Método de aprendizaje multi-instancia basado en bolsas de palabras

Yojacni Companioni García

Departamento de Ingeniería Informática

Universidad de Sancti Spíritus «José Martí Pérez»

Trabajo de diploma para optar por el título de

Ingeniero Informático

Tutor: MSc. Luis A. Quintero Domínguez
Lic. Jose A. Palmero Salazar

Sancti Spíritus, 2019

A mis seres queridos.

AGRADECIMIENTOS

Muchas han sido las personas que de una forma u otra han intervenido en la materialización de este sueño. Esto no hubiera sido posible sin la ayuda incondicional de un grupo de personas que no quiero dejar de mencionar:

A mi tutor Luis Quintero Domínguez por su valiosa atención y supervisión constante.

A mi familia y amigos más íntimos, en especial a mi madre, que siempre me animaron a superar los inconvenientes que se presentaban en el camino; por su apoyo y buenas vibras.

A todas las personas, compañeros de estudio y profesores, con los que compartí en estos últimos años, que con pequeños o grandes aportes me impulsaron en el camino transitado.

RESUMEN

Dentro del área del aprendizaje automático se encuentra el aprendizaje multi-instancia, que no es más que una variante del aprendizaje proposicional en el que cada ejemplo no está descrito ya por un único vector, sino por muchos vectores atributo-valor. Este tipo de representación ha sido usado con muy buen desempeño en ciertos problemas como la clasificación de imágenes, y la categorización de documentos.

Existen varios métodos de aprendizaje multi-instancia que transforman cada bolsa en una única instancia para luego aplicar métodos de aprendizaje supervisado estándar. Sin embargo, estas transformaciones provocan una disminución de la precisión de la clasificación.

En este trabajo se propone un nuevo método, que transforma los datos multi-instancia inspirado en la minería de textos. El método propuesto realiza la transformación de los datos a una representación atributo-valor tradicional, mediante la creación de un corpus de documentos formados por palabras artificiales para reducir la pérdida de información durante el proceso de transformación.

Además, se evaluó el método propuesto, de forma experimental, utilizando nueve conjuntos de datos multi-instancia y otros dos métodos que también transforman los datos multi-instancia a una representación atributo-valor tradicional. De acuerdo a los resultados obtenidos se puede indicar que, en términos de precisión de la clasificación, el método propuesto es competitivo con los métodos de aprendizaje utilizados en la comparación.

ABSTRACT

Within the area of machine learning is multi-instance learning, which is no more than a variant of propositional learning in which each example is not already described by a single vector, but by many attribute-value vectors. This type of representation has been used with very good results in certain problems such as classification of images, and categorization of documents.

There are several multi-instance learning methods that transform each bag into a single instance and then apply standard supervised learning methods. However, these transformations cause a decrease in the accuracy of the classification.

In this work a new method is proposed, which transforms the multi-instance data inspired by text mining. The proposed method performs the transformation of the data to a traditional attribute-value representation, through the creation of a corpus of documents formed by artificial words to reduce the loss of information during the transformation process.

In addition, the proposed method was evaluated, experimentally, using nine multi-instance data sets and two other methods that also transform the multi-instance data to a traditional attribute-value representation. According to the results obtained, it can be indicated that, in terms of classification accuracy, the proposed method is competitive with the learning methods used in the comparison.

ÍNDICE

FIGURAS

TABLAS

INTRODUCCIÓN

En la actualidad, todo sistema automatizado genera, de alguna forma, datos para su posterior diagnóstico o análisis. Esto ha provocado a que se genere a nivel mundial una gran cantidad de datos. Según (Aggarwal, 2015) la cantidad de datos almacenados alcanza fácilmente el orden de los exabytes. Esta información es conformada por fuentes tan disímiles como los miles de millones de documentos indexados en la *World Wide Web*, y los existentes en la *Dark Web* que es mucho mayor. Las transacciones financieras que se realizan a diario, ya sea por tarjeta de crédito o usando un cajero automático, pueden crear datos de forma automática. Las diversas formas de interacciones de usuarios crean grandes volúmenes de datos, las que surgen de los registros de las llamadas telefónicas, los mensajes de textos, las videollamadas y audiollamadas, etc. Además, en lo referente a la tecnología de sensores y el internet de las cosas, el número de dispositivos que se pueden comunicar entre sí excedieron al número de personas en el planeta en 2008 (Aggarwal, 2013).

Resultado de lo expuesto anteriormente, nos hemos ido acostumbrando al hecho de que existen grandes volúmenes de datos ocupando nuestras computadoras, redes y vidas. Las agencias gubernamentales, instituciones científicas y empresas, han dedicado enormes recursos para recolectar y almacenar datos. Sin embargo, en la práctica, solo una pequeña cantidad de dichos datos van a ser alguna vez usados porque, en muchos casos, el volumen es demasiado grande como para poder manipularlos, o la estructura de los datos en sí es muy complicada como para ser analizada de forma efectiva.

Los datos son unidades mínimas que por sí solas carecen de significado. Los datos almacenados en el disco duro de una computadora no ayudan al desarrollo de una empresa u organismo

en el que se encuentre. Por tanto, es necesario procesarlos para extraer información que resulte útil con la finalidad de obtener conocimiento útil. Un ser humano es capaz de analizar «pequeñas» cantidades de datos y llegar a determinadas conclusiones. Sin embargo, cuando tenemos cientos de GB o incluso de MB, las capacidades de procesamiento de los humanos resultan ineficaces. Por tanto, se hace necesario utilizar métodos estadísticos o de Inteligencia Artificial, para poder procesar las cantidades de datos que se generan en la actualidad.

Un campo de la ciencia de la computación que hace uso de dichos métodos es la minería de datos,que no es más que la búsqueda automática de patrones en grandes bases de datos, utilizando técnicas computacionales de estadística, aprendizaje automático y reconocimiento de patrones. Todo esto mediante la extracción no trivial de información implícita, previamente desconocida y potencialmente útil de los datos (Gorunescu, 2011). Según la naturaleza de los datos han existido dos amplias categorías para los algoritmos de aprendizaje automático, estas son supervisado y no supervisado. Muchos problemas de minería de datos se dirigen hacia un objetivo especializado que a veces se representa por el valor de una característica particular (etiqueta de clase) en los datos. Por tanto, dichos problemas son supervisados, en donde se aprenden las relaciones de las características restantes en los datos con respecto a esta característica especial. Los datos utilizados para aprender estas relaciones se conocen como datos de entrenamiento. El modelo aprendido se puede usar para determinar las etiquetas de clase estimadas para los registros, donde falta la etiqueta (Aggarwal, 2015).

En el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos, junto a una etiqueta de decisión. La tarea es aprender una función que prediga la etiqueta de decisión de un ejemplo, dado el vector de atributos (Borges Jiménez, 2015). Aunque este tipo de aprendizaje es ampliamente usado, existen problemas donde los datos tienen cierta estructura y relaciones entre sí que no pueden ser representados de esta forma. A raíz de este problema, surge el aprendizaje multi-instancia que fue propuesto por (Dietterich et al., 1997). En este tipo de aprendizaje cada ejemplo consiste en una bolsa de instancias que contiene las instancias sin orden, pudiendo existir incluso instancias repetidas dentro de una misma bolsa. En contraparte con el aprendizaje simple-instancia, las instancias en sí no están etiquetadas, sino que las etiquetas son a nivel de bolsa. Entonces, el problema de aprendizaje aquí es la construcción de un modelo, que se base en las bolsas de ejemplo dadas, que pueda predecir la etiqueta de decisión de futuras bolsas (Borges Jiménez, 2015).

El aprendizaje multi-instancia puede entenderse fácilmente cuando se explica el problema del

cerrajero simple. Consiste en tener una puerta cerrada con cerrojo, y se tiene *M* llaveros (cada llavero contiene un manojo de llaves). Se relaciona al llavero con una bolsa, y una llave con una instancia. Si el llavero contiene alguna llave que pueda abrir la cerradura, entonces el llavero es considerado útil. El problema de aprendizaje consiste en construir un modelo que pueda predecir cuando un llavero dado es útil o no (Díaz Figueredo, 2015).

Desde la presentación del aprendizaje multi-instancia por parte de Dietterich, se han propuesto una gran cantidad de métodos. No obstante, todavía existen problemas abiertos en el aprendizaje multi-instancia, ya que esta es un área relativamente joven dentro del aprendizaje automático.

Existen varios métodos de aprendizaje multi-instancia que transforman los problemas multi-instancia en problemas de aprendizaje tradicional remplazando cada bolsa con un vector de atributos consistente en un resumen estadístico derivado de las instancias en la bolsa. Este enfoque tiene como principal ventaja que posibilita la utilización de la amplia gama de métodos de aprendizaje automático tradicional que han demostrado su efectividad en múltiples problemas. Como algunos ejemplos de algoritmos que realizan este tipo de transformación se encuentran el SimpleMI y el MIWrapper (Witten y Frank, 2005), que luego de realizar dicha transformación, se le aplica un algoritmo de aprendizaje automático tradicional como el J48 (implementación en Weka del algoritmo C4.5) (Witten y Frank, 2005), RandomForrest (Breiman, 2001), SMO (Witten y Frank, 2005), etc. Estos métodos, pertenecientes a la categoría de envoltorio pueden provocar cierta pérdida de información y por tanto repercutir en una baja precisión de la clasificación (Fu y Robles-Kelly, 2009).

Por tanto, tomando en consideración lo anteriormente planteado, se define como **problema de investigación** el siguiente: limitaciones de los métodos de aprendizaje multi-instancia que transforman los datos a una representación atributo-valor que provocan una disminución de la precisión de la clasificación.

Para dar respuesta al problema de investigación, el objetivo general de esta investigación consiste en: desarrollar un nuevo método de aprendizaje multi-instancia basado en la transformación a una representación atributo-valor que mejore la precisión de la clasificación.

Para lograr este objetivo general se proponen los siguientes **objetivos específicos**:

1. Realizar un análisis del estado del arte relacionado con el aprendizaje multi-instancia

2. Diseñar un método de aprendizaje multi-instancia inspirado en la minería de textos.
3. Implementar el método en la herramienta de minería de datos Weka.
4. Evaluar el comportamiento del método propuesto en comparación con otros métodos del estado del arte.

Después de haber realizado el marco teórico se formuló la siguiente hipótesis de investigación: el desarrollo de un método de aprendizaje multi-instancia que reduzca la pérdida de información durante la transformación a una representación atributo-valor permite mejorar la precisión de la clasificación de los métodos de aprendizaje.

El informe de la investigación está estructurado en tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos.

El Capítulo 1 se divide en tres secciones, en la primera 1.1 se explica en qué consiste el aprendizaje automático, haciendo hincapié en el tipo de aprendizaje supervisado. Luego, en la segunda sección 1.2, se formaliza la clasificación multi-instancia, su importancia y aplicaciones en la actualidad, y se definen los enfoques existentes de aprendizaje en intentos de ordenar los métodos de clasificación multi-instancia. Finalmente, en la tercera sección 1.3, se exponen las conclusiones parciales de este capítulo.

El Capítulo 2 primero realiza una breve introducción a la minería de textos que inspiró el método propuesto, MIBoW. Luego, la sección 2.2 describe el algoritmo propuesto, además de los diferentes métodos de pesado que existen y los que fueron utilizados en el desarrollo de dicho algoritmo. En la sección 2.3 se describe la implementación del método propuesto en la herramienta Weka.

El Capítulo 3 realiza una descripción del estudio experimental realizado para evaluar el comportamiento del método MIBoW. Para esto se describe la configuración del estudio experimental y se realiza un detallado análisis de los resultados obtenidos.

CAPÍTULO 1

Fundamentación teórica

En este capítulo se introduce el marco teórico de la investigación referente al aprendizaje multi-instancia (*multiple instance learning*, MIL). Primeramente se da una introducción al aprendizaje automático, en específico el aprendizaje supervisado. Luego se formaliza el aprendizaje multi-instancia y se describen los principales enfoques en los que se clasifican estos algoritmos.

1.1. Aprendizaje automático

El aprendizaje automático puede definirse, según (Mitchel, 1997), como: Un programa de computadora aprende de la experiencia E respecto a alguna clase de tarea T y una medida del desempeño D, si su desempeño en la tarea T, medido a través de D, aumenta con la experiencia E.

Para plantear correctamente un problema de aprendizaje se necesita identificar el tipo de tarea a realizar por el aprendiz (T), la fuente de experiencia (E) y la medida del desempeño a mejorar (D). Un ejemplo del propio (Mitchel, 1997) aclara esta idea: Un programa de computadora que aprenda a reconocer palabras escritas a mano donde la tarea (T) es reconocer y clasificar palabras escritas a mano en imágenes, siendo la medida de desempeño (D) el porciento de palabras clasificadas correctamente y la fuente de experiencia (E) una base de datos de palabras escritas a mano con clasificaciones dadas. Estos tres elementos permiten determinar que sea

Tabla 1.1: Representación de la información en el aprendizaje supervisado tradicional

Ejemplos	Atributo 1	Atributo 2	...	Atributo N	Decisión
x_1	$x_{1,1}$	$x_{1,2}$	...	$x_{1,N}$	y_1
x_2	$x_{2,1}$	$x_{2,2}$	...	$x_{2,N}$	y_2
...	...	...	...	...	...
x_M	$x_{M,1}$	$x_{M,2}$	...	$x_{M,N}$	y_M

un proceso de aprendizaje, explicando el carácter automático del aprendizaje al ser llevado a cabo utilizando una computadora.

Existen varios tipos de aprendizaje automático. Entre ellos se puede encontrar el aprendizaje no supervisado, el semisupervisado y el supervisado. Este último es el más ampliamente utilizado, y se puede encontrar aplicado en diversos campos como Bioinformática y Quimioinformática, en la detección de correos *spam*, reconocimiento de patrones, entre muchos otros.

1.1.1. Definición de aprendizaje supervisado

En el aprendizaje supervisado los algoritmos requieren el suministro de un conjunto de ejemplos, donde cada ejemplo se describe por un conjunto de atributos y una etiqueta asociada que corresponde a alguna propiedad importante o decisión relativa al ejemplo. El algoritmo de aprendizaje tiene como asignación construir un modelo que genere predicciones con alto nivel de exactitud de las etiquetas de ejemplos posteriores (Sánchez Tarragó, 2014).

Según afirma (Sánchez Tarragó, 2014), en el aprendizaje supervisado una instancia (ejemplo) es un par (x_i, y_i), donde $x_i = \langle x_{i,1}, x_{i,2}, x_{i,3}, \ldots, x_{i,N} \rangle \in X$ es un vector de N atributos y $y_i \in Y$ se define como la etiqueta de decisión de la instancia. Los atributos y las etiquetas de decisión son, o atributos numéricos (pertenecientes al dominio de los números reales) o atributos nominales (conjuntos de nombres de un dominio específico). El espacio $N-$dimensional X de donde x_i extrae valores se conoce como espacio de instancias o espacio de atributos, y Y es el conjunto de etiquetas de decisión. La Tabla 1.1 ilustra cómo es representada la información en el aprendizaje supervisado. Se observa que cada ejemplo x_i se describe por un vector de atributos y la etiqueta de decisión y_i correspondiente a cada x_i.

El aprendizaje supervisado tiene como tarea encontrar $f(x) = y$, basado en un conjunto de ejemplos de entrenamiento $D = \{(x_1, y_1), (x_2, y_2), \ldots, (x_M, y_M)\}$. $f(x)$ es una función desco-

nocida que asigna a cada elemento del dominio una etiqueta de decisión. El proceso es denominado clasificación cuando la etiqueta es un atributo nominal, mientras que si es un atributo numérico, el proceso es llamado regresión. El proceso de clasificación subyacente $f(x)$ es conocido como un concepto en la terminología del aprendizaje automático. Dado un conjunto de ejemplos de entrenamiento D a partir de los cuales aprender, un algoritmo de aprendizaje supervisado devuelve un modelo de los datos $h(x)$ que aproxima a $f(x)$. Dicho modelo se conoce como hipótesis o descripción del concepto (Díaz Figueredo, 2015).

1.1.2. Métodos clásicos del aprendizaje automático supervisado

Existe actualmente una enorme cantidad de algoritmos de aprendizaje supervisado. Estos se pueden agrupar atendiendo a varios enfoques fundamentales. Entre los métodos más populares se pueden encontrar: aprendizaje basado en casos, máquinas de soporte vectorial, árboles de decisión y redes neuronales artificiales.

La esencia del aprendizaje basado en casos es devolver como solución a un problema dado, la solución conocida de un problema similar (Mitchel, 1997). En los algoritmos de aprendizaje basados en casos, se representa cada concepto mediante un conjunto de ejemplos, donde cada ejemplo puede ser un ejemplo individual del concepto o una abstracción de este. El algoritmo más utilizado es el de los k-vecinos más cercanos (*K-Nearest Neighbors*, KNN), esta técnica coloca un objeto de interés dentro de clases o grupos examinando sus atributos y agrupándolo con otros cuyos atributos son cercanos a él (Garre et al., 2007).

Las máquinas de soporte vectorial (*Support Vector Machines*, SVMs) predicen la clase correspondiente a partir de un hiperplano en alta dimensión que separa los elementos de las clases según su proximidad (Jiménez y Rengifo, 2010). Los árboles de decisión, como su nombre indica, son una estructura que se forma por las bifurcaciones en cada una de las decisiones, descubriendo reglas. Visualmente describe las condiciones a causa de las decisiones tomadas (Breiman, 1984). Como ejemplo se tiene a RandomForest, el cual es un clasificador consistente en una colección estructurada de clasificadores basados en árboles $\{h(x,\Theta_k), k = 1,\ldots\}$ donde $\{\Theta_k\}$ son vectores aleatorios independientes distribuidos de manera idéntica, donde cada árbol calcula un voto unitario para la clase más popular de la entrada x (Breiman, 2001). Este puede construirse por muestreo de conjuntos de rasgos, de datos o variando de forma aleatoria algunos parámetros del árbol.

Una red neuronal artificial (*Artificial Neural Network*, ANN) es una técnica implementada por algoritmos iterativos que forman un modelo matemático de predicción (Izaurieta y Saavedra, 2000). Esta implementación imita el funcionamiento interno de las neuronas humanas.

1.1.3. Sistemas computacionales usados para el aprendizaje automático

Antes de usar las técnicas de aprendizaje automático, es necesario seleccionar el programa correcto. Existen muchos lenguajes de programación y tipos de software que brindan al usuario herramientas para el aprendizaje automático. En la práctica, la parte más importante es reconocer cuál técnica utilizar y cómo construir nuevas, siendo el software solo una herramienta.

1.1.3.1. R y RStudio

R es un lenguaje de programación diseñado para el análisis de datos y aprendizaje automático. Es un lenguaje interpretado en el sentido de que ejecuta los comandos directamente, lo que hace que tenga una interfaz más amigable que otros lenguajes de programación. Es el lenguaje de programación más usado para la estadística, y tiene como soporte a una gran comunidad internacional.

Otra ventaja de R es RStudio, que es un entorno de desarrollo integrado (*Integrated Development Environment*, IDE) pensado para R. RStudio incluye una consola interactiva y herramientas que son usadas para acceder a la ayuda de R, visualizar gráficos, y para la depuración. R, combinado con RStudio, permite al usuario desarrollar poderosas soluciones de aprendizaje automático en un período relativamente corto de tiempo (Usuelli, 2014).

1.1.3.2. KEEL

KEEL (*Knowledge Extraction based on Evolutionary Learning*) fue desarrollado originalmente como una herramienta enfocada principalmente en la implementación de algoritmos evolutivos y técnicas de *soft computing* para problemas estándares de minería de datos como clasificación, regresión y reglas de asociación. Fue lanzado en 2009 y luego actualizado en 2011 como una *suite Java* no comercial.

KEEL ofrece una simple interfaz gráfica de usuario para el diseño de experimentos con diferentes conjuntos de datos y algoritmos de inteligencia computacional, lo que permite evaluar el comportamiento de estos últimos ante diferentes problemas. Además, fue diseñado con un doble objetivo: la investigación y la educación (Triguero et al., 2017).

1.1.3.3. KNIME

KNIME (*Konstanz Information Miner*) es un sistema de código abierto, de flujo de trabajo científico para el análisis inteligente de datos. Fue diseñado como una plataforma para la enseñanza, el aprendizaje y la colaboración, que habilita la integración simple de nuevos algoritmos y herramientas para la manipulación de datos o métodos de visualización en la forma de nuevos módulos o nodos.

Recientemente, una compañía líder en investigaciones de tecnologías de la información colocó a KNIME en una posición prominente entre las plataformas avanzadas de análisis de datos. Similar a KEEL, KNIME se acoge al paradigma de los flujos de trabajo para análisis de datos complejos. No obstante, posee un enfoque más general e incorpora cientos de nodos de procesamiento para entrada/salida, análisis de datos, preprocesamiento, minería de datos así como vistas interactivas y reportes. KNIME está basado en la plataforma Eclipse y permite la implementación de *plug-ins* para extender las funcionalidades mediante el desarrollo de nuevos nodos de procesamiento. Por ejemplo, posee extensiones que permiten la integración con Weka (ver la próxima sección) y la interoperatividad con R. Sus ventajas lo colocan en el centro del grupo de herramientas de análisis de datos (Mencar et al., 2017). Los principal defectos de KNIME son que es una herramienta privativa y su gran consumo de recursos.

1.1.3.4. Waikato Environment for Knowledge Analysis (Weka)

Weka es un entorno de software para la prueba y experimentación de diferentes algoritmos de aprendizaje automático. Es un proyecto libre auspiciado por la Universidad de Waikato, ubicada en Nueva Zelanda. Es desarrollado en Java y probado bajo Windows, Linux y Macintosh.

Incorpora un amplio conjunto de algoritmos para clasificación entre los que se encuentra CitationKNN, MIBoost, LinearRegression, entre otros. Adicionalmente, Weka posee paquetes de filtrado para el preprocesamiento de los datos y el postprocesamiento de los resultados.

Además, posee un sistema de interfaces gráficas de usuario que permiten la exploración de los datos y la experimentación entre los diversos algoritmos implementados. También posee una estructura bien organizada, posibilitando la extensión de los algoritmos existentes de forma relativamente fácil, redefiniendo una serie de clases establecidas (Witten and Frank, 2005). Weka también puede ser empleado por programadores en Java como una biblioteca, lo que posibilita su integración con aplicaciones que deseen utilizar sus funcionalidades para el análisis de datos.

1.2. Aprendizaje multi-instancia

MIL ha adquirido mucha popularidad en el aprendizaje supervisado, aunque también ha sido aplicado en problemas no supervisados. MIL es una variación del aprendizaje proposicional en el que cada ejemplo no está descrito ya por un único vector, sino por muchos vectores atributo-valor (Díaz Figueredo, 2015).

Mientras que en el aprendizaje supervisado estándar cada ejemplo es una instancia que consiste en un vector de atributos junto a una etiqueta de decisión, en MIL cada ejemplo está descrito como una bolsa de instancias donde cada bolsa tiene asociada una etiqueta de decisión o clase. En el primer tipo de aprendizaje la tarea es aprender una función que prediga la etiqueta de decisión de un ejemplo, dado el vector de atributos; mientras que en MIL el objetivo es construir un modelo que, basado en las bolsas dadas como ejemplo de entrenamiento, pueda predecir con precisión la etiqueta o clase de decisión de bolsas futuras (Borges Jiménez, 2015).

MIL es una generalización del aprendizaje proposicional tradicional ya que cualquier ejemplo de este último puede representarse en MIL como una bolsa que contiene una única instancia. El paradigma MIL fue introducido inicialmente en el trabajo de (Dietterich et al., 1997) motivado principalmente por su aplicación en bioquímica. El objetivo era predecir si una molécula se uniría a un receptor dado o no. Cada molécula, que puede ser considerada como bolsa, puede tomar muchas diferentes conformaciones espaciales, o sea, instancias. La metodología para resolver el problema MIL es designar un hiper-rectángulo (*axis-parallel hyper rectangle*, APR) en el espacio de funciones destinado a contener al menos una instancia positiva de cada bolsa positiva, mientras que se excluyen todas las instancias de bolsas negativas. Una molécula es clasificada como positiva si una de sus instancias pertenece al APR.

Tabla 1.2: Representación de la información en el aprendizaje multi-instancia

Ejemplo	Instancias	Atributo 1	Atributo 2	...	Atributo N	Decisión
x_1	$x_{1,1}$	$x_{1,1,1}$	$x_{1,1,2}$	...	$x_{1,1,N}$	y_1
	...	...	...	...	...	
	x_{1,T_1}	$x_{1,T_1,1}$	$x_{1,T_1,2}$	...	$x_{1,T_1,N}$	
...	...	...	...	...	...	...
x_M	$x_{M,1}$	$x_{M,1,1}$	$x_{M,1,2}$	...	$x_{M,1,N}$	y_M
	...	...	...	...	...	
	x_{M,T_M}	$x_{M,T_M,1}$	$x_{M,T_M,2}$	...	$x_{M,T_M,N}$	

1.2.1. Formalización de MIL

La clasificación multi-instancia es una de las tareas de MIL, donde cada ejemplo (bolsa o multiconjunto) está compuesto por varias instancias y una etiqueta de tipo nominal que representa la clase o categoría de la bolsa.

En la clasificación multi-instancia cada ejemplo se describe como un par (X, y) donde

$X = \{x_1, x_2, \ldots, x_T\} \in \mathbb{N}^x$ es un multiconjunto (bolsa, en argot matemático) de T instancias y $y \in Y$ es la etiqueta de clase asociada a cada bolsa. Se habla de multiconjunto y no de conjunto debido a que el multiconjunto permite tener instancias repetidas, no siendo así en este último. Las instancias $x_i \in X, i = 1 \ldots T$ son vectores de N atributos que no tienen etiquetas de decisión. El conjunto X es un espacio $N-$dimensional formado por el producto vectorial de los N atributos que describen a las instancias, y Y es el conjunto de etiquetas de clase (Calderón Muro, 2015). La mayoría de los trabajos desarrollados en clasificación multi-instancia asume que el atributo de decisión es binario, es decir, $Y = \{+, -\}$, pero, en general, Y puede tener más de dos elementos, en problemas de clasificación multiclase y en los problemas de regresión puede tener un número infinito de elementos (Sánchez Tarragó, 2013). La Tabla 1.2 ilustra cómo es representada la información en el aprendizaje multi-instancia.

La tarea de la clasificación multi-instancia es encontrar $f(X) = y$, basado en un conjunto de instancias de entrenamiento $D = \{f(X_1, y_1), \ldots, (X_M, y_M)\}$, donde $f : \mathbb{N}^x \rightarrow Y$ es un concepto multi-instancia. Un algoritmo MIL construye, partiendo de un conjunto de ejemplos de entrenamiento D, un clasificador $h(x)$, el cual es una hipótesis de $f(x)$ (Borges Jiménez, 2015).

1.2.2. Hipótesis multi-instancia

En muchos de los algoritmos de clasificación multi-instancia se asume la existencia de etiquetas de clase ocultas en las instancias, y la relación entre las etiquetas de clase de las instancias con las etiquetas de clase de la bolsa. Esto se conoce como hipótesis multi-instancia. Varias hipótesis multi-instancia han sido propuestas a medida que se han desarrollado nuevos métodos de solución para los problemas multi-instancia (Borges Jiménez, 2015). Algunas de estas se plantean explícitamente en los algoritmos en las que son implementadas, otras se encuentran implícitamente en diversos métodos de solución multi-instancia. Se dividen en dos grandes grupos: el de las hipótesis basadas en instancia y el de aquellas basadas en metadatos. En el primer grupo se encuentran las basadas en conteo, las cuales determinan la etiqueta de la bolsa atendiendo al número de instancias de cada concepto que exista en la bolsa; y las basadas en distribución de probabilidades, en la que las bolsas se toman como muestras aleatorias de cierta distribución de probabilidades, determinando el valor de clase esperado de la población de la bolsa, la etiqueta de dicha bolsa. El segundo grupo contiene las basadas en distancia y las basadas en estadísticas (Sánchez Tarragó, 2013).

La más usada es la conocida como hipótesis estándar planteada por Dietterich et al. (1997). Dicha hipótesis fue adoptada porque se creyó la más adecuada para el problema *musk*. Posteriormente, se demostró que otras hipótesis multi-instancia podían ofrecer soluciones con igual o superior calidad a este problema (Sánchez Tarragó, 2014).

La hipótesis estándar establece que una bolsa es positiva si tiene al menos una instancia positiva, y que una bolsa es negativa si todas sus instancias son negativas. Matemáticamente, se puede describir mediante la disyunción de las etiquetas de clase de las instancias. Representada en la siguiente fórmula, se asume que las etiquetas positivas toman valor verdadero y las negativas falso:

$$\begin{aligned} f(x) &= g(x_1) \bigvee g(x_2) \bigvee \ldots \bigvee g(x_n), x_i \in X \\ &= \bigvee_{x_i \in X} g(x_i) \end{aligned}$$

En el caso en que las etiquetas positivas se les asigne valor 1 y las negativas el valor 0 o -1 se

representa con la función máx:

$$f(x) = \max_{x_i \in x} g(x_i)$$

En la hipótesis estándar se debe tener bien claro cuál es la etiqueta positiva pues, si las etiquetas positiva y negativa se invierten, la hipótesis cobra un significado diferente (Sánchez Tarragó, 2013).

En numerosos algoritmos multi-instancia ha sido usado esta hipótesis, como ejemplos se tienen los árboles de decisión (Blockeel et al., 2005), máquinas de soporte vectorial (Andrews et al., 2002), *boosting* (Boulicaut et al., 2004), densidad diversa (Lozano-Pérez y Maron, 1998), redes neuronales (Ramon y De Raedt, 2000), etc.

De forma alternativa, han sido propuestos una amplia variedad de suposiciones multi-instancia (Amores, 2013). Sin embargo, no es posible afirmar que una suposición en particular es la más efectiva en cada dominio de problema (Herrera et al., 2016).

1.2.3. Aplicaciones de MIL

MIL se utiliza en problemas donde predomina la ambigüedad en los datos. A continuación se exponen algunos ejemplos de aplicaciones donde es usado la clasificación multi-instancia, que es una de las tareas de MIL, donde se identifican la relación entre bolsas e instancias.

La clasificación multi-instancia tiene mucha aplicación en la categorización de texto. El aumento exponencial de documentos que se produce día a día ha aumentado la importancia de la tarea de clasificación de textos de una forma automática (Ventura y Moral, 2011). La rama de la clasificación textual es ampliamente utilizada en un sinnúmero de campos de aplicación, que van desde la recuperación de información hasta la extracción de conocimiento de la Internet, etc.

La recomendación de páginas índice (páginas que contienen referencias o breves resúmenes de otras páginas) puede ser visto como un problema multi-instancia, donde el objetivo es etiquetar nuevas páginas índice como positivas o negativas en dependencia de si el usuario está interesado en al menos uno de sus enlaces o en ninguno de ellos. Se considera como bolsa a cada

página índice, mientras que sus páginas enlazadas serán las instancias de cada bolsa correspondiente. Cada instancia puede ser representada por un vector de términos $T = [t_1, t_2, \ldots, t_n]$ donde $t_i (i = 1, 2, \ldots, n)$ es uno de los n términos más frecuentes que aparecen en la correspondiente página enlazada. Se tiene que el número de instancias en cada bolsa puede ser diferente, ya que no tienen que tener cada página índice el mismo número de enlaces (Zafra et al., 2007).

Los problemas en biología y química pueden ser a menudo formulados utilizando MIL debido a la incapacidad de observar instancias de clase individuales. Desde el trabajo realizado por (Dietterich et al., 1997), se ha usado mucho en el diseño de fármacos y aplicaciones del campo de la biología. Similar a Dietterich, los complejos químicos o entidades biológicas (compuestos, moléculas, genes, etc.) son modelados como bolsas. Estas entidades se componen de partes o regiones que pueden inducir un efecto de interés. El objetivo es clasificar bolsas desconocidas para una mejor comprensión de los mecanismos subyacentes del fenómeno biológico o químico (Carbonneau, 2017). MIL ha sido usado para la predicción de la biodisponibilidad de una droga (Bergstra y Bengio, 2012), predecir la afinidad de unión de los péptidos a las moléculas complejas de histocompatibilidad principales (EL-Manzalawy et al., 2011), para descubrir sitios de unión que gobiernan la expresión genética (Bandyopadhyay et al., 2015; Palachanis, 2014) y para predecir funciones genéticas(Eksi et al., 2013).

MIL está ganando popularidad en aplicaciones médicas. Las etiquetas débiles, como el diagnóstico general de un sujeto, suelen ser más fáciles de obtener que las etiquetas fuertes, como los contornos de anomalías en una exploración médica. La estructura MIL es apropiada en esta situación dado que los pacientes tienen regiones anormales y saludables en su exploración médica, mientras que los sujetos sanos solo tienen regiones saludables (Carbonneau, 2017). Algunas aplicaciones incluyen la clasificación del cáncer en imágenes histopatológicas (Xu, 2014), diabetes en imágenes retinianas (Quellec, 2012), demencia en resonancia magnética cerebral (Tong et al., 2014), tuberculosis en imágenes de rayos-X (Melendez et al., 2015), entre otros.

1.2.4. Enfoques para aprendizaje multi-instancia

En la actualidad, los algoritmos de clasificación multi-instancia cubren todas las categorías de los métodos de aprendizaje automático. Debido al creciente número de estos algoritmos surgió la necesidad de listar sus características de una forma clara y sistemática. Varios sistemas de

categorización fueron propuestos para este propósito, tratando de extraer características que diferencie a un algoritmo de otro.

Uno de los primeros intentos de ordenar los métodos de clasificación multi-instancia fue presentado por (Xu, 2003). En esta taxonomía se pueden ver dos importantes categorías. La primera contiene a los métodos basados en instancia, que no son más que los que primero tratan de determinar las etiquetas de clase de una instancia, que luego son usadas para derivar las etiquetas de clase de la bolsa con una regla específica. La segunda contiene a los métodos basados en metadatos, que son los métodos que determinan las etiquetas de clase de la bolsa, desde información extraída directamente de las bolsas.

Luego, fue (Foulds, 2008) el que propuso una categorización muy diferente, considerando tres categorías ampliamente seguidas por la mayoría de los trabajos realizados desde el 2009 en adelante. Conforme a la relación con la clasificación simple-instancia tradicional, se establecieron las categorías: *Purpose-built*, *Upgraded* y *Wrappers*.

Recientemente, (Amores, 2013) presentó una taxonomía de 3 grupos de métodos, que actualmente es la más ampliamente usada. Son las siguientes: Paradigma del espacio de instancia, Paradigma del espacio de bolsa, y Paradigma de espacio embebido.

Basado en los estudios previos antes descritos, (Herrera et al., 2016) propuso una taxonomía adecuada que cubre la mayoría de los algoritmos y es fácil de utilizar para la comunidad dedicada al aprendizaje automático. Se divide en dos categorías principales: métodos basados en instancia y métodos basados en bolsa. Estos últimos son a su vez divididos en dos categorías que difieren en su uso de si un espacio es embebido o no. Se necesitó un segundo nivel de categorización, donde los algoritmos son divididos en categorías clásicas usadas en la clasificación simple-instancia tradicional.

1. Métodos basados en instancias: Algoritmos en los que el proceso de aprendizaje ocurre al nivel de instancias. Pueden ser directamente usados para tareas de clasificación de instancias, pero no funcionan cuando las instancias no tienen una clase bien definida. Se dividen en 6 paradigmas clásicos simple-instancia: redes neuronales, métodos basados en máxima probabilidad**,** reglas y árboles de decisión, máquinas de vectores de soporte a nivel de instancia**,** métodos evolutivos **y** métodos envoltorios. Son menos precisas que los métodos basados en bolsa.

2. Métodos basados en bolsas: Algoritmos en los que el proceso de aprendizaje ocurre a

nivel de bolsas. Considera a la bolsa como toda una entidad, y no intenta determinar etiquetas de clase para instancias individuales. Estos métodos son divididos en dos subcategorías, métodos que trabajan en el espacio de bolsa original y métodos que trabajan en un espacio de bolsa mapeado.

- Métodos basados en el espacio de bolsa original: Incluye a los clasificadores que definen una distancia o medida de similaridad entre bolsas para directamente trabajar en el espacio de bolsa original: métodos basados en vecinos más cercanos, máquinas de vectores de soporte a nivel de bolsa.
- Métodos basados en transformación de bolsas: Incluye a los clasificadores que transforman cada bolsa a una representación simple-instancia, tal que el aprendiz puede entrenar cualquier clasificador simple-instancia para etiquetar nuevas bolsas. Los métodos basados en mapeo se diferencian entre ellos en la forma específica en que realiza el proceso de mapeo: transformación basada en estadísticas, transformación basada en prototipos, transformación basada en conteo, transformación basada en distancias, transformación basada en distancia de bolsas.

1.2.5. Métodos basados en transformación de bolsas

Los métodos basados en transformación de bolsa buscan representar cada bolsa mediante un solo vector de atributos que resume la información estadística de la bolsa.

Se considera una bolsa $X = \{x_1, \ldots, x_n\}$ en la que cada instancia es descrita por d atributos,es decir, $\langle x_i^1, \ldots, x_i^d \rangle, \forall i \in [1, \ldots, n]$. La bolsa puede ser vista como un conjunto de d variables aleatorias con una distribución de probabilidad desconocida, para la que se tiene un ejemplo de tamaño n, y que pueden ser caracterizadas usando varias estadísticas. Existen varios ejemplos del tipo de transformación realizada a la bolsa, obteniéndose nuevos espacios de atributos donde cada atributo del espacio original está representado por uno o más valores estadísticos que intentan capturar la forma de distribución de probabilidad de la variable original dentro de la bolsa. Se puede encontrar a *Average Mapping* , *Min-Max Mapping* y *Moments Mapping*. La dimensión del nuevo espacio se puede obtener multiplicando el número de dimensiones del espacio original por el número de estadísticas usadas para describir cada variable (Herrera et al., 2016).

Existen varios métodos basados en transformación de bolsa. Uno de ellos es SimpleMI que, comparado con otros, es un método muy simple y tiene la gran ventaja de ser muy rápido. SimpleMI es un método que puede aplicar directamente algoritmos de aprendizaje simple-instancia a problemas multi-instancia. La idea que sigue es resumir cada bolsa de datos y construir una instancia que representa la bolsa completa. La etiqueta de la instancia se hereda de la bolsa correspondiente. A través del proceso de resumen, el conjunto de datos multi-instancia es convertido a un conjunto de datos simple-instancia, para luego poder aplicar cualquier tipo de método de aprendizaje simple-instancia sin ninguna restricción. De esta forma, el modelo puede obtenerse eficientemente basado en los datos transformados (Dong, 2006).

Otro de los métodos basados en estadísticas de bolsa es el MIWrapper. Este método es bastante sencillo y los resultados experimentales presentados por (Xu y Frank, 2004) muestran que es competitivo con otros algoritmos multi-instancia en los conjuntos de datos *Musk*. Los datos son representados en formato atributo-valor para poder trabajar con ellos. MIWrapper también ofrece una fácil solución para usar métodos de aprendizaje simple-instancia para resolver problemas multi-instancia (Dong, 2006).

1.3. Conclusiones parciales

En el presente capítulo se expuso lo referente a toda la base teórica que justifica esta investigación. Primeramente se realizó una revisión bibliográfica sentando las bases para, posteriormente, formalizar lo que se conoce como clasificación multi-instancia, introduciendo la notación matemática a ser usada en este informe. Se demuestra la importancia de la clasificación multi-instancia en los tiempos presentes, mediante algunos ejemplos de diversos campos donde se aplican con excelentes resultados. Se abordan los enfoques que existen para el aprendizaje, donde se intenta ordenar los métodos de clasificación multi-instancia, y se describen los más usados en la actualidad. Luego, se describen los métodos basados en estadísticas de bolsa, ya que el método propuesto responde a este tipo, y se exponen algunos métodos que convierten la representación multi-instancia en una representación atributo-valor para luego aplicar métodos de aprendizaje automático tradicional.

CAPÍTULO 2

Método de aprendizaje multi-instancia MIBoW

El método propuesto en la presente investigación está basado en la representación bolsa-de-palabras (*bag-of-words*, BoW) proveniente de la minería de textos. Debido a esto, para su mejor comprensión, se introduce brevemente en qué consiste la minería de textos y la representación BoW. Luego se describe detalladamente el método propuesto denominado MIBoW. Además, se ofrecen detalles de su diseño e implementación para la plataforma de aprendizaje automático y minería de datos, Weka.

2.1. Minería de textos y bolsa-de-palabras

La creciente cantidad de datos en formato texto disponible por las diferentes aplicaciones ha creado la necesidad de avances en el diseño de algoritmos que puedan aprender patrones interesantes de datos textuales en una forma escalable y dinámica. Mientras que los datos estructurados se manejan generalmente con sistemas de bases de datos, tradicionalmente los datos textuales son manejados mediante motores de búsqueda debido a la falta de estructuras. Un motor de búsqueda permite al usuario encontrar información útil a partir de palabras claves. Mejorar la efectividad y eficiencia de los motores de búsqueda ha sido el centro de las investigaciones en el campo de la recuperación de información. No obstante, la investigación en

recuperación de información se ha centrado en facilitar el acceso a la información, en lugar de analizar la información para descubrir patrones interesantes que es el objetivo principal de la minería de textos (Aggarwal y Zhai, 2012c).

El término minería de textos es usado, análogamente a minería de datos, cuando los datos son textos. Así como hay características específicas de los datos cuando se maneja texto, en comparación a manejar datos de bases de datos, la minería de textos tiene varios métodos y enfoques específicos. Algunos de ellos son extensiones de métodos de aprendizaje automático y minería de datos, mientras que otros son bastante específicos. Los enfoques de la minería de textos combinan métodos de muchos campos relacionados entre sí, entre los que se incluyen como es de suponer, el aprendizaje automático y la minería de datos, además del procesamiento del lenguaje natural, aprendizaje estadístico, recuperación de información y la Web Semántica (Sammut y Webb, 2011).

Existen varias tareas en las cuales se ha centrado la minería de texto. A continuación se describen algunas de estas tareas.

Una de estas tareas es la minería de opinión. Una considerable cantidad de la información textual en los sitios web ocurre en el contexto de la revisión de productos u opiniones de diferentes usuarios. Analizar estas opiniones textuales para revelar y resumir opiniones sobre un tópico tiene amplias aplicaciones, como por ejemplo el soporte a los consumidores para optimizar decisiones (Liu y Zhang, 2012).

Dentro de la minería de textos también se ha desarrollado la detección de tópicos. En esta tarea se detectan los temas abordados en un corpus de documentos. Para esto a cada documento del corpus se le asigna una probabilidad de pertenencia a cada grupo de documentos que representa cada tema. De esta manera se puede representar cuando un documento aborda múltiples tópicos (Aggarwal y Zhai, 2012b).

Otra de las tareas más desarrolladas en la minería de textos es la clasificación de textos. Muchos métodos de aprendizaje automático supervisado han sido adaptados para su utilización en el contexto de información textual. Para la clasificación de textos una de las técnicas más utilizadas ha sido la representación BoW (Aggarwal y Zhai, 2012a).

2.1.1. Minería de textos usando el modelo bolsa-de-palabras

La clasificación de documentos usualmente se ejecuta mediante la utilización de la representación BoW (también conocida como representación *vector space model*) de los documentos y utilizan documentos que han sido clasificados manualmente para generar un modelo que permita clasificar nuevos documentos.

En la representación BoW de un documento, se forma un vector de pesos de palabras tomando en cuenta todas las palabras que aparecen en todos los documentos. La mayoría de los investigadores utilizan palabras individuales para representar los documentos, pero también existen investigaciones donde se propone utilizar información adicional para mejorar los resultados de la clasificación (Sammut y Webb, 2011).

Es común representar documentos como BoW, tomando en cuenta el número de ocurrencias de cada término pero ignorando el orden. Esta representación equilibra eficiencia computacional con la necesidad de retener el contenido del documento. Esto resulta en una representación vectorial que puede ser analizada con técnicas de aprendizaje automático. A esta representación también se le pueden aplicar técnicas de reducción de dimensionalidad (selección de atributos) para obtener un conjunto de datos menor que preserve las propiedades importantes.

La utilización de la representación BoW resulta en una minería de textos eficiente porque factores más complejos como gramática y orden de palabras no son tomados en cuenta (Aggarwal y Zhai, 2012c).

2.1.2. Métodos de pesado

En el modelo BoW, una palabra es representada como una variable separada que tiene un peso numérico de importancia. Según (Aggarwal y Zhai, 2012c), el método más popular de pesado es *Term Frequency / Inverse Document Frequency* (TF-IDF):

$$tfidf(w) = tf * \log \frac{N}{df(w)}$$

donde:

- $tf(w)$ es el número de ocurrencias de la palabra en un documento (*term frequency*)
- $df(w)$ es el número de documentos que contienen la palabra (*document frequency*)
- N es el número de documentos en el corpus.

Alternativamente, se pueden aplicar esquemas más simples como el $tf(w)$ (*term frequency*), que no es más que el número de veces en la que un término aparece en el documento, o el esquema binario, que solo expresa si el término aparece en el documento (1) o no (0) (Perovšek et al., 2015).

2.1.3. Aplicación de la representación bolsa-de-palabras en otros campos

El éxito de la representación BoW en la minería de textos ha inspirado nuevas técnicas en otros dominios tales como clasificación de imágenes (*Bag-of-Visual-Words*, BoVW) y recuperación de información a partir de audio (*Bag-of-Audio-Words*, BoAW), en los cuales también se han alcanzado buenos resultados. En los documentos textuales las palabras ocurren en lenguaje natural, sin embargo, en clasificación de imágenes y recuperación de información a partir de audio se necesita construir las palabras de manera artificial.

La influencia de la representación BoW también ha alcanzado a la Minería de Datos Multi-Relacional (*Multi-Relational Data Mining*, MRDM). Recientemente, algunos métodos de proposicionalización se han inspirado en la representación BoW. Un ejemplo es el método propuesto por Perovšek et al. (2015), este método crea palabras sintéticas usando el nombre de las tablas y las columnas de la base de datos, así como los valores almacenados en las tablas. Además, (Quintero-Domínguez et al., 2019) propusieron un método de proposicionalización multi-instancia también basado en la representación BoW. Precisamente estos métodos son los que han inspirado el método propuesto en la presente investigación.

2.2. Descripción del método MIBoW

Como se ha dicho anteriormente, el método propuesto pretende lograr reducir la pérdida de información durante la transformación de los datos multi-instancia en una representación

Algorithm 1: MIBoW(*Examples*,*tipoPesado*)

Input : dataset multi-instancia *examples*, *tipoPesado*
Output: dataset *examples* transformado a representación atributo-valor

```
1 corpus ← []
2 W ← {}
3 for example ∈ Examples do
4 |   d ←crearDocumento(example)
5 |   corpus ← corpus ⋃ d
6 |   W ← W ⋃ llaves(d)
7 examplesTransformado ←calcularPesos(corpus,W,tipoPesado)
8 return examplesTransformado
```

Algorithm 2: crearDocumento(T)

Input : *example*
Output: documento relativo a *example*

```
1 d ← {}
2 for instancia ∈ example.instancias() do
3 |   for atributo ∈ example.atributos() do
4 |   |   nombre ← instancia.nombre(atributo)
5 |   |   valor ← instancia.valor(atributo)
6 |   |   palabra ← nombre + "−" + valor
7 |   |   d[palabra] ← d[palabra] + 1
8 return d
```

atributo-valor.

MIBoW transforma el conjunto de datos multi-instancia en un corpus de documentos en formato BoW. Cada bolsa de instancias se transforma en un documento formado por palabras artificiales, que se construyen mediante la combinación de los nombres de los atributos con su valor discreto correspondiente en el formato: [nombre del atributo]_[valor discreto]. Con el objetivo de que los valores numéricos no provoquen la generación de un número excesivo de palabras artificiales, los atributos necesitan ser discretizados con anterioridad. MIBoW recorre cada instancia de la bolsa y genera las palabras artificiales con cada uno de los atributos, para que quede conformado el conjunto de palabras que compondrán el documento correspondiente a la bolsa. Luego se construyen los vectores de pares atributo-valor que representarán a las bolsas de la representación multi-instancia original. Para esto, cada una de las palabras artificiales que fue generada en el corpus de documentos, es tomada como un atributo en la nueva representación (omitiéndose las palabras repetidas), donde el valor correspondiente a cada documento (instancia) para cada palabra (atributo) es la frecuencia con que dicha palabra ocurrió en el documento, y su etiqueta es la etiqueta de la bolsa correspondiente. En este ejemplo se utiliza la frecuencia de aparición de las palabras para asignarlos a los atributos de cada documento, no obstante, al método se le incorporó las opciones de elegir entre otros dos tipos de pesado, el TF-IDF y el binario, descritos con anterioridad. Luego de aplicar la transformación, se procede a aplicarle al conjunto de datos, diferentes métodos de aprendizaje supervisado tradicional.

Para una mejor comprensión del procedimiento a realizar por MIBoW se considera la versión simplificada del problema *East-West Trains*, donde se tiene dos trenes, cada tren tiene dos vagones que presentan formas diferentes (ver Tabla 2.1a). La tarea en este problema, originalmente propuesto por *(Michalski, 1980),* es predecir si un tren, no visto con anterioridad, tiene dirección este u oeste (columna ***direction***). Se procede a describir en detalle los principales pasos en este problema. MIBoW recibe como parámetros la representación multi-instancia del problema (ver Tabla 2.1a) y el método de pesado TF.

1. MIBoW primero crea un corpus vacío y una lista para almacenar todas la palabras del corpus también vacía.

2. Luego, MIBoW recorre cada bolsa de instancias para generar un documento por cada bolsa (líneas 3-6 de Algoritmo 1).

Tabla 2.1: Ejemplo del funcionamiento de MIBoW

(a) Representación MI original

id	loadshape	roof	direction
1	triangle	peaked	east
	circle	peaked	
2	circle	flat	west
	diamond	peaked	

(b) Corpus de documentos creado

documento para la bolsa con $id = 1$	loadshape_triangle, loadshape_circle, roof_peaked, roof_peaked
documento para la bolsa con $id = 2$	loadshape_circle, loadshape_diamond, roof_flat, roof_peaked

(c) Representación simple-instancia luego de la transformación

loadshape_triangle	loadshape_circle	loadshape_diamond	roof_peaked	roof_flat	direction
1	1	0	2	0	east
0	1	1	1	1	west

a) Para la bolsa con id=1 se genera un documento, que se conforma por palabras artificiales (creadas a partir de la combinación de los nombres de los atributos con su valor correspondiente) con el formato [nombre del atributo]_[valor discreto] (vea primera fila de Tabla 2.1b) (líneas 2-7 de Algoritmo 2). En el documento no se añaden palabras repetidas sino que se cuenta la cantidad de veces que aparecen.

b) El documento obtenido es añadido al corpus de documentos que sería lo mostrado en la Tabla 2.1b para este ejemplo (línea 5 de Algoritmo 1).

c) Se añaden todas las palabras del documento a la lista de palabras que aparecen en el corpus (línea 6 de Algoritmo 1).

d) Se repiten los pasos a) , b) y c) para cada bolsa de la Tabla 2.1a. En este caso solo sería aplicarlo a la bolsa con $id = 2$ (ver segunda fila de Tabla 2.1b).

3. En el paso siguiente, se construyen los vectores de pares atributo-valor relacionado a cada bolsa de la representación multi-instancia original. Como se observa en Tabla 2.1c, se toma cada palabra artificial de las generadas en el corpus de documentos, y se considera

como un atributo en la nueva representación. Luego, dependiendo del tipo de pesado elegido, se procede a darle valor a cada atributo. Como en este ejemplo se escogió el pesado TF, el valor correspondiente a cada documento (instancia) para cada palabra (atributo) es la frecuencia con que dicha palabra ocurrió en el documento. La etiqueta de cada documento es la de su bolsa correspondiente, obteniéndose la nueva representación. Para este ejemplo quedaría como muestra la Tabla 2.1c. (línea 7-8 del Algoritmo 1)

2.3. Paquete MIBoW para Weka

La plataforma Weka posee un manejador de paquetes, que permite al usuario la instalación y desinstalación de paquetes a la versión que esté instalada en la computadora, permitiendo que algoritmos desarrollados por otros programadores estén en manos de cualquier usuario que necesite algún determinado paquete. Esta flexibilidad hace que se distribuyan los resultados dentro de la comunidad científica de una forma bastante fluida, siendo útil entonces conocer la forma en que se elabora un paquete para este tipo de repositorio.

2.3.1. Implementación del método MIBoW en Weka

Gracias a que Weka es una plataforma de código abierto y a la vez extensible, se hace posible implementar un nuevo clasificador en este ambiente. Las clases de Weka se organizan en paquetes (agrupación de clases o interfaces donde las clases que lo formen estén ubicadas en un mismo directorio y exista relación entre ellas). De esta forma, se hace un poco más fácil cualquier tarea de añadir, eliminar o modificar algún elemento.Según (Guevara Yanes, 2007) está formado por 10 paquetes globales:

associations: que contiene las clases que implementan los algoritmos de asociación.

attributeSelection: que contiene las clases que implementan técnicas de selección de atributos.

classifiers: que agrupa todas las clases que implementan algoritmos de clasificación y estas a su vez se organizan en subpaquetes de acuerdo al tipo de clasificador.

clusterers: que contiene las clases que implementan algoritmos de agrupamiento.

core: que es el paquete central, contenedor de las clases controladoras del sistema y usado en la mayoría de las clases existentes.

datagenerators: que es el paquete que contiene clases útiles en la generación de conjuntos de datos atendiendo al tipo de algoritmo que será usado.

estimators: que son las clases que realizan estimaciones(probabilísticas en su mayoría) sobre los datos.

experiment: que contiene las clases controladoras que permiten la realización de experimentos con varias bases y diferentes algoritmos.

filters: que está constituido por las clases que implementan algoritmos de preprocesamiento.

gui: que contiene todas las clases que implementan la interfaz visual con el usuario.

Para lograr incorporar un nuevo clasificador es necesario conocer la clase *AbstractClassifier* (*Classifier*, en versiones anteriores) que es una superclase de todos los clasificadores que existen. Aunque esta clase ya tiene implementado los métodos principales que un clasificador debe tener, algunos métodos hay que redefinirlos. Según (Gallardo Segura, 2014), de los métodos que se exponen a continuación, es estrictamente necesario redefinir el primero mientras solo es necesario definir al menos uno de los dos últimos:

buildClassifier(): encargado de construir el modelo del clasificador tomando como parámetro las instancias de entrenamiento. Debe inicializar todas las variables que correspondan a las opciones específicas del esquema. Nunca debe modificar ningún valor de las instancias. Después de terminada la ejecución de este método el clasificador debe ser capaz de predecir la clase de cualquier nueva instancia.

classifyInstance(): permite la clasificación de una instancia concreta. Devuelve la clase en la que se ha clasificado o «desconocido» si no se consigue clasificar.

distributionForInstance(): devuelve un vector con las probabilidades de pertenencia de la instancia a cada una de las clases.

El clasificador implementado en esta investigación fue ubicado en el subpaquete *classifiers.mi,* donde la clase principal que la representa se denomina MIBoW. Para poder observar mejor la organización de dichas clases, en A podemos ver un diagrama de clases del paquete en cuestión.

(Guevara Yanes, 2007)establece un conjunto de pasos para la creación de un nuevo clasificador:

1. Crear la clase con el nombre del clasificador. Debe heredar de la clase abstracta *Classifier*, se importa previamente el paquete *weka.classifiers.**
2. Implementar los métodos imprescindibles descritos anteriormente de tal forma que *buildClassifier* implemente el método de entrenamiento y *distributionForInstance* y/o *classifyInstance* implementen la forma de evaluar una nueva instancia después de entrenado el clasificador.
3. Para agregar parámetros necesarios para el entrenamiento se necesita modificar los métodos *listOptions, getOptions, setOptions*
4. Implementar la función de la forma siguiente:

public static void main(String[] argv) {

runClassifier(new <nombre del clasificador en cuestion>(), argv);

}

para llamar al clasificador con las opciones (argv) que se deseen.

2.3.2. Creación de un paquete para Weka con MIBoW

Un paquete Weka es un archivo con extensión *.zip* comprimido en el directorio actual que contiene recursos como la compilación del código, código fuente, documentación correspondiente, descripción del paquete. archivos, bibliotecas de terceros y archivos de configuración. Para la instalación del algoritmo, el gestor de paquete hace uso de la información contenida en el campo «nombre» del fichero **Description.props**, además de que debe existir en el paquete al menos un archivo con extensión *jar* con clases de Java compiladas (Gallardo Segura, 2014).

En el caso específico del paquete para la clasificación multi-instancia la vista del directorio quedaría como muestra la Figura 2.1:

El gestor de paquetes hace uso del fichero **Description.props**, tomando la información de su campo «nombre» para la creación del directorio en **$WekaHome/package**. Dentro de la

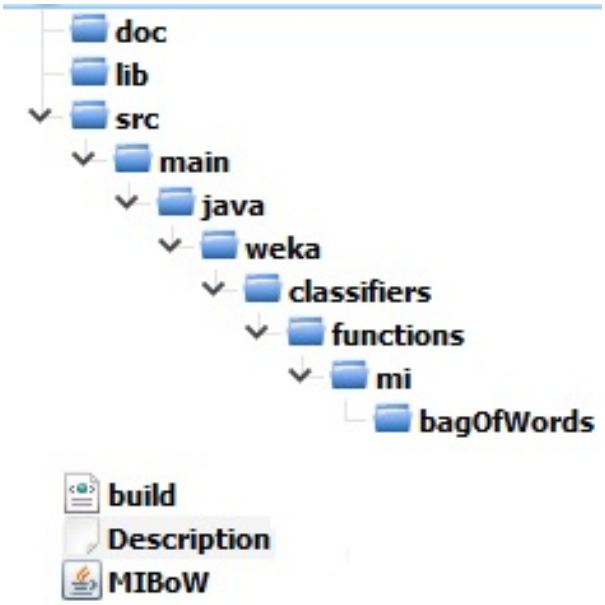


Figura 2.1: Estructura del directorio del paquete para la clasificación multi-instancia

carpeta **doc** se incluye la documentación java (javadoc) de las clases que están en el paquete. Dentro de la carpeta **lib** se incluyen las librerías de terceros que fueron utilizadas. El manejador de paquetes intentará cargar todos los archivos contenidos en el directorio raíz y el directorio **lib** que tengan extensión **.jar**. Es importante que el paquete tenga al menos un archivo **.jar** con clases de Java compiladas.

En la Figura 2.2 se muestra el fichero **Description.props**. Este archivo es el que proporciona los metadatos del paquete, así que es necesario para que dicho paquete sea válido. Las líneas que comienzan con **#** son comentarios, y los comentarios que contienen **required** es obligatorio llenarlos. Es relativamente fácil crear este fichero, pues es bastante descriptivo. En **PackageName** se pone el nombre del paquete el cual no puede contener espacios y puede ser compuesto por letras y números. **Title** es una sola oración que describe al paquete. Ya en **Description** se describe con más detalle la funcionalidad del paquete. En el campo **Manteiner** se incluye el correo electrónico de los que mantienen dicho paquete, para el reporte de errores. En (Bouckaert et al., 2016) se describen los restantes campos a ser llenados.

```
# Template Description file for a Weka package
# Package name (required)
PackageName=MiBoW
# Version (required)
Version=1.0.0
#Date (year-month-day)
Date=2019-03-10
# Title (required)
Title=Multi-Instance Bag of Words
# Category (recommended)
Category=Classifier
# Author (required)
Author=Yojacni Companioni Garcia,Luis Quintero Dominguez
# Maintainer (required)
Maintainer=Yojacni Companioni Garcia <yjesus@uniss.edu.cu>
# License (required)
License=GPL 3.0
# Description (required)
Description= Clasificador multi-instancia basado en bolsa de palabras
# Package URL for obtaining the package archive (required)
PackageURL=http://userpage.fuberlin.de/semu/software/weka/anonymization1.0.1.zip
# URL for further information
URL=http://userpage.fu-berlin.de/semu/software/weka/
# Dependencies (format: packageName (equality/inequality
version_number)
Depends=weka (>=3.8.1)
```

Figura 2.2: Description.props

2.3.3. Utilización del paquete instalado

(Gallardo Segura, 2014) explica detalladamente las diferentes formas de instalar un paquete en la versión de Weka instalada en la PC. Luego de tener instalado este paquete (ya sea por vía manual, vía interfaz gráfica de usuario, por línea de comandos o utilizando un proxy http) se procede a abrir la aplicación Weka. En la ventana principal (Figura 2.3) se selecciona la opción **Explorer**.

Después, se selecciona la base de casos a utilizar. En este caso se seleccionó EastWest (previamente discretizada) en la pestaña **Preprocess** (Figura 2.4). Aquí se puede visualizar información de la base de casos seleccionada: nombre de la relación, número de instancias, cantidad de atributos, etc.

Figura 2.3: Ventana principal del Weka

Luego, se selecciona el modelo de clasificación a utilizar para el problema (MIBoW en este caso) en la pestaña **Classify** (Figura 2.5). En el botón **Choose** se elige MIBoW, que se encuentra dentro de **mi** (contenedora de los clasificadores multi-instancia), como se observa en la Figura 2.6. Cuando ya se seleccionó el clasificador, es necesario ajustar algunos parámetros para un funcionamiento óptimo. Para esto se da *click* al nombre del clasificador y en las opciones de configuración se elige el clasificador base que se va a usar en combinación con MIBoW (RandomForest, como se muestra en la Figura B.1); y también se elige el tipo de pesado a utilizar, que en este ejemplo se usó TF (ver Figura B.2). Luego se acepta la selección y se pincha en el botón **Start**, obteniéndose los resultados mostrados en la Figura 2.7, donde se brindan una serie de estadísticas, entre las que se encuentran la matriz de confusión y los porcientos de clasificados correctos.

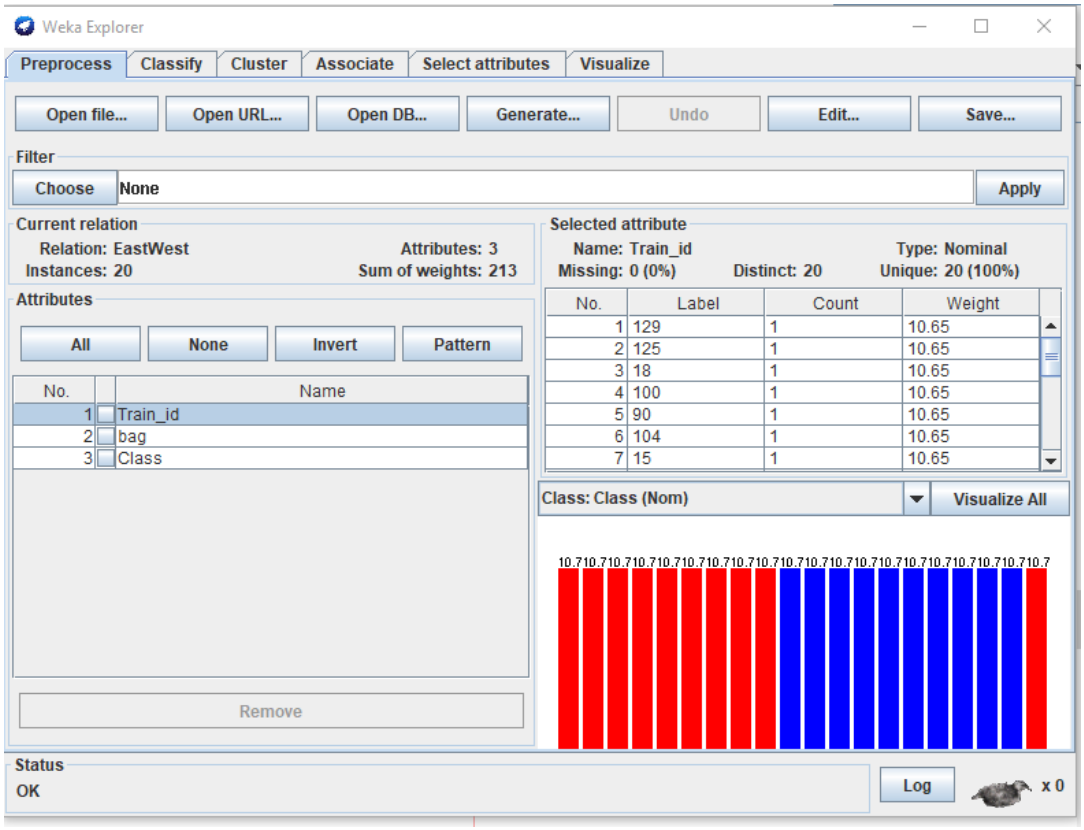


Figura 2.4: Pestaña Preprocess

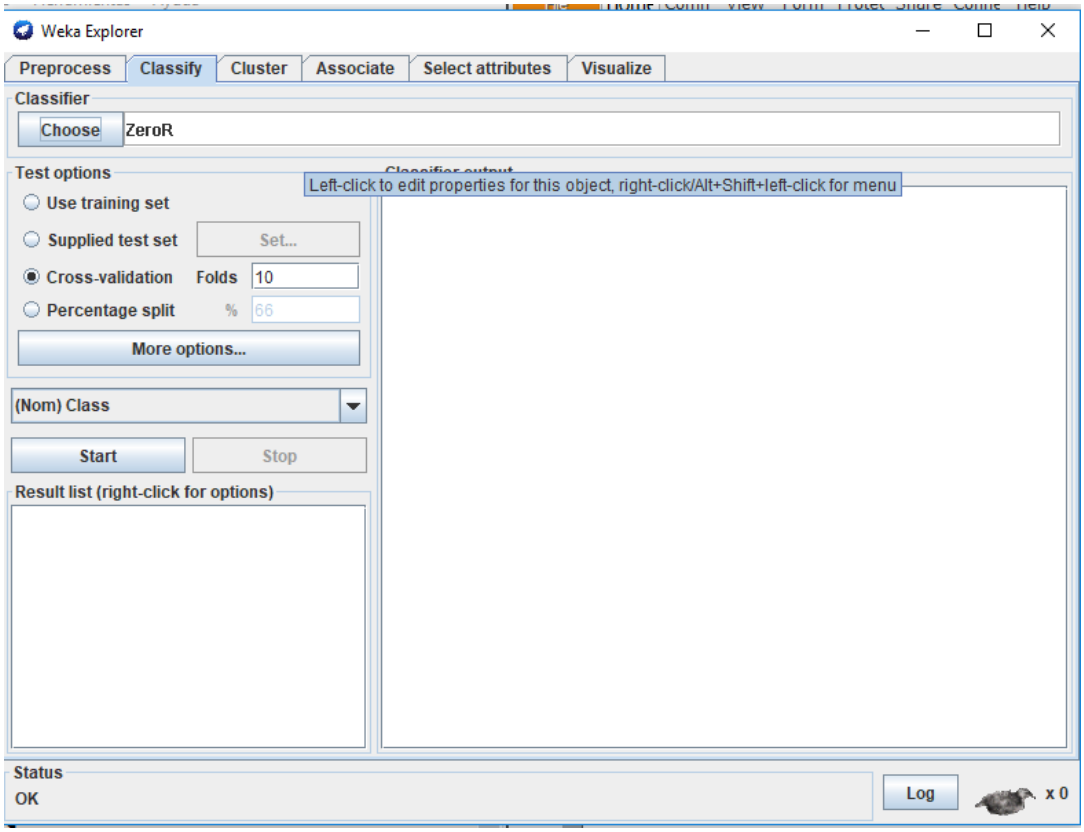


Figura 2.5: Pestaña Classify

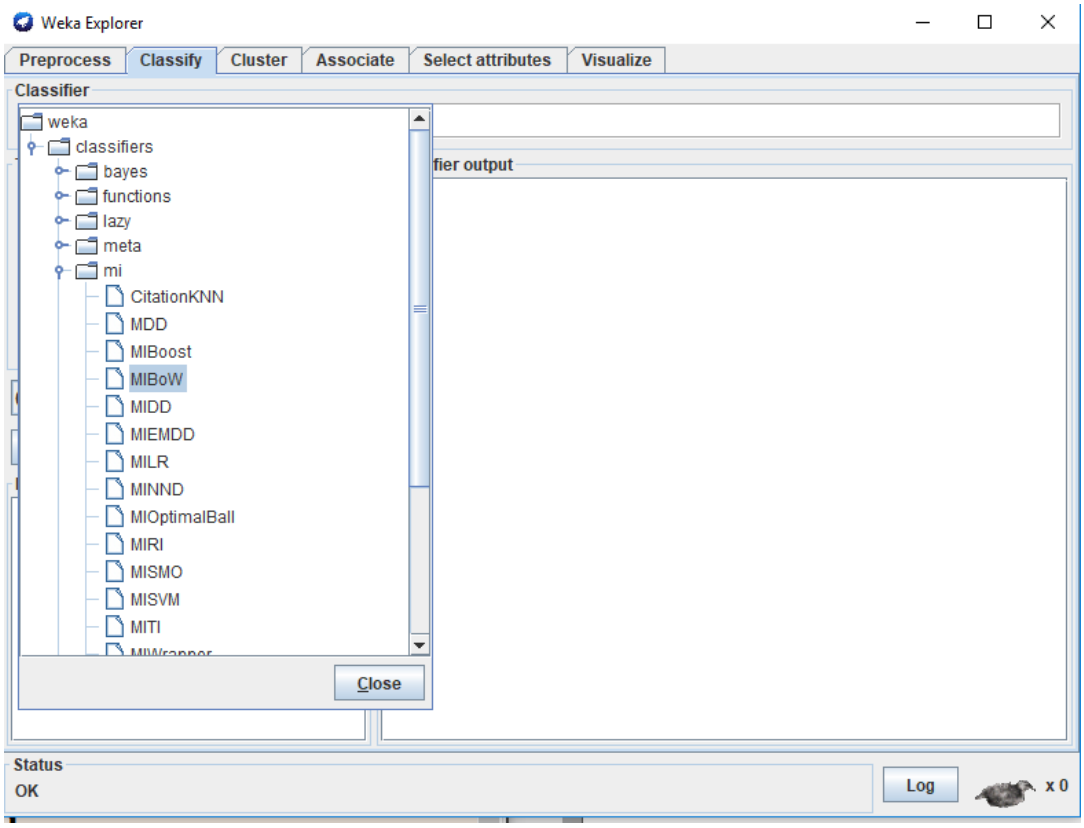


Figura 2.6: Selección de MIBoW

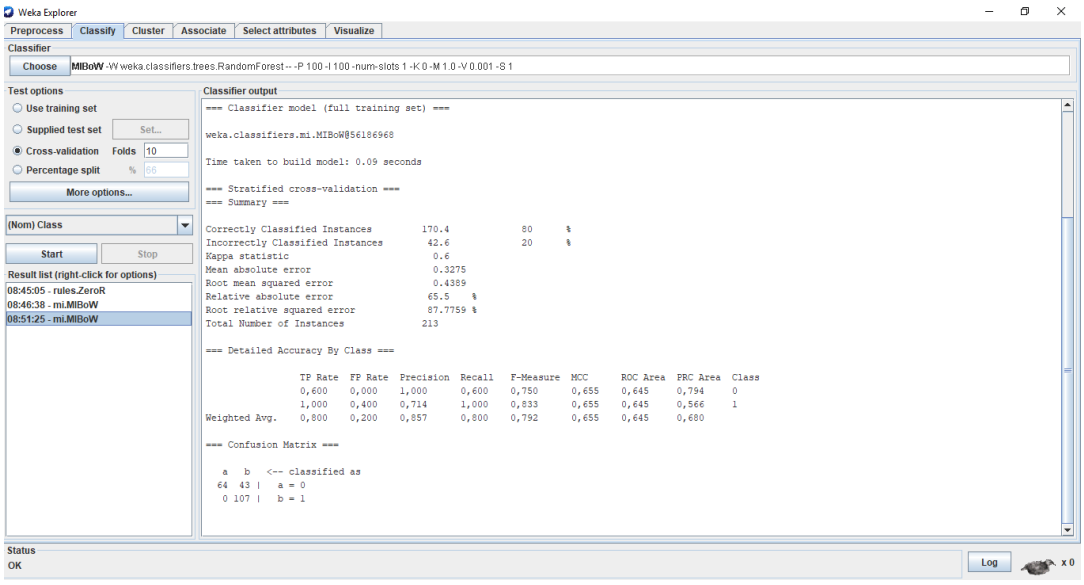


Figura 2.7: Resultados TF

2.4. Conclusiones parciales

En el presente capítulo se describió brevemente el concepto de minería de textos y la representación BOW, así como sus aplicaciones en la actualidad, para una mejor comprensión del método propuesto. Se diseñó el algoritmo MIBoW para su incorporación a la plataforma Weka, y se describió en detalle su funcionamiento mediante el problema *Trains* simplificado. Se describió la metodología utilizada para la incorporación del clasificador propuesto como paquete a la plataforma Weka. Además, se describieron los pasos necesarios para la correcta utilización de este clasificador, una vez esté incorporado en la versión de Weka instalada en la PC.

CAPÍTULO 3

Evaluación experimental del método MIBoW

En este capítulo se presenta el estudio experimental inicial con el objetivo de evaluar la eficacia del método propuesto.

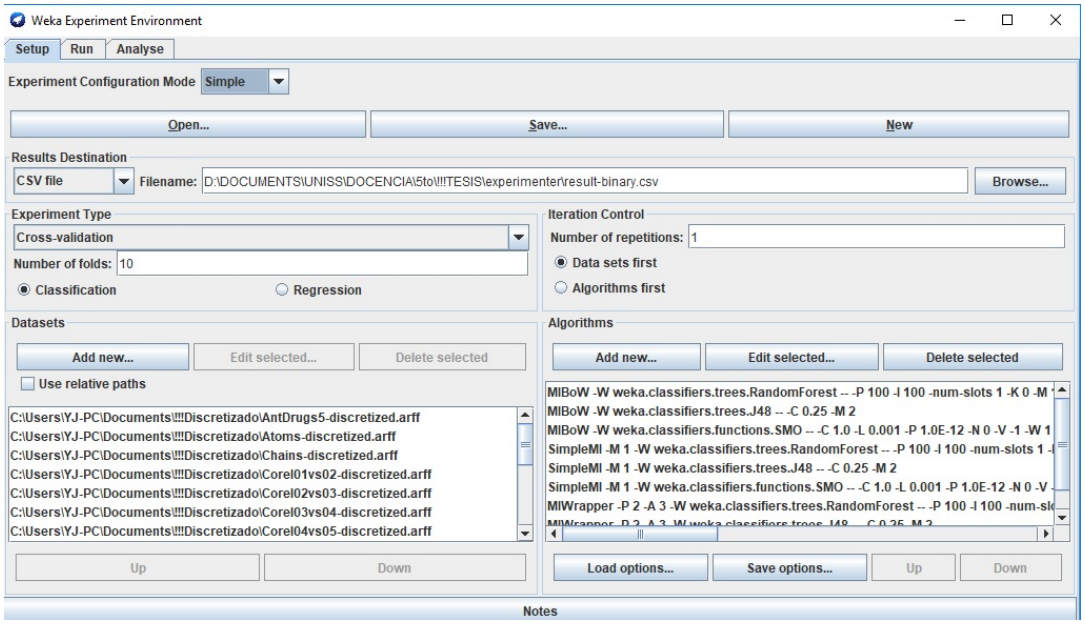
3.1. Configuración del estudio experimental

Para la evaluación experimental fueron utilizados nueve conjuntos de datos multi-instancia, descritos en la Tabla 3.1. Además del método MIBoW, se utilizó para la comparación SimpleMI y MIWrapper, que son dos algoritmos que también realizan una transformación de los datos a una representación atributo-valor tradicional.

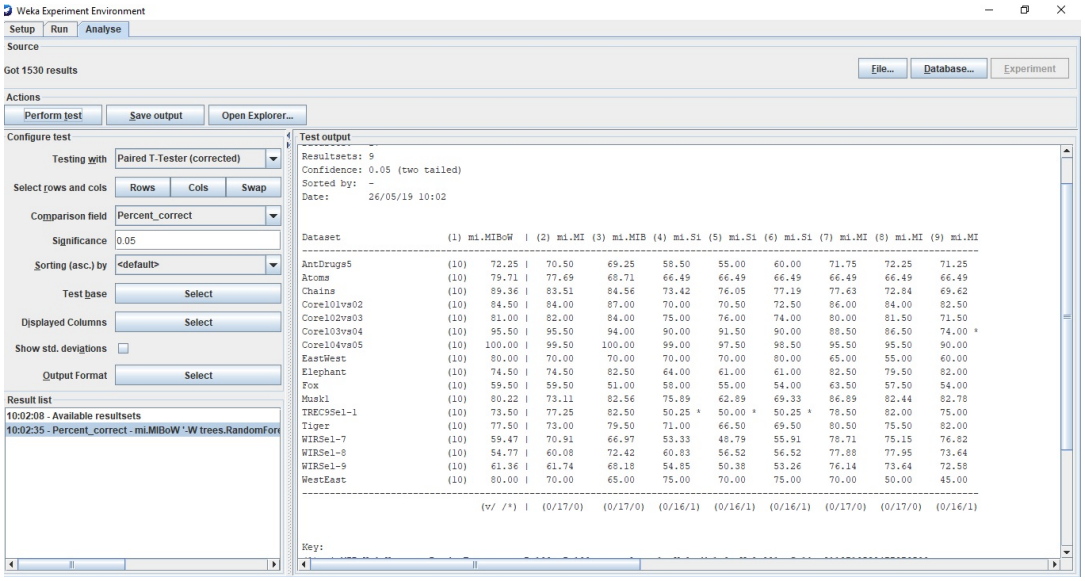
Estos tres algoritmos, luego de la transformación de los conjuntos de datos multi-instancia a una representación atributo-valor, utilizan métodos de clasificación tradicional. Por este motivo, la comparación experimental se realizó utilizando los clasificadores base RandomForest y SMO.

Se utilizó Weka como herramienta para la realización de la evaluación experimental. Específicamente, se utilizó el módulo Experimenter de Weka (ver Figuras 3.1a y 3.1b).

Además, la medida empleada para medir la eficacia de los métodos fue la exactitud de la



(a) Configuración



(b) Resultados

Figura 3.1: Experiementer de Weka.

Tabla 3.1: Características de los conjuntos de datos utilizados en la experimentación.

Dataset	Atributos	Bolsas positivas	Bolsas negativas	Total bolsas
AntDrugs5	5	198	202	400
Atoms	10	125	63	188
Chains	24	125	63	188
Corel01vs02	9	100	100	200
Corel01vs03	9	100	100	200
Corel01vs04	9	100	100	200
Corel01vs05	9	100	100	200
EastWest	24	10	10	20
TREC9Sel-1	299	200	200	400

clasificación. El método propuesto necesita que los valores de los atributos sean discretos, por esta razón los nueve conjuntos de datos fueron previamente discretizados utilizando la subdivisión del rango de cada atributo en tres intervalos de igual longitud. Los métodos de aprendizaje se utilizaron con los valores de los parámetros por defecto del Weka.

3.2. Resultados y discusión

Como se mencionó anteriormente, en el estudio experimental fue comparado el método MIBoW propuesto con los métodos SimpleMI y MIWrapper, utilizados en combinación con los algoritmos de clasificación tradicional RandomForest y SMO. A continuación se presentan los resultados obtenidos utilizando cada uno de los métodos de pesado posibles en MIBoW (TF, TF-IDF y binario).

3.2.1. Resultados utilizando el método de pesado frecuencia total

La Tabla 3.2 muestra los resultados de la evaluación experimental utilizando el método de pesado frecuencia total (TF), en términos de exactitud de la clasificación. En la Tabla 3.3a se observa que, utilizando RandomForest como clasificador base, MIBoW obtiene el mejor valor de exactitud de la clasificación en siete de los nueve conjuntos de datos.

Para comprobar si son estadísticamente significativas estas diferencias, se realizaron pruebas

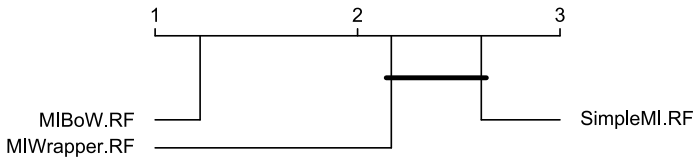


Figura 3.2: Comparación utilizando pesado TF y aprendizaje con RandomForest

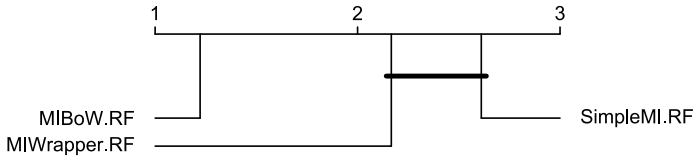


Figura 3.3: Comparación utilizando pesado TF y aprendizaje con SMO

Tabla 3.2: Resultados de la evaluación experimental utilizando TF.

Dataset	MIBoW-RF	SimpleMI-RF	MIWrapper-RF
AntDrugs5	**72.25**	58.50	71.75
Atoms	**79.71**	66.49	66.49
Chains	**89.36**	73.42	77.63
Corel01vs02	84.50	70.00	**86.00**
Corel01vs03	**81.00**	75.00	80.00
Corel01vs04	**95.50**	90.00	88.50
Corel01vs05	**100.00**	99.00	95.50
EastWest	**80.00**	70.00	65.00
TREC9Sel-1	73.50	50.25	**78.50**

(a) Resultados usando RandomForest como clasificador base

Dataset	MIBoW-SMO	SimpleMI-SMO	MIWrapper-SMO
AntDrugs5	69.25	60.00	**71.25**
Atoms	**68.71**	66.49	66.49
Chains	**84.56**	77.19	69.62
Corel01vs02	**87.00**	72.50	82.50
Corel01vs03	**84.00**	74.00	71.50
Corel01vs04	**94.00**	90.00	74.00
Corel01vs05	**100.00**	98.50	90.00
EastWest	70.00	**80.00**	60.00
TREC9Sel-1	**82.50**	50.25	75.00

(b) Resultados usando SMO como clasificador base

estadísticas siguiendo la metodología propuesta por (García y Herrera, 2008) para comparar varios clasificadores sobre varios conjuntos de datos. En la Figura 3.2 se muestra la comparación entre las combinaciones con RandomForest utilizando la prueba de Friedman y el procedimiento de Shaffer para el análisis post hoc con un valor α=0.05. En la gráfica cuando dos métodos están conectados por una línea horizontal indica que no existen diferencias significativas entre ellos, si no están conectados entonces si existen diferencias estadísticamente significativas de acuerdo al procedimiento de Shaffer. En esta figura se puede apreciar que la combinación con MIBoW es significativamente superior a las que se realizan con SimpleMI y MIWrapper.

Cuando se utiliza SMO como clasificador base, MIBoW también obtiene el mejor resultado en siete de los nueve conjuntos de datos, como se puede apreciar en la Tabla 3.3b. De forma

similar a la metodología seguida con las combinaciones con RandomForest para comprobar si estas diferencias son estadísticamente significativas, se utilizó la prueba de Friedman y el procedimiento de Shaffer para el análisis post hoc con un valor α=0.05. En la Figura 3.3 se puede observar que la combinación con MIBoW obtuvo el primer lugar en el *ranking* de Friedman y es significativamente superior a las que se realizan con SimpleMI y MIWrapper.

3.2.2. Resultados utilizando el método de pesado TF-IDF

En la Tabla 3.3 se muestran los resultados de la experimentación utilizando el método de pesado TF-IDF, en términos de exactitud de la clasificación. A primera vista se puede apreciar en la Tabla 3.4a que utilizando RandomForest como clasificador base, el método propuesto solo logra igualar el mejor valor de exactitud de la clasificación en uno de los nueve conjuntos de datos. De manera similar, utilizando SMO como clasificador base, MIBoW solo logra el mejor valor de exactitud de la clasificación en uno de los nueve conjuntos de datos, igualando a MIWrapper y SimpleMI.

Tabla 3.3: Resultados de la evaluación experimental utilizando TF-IDF.

Dataset	MIBoW-RF	SimpleMI-RF	MIWrapper-RF
AntDrugs5	49.50	58.50	**71.75**
Atoms	**66.49**	**66.49**	**66.49**
Chains	33.51	73.42	**77.63**
Corel01vs02	50.00	70.00	**86.00**
Corel01vs03	50.00	75.00	**80.00**
Corel01vs04	50.00	**90.00**	88.50
Corel01vs05	50.00	**99.00**	95.50
EastWest	50.00	**70.00**	65.00
TREC9Sel-1	50.00	50.25	**78.50**

(a) Resultados usando RandomForest como clasificador base

Dataset	MIBoW-SMO	SimpleMI-SMO	MIWrapper-SMO
AntDrugs5	49.50	60.00	**71.25**
Atoms	**66.49**	**66.49**	**66.49**
Chains	33.51	**77.19**	69.62
Corel01vs02	50.00	72.50	**82.50**
Corel01vs03	50.00	**74.00**	71.50
Corel01vs04	50.00	**90.00**	74.00
Corel01vs05	50.00	**98.50**	90.00
EastWest	50.00	**80.00**	60.00
TREC9Sel-1	50.00	50.25	**75.00**

(b) Resultados usando SMO como clasificador base

Para comprobar si son estadísticamente significativas estas diferencias, también se siguió la metodología propuesta por (García y Herrera, 2008). Como se puede apreciar en las Figuras 3.4 y 3.5, los métodos MIWrapper y SimpleMI fueron significativamente superiores a MIBoW utilizando el pesado TF-IDF, tanto para el aprendizaje con RandomForest como con SMO.

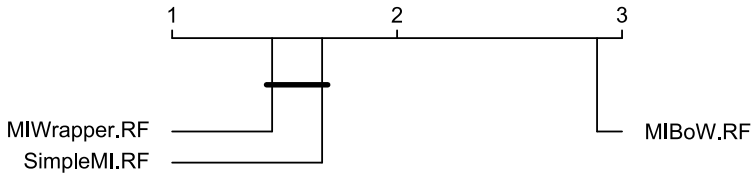


Figura 3.4: Comparación utilizando pesado TF-IDF y aprendizaje con RandomForest.

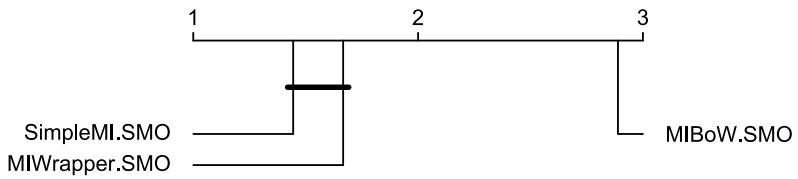


Figura 3.5: Comparación utilizando pesado TF-IDF y aprendizaje con SMO.

3.2.3. Resultados utilizando el método de pesado binario

Los resultados de la experimentación utilizando el pesado binario se muestran en la Tabla 3.4. En la Tabla 3.5a se puede observar que utilizando RandomForest como clasificador base,

Tabla 3.4: Resultados de la evaluación experimental utilizando pesado binario

Dataset	MIBoW-RF	SimpleMI-RF	MIWrapper-RF
AntDrugs5	**73.25**	58.50	71.75
Atoms	**69.12**	66.49	66.49
Chains	73.95	73.42	**77.63**
Corel01vs02	83.50	70.00	**86.00**
Corel01vs03	72.50	75.00	**80.00**
Corel01vs04	**92.00**	90.00	88.50
Corel01vs05	**100.00**	99.00	95.50
EastWest	**70.00**	**70.00**	65.00
TREC9Sel-1	**82.50**	50.25	78.50

(a) Resultados usando RandomForest como clasificador base

Dataset	MIBoW-SMO	SimpleMI-SMO	MIWrapper-SMO
AntDrugs5	**72.00**	60.00	71.25
Atoms	**67.54**	66.49	66.49
Chains	70.70	**77.19**	69.62
Corel01vs02	**85.00**	72.50	82.50
Corel01vs03	66.00	**74.00**	71.50
Corel01vs04	**95.00**	90.00	74.00
Corel01vs05	**99.50**	98.50	90.00
EastWest	60	**80.00**	60.00
TREC9Sel-1	**88.22**	50.25	75.00

(b) Resultados usando SMO como clasificador base

MIBoW obtiene el mejor valor de exactitud de la clasificación en cinco conjuntos de datos e iguala con SimpleMI como el mejor en otro.

De manera similar a lo realizado para los resultados con los métodos de pesado TF y TF-IDF, se realizaron pruebas estadísticas siguiendo la metodología propuesta por García y Herrera (2008). La prueba de Friedman, realizada con valor α=0.05, arrojó un valor de *p*-value de 0,1316. Este valor de *p*-value al ser mayor que 0,05 indica que no existen diferencias significativas entre los tres algoritmos, a pesar de que con RandomForest, MIBoW fue el mejor en cinco conjuntos de datos, mientras que MIWrapper y SimpleMI solo ganaron en tres y uno respectivamente.

Cuando se utiliza SMO como clasificador base, MIBoW obtiene la mejor exactitud de la cla-

sificación en seis de los nueve conjuntos de datos, como se puede ver en la Tabla 3.5b. Al aplicar la prueba de Friedman, con valor α=0.05, se obtuvo un valor de p-value de 0,1316. Este p-value indica que no existen diferencias significativas entre los tres algoritmos, de manera similar a como ocurrió con RandomForest.

3.2.4. Análisis general entre todos los métodos de pesado

Luego de hacer un análisis de los resultados obtenidos por el método propuesto con los diferentes métodos de pesado, en esta sección se va a realizar un análisis general de los resultados. En la Figura 3.6 se muestran los resultados utilizando el método de aprendizaje RandomForest combinado con MIBoW con cada método de pesado así como con MIWrapper y SimpleMI. Como se puede apreciar en la figura, la mejor exactitud de la clasificación es obtenida por MIBoW, con pesado TF o binario, en ocho de los nueve conjuntos de datos. Solo en el caso de Corel01vs02, el mejor resultado es obtenido por MIWrapper.

La Figura 3.7 muestra los resultados utilizando el método de aprendizaje SMO. Se puede observar que también MIBoW, con pesado TF o binario, obtiene los mejores valores de exactitud de la clasificación en ocho de los nueve conjuntos de datos. En este caso, solo para EastWest el método SimpleMI logró ser superior.

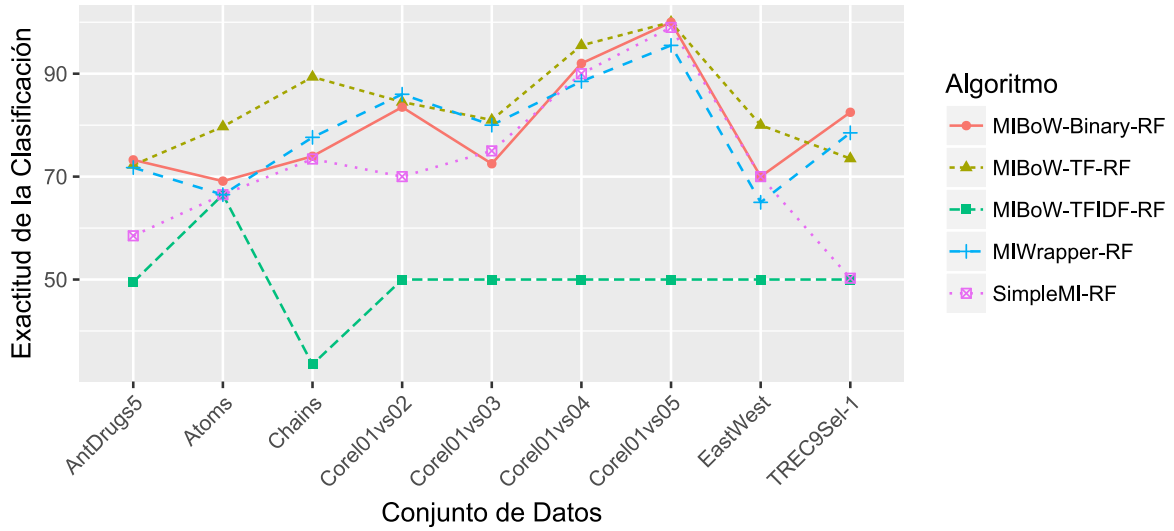


Figura 3.6: Resumen de los resultados con RandomForest.

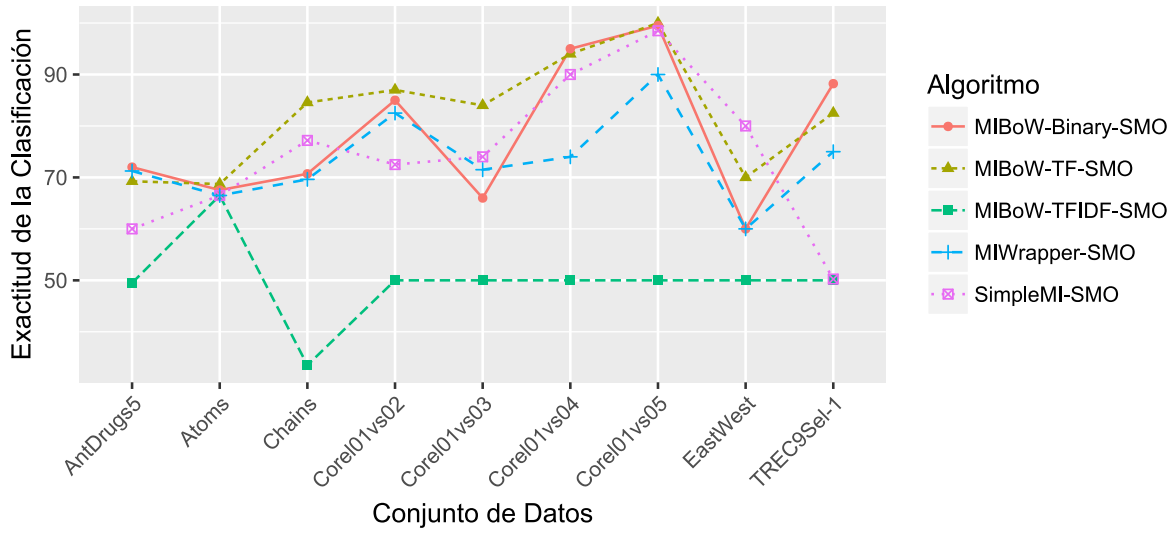


Figura 3.7: Resumen de los resultados con SMO.

También se debe resaltar que MIBoW con el método de pesado TF-IDF es el que obtiene los peores resultados tanto con RandomForest como con SMO. Esto indica que a pesar de ser el más utilizado en minería de textos, no es una buena opción para utilizarlo junto con el método de aprendizaje multi-instancia propuesto en este trabajo, MIBoW.

3.3. Conclusiones parciales

En este capítulo se presentó el extenso estudio experimental realizado para evaluar la eficacia del método propuesto. Este estudio experimental se realizó utilizando nueve conjunto de datos multi-instancia. Para la comparación con los métodos MIWrapper y SimpleMI se utilizó la metodología propuesta por García y Herrera para la comparación de múltiples algoritmos utilizando múltiples conjuntos de datos. Este estudio experimental demostró que MIBoW utilizando el pesado TF o el binario es superior a MIWrapper y SimpleMI, no siendo así para el pesado TF-IDF. Además, a partir de los resultados experimentales se puede concluir que TF es el mejor método de pesado para utilizar con MIBoW.

CONCLUSIONES

La revisión bibliográfica realizada, permitió realizar un análisis del estado del arte. De esta manera se logró realizar una descripción general del aprendizaje automático, sentando las bases para formalizar lo que se conoce como aprendizaje multi-instancia. Este estudio del estado del arte también permitió diferentes clasificaciones de algoritmos de aprendizaje multi-instancia y determinar que la herramienta de minería de datos Weka es la adecuada para implementar el algoritmo de aprendizaje multi-instancia propuesto.

Además, se diseñó un nuevo método de aprendizaje multi-instancia, inspirado en la minería de textos, para reducir la pérdida de información que ocurre en los métodos que transforman la representación multi–instancia en una representación atributo valor.

El diseño realizado permitió implementar el algoritmo propuesto para la herramienta de minería de datos Weka. Además, se creó un paquete para Weka que incluye el algoritmo propuesto, lo que facilita su utilización por cualquier usuario que tenga instalada esta herramienta de minería de datos.

Luego, se evaluó experimentalmente el comportamiento del método propuesto en comparación con los métodos del estado del arte SimpleMI y MIWrapper, para lo que se utilizaron nueve conjuntos de datos multi-instancia reales. Este estudio experimental, indica que el método propuesto es superior a SimpleMI y MIWrapper, en términos de exactitud de la clasificación.

RECOMENDACIONES

Se recomienda realizar un estudio experimental más amplio, que permita comparar el comportamiento del método propuesto con otros métodos de aprendizaje multi-instancia que no transformen la representación multi-instancia.

Además, se recomienda utilizar el método propuesto en la solución de problemas reales de la economía y la ciencia.

REFERENCIAS

Aggarwal, C. C. (2013). *Managing and mining sensor data*, Springer.

Aggarwal, C. C. (2015). *Data Mining. The Textbook*, Springer, New York, NY, USA.

Aggarwal, C. C. y Zhai, C. (2012a). A Survey of Text Classification Algorithms, *in* C. C. Aggarwal y C. Zhai (eds), *Mining Text Data*, Springer US, pp. 163–222.
URL: *http://link.springer.com/chapter/10.1007/978-1-4614-3223-4*$_6$

Aggarwal, C. C. y Zhai, C. (2012b). A Survey of Text Clustering Algorithms, *in* C. C. Aggarwal y C. Zhai (eds), *Mining Text Data*, Springer US, pp. 163–222.
URL: *http://link.springer.com/chapter/10.1007/978-1-4614-3223-4*$_6$

Aggarwal, C. C. y Zhai, C. (eds) (2012c). *Mining Text Data*, Springer US, Boston, MA.
URL: *http://link.springer.com/10.1007/978-1-4614-3223-4*

Amores, J. (2013). Multiple instance classification: Review, taxonomy and comparative study, *Artificial Intelligence* **201**: 81–105.
URL: *https://www.sciencedirect.com/science/article/pii/S0004370213000581*

Andrews, S., Tsochantaridis, I. y Hofmann, T. (2002). Support vector machines for multiple-instance learning, *Advances in neural information processing systems*, pp. 561–568.

Bandyopadhyay, S., Ghosh, D., Mitra, R. y Zhao, Z. (2015). MBSTAR: multiple instance learning for predicting specific functional binding sites in microRNA targets.

Bergstra, J. y Bengio, Y. (2012). Random Search for Hyper-parameter Optimization, pp. 281–305.

Blockeel, H., Page, D. y Srinivasan, A. (2005). Multi-instance tree learning, *Proceedings of the 22nd international conference on Machine learning*, ACM, pp. 57–64.
URL: *http://dl.acm.org/citation.cfm?id=1102359*

Borges Jiménez, J. L. (2015). *Métodos avanzados de preprocesamiento de datos para mitigar el problema de clases desbalanceadas en la clasificación multinstancia*, PhD thesis, Marta Abreu de Las Villas, Santa Clara, Villa Clara, Cuba.

Bouckaert, R. R., Frank, E., Hall, M., Kirkby, R., Peter, R., Seewald, A. y Scuse, D. (2016). *WEKA Manual for Version 3-8-1*, The University of Waikato.

Boulicaut, J. F., Esposito, F. y Pedreschi, D. (2004). *A boosting approach to multiple instance learning*, Springer Berlin Heidelberg.

Breiman, L. (1984). *Classification and regression trees*, CRC Press.

Breiman, L. (2001). Random Forests, *Kluwe Academic Publishers* **45**(1): 5–332.

Calderón Muro, C. (2015). *Desarrollo de clasificadores ensamblados robustos ante el problema de clases desbalanceadas en la clasificación multinstancia*, PhD thesis, Marta Abreu de Las Villas, Santa Clara, Villa Clara, Cuba.

Carbonneau, M.-A. (2017). *Multiple Instance Learning Under Real-World Conditions*, PhD thesis, ÉCOLE DE TECHNOLOGIE SUPÉRIEURE UNIVERSITÉ DU QUÉBEC, Montreal, Canadá.

Díaz Figueredo, C. A. (2015). *Desarrollo de clasificadores multinstancia para aplicaciones textuales basados en la fórmula de Rocchio*, PhD thesis, Marta Abreu de Las Villas, Santa Clara, Villa Clara, Cuba.

Dietterich, T. G., Lathrop, R. H. y Lozano-Pérez, T. (1997). Solving the multiple instance problem with axis-parallel rectangles, *Artificial Intelligence* **89**(1): 31–71.
URL: *http://www.sciencedirect.com/science/article/pii/S0004370296000343*

Dong, L. (2006). *A comparison of multi-instance learning algorithms*, Master's thesis, The University of Waikato.

Eksi, R., Li, H., Menon, R., Wen, Y., Omenn, G. S., Kretzler, M. y Guan, Y. (2013). Systematically differentiating functions for alternatively spliced isoforms through integrating RNA-seq data.

EL-Manzalawy, Y., Dobbs, D. y Honavar, V. (2011). Predicting MHC-II Binding Affinity Using Multiple Instance Regression, pp. 1067–1079.

Foulds, J. R. (2008). *Learning instance weights in multi-instance learning*, Master's thesis, The University of Waikato, New Zealand.

Fu, Z. y Robles-Kelly, A. (2009). An instance selection approach to Multiple instance Learning, *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 911–918.

Gallardo Segura, A. (2014). *Metodología para la incorporación de un paquete de clasificación a la plataforma WEKA. Implementación utilizando un paquete para redes neuronales recurrentes.*, PhD thesis, Marta Abreu de Las Villas, Santa Clara, Villa Clara, Cuba.

García, S. y Herrera, F. (2008). An Extension on "Statistical Comparisons of Classifiers over Multiple Data Sets" for all Pairwise Comparisons, *Journal of Machine Learning Research* **9**: 2677–2694.

Garre, M., Cuadrado, J., Sicilia, M., Rodriguez, D. y Rejas, R. (2007). Comparacion de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software, *Revista Española de Inovación, calidad e Ingenieria del Software* pp. 6–22.

Gorunescu, F. (2011). *Data Mining: Concepts, Models and Techniques*, Intelligent Systems Reference Library, Springer-Verlag, Berlin Heidelberg.
URL: *https://www.springer.com/gp/book/9783642197208*

Guevara Yanes, L. E. (2007). *Extensión del sistema Weka con la incorporación de Redes Neuronales Recurrentes*, PhD thesis, Marta Abreu de Las Villas, Santa Clara, Villa Clara, Cuba.

Herrera, F., Ventura, S., Bello-Pérez, R., Cornelis, C., Zafra Gómez, A., Sánchez-Tarragó, D. y Vluymans, S. (2016). *Multiple Instance Learning. Foundations and Algorithms*, Springer International Publishing.
URL: *http://www.springer.com/us/book/9783319477589*

Izaurieta, F. y Saavedra, C. (2000). Redes neuronales, *Technical report*, Concepción, Chile, Departamento de Física.

Jiménez, L. y Rengifo, P. (2010). Al interior de una maquina de soporte vectorial., *Revista de Ciencia* **14**: 73–85.

Liu, B. y Zhang, L. (2012). A Survey of Opinion Mining and Sentiment Analysis, *in* C. C. Aggarwal y C. Zhai (eds), *Mining Text Data*, Springer US, pp. 163–222.
URL: *http://link.springer.com/chapter/10.1007/978-1-4614-3223-4*$_6$

Lozano-Pérez, T. y Maron, O. (1998). A framework for multiple-instance learning., *Advances in Neural Information Processing Systems (NIPS Conference)* pp. 570–576.

Melendez, J., van Ginneken, B., Maduskar, P., Philipsen, R. H. H. M., Reither, K., Breuninger, M., Adetifa, I. M. O., Maane, R., Ayles, H. y Sánchez, C. I. (2015). A novel multipleinstance learning-based approach to computer-aided detection of tuberculosis on chest x-rays., **34**(1): 179–192.

Mencar, C., Fanelli, A. M. y Russo, M. (2017). KEEL meets KNIME, Vol. 17, IEEE.

Michalski, R. S. (1980). Pattern Recognition as RuleGuided Inductive Inference, *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI* **2**(4): 349–361.

Mitchel, T. M. (1997). *Machine Learning*, McGraw-Hill Science/Engineering/Math.

Palachanis, D. (2014). *Using the Multiple Instance Learning framework to address differential regulation*, PhD thesis, Delft University of Technology.

Perovšek, M., Vavpetič, A., Kranjc, J., Cestnik, B. y Lavrač, N. (2015). Wordification: Propositionalization by unfolding relational data into bags of words, *Expert Systems with Applications* **42**(17): 6442–6456.
URL: *http://www.sciencedirect.com/science/article/pii/S095741741500247X*

Quellec, G. (2012). A multiple-instance learning framework for diabetic retinopathy screening., **16**(6): 1228–1240.

Quintero-Domínguez, L. A., Morell, C. y Ventura, S. (2019). WordificationMI: multi-relational data mining through multiple-instance propositionalization, *Progress in Artificial Intelligence* .
URL: *https://doi.org/10.1007/s13748-019-00186-y*

Ramon, J. y De Raedt, L. (2000). Multi instance neural networks., *Attribute-value and relational learning: Crossing the boundaries* .

Sammut, C. y Webb, G. I. (eds) (2011). *Encyclopedia of Machine Learning*, Springer.

Sánchez Tarragó, D. (2013). Monografía Aprendizaje Multinstancia.

Sánchez Tarragó, D. (2014). *Algoritmos para la Clasificación Multinstancia*, PhD Thesis, Universidad de Granada, Granada, España.

Tong, T., Wolz, R., Gao, Q., Guerrero, R., Hajnal, J. V., Rueckert, D. y Initiative, A. D. N. (2014). Multiple instance learning for classification of dementia in brain mri., **18**(5): 808–818.

Triguero, González, S., Moyano, J. M., García, S., Alcalá-Fdez, J., Luengo, J., Fernández, A., del Jesús, M. J., Sánchez, L. y Herrera, F. (2017). KEEL 3.0: An Open Source Software for Multi-Stage Analysis in Data Mining, *International Journal of Computational Intelligence Systems* **10**: 1238–1249.

Usuelli, M. (2014). *R Machine Learning Essentials*, Pack Publishing Ltd.

Ventura, S. y Moral, R. (2011). Estudio de la clasificación de textos usando aprendizaje con múltiples instancias.

Witten, I. y Frank, E. (2005). *Data Mining: Practical Machine Learning Tools and Techniques, Second Edition*, The Morgan Kaufmann Series in Data Management Systems, Elsevier Science.
URL: *https://books.google.com.cu/books?id=QTnOcZJzlUoC*

Xu, X. (2003). *Statistical learning in multiple instance problems*, Master's thesis, Citeseer.
URL: *http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.224.9104rep=rep1type=pdf*

Xu, X. y Frank, E. (2004). Logistic regression and boosting for labeled bags of instances., Vol. 04, Pacific-Asia, pp. 272–281.

Xu, Y. (2014). Weakly supervised histopathology cancer image segmentation and classification., **18**(3): 591–604.

Zafra, A., Ventura, S. y Herrera-Viedma, E. (2007). Aprendizaje Multi-instancia con Programación Genética para Web Mining, España, pp. 285–292.

ANEXO A

Diseño de clases de MIBoW

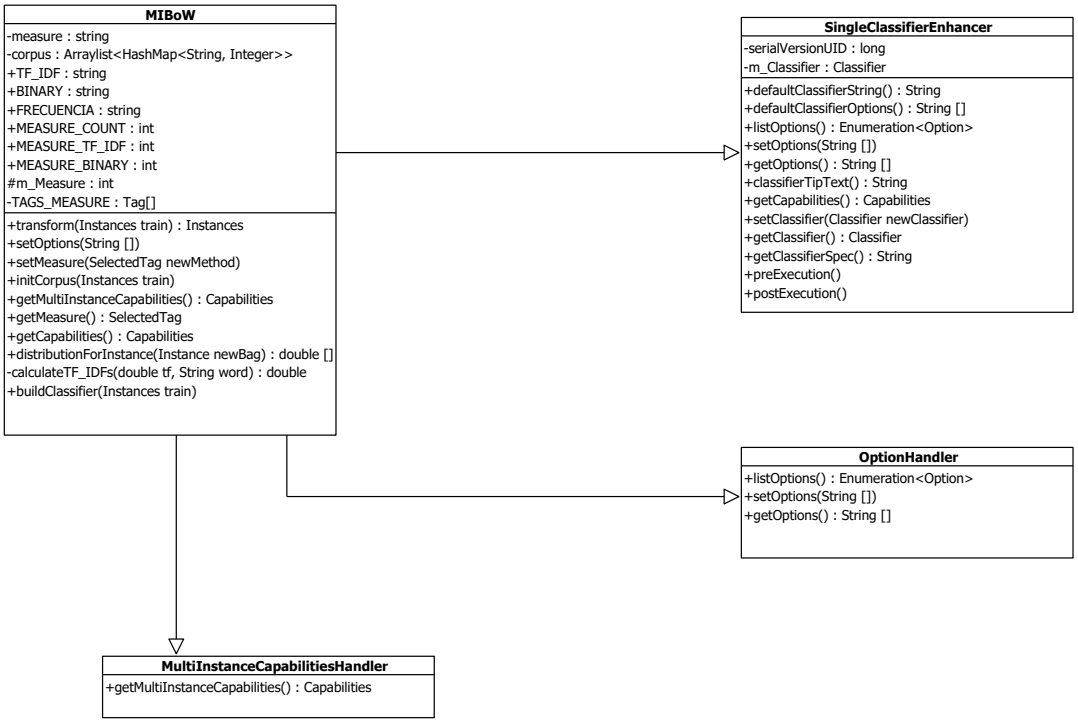


Figura A.1: Diseño de clases del método propuesto

ANEXO B

Configuración de MIBoW

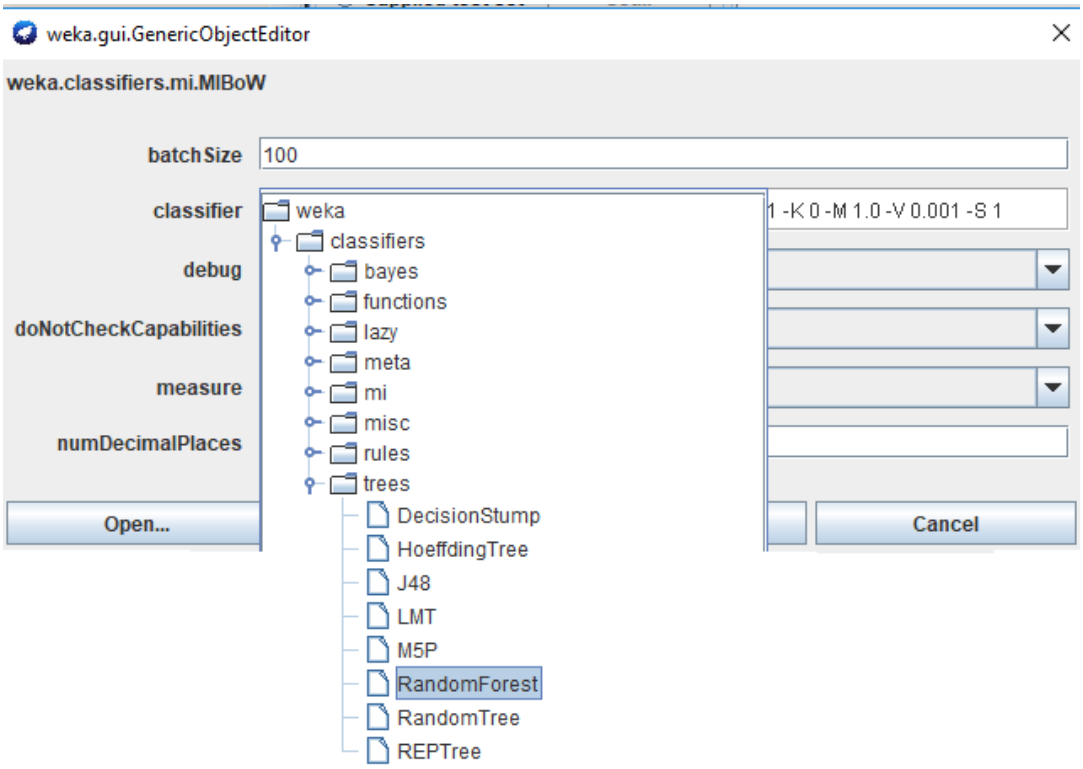


Figura B.1: Ajuste de parámetros

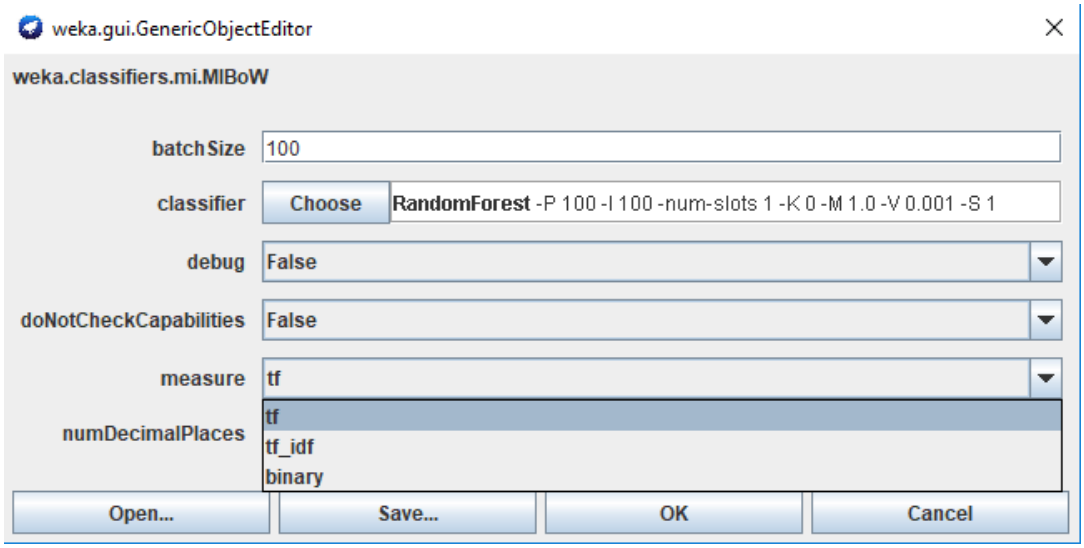


Figura B.2: Selección de tipo de pesado TF