tabla 41: Tabla de resultados completa - Representación BOW

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.478	0.019	0.571	0.478	0.489	0.985
J48	0.815	0.006	0.869	0.815	0.821	0.975
RandomForest	0.816	0.006	0.87	0.816	0.822	0.976

Tabla 42: Tabla de resultados completa - Representación nominal

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.1814	0.028	0.189	0.184	0.133	0.927
J48	0.909	0	0.905	0.909	0.902	1
RandomForest	0.97	0	0.969	0.97	0.967	1

Tabla 43: Tabla de resultados no honestos completa - Representación BOW

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.489	0.019	0.571	0.489	0.497	1
J48	0.974	0	0.975	0.974	0.976	1
RandomForest	0.974	0	0.975	0.974	0.973	1

Tabla 44: Tabla de resultados no honestos completa - Representación nominal

A.1.3 Nuevos atributos

A continuación se muestran las tablas de resultados de los experimentos de adición de nuevos atributos. Se amplian los resultados expuestos en la sección 7.2.3. Se muestran los resultados correspondientes a la clasificación utilizando *NaiveBayes*, *J48* y *RandomForest*, clasificadores explicados en el capítulo 8, con los mismos datos normalizados y representados mediante la técnica nominal. Utilizando los atributos originales (Text y Clase) en la tabla 45, todos los atributos disponibles, Originales (Text, Clase) + Nuevos (Término del ICD-9-CM, sinónimos, distancia de Levenshtein, número de N-gramms compartidos, número de palabras compartidas y número de palabras compartidas en la misma posición), en la tabla 46, y los atributos originales más el atributo sinónimos en la tabla 47. En las tablas 48, 49, y 50, se muestran los resultados correspondientes a los mismos experimentos utilizando una evaluación "No honesta".

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.478	0.019	0.571	0.478	0.489	0.985
J48	0.815	0.006	0.869	0.815	0.821	0.975
RandomForest	0.816	0.006	0.87	0.816	0.822	0.976

Tabla 45: Tabla de resultados completa - Sin atributos añadidos

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.804	0.003	0.739	0.804	0.751	0.995
J48	0.979	0.001	0.964	0.979	0.97	0.989
RandomForest	0.975	0.001	0.961	0.975	0.966	0.989

Tabla 46: Tabla de resultados completa - Con atributos añadidos

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.664	0.012	0.746	0.664	0.674	0.985
J48	0.815	0.006	0869	0.815	0.821	0.975
RandomForest	0.826	0.006	0.872	0.826	0.83	0.977

Tabla 47: Tabla de resultados completa - Con atributo sinónimos añadido

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.489	0.019	0.571	0.489	0.497	1
J48	0.974	0	0.975	0.974	0.976	1
RandomForest	0.974	0	0.975	0.974	0.973	1

Tabla 48: Tabla de resultados no honestos completa - Sin atributos añadidos

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.851	0.002	0.794	0.851	0.81	0.999
J48	1	0	1	1	1	1
RandomForest	1	0	1	1	1	1

Tabla 49: Tabla de resultados no honestos completa - Con atributos añadidos

CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
NaiveBayes	0.702	0.011	0.77	0.702	0.709	0.999
J48	0.974	0	0.975	0.974	0.973	1
RandomForest						

Tabla 50: Tabla de resultados no honestos completa - Con atributo sinónimos añadido

A.1.4 Reducción dimensional

A continuación se muestran las tablas de resultados de los experimentos de Reducción dimensional, se corresponden con los resultados expuestos en la sección 7.2.4. Los experimentos realizados en esta sección han sido llevados a cabo únicamente con el clasificador *Random Forest* puesto que así lo hemos creido conveniente por temas de eficiencia en cuanto a coste computacional y temporal. Se han realizado todos los experimentos utilizando los datos normalizados y representados mediante la técnica nominal, con todos los atributos disponibles y utilizando sólo los atributos originales y sinónimos. En las tablas 51 y 58 sin aplicarles ninguna técnica de reducción, en las tablas 52 y 59 se les ha aplicado la técnica *InfoGain*, en las tablas 53 y 60 se les ha aplicado la técnica *TF-IDF*, en las tablas 54 y 61 *CfsSubsetEval*, en las tablas 55 y 62 *ClassifierSubsetEval*, en la tabla 56 *WrapperSubsetEval* y en las tablas 57 y 63 el filtro *Remove*. Para La técnica *Wrapper* sólo hay resultados para los datos con todos los atributos disponibles, ésto se debe a que por problemas de capacidad de los equipos no conseguimos realizar los experimentos para los datos con los atributos originales y sinónimos.

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.975	0.001	0.961	0.975	0.966	0.989

Tabla 51: Tabla de resultados completa - Sin ninguna técnica de reducción dimensional

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.978	0.001	0.964	0.978	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
1. Término ICD-9-CM 2. Text 3. Sinónimos 4. Dist. Levenshtein 5. N-gramms comp. 6. Palabras comp. 7. Palabras en posición comp.	7	-

Tabla 52: Tabla de resultados completa - InfoGain.Ranking

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.976	0.001	0.959	0.976	0.966	0.989

Tabla 53: Tabla de resultados completa - TF-IDF

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	2	Término del ICD-9-CM, sinónimos

Tabla 54: Tabla de resultados completa - Cfs.Best first search

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	2	Término del ICD-9-CM, sinónimos

Tabla 55: Tabla de resultados completa - Classifier.Best first search

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.979	0.001	0.964	0.979	0.97	0.989

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Término del ICD-9-CM

Tabla 56: Tabla de resultados completa - Wrapper.Best first search

ATR. SELEC.	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Text Térm. ICD-9-CM Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.975	0.001	0.961	0.975	0.966	0.989
Térm. ICD-9-CM Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.979	0.001	0.964	0.979	**0.97**	0.989
Sinónimos Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.851	0.002	0.825	0.851	0.824	0.982
Dist. Levenshtein N-gramms comp. Palabras comp. Pal. Pos. comp.	0.395	0.008	0.286	0.395	0.309	0.935
N-gramms comp. Palabras comp. Pal. Pos. comp.	0.1	0.031	0.042	0.1	0.044	0.725
Palabras comp. Pal. Pos. comp.	0.094	0.032	0.023	0.094	0.032	0.721
Pal. Pos. comp.	0.069	0.034	0.008	0.069	0.014	0.675

Tabla 57: Tabla de resultados completa - Remove

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

Tabla 58: Tabla de resultados completa - Sin ninguna técnica de reducción dimensional

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
1. Text 2. Sinónimos	2	-

Tabla 59: Tabla de resultados completa - InfoGain.Ranking

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.826	0.006	0.872	0.826	0.83	0.977

Tabla 60: Tabla de resultados completa - TF-IDF

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.824	0.006	0.871	0.824	0.828	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Sinónimos

Tabla 61: Tabla de resultados completa - Cfs.Best first search

TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
0.824	0.006	0.871	0.824	0.828	0.977

RANKING	N° ATR. SEL.	ATRIBUTOS SELECCIONADOS
-	1	Sinónimos

Tabla 62: Tabla de resultados completa - Classifier.Best first search

ATR. SELEC.	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
Text Sinónimos	0.826	0.006	0.872	0.826	**0.83**	0.977
Sinónimos	0.824	0.006	0.871	0.824	0.828	0.977

Tabla 63: Tabla de resultados completa - Remove

A.2 EXPERIMENTOS CLASIFICACIÓN

En esta sección se muestran las tablas completas de resultados correspondientes a los experimentos de la Clasificación expuestos en la sección 8.2, dedicada al Marco Experimental de la Clasificación.

A.2.1 Resultados de los conjuntos de datos, representados de forma de *BagOfWords*

SET	EVALUATION	++FEATURES	CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
2+3	re-substitution	NO	NaiveBayes	0.1814	0.028	0.189	0.184	0.133	0.927
			J48	0.909	0	0.905	0.909	0.902	1
			RandomForest	0.97	0	0.969	0.97	0.967	1
			SMO	0.964	0	0.959	0.964	0.959	1
		YES	NaiveBayes	0.393	0.021	0.345	0.393	0.311	0.976
			J48	0.985	0	0.977	0.985	0.98	1
			RandomForest	1	0	1	1	1	1
			SMO						
	10fCV	NO	NaiveBayes	train - 0.109	0.032	0.11	0.109	0.066	0.864
				dev - 0.036	0.036	0.001	0.036	0.002	0.703
			J48	0.858	0.001	0.853	0.858	0.85	0.965
			RandomForest						
			SMO						
		YES	NaiveBayes						
			J48						
			RandomForest						
			SMO						
	dev	NO	NaiveBayes	0.181	0.028	0.163	0.181	0.131	0.913
			J48	0.851	0	0.854	0.851	0.843	0.962
			RandomForest	0.881	0	0.883	0.881	0.876	0.959
			SMO	0.889	0	0.88	0.889	0.878	0.98
		YES	NaiveBayes	0.394	0.02	0.328	0.394	0.312	0.959
			J48	0.951	0	0.947	0.951	0.947	0.981
			RandomForest	0.939	0	0.942	0.939	0.937	0.976
			SMO	0.97	0	0.96	0.97	0.964	0.989

SET	EVALUATION	++FEATURES	CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
2	re-substitution	NO	NaiveBayes	0.136	0.028	0.11	0.136	0.086	0.905
			J48	0.833	0.002	0.809	0.833	0.804	0.998
			RandomForest	0.949	0.001	0.95	0.949	0.939	0.999
			SMO	0.94	0.001	0.933	0.94	0.929	0.999
		YES	NaiveBayes						
			J48	0.956	0	0.934	0.956	0.942	1
			RandomForest	0.998	0	0.997	0.998	0.997	1
			SMO	1	0	1	1	1	1
	10fCV	NO	NaiveBayes						
			J48						
			RandomForest						
			SMO						
		YES	NaiveBayes						
			J48						
			RandomForest						
			SMO						
	dev	NO	NaiveBayes	0.227	0.03	0.114	0.227	0.137	0.987
			J48	0.542	0.002	0.421	0.542	0.457	0.997
			RandomForest	0.506	0.002	0.391	0.506	0.425	0.997
			SMO						
		YES	NaiveBayes						
			J48						
			RandomForest						
			SMO						

A.2.2 Resultados de los conjuntos de datos, representados de forma nominal

SET	EVALUATION	++FEATURES	CLASSIFIER	TPR	FPR	PRECISSION	RECALL	F-MEASURE	ROC AREA
2+3	re-substitution	NO	NaiveBayes	0.489	0.019	0.571	0.489	0.497	1
			J48	0.974	0	0.975	0.974	0.976	1
			RandomForest	0.974	0	0.975	0.974	0.973	1
			SMO						
		YES	NaiveBayes	0.851	0.002	0.794	0.851	0.81	0.999
			J48	1	0	1	1	1	1
			RandomForest	1	0	1	1	1	1
			SMO						
	10fCV	NO	NaiveBayes	0.432	0.021	0.541	0.432	0.446	0.955
			J48	0.036	0.036	0.001	0.036	0.003	0.448
			RandomForest						
			SMO						
		YES	NaiveBayes	0.773	0.004	0.725	0.773	0.721	0.99
			J48						
			RandomForest						
			SMO						
	dev	NO	NaiveBayes	0.478	0.019	0.571	0.478	0.489	0.985
			J48	0.815	0.006	0.869	0.815	0.821	0.975
			RandomForest	0.816	0.006	0.87	0.816	0.822	0.976
			SMO						
		YES	NaiveBayes	0.804	0.003	0.739	0.804	0.751	0.995
			J48	0.979	0.001	0.964	0.979	0.97	0.989
			RandomForest	0.975	0.001	0.961	0.975	0.966	0.989
			SMO						

B GLOSARIO

Accuracy

. Figura de mérito utilizada para medir la calidad de un clasificador, indica la tasa de aciertos de este.

Atributo

. Dentro del ámbito de la minería de datos se denomina atributos al conjunto de elementos que caracterizan una instancia. Se le llama atributo, característica o feature. La clase que identifica una instancia es un tipo especial de atributo.

Bag Of Words (BOW)

. Técnica de representación de textos que solo tiene en cuenta la presencia o ausencia de las palabras ignorando el orden de estas en el texto. Tras aplicar esta técnica ya no se dispone del texto con una estructura sintáctica. Se tiene una lista de palabras que aparecen en él pero no se conoce el orden.

Corpus

. Conjunto de datos utilizado para la realización de un proyecto.

Distancia de Mahalanobis

. Medida de distancia que permite determinar la similitud entre dos variables aleatorias multidimensionales. Tiene en cuenta la correlación entre las variables aleatorias.
Formalmente, la distancia de Mahalanobis entre dos variables aleatorias con la misma distribución de probabilidad $\vec{x}$ y $\vec{y}$ con matriz de covarianza Σ se define como: $d_m(\vec{x},\vec{y})=\sqrt{(\vec{x}-\vec{y})^T\Sigma^{-1}(\vec{x}-\vec{y})}$.

Entropía

. Medida de incertidumbre promedio, calculada a partir de la probabilidad de ocurrencia de cada uno de los eventos. (Teoría de la Información).
$-\sum_{k=1}^{K} p_k \log P_k$

Instancia

. Se llama instancia a cada entrada o ejemplo presente en los datos.

Matriz de confusión

. Herramienta para la visualización de la calidad de un clasificador utilizada en clasificación supervisada. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real.

N-gram Tokenization

. Técnica de selección de atributos que consiste en segmentar el texto en unidades básicas (tokens). Este proceso precede a casi todo lo demás que hagamos durante el procesamiento del texto. Existen distintos tipos de algoritmos (tokenizadores) para llevar a cabo esta técnica dependiendo de en qué tipo de unidades queramos dividir el texto, como por ejemplo, tokenizador por frases, tokenizador por espacios y tokenizador por palabras.
Eligiendo el tokenizador por palabras como muestra, la frase "Dividiendo el texto en unidades básicas" quedaría :
"Dividiendo", "el", "texto", "en", "unidades", "básicas".
Utilizando esta técnica perdemos dos tipos de información, la relación entre dos palabras y las palabras compuestas. Para solucionar esto podemos cambiar la estrategia de división. En vez de hacerlo por palabras individuales (Estrategia Unigram), lo podemos hacer por pares de palabras (Estrategia Bigram) o incluso de tres en tres (Estrategia Trigram).

Stopwords

. Técnica de selección de atributos que consiste en eliminar palabras que no son relevantes para la información del texto, es decir, conjunciones, preposiciones, determinantes o pronombres que aparecen numerables veces en el texto y no aportan información de interés para la clasificación del mismo. Para detectar estas palabras se hace un analisis de frecuencia de cada palabra y se ordenan descendientemente. Se establece un valor y todas las palabras que aparezcan más veces que este son eliminadas. Para establecer este valor umbral, habrá que tener en cuenta que si es demasiado alto muy pocas palabras serán eliminadas, y si es demasiado bajo perderemos demasiadas palabras.

TPR

. True Positive Ratio. Figura de mérito que mide el porcentaje de aciertos entre los clasificados como positivo.

Word Stemming

. Técnica que consiste en reducir una palabra a sus letras más características, a menudo coincide con la raíz de la palabra, aunque es importante saber que no tiene porqué serlo. Se eliminan prefijos y sufijos para que todas las palabras que se refieran al mismo concepto aunque tengan diferencias en la forma gramatical, sean tratadas de igual manera. Al utilizar esta técnica nos arriesgamos a perder información gramatical de las palabras. Así las palabras, selección, seleccionar, selecciona, seleccionando y todas las derivaciones de estas, quedarán representadas como selección.

C | LISTA DE ACRÓNIMOS

ICD-9-CM

. *International Statistical Classification of Diseases, Ninth Revision, Clinical Modification* o en castellano; "Clasificación Internacional de Enfermedades, 9ª Edición, Modificación Clínica" (CIE-9-MC).

EDT

. Estructura de Descomposición del Trabajo. Típicamente representado mediante un diagrama en el que se observa el desglose de todas las tareas y subtareas que componen un proyecto.

VPN

.*Virtual Private Network* o Red Privada Virtual una red VPN está definida por un conjunto de nodos asociados a una "tubería" que proporciona un puntos de acceso a la red a través de los nodos.

ssh

. (o Secure SHell) es un protocolo que facilita las comunicaciones seguras entre dos sistemas usando una arquitectura cliente/servidor y que permite a los usuarios conectarse a un host remotamente. A diferencia de otros protocolos de comunicación remota tales como FTP o Telnet, SSH encripta la sesión de conexión, haciendo imposible que alguien pueda obtener contraseñas no encriptadas. Definición obtenida del *Red Hat Enterprise Linux 4: Manual de referencia*.

MTI

.*Medical Text Indexer* o Indexador de Textos Médicos es un sistema para convertir textos clínicos en su correspondiente código ICD-9-CM.

KNN

. "K Nearest Neighbour" Clasificador que decide la clase de la instancia a clasificar buscando los k vecinos más cercanos en el espacio de atributos, y eligiendo seún algún criterio establecido. Típicamente se elige la clase mayoritaria entre los vecinos más cercanos.

D | FIGURAS

Debido a su tamaño, algunas figuras presentes en la memoria no se visualizan con total claridad. Por tanto, se encuentran todas ellas disponibles online en el link http://goo.gl/78VfYO ordenadas por capítulos.

CRÉDITOS

Este documento está basado en los estilos tipográficos "classicthesis" y "arsclassica", ambos disponibles para LaTeX vía CTAN. Para más información consultar A. Miede o L. Pantiere en http://www.miede.de/index.php?page=classicthesis o http://www.lorenzopantieri.net/LaTeX.html respectivamente.

[5 de septiembre de 2014]

BIBLIOGRAFÍA

[Alpaydin(2010)] Alpaydin, E. (2010), *IntroductiontoMachineLearning*, MIT Press.

[Aronson & et al.(2007)] Aronson, A. & et al., O. B. (2007), *Indexing the Biomedical Literature to Coding Clinical Text: Experience with MTI and Machine Learning Approaches*.

[Black & Pieterse(2013)] Black, P. E. & Pieterse, V. (2013), *DictionaryofAlgorithmsandDataStructures*, CRC Press LLC.

[C. Manning & Schutze(2008)] C. Manning, P. R. & Schutze, H. (2008), *Introduction to Information Retrieval*, Cambridge University Press.

[Cacoo(2014)] Cacoo (2014), URL https://cacoo.com/.

[Contributors(2014)] Contributors, W. (2014), *Algorithm Implementation*, Wikibooks, The Free Textbook Project, URL http://en.wikibooks.org/wiki/Algorithm_Implementation.

[Dropbox(2014)] Dropbox (2014), URL https://www.dropbox.com/.

[Eclipse(2014)] Eclipse (2014), URL https://www.eclipse.org/.

[Gelbukh(2002)] Gelbukh, A. (2002), «Tendencias recientes en el procesamiento de lenguaje natural», .

[Hall(1999)] Hall, M. A. (1999), *Correlation-based Feature Selection for Machine Learning*, PhD. Thesis, University of Waikato, New Zealand.

[Han & Kamber(2006)] Han & Kamber (2006), *Data Mining. Concepts and Techniques*, Morgan Kaufman Publishers.

[Ian H. Witten(2011)] Ian H. Witten, M. A. H., Eibe Frank (2011), *Data Mining: Practical Machine Learning Tools and Techniques*, The Morgan Kaufmann Series in Data Management Systems, 3rd ed.

[Java(2014)] Java (2014), URL https://www.java.com.

[J.Medori(2010)] J.Medori, C. (2010), «Machine learning and features selection for semi-automatic ICD-9-CM encoding», *Proceedings of the NAACL HLT*, vol. Second Louhi Workshop on Text and Data Mining of Health Documents, p. 84–89, 2010.

[José Hernandez Orallo(2004)] José Hernandez Orallo, e. a. (2004), *Introducción a la minería de datos*.

[LaTeX(2014)] LaTeX (2014), URL http://kile.sourceforge.net/,http://www.tug.org/texworks/,http://www.winedt.com/.

[Mitchel(1997)] Mitchel, T. M. (1997), *Machine Learning*.

[Santiso(2014)] Santiso, S. (2014), *Detección de efectos adversos a medicamentos en documentos médicos*, Degree Memory, Escuela de Ingeniería Técnica Industrial Bilbao.

[WHO(2014)] WHO (2014), «World Health Organization», URL http://www.who.int/.

[Zhang & Patrick(2008)] Zhang, W. H. & Patrick, J. (2008), *Automatic Classification of Pathology Reports into SNOMED Codes Automatic Classification of Pathology Reports into SNOMED Codes Automatic classification of pathology reports into SNOMED codes*, Degree Memory, University of Sidney.

[Zhang(2006)] Zhang, Y. (2006), «A Hierarchical Approach to Encoding Medical Concepts for Clinical Notes», .