UNIVERSIDAD POLITÉCNICA DE VALENCIA
DEPARTAMENTO DE SISTEMAS INFORMÁTICOS Y COMPUTACIÓN

Aplicación de distancias entre términos para datos planos y jerárquicos

Jorge Alonso Bedoya Puerta

Tesis presentada para optar al título de:

Máster en ingeniería de software, métodos formales y sistemas de información

Dirigida por:

José Hernández Orallo

Valencia, septiembre de 2011

En memoria de mi abuela:
María Jesús González Ángel

Resumen

El aprendizaje automático se basa en el diseño de algoritmos que utilizan la información histórica para aprender. Algunos de estos algoritmos usan las distancias para medir el grado de similitud (o disimilitud) entre objetos; estos métodos, basados en distancias, son un enfoque muy conocido y potente para la inferencia inductiva, ya que las distancias son una forma adecuada para medir la disimilitud. Comúnmente, todos estos algoritmos, basados en distancias y otros enfoques, utilizan información que es representada mediante datos proposicionales (tuplas de componentes numéricos o categóricos, los cuales son fácilmente representados como una tabla); sin embargo, este tipo de representaciones pueden ser un poco restrictivas y, en muchas ocasiones, se requiere de estructuras más complejas que permitan representar los datos de una manera más natural.

Por otra parte, los términos son la base para la representación de la programación lógica y funcional. A la vez, la programación lógica y la programación funcional pueden ser utilizadas para la representación del conocimiento en variedad de aplicaciones (sistemas basados en el conocimiento, minería de datos, etc.). Las distancias entre términos son una herramienta muy útil, no solo para comparar términos, sino también para determinar el espacio de búsqueda en muchas de esas aplicaciones.

Este trabajo aplica las distancias entre términos, aprovechando las características propias de cada distancia y la posibilidad de comparar desde tipos de datos proposicionales hasta representaciones jerárquicas. Las distancias entre términos se aplican por medio del algoritmo de clasificación k-NN (k-vecinos más cercanos) utilizando un lenguaje de representación común; el lenguaje utilizado es XML, que además de su popularidad, ofrece una alta flexibilidad para representación de datos proposicionales y términos desde diferentes niveles de jerarquía. Para poder representar estos datos en una estructura XML y con el fin de aprovechar los beneficios de las distancias entre términos, es necesario aplicar algunas transformaciones. Estas transforma-

ciones permiten convertir datos planos en datos jerárquicos, representados en XML, aplicando algunas técnicas basadas en asociaciones intuitivas entre los valores y nombres de las variables y asociaciones basadas en la similitud de los atributos.

Adicionalmente, es posible aplicar estas distancias entre términos sobre estructuras originalmente jerárquicas. Conservando XML como lenguaje de representación, el proceso de transformación consiste, principalmente, en mantener el nivel de profundidad de los nombres de las variables y sus valores y garantizar el orden, de acuerdo con características de formación de XML.

Consecuentemente, en este trabajo se realizan experimentos con la distancia entre términos de Nienhuys-Cheng [28] y la distancia de Estruch et al. [10]. Para el caso de datos originalmente proposicionales, estas distancias son comparadas con la distancia euclídea. En todos los casos, los experimentos se realizan con distancias ponderadas del algoritmo de k-vecinos más cercanos, utilizando varios exponentes para la función de *atracción* (distancia ponderada). Se puede ver que, en algunos casos, estas distancias entre términos pueden mejorar significativamente los resultados sobre enfoques aplicados para representaciones planas. Para el caso de estructuras originalmente jerárquicas, también se pueden resaltar diferencias obtenidas en el proceso de clasificación dependiendo del tipo de distancia (entre términos) aplicada.

Palabras clave: funciones de distancia, clasificación, representación basada en términos, datos jerárquicos, datos XML, programación inductiva.

Abstract

Machine learning is based on the design of algorithms that use historical information to learn. Some of these algorithms use distances to measure the degree of similarity (or dissimilarity) between objects; these methods, based on distances, are a well known and powerful approach to inductive inference, since the distances are a proper way to measure dissimilarity. Usually, these distance-based algorithms, and other approaches, use information that is represented by propositional data (tuples of numerical or categorical components, which are easily represented as a table), however, this kind of representation can be quite restrictive and, in many cases, it requires more complex structures in order to represent data in a more natural way.

On the other hand, the terms are the basis for functional and logic programming representation. At the same time, logic and functional programming can be used for knowledge representation in a variety of applications (knowledge-based systems, data mining, etc.). Distances between terms are a useful tool not only to compare terms, but also to determine the search space in many of these applications.

This dissertation applies distances between terms, exploiting the features of each distance and the possibility to compare from propositional data types to hierarchical representations. The distances between terms are applied through the k-NN (k-nearest neighbor) classification algorithm using a common language representation; the language used is XML which, in addition to its popularity, offers high flexibility for propositional data and term representation from different hierarchy levels. To be able to represent these data in an XML structure and to take advantage of the benefits of distance between terms, it is necessary to apply some transformations. These transformations allow the conversion of flat data into hierarchical data represented in XML, using some techniques based on intuitive associations between the names and values of variables and associations based on attribute similarity.

Additionally, it is possible to apply these distances between terms over originally hierarchical structures. Preserving XML as a representation language, the transformation process is mainly to maintain the depth level of the variables' names and their values and ensure the order, according to XML formation features.

Consequently, several experiments with the distances between terms of Nienhuys-Cheng [28] and Estruch et al. [10] were performed in this dissertation. In the case of originally propositional data, these distances are compared to the Euclidean distance. In all cases, the experiments were performed with the distance-weighted k-nearest neighbor algorithm, using several exponents for the *attraction* function (weighted distance). It can be seen that in some cases, the term distances can significantly improve the results on approaches applied to flat representations. In the case of originally hierarchical structures, differences obtained in the classification process depending on the distance type applied (between terms) can also be highlighted.

Keywords: Distance functions, classification, term-based representation, hierarchical data, XML data, inductive programming.

Agradecimientos

Este trabajo ha sido realizado con la ayuda, apoyo, comprensión y paciencia de muchas personas y sería injusto no hacer un reconocimiento a todos quienes, desde diferentes ámbitos, han aceptado compartir conmigo las dificultades propias de un trabajo de investigación de fin de máster, máxime con las implicaciones del desarraigo cultural que lo hace más complejo, pero al mismo tiempo más satisfactorio.

A mi madre y mi padre por convertir mis sueños en sus sueños, con todas las consecuencias que esto conlleva.

A José Hernández Orallo, por aceptar ser mi asesor, en el sentido pleno de la palabra; su calidez humana, conocimiento, disposición, apoyo, diligencia y paciencia han hecho posible que, no sólo finalice este proceso satisfactoriamente, sino que amplíe mi conocimiento profesional.

A María José Ramírez y César Ferri, por brindar su experiencia y conocimiento para apoyar activamente este proyecto y ayudarme a mejorarlo.

A mis amigos en Valencia: Virginia, Antonio, Fabio, Oscar y Pablo por su generosidad, amistad y hospitalidad durante todo este proceso, haciéndome sentir un miembro de sus familias. Gracias a ellos, no ha sido tan difícil estar lejos de casa.

A mi hermano por su consejo simple pero contundente que permite simplificar las dificultades cuando las cosas no van bien.

A todos los familiares, amigos y compañeros del trabajo que de alguna manera me motivaron en este proyecto y de diferentes maneras me apoyaron.

Y por supuesto a Gloria, mi novia, esta bella persona que con su amor y compañía hace que todo sea más sencillo.

Agradezco igualmente a Arkix S.A., por permitirme continuar creciendo desde situaciones prácticas; al grupo DMIP (*Data Mining and Inductive Programming*) por acogerme en su equipo, apoyarme y hacerme parte de sus actividades y a la Universidad Politécnica de Valencia, por ser mi casa lejos de casa y permitirme crecer como ser humano y como profesional.

Jorge A. Bedoya P.
Valencia, 2011

Tabla de Contenido

Lista de Tablas

Lista de Figuras

1 Introducción

1.1 Motivación

La información histórica es fundamental para la aplicación de algoritmos de aprendizaje automático; esta información puede ser representada de múltiples formas que van desde estructuras rígidas y estrictamente ordenadas (por ejemplo datos tabulares) hasta representaciones más naturales (textos, páginas web, etc.). Muchos de estos métodos de aprendizaje, típicos en aplicaciones de minería de datos, están diseñados para trabajar con representación de datos proposicionales que consisten en tuplas de atributos numéricos o categóricos, comúnmente representados como una tabla. Los resultados obtenidos por la aplicación de estos algoritmos se ven afectados, entre otros aspectos, por la manera en que se representan los datos; en muchas ocasiones, los datos deben ser aplanados para ajustarse a representaciones proposicionales; lo anterior hace que se requiera de un lenguaje de representación más expresivo basado en tipos de datos estructurados como conjuntos, árboles, grafos, etc. Dado que el aprendizaje proposicional no se puede tratar con estos tipos de datos, nuevos métodos han sido diseñados para extraer patrones desde descripciones más complejas [8].

Un subcampo del aprendizaje automático, relacionado con el aprendizaje a partir de la lógica de primer orden, es conocido como programación lógica inductiva (ILP), que puede ser aplicada para la representación de la información de una forma más natural y expresiva que otros paradigmas, que solo aprenden de ejemplos con estructuras planas. Este área ha sido extendida a otros paradigmas, como la programación funcional, dando lugar a la programación funcional inductiva o a la programación lógico-funcional inductiva [20] [21] [13] o al área general de programación inductiva [29] [14] [23]. En programación lógica y en programación funcional,

los datos se representan en base a una estructura de datos que se denomina término, a partir de los que se definen los átomos , cláusulas o reglas y, en definitiva, los programas.

El aprendizaje basado en instancias utiliza el principio del razonamiento basado en casos (CBR), donde "problemas similares tienen soluciones similares". Este aprendizaje utiliza la distancia para conocer la similitud (o disimilitud) entre instancias. Las distancias (también llamadas métricas) son medidas de disimilitud con algunas propiedades especiales, como la simetría y la desigualdad triangular, las cuales son muy ventajosas para muchos algoritmos porque el espacio de búsqueda puede ser reducido a una triangulación. Los métodos basados en distancias son muy eficaces para la inferencia inductiva; la gran ventaja de estos métodos es que también son algoritmos o técnicas que pueden ser aplicadas sobre diferentes ordenes de tipos de datos, siempre que una función de similitud haya sido definida previamente para estos tipos de datos [26].

Por otra parte, existen lenguajes muy populares que se basan en árboles o jerarquías, como XML y estructuras relacionadas funcionalmente similares [5], que pueden ser utilizados para representar la información o el conocimiento (por ejemplo ontologías). Las estructuras en árboles y los términos funcionales tienen fuertes similitudes; un término en programación funcional (o lógica) puede ser representado como un árbol ordenado. Aunque la semántica es crucial para entender el rol de un término o átomo con relación a un programa, también es posible analizar términos de manera aislada (analizando únicamente su parte sintáctica).

Existen distancias para virtualmente cualquier tipo de objeto, incluyendo complejos o altamente estructurados, como tuplas, listas, árboles, grafos, imágenes, sonidos, páginas web, ontologías, etc. Un nuevo reto en el aprendizaje automático, pero más especialmente en el área de la programación inductiva, es la distancia entre términos o átomos de primer orden. Aunque los términos pueden ser usados para representar muchos de los tipos de datos previos (y consecuentemente, una distancia entre términos virtualmente llega a ser una distancia para cualquier dato complejo y estructurado), estos tipos de datos son especialmente adaptados para representaciones basadas en términos o árboles. De esta manera, estas distancias no solo pueden ser usadas en el área de la programación lógica inductiva (ILP) [27] (por ejemplo segmentación de primer orden [3]) o en general, programación inductiva, sino también para áreas donde está involucrada la información estructurada (jerárquica) como aprendizaje de ontologías o documentos XML.

En comparación con las distancias tradicionales, las distancias entre términos cuentan con propiedades adicionales (como la sensibilidad del contexto, el tamaño de las diferencias, las diferencias repetidas, etc.) que favorecen la comparación de estructuras jerárquicas; en esta me-

dida, estas distancias, permiten comparar desde tipos de datos proposicionales hasta estructuras jerárquicas. Una manera de lograr mejores resultados en las tareas de aprendizaje basado en distancias es aplicando transformaciones que conviertan datos proposicionales en jerárquicos o mantener la jerarquía original y así obtener un mayor provecho de las propiedades de las distancias entre términos.

1.2 Objetivos

El propósito general de este trabajo es aplicar distancias entre términos sobre conjuntos de datos jerárquicos que, por medio de transformaciones apropiadas sobre datos planos, puedan ser utilizadas en una gama más amplia de aplicaciones y permitan mejorar los procesos de clasificación. Específicamente, se utilizan las distancias entre términos de Nienhuys-Cheng y Estruch et al. para el algoritmo de k-vecinos más cercanos, aplicadas para datos categóricos sobre dos tipos de representaciones: primero, jerarquías construidas a partir de datos planos, utilizando relaciones entre atributos inducidas por nombre o igualdad de valor y métricas de similitud entre atributos; los resultados son comparados utilizando la distancia euclídea; segundo, datos originalmente jerárquicos. Para estas dos situaciones se utiliza XML como lenguaje de representación.

Lo anterior es detallado en los siguientes objetivos específicos:

- Estudiar y entender las propiedades de las distancias entre términos utilizando como referencia de inicio de este trabajo el artículo *A New Context-Sensitive and Composable Distance for First-Order Terms. Technical Report* [9], revisando las distancias entre términos, particularmente la distancias de Nienhuys-Cheng y Estruch et al. y sus propiedades.
- Considerar XML como un lenguaje de representación de datos sobre el cual se calcule la distancia entre términos, permitiendo así un formato de entrada más estándar y flexible para las transformaciones y jerarquías propuestas.
- Implementar las distancias entre términos y un algoritmo de clasificación para evaluar dichas distancias. En concreto, se propone implementar las distancias entre términos y la distancia euclídea.

- Explorar enfoques de agrupación de variables para construir jerarquías sobre datos planos. Partiendo de los posibles enfoques de agrupación de atributos (inducida por nombre o igualdad de valor y métricas de similitud entre atributos), se crearán jerarquías que posteriormente puedan ser utilizadas por el algoritmo de clasificación.
- Aplicación de distancias sobre conjuntos de datos y análisis de resultados. Con las jerarquías inducidas a partir de datos planos y los datos originalmente jerárquicos, se realizarán experimentos sobre los conjuntos de datos, se analizarán los resultados devueltos por el algoritmo de clasificación y se establecerán las conclusiones de cada proceso con el fin de entender mejor el comportamiento de estas distancias y de los procesos de transformación propuestos.

1.3 Organización de la tesis

Este trabajo se encuentra organizado de la siguiente manera:

- **Capítulo 2: Antecedentes.** Este capítulo describe aspectos generales del aprendizaje automático, características y propiedades de los métodos basados en distancias, algoritmos basados en distancias (particularmente el algoritmo k-NN y algunas mejoras) y distancias para diferentes tipos de datos estructurados, representación de términos usando XML y, finalmente, se describen las propiedades de algunas distancias entre términos y su definición formal (profundizando en la distancia de Nienhuys-Cheng y Estruch et al.).
- **Capítulo 3: Transfomación de datos semi-estructurados en una representación basada en términos usando XML.** En el capítulo 3 se hace una descripción de las transformaciones realizadas para convertir datos planos en datos jerárquicos utilizando como lenguaje de representación común XML. Para este proceso se consideran tres fuentes de estructuras: igualdad de valor, jerarquía inducida por nombres y jerarquía de similitud de atributos. Adicionalmente, se describe brevemente algunos aspectos a considerar para la derivación de esquemas XML jerárquicos a partir de datos originalmente jerárquicos.
- **Capítulo 4: Experimentos.** Este capítulo muestra los resultados obtenidos a partir de la ejecución del algoritmo de clasificación k-NN, utilizando varios exponentes para la función de *atracción* con diferentes tipos de jerarquías derivadas por medio de las transformaciones

mostradas en el capítulo 3, aplicando la distancia entre términos de Nienhuys-Cheng y Estruch et al. y comparándolas con la distancia euclídea. Adicionalmente, se muestran los resultados sobre la aplicación de distancias entre términos sobre un conjunto de datos originalmente jerárquico.

Este trabajo recopila, en el **capítulo 5**, las conclusiones obtenidas a partir de los resultados de los experimentos, algunos trabajos futuros y la publicación realizada.

Finalmente, se adjuntan dos **apéndices** con el detalle de los resultados del capítulo 4 y una descripción de la implementación de las distancias entre términos.

2 Antecedentes

En este capitulo se describen algunos conceptos básicos del aprendizaje automático, además de algunos métodos, representaciones y medidas basadas en la similitud de instancias que sirven para comprender e ilustrar los capítulos porteriores sobre la aplicación de distancias entre términos para datos planos y jerárquicos.

2.1 Aprendizaje automático

El aprendizaje automático es el estudio de métodos computacionales que pueden aprender y mejorar a partir de la experiencia. Durante los últimos años, varios métodos de aprendizaje han sido desarrollados y utilizados en muchas aplicaciones prácticas. En [26] se presentan algunos de estos métodos: árboles de decisión, redes neuronales, aprendizaje bayesiano y métodos estadísticos, aprendizaje basado en instancias, algoritmos genéticos, reglas de decisión y programación lógica inductiva (ILP); existen otras técnicas también mencionadas por otros autores como son los conjuntos de clasificadores [7] y las máquinas de soporte vectorial [6].

Dentro de estos métodos es posible encontrar diferentes tipos de algoritmos de acuerdo con su función de salida o tarea. De ellos se pueden destacar los algoritmos de aprendizaje supervisado y algoritmos de aprendizaje no supervisado. El aprendizaje supervisado produce una función que establece una correspondencia entre las variables de entrada y la variable de salida deseada; el objetivo de su función es predecir esta variable de salida. Si la variable de salida es discreta es llamado clasificación y si la variable de salida es un valor continuo es regresión. Los algoritmos de aprendizaje no supervisado están conformados únicamente por variables de entrada, su objetivo es identificar relaciones entre variables o entre ejemplos; las relaciones entre las variables pueden

ser asociaciones, dependencias o correlaciones; las relaciones entre ejemplos son agrupamientos.

Otro tipo de clasificación de algoritmos de aprendizaje automático se basa en la inteligibilidad de los modelos generados. La principal ventaja de un modelo inteligible es que ofrece una explicación parcial de la realidad proporcionando conocimiento sobre cómo las predicciones están hechas. Los algoritmos con esta característica pueden ser divididos en dos grupos, dependiendo del lenguaje de representación que ellos usen. El primer grupo está formado por aprendizaje de árboles de decisión y métodos de aprendizaje de reglas clásicas; todos estos emplean reglas proposicionales que tienen una expresividad restrictiva porque no pueden tener variables. Un segundo grupo está compuesto por métodos que usan lenguajes de representación muy expresivos como cláusulas de Horn de primer orden, que permiten representar las reglas con características avanzadas como variables, predicados y llamado de funciones. Esta segunda familia es usualmente llamada programación lógica inductiva (ILP) [12] o, más en general, programación inductiva [29] [14] [23].

Finalmente, los algoritmos de aprendizaje también son clasificados como métodos *perezosos* o *ansiosos*. Los algoritmos *perezosos* son métodos basados en instancias, como por ejemplo el k-NN (k-vecinos más cercanos). Este tipo de algoritmos utilizan enfoques conceptualmente sencillos para las aproximaciones de valores reales o discretos de las funciones de salida. Aprender en estos modelos consiste en almacenar los datos de entrenamiento presentados y, cuando una nueva instancia es encontrada, un grupo de ejemplos similares relacionados son recuperados de memoria y usados para clasificar la nueva instancia consultada. Una diferencia clave en estos enfoques, con respecto a otros métodos, es que pueden construir una aproximación diferente de la función de salida para cada ejemplo que debe ser clasificado. De hecho, muchas técnicas construyen solo una aproximación local de la función de salida que se aplica en la vecindad de una nueva instancia y nunca construyen una aproximación diseñada para tener un buen rendimiento sobre todo el espacio de instancias de entrada. Esto tiene ventajas significantes cuando la función objetivo es muy compleja y puede ser descrita por una colección de aproximaciones locales menos complejas [26].

Una desventaja de los enfoques basados en instancias [26], es que el costo de clasificación de un ejemplo nuevo puede ser muy alto; esto se debe al hecho de que casi todo el cálculo tiene lugar en tiempo de clasificación y no cuando los ejemplos de entrenamiento son encontrados previamente. Por lo tanto, las técnicas para la indexación eficiente de ejemplos de entrenamiento son un tema significativamente práctico en la reducción del cómputo requerido en el momento de la consulta. Una segunda desventaja para muchos de estos enfoques basados en instancias, especialmente los enfoques del k-NN, es que se suelen considerar todos los atributos cuando se

trata de recuperar en memoria los ejemplos de entrenamiento similares. Si el valor de salida depende únicamente de unos pocos de los muchos atributos disponibles, entonces las instancias que son realmente más "similares"pueden tener una gran distancia de separación.

La tabla 2.1 muestra una clasificación de algunos algoritmos de aprendizaje automático de acuerdo con su inteligibilidad y la manera en que construyen su función de salida.

	Inteligible	No inteligible
Ansioso	-Árboles de decisión -Aprendizaje de reglas -Programación Lógica inductiva (ILP) -k-medias	-Redes neuronales artificiales (ANN) -Máquinas de soporte vectorial -Métodos Kernel -Clasificadores de bayesianos -Perceptrón
Perezoso		-k-vecinos más cercanos (k-NN) -Razonamiento basado en casos (CBR) -Regresión lineal ponderada local

Tabla 2.1: Clasificación de algoritmos de aprendizaje automático

2.2 Métodos basados en distancias

El razonamiento basado en casos (CBR) es un tipo de método basado en instancias que está relacionado con el aprendizaje basado en distancias. CBR supone que problemas similares tienen soluciones similares. En el aprendizaje automático es muy común utilizar la similitud entre objetos y existen muchos métodos que se fundamentan en establecer si un nuevo objeto es similar a uno previamente conocido. Una manera de hacerlo es cuantificando la similitud (o disimilitud) entre dos objetos por medio de una medida de distancia.

Más precisamente, una función $s : X \times X \to \mathbb{R}$ se dice que es una función de similitud si $s(x_i, x_j)$ es mayor cuando los objetos x_i y x_j son más similares. Por ejemplo, a continuación se define la función s, para atributos nominales, que cuenta el número de coincidencias ocurridas en la misma posición de una tupla [8]:

$$\begin{array}{rcl} s : X \times X & \rightarrow & \mathbb{R} \\ s(x_i, x_j) & \rightarrow & \sum_{k=1,2} s'(x_{ik}, x_{jk}) \end{array}$$

donde $x_i = (x_{i1}, x_{i2})$, $x_j = (x_{j1}, x_{j2})$ y

$$s'(x_{ik}, x_{jk}) = \begin{cases} 1, & \text{si } x_{ik} = x_{jk} \\ 0, & \text{otro caso} \end{cases}$$

Suponiendo que $x_1 = (a, a)$, $x_2 = (a, b)$ y $x_3 = (b, c)$ entonces $s(x_1, x_2) = 1$ y $s(x_1, x_3) = s(x_1, x_3) = 0$. Por lo tanto, x_1 y x_2 son más similares entre ellos que x_1 ó x_2 con respecto a x_3.

Al igual que la similitud, es posible cuantificar la disimilitud de dos objetos. Una función $s : X \times X \rightarrow \mathbb{R}$ se dice que es una función de disimilitud si $d(x_i, x_j)$ es mayor cuando x_i y x_j son menos similares [8].

La relación entre la similitud y la disimilitud es más clara cuando se trabaja con una función de similitud (o disimilitud) normalizada [8]; es decir cuando $0 \leq s, d \leq 1$. Esto además permite expresar una función en términos de la otra; es decir $d = 1 - s$ ó $s = 1 - d$.

La disimilitud es llamada distancia o métrica d cuando satisface las propiedades de *no negatividad*, *reflexivilidad*, *simetría* y *desigualdad triangular*, definidas de la siguiente manera:

1. No negatividad: $d(x_i, x_j) \geq 0, \forall x_i, x_j \in X$.
2. Reflexividad: $d(x_i, x_j) = 0 \Leftrightarrow x_i = x_j$.
3. Simetría: $d(x_i, x_j) = d(x_j, x_i)$.
4. Desigualdad triangular: $d(x_i; x_j) \leq d(x_i, x_k) + d(x_k, x_j), \forall x_i, x_j; x_k \in X$.

Estas propiedades muestran la interpretación de la distancia de manera más natural que una función de disimilitud. La propiedad de no negatividad nos garantiza que la distancia entre dos objetos x_i y x_j siempre es mayor o igual a cero; la reflexividad asegura que si la distancia entre

dos objetos es cero, entonces x_i y x_j son iguales. De igual manera, la simetría garantiza que la distancia entre dos objetos no depende del orden en que son comparados. Finalmente, si no se cumple la desigualdad triangular puede suceder que se obtienen resultados no deseables, por ejemplo, si los objetos x_i y x_j son muy similares a x_k, debe esperarse que x_i y x_j también sean muy similares.

Una de las principales ventajas de los métodos basados en distancias es que el algoritmo se puede ajustar para un problema específico por medio de la definición de una distancia adecuada; es decir, la distancia es un parámetro del algoritmo de aprendizaje. Igualmente, un algoritmo puede ser usado para cualquier representación de datos si una función de distancia se define sobre él. Por ejemplo, si se utiliza las distancias para datos proposicionales, en caso de tener datos estructurados, la mayoría de los métodos de aprendizaje se pueden usar directamente utilizando una definición de distancia para el tipo de datos manejado. Lo anterior implica que el rendimiento del algoritmo puede variar en función de la distancia utilizada [8].

Las funciones de distancia más comunes utilizadas para representaciones basadas en tuplas son la distancia euclídea, la distancia de Manhattan, la distancia de Chebychev, la distancia Minkowski y la distancia de Mahalanobis. Formalmente, definimos estas distancias de la siguiente manera:

Sea x una instancia arbitrariamente descrita por el vector de características [26]:

$$\langle a_1(x), a_2(x), \ldots, a_n(x) \rangle$$

donde $a_r(x)$ denota el valor de r-ésimo atributo de la instancia x; entonces la distancia entre dos instancias x_i y x_j, es definida por $d(x_i, x_j)$, para cada una de las siguientes distancias:

- Distancia euclídea

$$d(x_i, x_j) = \sqrt{\sum_{i=1}^{n} (a_r(x_i) - a_r(x_j))^2}$$

- Distancia de Manhattan

$$d(x_i, x_j) = \sum_{i=1}^{n} |(a_r(x_i) - a_r(x_j))|$$

- Distancia de Chebychev

$$d(x_i, x_j) = max_{i=1,\ldots,n} \left|(a_r(x_i) - a_r(x_j))\right|$$

- Distancia de Minkowski

$$d(x_i, x_j) = \left(\sum_{i=1}^{n} \left|(a_r(x_i) - a_r(x_j))\right|^p\right)^{1/p}$$

- Distancia de Mahalanobis

$$d(x_i, x_j) = \left((a_r(x_i) - a_r(x_j))^t \, S^{-1} \, (a_r(x_i) - a_r(x_j))\right)^{1/2}$$

  donde S es la matriz de covarianza

La distancia de Minkowski es una generalización de las diatancias euclídea, Manhattan y Chebychev, donde un parámetro p debe ser definido. Si $p = 1$, es la distancia de Manhattan, si $p = 2$, es la distancia euclídea y finalmente si $p = \infty$, es la distancia de Chebychev. Adicionalmente, la distancia euclídea es un caso particular de la distancia de Mahalanobis: en la distancia euclídea no se tiene en cuenta la correlación entre los atributos.

Los métodos basados en distancias consideran que cada ejemplo corresponde a un punto en el espacio métrico y el rendimiento de sus predicciones, mediante la comparación de la proximidad, difieren, en gran parte, en la manera en que son elegidos los ejemplos. A continuación se hace una breve descripción de los métodos más comunes; el k-vecinos más cercanos, el discriminante de Fisher y el aprendizaje por cuantificación vectorial son métodos supervisados y la agrupación jerárquica y k-medias son métodos no supervisados.

- **k-vecinos más cercanos:** es uno de los algoritmos más analizados en aprendizaje automático. Este algoritmo permite aproximar funciones de salida para valores discretos o continuos. Asume que las instancias corresponden a un punto en un espacio métrico n-dimensional. El valor de la función de salida para un nuevo ejemplo es estimado de los valores conocidos de los k ejemplos de entrenamiento más cercanos [26]. Este algoritmo se describe con más detalle en la sección 2.3.
- **Discriminante de Fisher:** cada clase es representada por medio un centroide que es un punto que minimiza la suma de la distancia para los elementos que pertenecen a una clase;

un nuevo ejemplo es marcado de acuerdo con la etiqueta de su centroide más cercano. Este método es *ansioso* y los ejemplos son removidos de la memoria una vez que el modelo (conjunto de centroides) haya sido obtenido. Dado que el espacio de representación está compuesto por tuplas de datos nominales o numéricos, se puede derivar un modelo más expresivo. Los centriodes de las clases adyacentes son unidos por líneas rectas y la mediana de estas líneas son las reglas discriminantes [8].

- **Aprendizaje por cuantificación vectorial (LVQ):** El modelo aprendido es una colección de prototipos donde un ejemplo es clasificado de acuerdo con la proximidad a estos prototipos. Inicialmente los prototipos son elegidos al azar, y sus posiciones se actualizan hasta que se alcanza un umbral. Este proceso se realiza de manera iterativa. Primero, un ejemplo del conjunto de entrenamiento es elegido y el prototipo más cercano es determinado. Dependiendo de las etiquetas, tanto del ejemplo como las del prototipo, la posición del prototipo cambia. Si las ambas etiquetas emparejan, el prototipo se acerca más al ejemplo [8].

- **Agrupamiento jerárquico:** este algoritmo se basa en la construcción de un árbol en el que las hojas son los elementos del conjunto de ejemplos, el resto son nodos con subconjuntos de ejemplos que pueden ser utilizados como particionamiento del espacio. Este árbol de grupos es también llamado dendrograma. Existen dos métodos para construir el árbol: 1) *Aglomerativo*. El árbol se va construyendo empezando por la hojas, hasta llegar a la raíz. Inicialmente, cada ejemplo es un grupo y se van aglomerando los grupos para formar conjuntos más numerosos hasta la raíz, que contiene todos los ejemplos. 2) *Desaglomerativo*. Se parte de la raíz, que es un solo grupo conteniendo a todos los ejemplos, y se hacen divisiones paulatinas hasta llegar a las hojas que representa a la situación en que cada ejemplo es un grupo [22].

  La forma en que los grupos son divididos o fusionados está guiada por el principio en que los elementos más cercanos deben permanecer en el mismo grupo. En consecuencia, es necesario conocer si dos grupos son cercanos. Teniendo en cuenta que la función de distancia es definida solo para un par de elementos, es necesario extenderla para medir la distancia entre grupos [8][22]. Las más comunes son:

  - **Enlace simple:** En este método, la distancia entre dos grupos es la distancia entre sus miembros más próximos, es decir, si U y V son dos grupos, entonces:

$$d_{UV} = min\{d_{ij} : i \in U, j \in V\}$$

- **Enlace completo:** la distancia entre dos grupos es la distancia entre sus miembros más alejados, es decir:

$$d_{UV} = max\left\{d_{ij} : i \in U, j \in V\right\}$$

- **Enlace promedio:** la distancia entre dos grupos es la distancia media entre todos los pares de unidades, donde un elemento del par es de un grupo, y el otro elemento pertenece al otro grupo, es decir, si n_u es el número de unidades en U, y n_v es el número de unidades en V , entonces:

$$d_{UV} = \frac{1}{n_u n_v} \sum_{i \in U} \sum_{j \in V} d_{ij}$$

En cuanto a los métodos *aglomerativos* , el par de grupos que están más cerca se fusionaran de acuerdo a la distancia de enlace. Si la distancia es el enlace promedio entonces los centroides deben ser recalculados. El proceso continúa hasta que un número previamente fijado de los grupos es logrado o el dendrograma completo es construido [8][22].

- **k-medias:** Los métodos de particionamiento son otro tipo importante de técnicas de agrupamiento. Los grupos son mejorados gradualmente de acuerdo con una función de optimización. Entre ellos el más conocido es k-medias. Inicialmente un conjunto de centroides son elegidos al azar; luego, los ejemplos son asignados a su centroide más cercano formando grupos y el centriode de los grupos nuevos es calculado. Este proceso se repite hasta que los centroides no cambien. Aunque este algoritmo ha tenido éxito en aplicaciones industriales y científicas, tiene algunos inconvenientes; el más significativo es que su desempeño depende fuertemente de los supuestos iniciales y del número de centroides propuestos [8].

Como se ha dicho, el algoritmo de aprendizaje de k-vecinos más cercanos, es uno de los algoritmos más analizados en para el aprendizaje automático. A continuación se introduce este algoritmo, incluyendo algunas variantes comúnmente utilizadas para este método.

2.3 k-vecinos más cercanos k-NN

El algoritmo k-NN, presentado en [26], asume que todas las instancias corresponden a puntos en un espacio n-dimensional $\Re^n$, aunque el algoritmo funciona igualmente para cualquier otro tipo de espacio, incluso sino es métrico. Los vecinos más cercanos de un ejemplo son definidos en términos de una distancia. Usualmente se utiliza la distancia euclídea; sin embargo, como se mencionó anteriormente, por ser un algoritmo basado en distancias, es posible utilizar cualquier otra distancia: la distancia de Manhattan, la distancia de Chebychev, etc.

En el aprendizaje del vecino más cercano, la función de salida puede ser un valor discreto (clasificación) o continuo (regresión). Considerando primero el aprendizaje para funciones de salida de valores discretos, la tabla 2.2 muestra el algoritmo de k-NN para aproximar esta función definida de la forma $f : \Re^n \longrightarrow V$, donde V es un conjunto finito de clases $\{v_1, \ldots, v_s\}$. El valor $\hat{f}(x_q)$, devuelto por el algoritmo como su estimación de $f(x_q)$, es el valor más común de f entre los k ejemplos de entrenamiento más cercanos a x_q. Si se elige $k = 1$, entonces el algoritmo asigna a $\hat{f}(x_q)$ el valor de $f(x_i)$ donde x_i es la instancia de entrenamiento más cercana a x_q. Para valores grandes de k, el algoritmo asigna el valor más común entre los k ejemplos más cercanos.

Algoritmo de entrenamiento:
- Por cada ejemplo de entrenamiento $\langle x, f(x)\rangle$, se adiciona un ejemplo a la lista de ejemplos de entrenamiento.

Algoritmo de clasificación:
- Dada una instancia de consulta x_q para ser clasificada,
  - Si $x_1, \ldots, x_k$ denota las k instancias de ejemplos de entrenamiento que son más cercanas a x_q.
  - Retorna

$$\hat{f}(x_q) \longleftarrow argmax_{v \in V} \sum_{i=1}^{k} \delta(v, f(x_i))$$

donde $\delta(a, b) = 1$ si $a = b$ y donde $\delta(a, b) = 0$ en otro caso.

Tabla 2.2: Algoritmo de k-NN para aproximar una función de valores discretos

Para funciones de salida de valores continuos, el algoritmo es fácilmente adaptable; el algoritmo calcula el valor medio de los k ejemplos de entrenamiento más cercanos, en lugar de calcular el valor más común. Más precisamente, para aproximar la función de salida de valores reales $f : \Re^n \longrightarrow \Re$ se remplaza $\hat{f}(x_q)$, utilizado en el algoritmo de valores discretos, por:

$$\hat{f}(x_q) \longleftarrow \frac{\sum_{i=1}^{k} f(x_i)}{k}$$

Un refinamiento para el algoritmo de k-NN puede ser realizado por medio de la función de *atracción*, denominada w; este refinamiento consiste en poderar la contribución de cada uno de los k-vecinos de acuerdo con su distancia para el punto de consulta x_q, dando mayor peso a los vecinos más cercanos. Por ejemplo, en el algoritmo de la tabla 2.2, el cual aproxima funciones de salida de valores discretos, es posible ponderar el voto de cada vecino de acuerdo con el cuadrado inverso de su distancia de x_q. Esto puede ser logrado reemplazando $\hat{f}(x_q)$ por:

$$\hat{f}(x_q) \longleftarrow argmax_{v \in V} \sum_{i=1}^{k} w_i \delta(v, f(x_i))$$

donde,

$$w_i \equiv \frac{1}{d(x_q, x_i)^2}$$

Esta mejora de ponderar los k vecinos más cercanos para un nuevo ejemplo, puede suavizar el impacto de los ejemplos de entrenamiento aislados o con ruido. Adicionalmente, es posible considerar que el exponente cuadrático, utilizado en el denominador de w, puede ser modificado por una variable denominada *parámetro de atracción* que permite incrementar o decrementar la ponderación w_i de manera que, mientras mayor sea esta variable, menos importante es k.

De igual manera, para funciones de salida con valores reales, la distancia ponderada de instancias utiliza un denominador constante que normaliza las contribuciones de varias ponderaciones.

De este modo, la función $\hat{f}(x_q)$ puede ser remplazada por:

$$\hat{f}(x_q) \longleftarrow \frac{\sum_{i=1}^{k} w_i f(x_i)}{\sum_{i=1}^{k} w_i}$$

El algoritmo k-vecinos más cercanos es uno de los algoritmos más analizados en el aprendizaje automático [26], esto se debe a su simplicidad y, por otra parte, a la edad que tiene. Es un método inductivo de inferencia muy eficaz en muchos problemas prácticos, especialmente cuando utiliza mejoras como la distancia ponderada; es robusto ante los ruidos de datos y suficientemente efectivo en conjuntos de datos grandes.

2.4 Distancias para datos estructurados y semi-estructurados

En el aprendizaje automático, los métodos basados en distancias pueden incorporar funciones de similitud definidas sobre datos estructurados (por ejemplo una tabla), semi-estructurados (por ejemplo un documento XML) o no estructurados (por ejemplo un documento de texto). Algunos métodos basados en similitud de datos estructurados son aplicados, en gran medida, a semi-estructurados; sin embargo, estos datos que no son completamente estructurados, es necesario aplicar técnicas que requieren procesos más complejos. Las distancias entre grafos (o en particular, distancias entre árboles) son comúnmente utilizadas como una medida de similitud para datos semi-estructurados. A continuación se describen algunos aspectos generales sobre datos estructurados y semi-estructurados y, finalmente, se detallan métodos basados en similitud de acuerdo con varios tipos de datos.

2.4.1 Datos estructurados y semi-estructurados

La principal diferencia que existe entre los datos estructurados y semi-estructurados está relacionada, obviamente, con la rigidez de su estructura; para los datos estructurados, un conjunto de instancias son definidas por variables con tipos de datos iguales y únicamente cambia el valor de esta variable (por ejemplo una tupla de tamaño fijo de valores reales, una imagen de $n \times m$ píxeles, etc.); los datos semi-estructurados estan conformados fundamentalmente por etiquetas que, aunque ofrecen cierto nivel estructural, también son altamente flexibles, con diferentes tipos de datos y dominios, además de datos no estructurados (como textos, imagenes, etc.) que adicionan dificultad a tareas de consulta y por lo tanto a la aplicación de médotos de aprendizaje (por ejemplo un documento en XML, un término lógico-funcional, una secuencia de ADN, etc.).

La definición de funciones de similitud entre objetos semi-estructurados se basa, en gran medida, en su parte estructural y la utilización de aquellas etiquetas existentes para un objeto; no obstante, existen técnicas para tratar los segmentos no estructurados de acuerdo con su contenido; por ejemplo, el procesamiento de lenguaje natural ofrece técnicas que permiten medir la similitud lingüística de objetos por medio de análisis léxico, sintáctico y semántico.

En un documento semi-estructurado, en aquellas partes que contienen algún tipo de estructura, se pueden aplicar funciones de similitud existentes para datos estructurados; aunque es posible encontrar diferentes dominios sobre variables conceptualmente iguales que obliga, en algunas ocasiones, a utilizar una serie de transformaciones y aplicar técnicas que permitan crear dominios comparables.

En los datos semi-estructurados, por sus características de etiquetado, es usual que se creen estructuras jerárquicas. Esto ha hecho que muchos casos hayan sido abordados mediante representaciones basadas en grafos o árboles; en consecuencia, es común que las funciones de distancia empleadas para medir la similitud entre instancias se basen también en distancias para este mismo tipo de datos. La similitud sobre datos semi-estructurados, han sido utilizadas principalmente para agrupación de documentos, y para detectar cambios entre estos. Algunos enfoques pueden ser vistos en [33], [15], [34].

2.4.2 Distancias entre datos

A continuación se decriben algunos métodos basados en similitud que son empleados acuerdo con los tipos datos más comunes (prensentados por [8], [11] y [4]): conjuntos, listas, árboles y grafos. Las distancias entre términos son un tema central en este trabajo, por esta razón son tratadas en la sección 2.6.

- **Conjuntos**:

  La cardinalidad de la diferencia simétrica entre dos conjuntos finitos A y B es una función de distancia $\mid (A-B) \cup (B-A) \mid$ que satisface la propiedad de identidad y simetría. La desigualdad triangular puede verse demostrando que para cualquier conjunto finito C, si el elemento x está en $\mid (A-B) \cup (B-A) \mid$ entonces x está en $\mid (A-C) \cup (C-A) \mid$ o en $\mid (C-B) \cup (B-C) \mid$, lo que implica que $\mid (A-B) \cup (B-A) \mid \leq \mid (A-C) \cup (C-A) \mid + \mid (C-B) \cup (B-C) \mid$. Cuando esta métrica es empleada, la distancia entre dos conjuntos es dada por el número de elementos que tienen en común. Los elementos más comunes y los elementos más cercanos. Por ejemplo, dado el siguiente conjunto de secuencias, $A = \{ab, a^4\}$, $B = \{ab, d^4\}$ y $C = \{ab, a^3\}$ entonces $d(A,B) = d(A,C) = 2$. Esta función asume que la distancia 0 ó 1 ($d(x,y) = 1$ si $x \neq y$, 0 otro caso) ha sido previamente definida sobre los elementos construidos en el conjunto; por lo tanto, la diferencia simétrica es una distancia que considera que cualquier elemento del conjunto es igualmente diferente al resto. Por ejemplo, esta distancia muestra que el elemento $d^4 \in B$ y $a^3 \in C$ son diferentes a $a^4 \in A$, cuando intuitivamente a^3 es más similar a a^4 que d^4.

  En algunos contextos es importante capturar las diferencias entre los elementos de los conjuntos de una manera más precisa a como lo hace la diferencia simétrica. La distancia de *Hausdorff*, por ejemplo, es una distancia que tiene en cuenta los elementos; y esta distancia está definida como:

$$d_H(A,B) = \text{máx} \left\{ \begin{array}{lll} \sup\limits_{a \in A} \inf\limits_{b \in B} & d(a,b), & \sup\limits_{b \in B} \inf\limits_{a \in A} \quad d(a,b) \end{array} \right\}$$

  donde A y B son dos conjuntos y $d(\cdot,\cdot)$ es una distancia definida sobre elementos.

- **Listas**:

  Una de las funciones para listas es la distancia de *Hamming*; Esta función es usada cuando las listas tienen longitud igual; la distancia entre dos listas es el número de posiciones para las cuales los símbolos son diferentes. Por ejemplo, dada la secuencia $s_1 = aabb$ y $s_2 = accb$ entonces la distancia $d(s_1, s_2) = 2$. La distancia de edición (tambien conocida como *Levenshtein*), es una generalización de la distancia de *Hamming*, que permite manejar secuencias con longitudes variables. Esta distancia cuenta el número mínimo de operaciones de eliminación, inserción y sustitución requeridas para transformar una secuencia en otra. Estas operaciones pueden ser ponderadas de acuerdo con el contexto del problema. Por lo tanto, la distancia corresponde a la transformación con el costo más bajo. A continuación se describe un ejemplo para esta distancia:

  Sea alfabeto $\Sigma = \{a, b, c\}$ y Σ^* un espacio finito de todas las listas construidas sobre los símbolos a,b y c. Se asigna un costo de 1 para inserciones y eliminaciones y, 2 para las sustituciones. La distancia de edición entre dos secuencias $s_1 = aabb$ y $s_2 = bbcc$ puede ser especificada por:

$$\begin{array}{cccccc} a & a & b & b & & \\ & & b & b & c & c \end{array}$$

  Entonces s_1 es transformada en s_2 borrando los dos primeros símbolos y luego adicionando dos símbolos al final:

$$s_1 = aabb \rightarrow abb \rightarrow bb \rightarrow bbc \rightarrow bbcc = s_2$$

  Lo anterior hace que $d(s_1, s_2) = 4$, esto es equivalente a transformar s_2 en s_1 borrando los dos símbolos b y luego adicionando los dos símbolos a. Las otras transformaciones pueden no ser óptimas, por ejemplo:

$$\begin{array}{cccccc} a & a & b & b & & \\ & b & b & c & c & \end{array}$$

La secuencia s_1 es transformada a s_2 removiendo el primer símbolo a (con costo 1); se sustituye el segundo símbolo a por b (con costo 2), luego se cambia el tercer símbolo b por c (con costo 2), finalmente se adiciona c al final de la secuencia (con costo 1)

$$s_1 = aabb \rightarrow abb \rightarrow bbb \rightarrow bbc \rightarrow bbcc = s_2$$

El costo total de esta transformación es 6; que es mayor a la primera transformación que tenía un costo de 4; por esa razón, esta última no es una trasformación óptima.

- **Árboles y grafos**:

La distancia de edición también puede ser utilizada para grafos y árboles etiquetados. Formalmente, un grafo puede ser definido $g(V, \alpha, \beta)$ donde V es un conjunto finito de vértices y $\alpha : V \rightarrow L$ es un nodo y $\beta : V \times V \leftarrow L$ es una función de etiquetado de aristas. En este contexto, los grafos son siempre completos si las aristas faltantes son consideradas como una etiqueta especial vacía.

Frecuentemente, se utiliza el concepto de *ecgm* (*error-correcting graph matching*) el cual es un conjunto de operaciones de edición para transformar un grafo en otro y el costo asociado a un *ecgm*.

Formalmente, sean $g_1 = (V_1, \alpha_1, \beta_1)$ y $g_2 = (V_2, \alpha_2, \beta_2)$ dos grafos. El *ecgm* de g_1 a g_2 es una función biyectiva $f : \hat{V}_1 \rightarrow \hat{V}_2$ donde $\hat{V}_1 \subset V_1$ y $\hat{V}_2 \subset V_2$.

El costo de un *ecgm* $f : \hat{V}_1 \rightarrow \hat{V}_2$ de un grafo $g_1 = (V_1, \alpha_1, \beta_1)$ y$g_2 = (V_2, \alpha_2, \beta_2)$ es dado por:

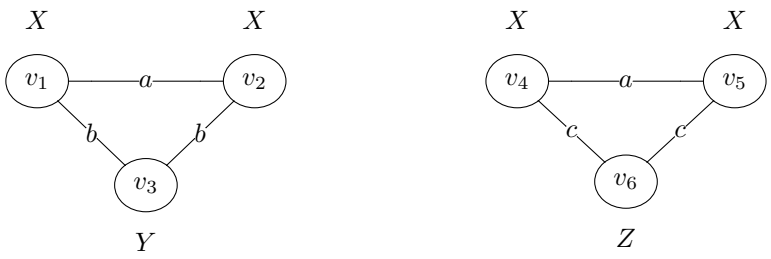
$$\begin{aligned} c(f) \quad = \quad & \textstyle\sum_{v \in \hat{V}_1} c_{ns}(v) + \sum_{v \in V_1 - \hat{V}_1} c_{nd}(v) + \sum_{v \in V_2 - \hat{V}_2} c_{ni}(v) \\ & + \textstyle\sum_{e \in \hat{E}_1} c_{es}(e) + \sum_{e \in E_1 - \hat{E}_1} c_{ed}(e) + \sum_{e \in E_2 - \hat{E}_2} c_{ei}(e) \end{aligned}$$

donde c_{ns}, c_{nd} y c_{ni} son el costo de sustitución, eliminación e inserción de un nodo,

respectivamente; e igualmente, c_{es}, c_{ed} y c_{ei} son el costo de sustitución, eliminación e inserción de una arista.

Finalmente, la mínima distancia entre dos grafos g_1 y g_2 es el mínimo costo obtenido sobre todos los *ecgm* de g_1 y g_2. Por ejemplo:

Sean $g_1 = (V_1, \alpha_1, \beta_1)$ y $g_2 = (V_2, \alpha_2, \beta_2)$ dos grafos representados en la figura 2.1:



(a) Grafo g_1 (b) Grafo g_2

Figura 2.1: Ejemplo de distancia entre dos grafos

- $V_1 = 1, 2, 3$; $V_2 = \{4, 5, 6\}$; $L = \{X, Y, Z, a, b, c, null\}$.
- $\alpha_1 : 1 \mapsto X, 2 \mapsto X, 3 \mapsto Y$
- $\alpha_2 : 4 \mapsto X, 5 \mapsto X, 6 \mapsto Z$
- $\beta_1 : (1, 2) \mapsto a, (1, 3) \mapsto b, (2, 3) \mapsto b$
- $\beta_2 : (4, 5) \mapsto a, (4, 6) \mapsto a, (5, 6) \mapsto c$

Un posible *ecgm* es $f : 1 \mapsto 4, 2 \mapsto 5$ con $\hat{V}_1 = \{1, 2\}$ y $\hat{V}_2 = \{4, 5\}$. Aplicando el *ecgm* los nodos 1 y 2 son sustituidos por 4 y 5, respectivamente. En consecuencia, la arista $(1, 2)$ es sustituida por $(4, 5)$. Todas estas sustituciones son idénticas en el sentido en que no hay cambios en la etiquetas; El nodo 3 y las aristas $1, 3$ y $2, 3$ son eliminadas y el nodo 6 junto con sus aristas $(4, 6)$ y $(5, 6)$ son insertados.

Si el costo de la función es definido de la siguiente manera:

$$c_{ns}(v) = \begin{cases} 0, & \text{si } \alpha_1(v) = \alpha_2(v) \\ \infty, & \text{otro caso} \end{cases}$$

$$c_{nd}(v) = 1, \forall v \in V_1 - \hat{V}_1$$

$$c_{ni}(v) = 1, \forall v \in V_2 - \hat{V}_2$$

$$c_{es}(e) = \begin{cases} 0, & \text{si } \beta_1(v) = \beta_2(v) \\ \infty, & \text{otro caso} \end{cases}$$

$$c_{ed}(v) = 0, \forall e \in E_1 - \hat{E}_1$$

$$c_{ci}(v) = 0, \forall e \in E_2 - \hat{E}_2$$

Se puede ver fácilmente que $d(g_1, g_2) = c(f) = 2$, puesto que el costo de eliminación o inserción de un nodo es igual a 1 (mientras que el costo de inserción o eliminacion de una arista es 0).

2.5 Representación basada en términos y XML

Además de la popularidad alcanzada por XML para la representación y el intercambio de datos en la Web [18], el principal interés de utilizar XML en este trabajo es la capacidad de representar estructuras como listas y árboles; por lo anterior, XML permite representar objetos basados en términos. A continuación se hace una corta descripción de este tipo de representaciones.

2.5.1 Representación basada en términos

La representación basada en términos busca obtener representaciones compactas donde los objetos están representados por medio de funtores los cuales son símbolos de función no-evaluadas cuyos argumentos son términos tan complejos como un problema lo requiera. En consecuencia, los funtores pueden ser usados para representar tamaños flexibles de tipos de datos ordenados tales como listas y árboles. Esto tiene la ventaja de que todos los datos relativos a un objeto se mantienen unidos [8][25]. El siguiente ejemplo muestra una representación basada en términos de un objeto del conjunto de datos de *Mushroom* (UCI *machine learning repository* [16]):

```
Mushroom(
    cap(CONVEX, SMOOTH, WHITE),
    BRUISES, ALMOND,
    gill(FREE, CROWDED, NARROW, WHITE),
    stalk(TAPERING, BULBOUS, surface(SMOOTH, SMOOTH), color(WHITE, WHITE)),
    veil(PARTIAL, WHITE),
    ring(ONE, PENDANT),
    spore(print(BROWN)),
    SEVERAL, WOODS)
```

El símbolo de predicado solo se refiere al objeto, mientras los funtores se refieren a partes de los objetos y las constantes se refieren a propiedades de estas partes. Por lo tanto, *Mushroom* es una tupla compuesta por *cap*, *gill*, *stalk*, *veil*, *ring* y *spore*, y varios atributos adicionales. A su vez, cada tupla puede estar compuesta de manera anidada por otras tuplas y atributos. La figura 2.2 muestra una representación en árbol del ejemplo anterior.

Una de las principales características de la representación basada en términos es que puede ser aplicada de una manera más natural que otros paradigmas que utilizan ejemplos con estructuras planas como datos tabulares, textos, etc. Las estructuras planas pueden obligar a tener campos en blanco (por ausencia de características) y no representar claramente relaciones jerárquicas entre variables.

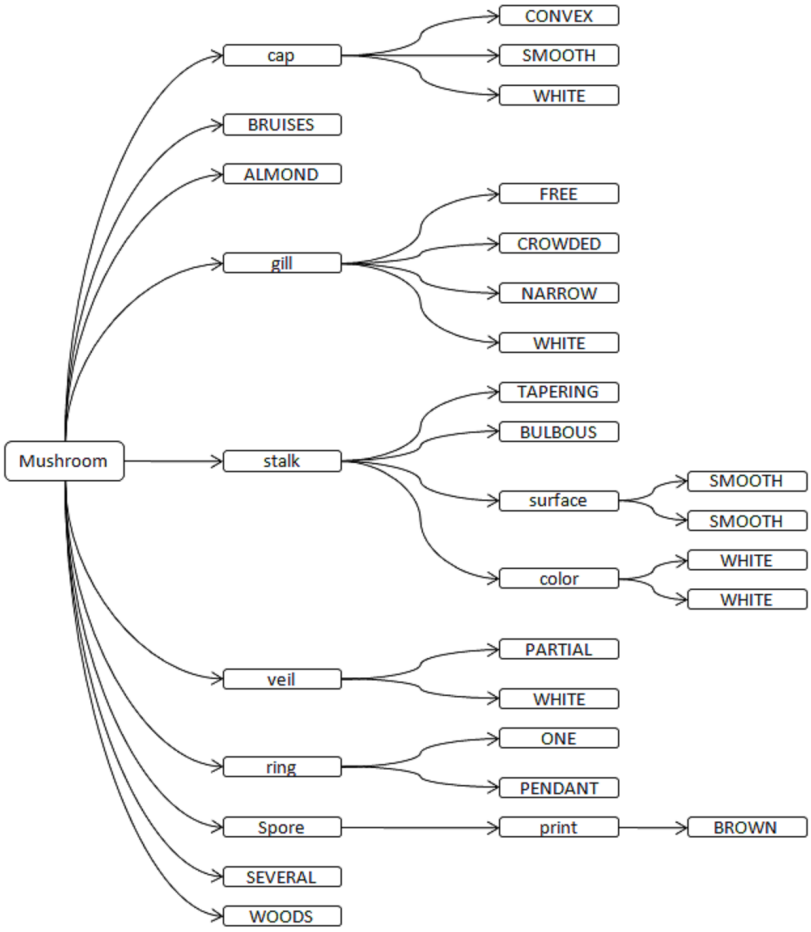


Figura 2.2: Representación en árbol de un objeto de *Mushroom*.

2.5.2 XML

XML es un lenguaje textual que ha ganado popularidad para la representación y el intercambio de datos en la Web. Los componentes básicos son elementos etiquetados que tienen una secuencia de cero o más pares atributo-valor y una secuencia de cero o más subelementos. Los subelementos pueden ser ellos mismos los elementos marcados o pueden ser segmentos de datos de texto sin etiqueta. Debido a que XML se define como un lenguaje textual y no como un modelo de datos, un documento XML siempre tiene un orden implícito (el orden puede o no ser relevante, sin embargo, es inevitable en una representación textual) [18].

XML permite jerarquizar y estructurar la información, describir los contenidos dentro del mismo documento y reutilizar partes del mismo; por esta razón un documento XML permite la representación de jerarquías y estructuras complejas que, al igual que la representación basada en términos, es más natural que estructuras planas; sin embargo, XML no es un término y deben considerarse aspectos como el orden de los elementos y la ausencia de características.

Un documento XML puede ser convertido en un árbol de términos funcionales usando listas y este puede ser tratado en el marco de programación lógico-funcional [22]. La figura 2.3 muestra una representacion en XML del objeto del ejemplo anterior.

```
<Mushroom>
  <cap>
    <shape>CONVEX</shape>
    <surface>SMOOTH</surface>
    <color>WHITE</color>
  </cap>
  <bruises>BRUISES</bruises>
  <odor>ALMOND</odor>
  <gill>
    <attachment>FREE</attachment>
    <spacing>CROWDED</spacing>
    <size>NARROW</size>
    <color>WHITE</color>
  </gill>
  <stalk>
    <shape>TAPERING</shape>
    <root>BULBOUS</root>
    <surface>
      <above_ring>SMOOTH</above_ring>
      <below_ring>SMOOTH</below_ring>
    </surface>
    <color>
      <above_ring>WHITE</above_ring>
      <below_ring>WHITE</below_ring>
    </color>
  </stalk>
  <veil>
    <type>PARTIAL</type>
    <color>WHITE</color>
  </veil>
  <ring>
    <number>ONE</number>
    <type>PENDANT</type>
  </ring>
  <spore>
    <print>
      <color>BROWN</color>
    </print>
  </spore>
  <population>SEVERAL</population>
  <habitat>WOODS</habitat>
</Mushroom>
```

Figura 2.3: Representación en XML de un objeto basado en términos.

2.6 Distancias entre términos

Las distancias entre términos son distancias muy adecuadas para la inferencia inductiva porque incorporan propiedades adicionales sobre las distancias tradicionales. Las distancias entre términos más relevantes son la distancia propuesta por Nienhuys-Cheng [28] y la distancia de J. Ramon et al. [31]. Cada una de estas dos distancias cuenta con propiedades específicas que las hacen muy diferentes entre sí. Por otra parte, V. Estruch [10] propone una distancia entre átomos/términos que busca integrar los beneficios de las distancias anteriores.

A continuación se describen las propiedades generales de las distancias entre términos y a qué distancias están asociadas estas propiedades; posteriormente, se presenta la notación para definir las distancias de Nienhuys-Cheng, J. Ramon et al. y Estruch et al.; finalmente se definen estas distancias (la distancia de Nienhuys-Cheng y Estruch et al. son definidas en las secciones 2.7 y 2.8, respectivamente).

2.6.1 Propiedades generales para las distancias entre términos

Las propiedades generales de las distancias entre términos (definidas en [10]) son:

- **Sensibilidad del contexto:** Es la posibilidad de considerar dónde ocurren las diferencias entre dos términos, de manera que las diferencias de las ocurrencias de símbolos que ocurren en posiciones más profundas cuentan menos, ya a que estas ofrecen menos información. Por ejemplo, la distancia entre $p(a)$ y $p(b)$ debe ser mayor que la distancia entre $p(f(a))$ y $p(f(b))$.
- **Fácil normalización:** es muy útil trabajar con distancias que puedan ser normalizadas. Por ejemplo aquellas que devuelven un valor real pueden ser fácilmente normalizadas.
- **Diferencias repetidas:** consiste en manejar apropiadamente las diferencias repetidas entre los términos. Por ejemplo si $r = p(a, a)$, $s = p(b, b)$ y $t = p(c, d)$, intuitivamente se espera que los términos r y s sean más cercanos que r y t (ó s y t).
- **Tamaño de las diferencias:** esta propiedad consiste en que si el tamaño de los términos se incrementa, entonces la distancia debe ser mayor. Por ejemplo, si la distancia $d(p(a), p(b)) = 1/2$ entonces la distancia $d(p(a), p(f(c)))$ debe ser mayor a $1/2$.

- **Manejo de las variables:** las variables son muy útiles cuando falta parte de la estructura de un objeto, ya que en caso de no existir, se requiere de conceptos extras para manejar símbolos de variables.
- **Componibilidad:** permite definir las funciones de distancia para tuplas, combinando las funciones de distancia definidas sobre tipos básicos sobre los cuales se construye esta tupla. Comúnmente, esta combinación es hecha como una combinación de distancias base.
- **Ponderaciones:** consiste en dar mayor o menor ponderación para posiciones específicas de un término sobre otros. Por ejemplo, la distancia entre $f(a)$ y $f(b)$ puede incrementarse si estos términos son reescritos como $f(d_1(d_2(a)))$ y $f(d_1(d_2(b)))$.

La tabla 2.3 muestra las propiedades para las distancias de Nienhuys-Cheng, J. Ramon et al. y Estruch et al. La distancia de Nienhuys-Cheng solo tiene en cuenta la componibilidad, la normalización y, aunque no siempre, el contexto en el que las diferencias se producen. Por el contrario, la distancia de J. Ramon et al. a pesar de cumplir con muchas de las propiedades, adolece de no cumplir fácilmente la propiedad de normalización y componibilidad y, al igual que la distancia de Nienhuys-Cheng, no siempre cumple la sensibilidad de contexto. Obviamente, la distancia de Estruch et al., por tratarse de un distancia que integra las propiedades de las dos distancias anteriores cumple, directa o indirectamente, todas las propiedades.

	Nienhuys-Cheng	J. Ramon et al.	Estruch et al.
Contexto	No siempre	No siempre (dependiendo de la ponderación usada)	Sí
Normalización	Sí	No es fácil	Sí
Repeticiones	No	Sí	Sí
Tamaño	No	Sí	Sí
Variables	Indirectamente	Sí	Indirectamente
Componibilidad	Sí	Difícil	Sí
Ponderaciones	No	Sí	Indirectamente

Tabla 2.3: Propiedades de las distancias entre términos [10].

2.6.2 Notación [10]

Sea $\mathcal{L}$ un lenguaje de primer orden definido sobre $\Sigma = \langle \mathcal{C}, \mathcal{F}, \Pi \rangle$, donde $\mathcal{C}$ es un conjunto de constantes, y $\mathcal{F}$ (Π, respectivamente) es una familia indexada en $\mathbb{N}$ (enteros no negativos), donde $\mathcal{F}_n$ (Π_n) es un conjunto de la función $n-$aria de símbolos de predicado. Los átomos y términos son construidos, como es usual, de Σ. El símbolo raíz y la aridad de una expresión t está dado por las funciones $Root(t)$ y $Arity(t)$, respectivamente. Por lo tanto, siendo $t = p(a, f(b))$, $Root(t) = p$ y $Arity(t) = 2$.

Considerando la representación habitual de t como un árbol etiquetado, las ocurrencias son secuencias finitas de números positivos (separados por puntos) que representan una ruta de acceso en t. Se supone que cada ocurrencia siempre está encabezada por un símbolo especial (implícito) λ, el cual denota la ocurrencia vacía. El conjunto de todas las ocurrencias de t es denotado por $O(t)$. Para este caso, $O(t) = \{\lambda, 1, 2, 2.1\}$. Se utilizan letras minúsculas (indexadas) $o', o, o_1, o_2, \ldots$ para representar las ocurrencias. La longitud de una ocurrencia o, $Length(o)$, es el número de elementos en o (excluyendo λ). Por ejemplo, $Length(2.1) = 2$, $Length(2) = 1$ y $Length(\lambda) = 0$. Adicionalmente, si $o \in O(t)$ entonces $t|_o$ representa el subtérmino de t en la ocurrencia o. En el ejemplo anterior, donde $t = p(a, f(b))$, $Root(t) = p$, $t|_1 = a$, $t|_2 = f(b)$, $t|_{2.1} = b$. En cualquier caso, siempre se tiene que $t|_\lambda = t$. Para $Pre(o)$, se denota el conjunto de todas los prefijos de las ocurrencias de o diferentes de o. Por ejemplo, $Pre(2.1) = \{\lambda, 2\}$, $Pre(2) = \{\lambda\}$ and $Pre(\lambda) = \emptyset$. Dos expresiones s y t son compatibles (denotado por la función booleana $Compatible(s, t)$) si $Root(s) = Root(t)$ y $Arity(s) = Arity(t)$. De lo contrario, se dice que s y t son incompatibles ($\neg Compatible(s, t)$).

2.6.3 Definición de las distancias entre términos

A continuación se define la distancia de J. Ramon et al. Las distancias de Nienhuys-Cheng y Estruch et al. son definidas en el sección 2.7 y 2.8.

Distancia de J. Ramon et al.

La distancia J. Ramon et al. [10], se basa en las diferencias sintácticas con relación a su operador *lgg* (*least general generalisation* [30]); adicionalmente utiliza una función de tamaño para calcular esta distancia. El tamaño es definido como $Size(t) = (F, V)$, donde F cuenta el

número de predicados y símbolos de función que ocurren en t; V es la suma de la frecuencia al cuadrado de la aparición de cada variable en t.

Dados dos términos s y t la distancia de J. Ramon et al., denotada por d_R, es definida de la siguiente manera:

$$d_R(s,t) = [Size(s) - Size(lgg(s,t))] + [Size(t) - Size(lgg(s,t))]$$

Esta distancia devuelve un par de valores (F,V) que expresa qué tan diferentes son los términos de función y los símbolos de variables. Por ejemplo, si $s = p(a,b)$ y $t = p(c,d)$ y se conoce que el $lgg(s,t) = p(X,Y)$, entonces:

$$\begin{aligned}
&Size(s) = (3,0)\\
&Size(t) = (3,0)\\
&Size(lgg(s,t)) = (1,2)\\
&d_R(s,t) = [(3,0) - (1,2)] + [(3,0) - (1,2)] = (2,-2) + (2,-2) = (4,-4)
\end{aligned}$$

Debido a que su función de salida no devuelve un valor numérico, sino un par de valores, es difícil la aplicación directa de esta distancia sobre algoritmos de clasificación tradicionales basados en distancias.

2.7 Distancia de Nienhuys-Cheng [28]

La distancia de Nienhuys-Cheng, como se mencionó anteriormente en la sección 2.6.1, tiene en cuenta la profundidad de las ocurrencias de símbolos, de manera que aquellas diferencias que ocurren más cerca del elemento raíz cuenta más.

Dadas dos expresiones, $s = s_0(s_1,\ldots,s_n)$ y $t = t_0(t_1,\ldots,t_n)$, la distancia de Nienhuys-Cheng, denotada por $d_N(s,t)$, es definida recursivamente como:

$$d_N(s,t) = \begin{cases} 0, & \text{si } s = t \\ 1, & \text{si } \neg Compatible(s,t) \\ \frac{1}{2n}\sum_{i=1}^{n} d(s_i,t_i), & \text{otro caso} \end{cases}$$

Por ejemplo, sean $s = p(a, a)$ y $t = p(f(b), f(b))$ dos expresiones; entonces,

$$d_N(s,t) = \frac{1}{4} \cdot \big(d(a, f(b)) + d(a, f(b))\big) = \frac{1}{4}\big(1+1\big) = \frac{1}{2}$$

Debe destacarse que la función de salida de esta distancia devuelve un valor númerico que puede ser facilmente utilizado por algoritmos tradicionales de aprendizaje basados en distancias.

2.8 Distancia de Estruch et al. [10]

La distancia entre átomos/términos de Estruch et al. integra las propiedades de las distancias de Nienhuys-Cheng y J. Ramon et al. para ello los autores definen el concepto de diferencias sintácticas de la expresión, el tamaño y el valor del contexto. A continuación se definen estos aspectos y finalmente esta distancia.

Sean s y t dos expresiones, el conjunto de las diferencias sintácticas, denotado por $O^\star(s,t)$, es definido como:

$$\begin{aligned} O^\star(s,t) \quad = \quad \{o \in O(s) \cap O(t) : \quad & \neg Compatible(s|_o, t|_o) \text{ y} \\ & Compatible(s|_{o'}, t|_{o'}), \forall o' \in Pre(o)\} \end{aligned}$$

Luego, la complejidad de las diferencias sintácticas entre s y t son calculadas sobre el número de símbolos de subtérminos (en s y t) en las ocurrencias que tiene $o \in O^\star(s,t)$. Para este propósito, se utiliza una función especial de tamaño de una expresión denotada por $Size'(t)$ definida de la siguiente manera:

Dada una expresión $t = t_0(t_1, \ldots, t_n)$ entonces $Size'(t) = \frac{1}{4}Size(t)$ donde,

$$Size(t_0(t_1, \ldots, t_n)) = \begin{cases} 1, n = 0 \\ 1 + \frac{\sum_{i=1}^{n} Size(t_i)}{2(n+1)}, n > 0 \end{cases}$$

Por ejemplo, considerando $s = f(f(a), h(b), b)$, entonces $Size(a) = Size(b) = 1$, $Size(f(a)) =$

$Size(h(b)) = 1 + 1/4 = 5/4$, $Size(s) = 1 + (5/4 + 5/4 + 1)/8 = 23/16$ y finalmente, $Size'(s) = 23/64$.

El valor de contexto de una ocurrencia o en una expresión t, $C(o;t)$, es usado para considerar la relación entre $t|_o$ y t en el sentido que, un alto valor de $C(o;t)$ corresponde a una posición de profundidad de $t|_o$ en t o la existencia de supertérminos de $t|_o$ con un gran número de argumentos. Este concepto se formaliza de la siguiente manera:

Sea t una expresión. Dada una ocurrencia $o \in O(t)$, el valor de contexto de o en t, denotado por $C(o;t)$, es definido como:

$$C(o;t) = \begin{cases} 1, \, o = \lambda \\ 2^{Length(o)} \cdot \prod_{\forall o' \in Pre(o)} (Arity(t|_{o'}) + 1), \text{ otro caso} \end{cases}$$

Está demostrado en [10] que si $o \in O^\star(s,t)$ entonces $C(o;s) = C(o;t)$. De esta manera, en estos casos, el contexto de una ocurrenca $o \in O^\star(s,t)$ es denotado por $C(o)$. Las difrencias repetidas se manejan a través de una relación de equivalencia ($\sim$) sobre el conjunto $O^\star(s,t)$ definido de la siguiente manera:

$$\forall o_i, o_j \in O^\star(s,t), \; o_i \sim o_j \Leftrightarrow s|_{o_i} = s|_{o_j} \text{ y } t|_{o_i} = t|_{o_j}$$

el cual produce una particion $O^\star(s,t)$ que no se sobrepone en la equivalencia de clases. También para la relación ($\leq$), en cada equivalencia de clase $O_i^\star(s,t)$, es definida como $\forall o_j, o_k \in O_i^\star(s,t), \; o_j \leq o_k \Leftrightarrow C(o_j) \leq C(o_k)$.

Adicionalmente, los conceptos previos son usados para definir otra función la cual simplemente asocia pesos para las ocurrencias de manera que al mayor $C(o)$, se asigna el menor de los pesos de o, es decir, el menos significativo de la diferencia sintáctica referidos en o. Por lo tanto, dadas dos expresiones s y t, la función de peso w es:

$$\forall o \in O_i^\star(s,t), \; w(o) = \frac{3f_i(o) + 1}{4f_i(o)}$$

donde $i = \pi(o)$, $\pi(o)$ es el índice de la equivalencia de clase perteneciente a o, y $f_i(o)$ es la posición que tiene o de acuerdo con $\leq$.

Finalmente, la distancia de Estruch et al. es definida como:

Sean s y t dos expresiones, la distancia entre s y t es,

$$d_E(s,t) = \sum_{o \in O^\star(s,t)} \frac{w(o)}{C(o)} \big(Size'(s|_o) + Size'(t|_o) \big)$$

Por ejemplo, sean $s = p(a,a)$ y $t = p(f(b), f(b))$ dos expresiones. Entonces, $O^\star(s,t) = \{1,2\}$. Además,

$$C(1) = C(2) = 2 \cdot (2+1) = 6$$

Los tamaños de los subtérminos involucrados en el cálculo de la distancia son:

$$Size'(a) = 1/4 \text{ and } Size'(f(b)) = 5/16$$

Hay solo una clase de equivalencia $O^\star = O_1^\star(s,t)$. Se supone que la ocurrencia 1 ocupa el primer lugar,

$$w(1) = 1 \text{ and } w(2) = 7/8$$

Finalmente,

$$d_E(s,t) = \frac{1}{6}\big(\frac{1}{4} + \frac{5}{16}\big) + \frac{7}{48}\big(\frac{1}{4} + \frac{5}{16}\big)$$

3 Transformación de datos semi-estructurados en una representación basada en términos usando XML.

Para aplicar distancias entre términos para diferentes tipos de datos se requiere un lenguaje común que permita su representación. XML, por sus características propias de su estructura, es un lenguaje flexible que permite la representación desde datos planos hasta datos jerárquicos y sobre el cual es posible aplicar distancias entre términos; sin embargo, es necesario controlar algunos aspectos su estructura; por consiguiente, con respecto a la estructura, dos situaciones extremas pueden ser identificadas (obviamente, existen casos intermedios que pueden ser manejados utilizando una mezcla de estas dos situaciones):

- **Datos planos:** muchos conjuntos de datos son dados como una tabla de pares atributo-valor. Sin embargo, si se hace una inspección detallada es posible identificar que algunos atributos se relacionan entre sí y pueden inducirse jerarquías sobre ellos.
- **Datos jerárquicos:** otros conjuntos de datos consisten en datos con una estructura jerárquica que está representada por un árbol o un término funcional (por ejemplo en Haskell o LISP).

A continuación se hace una descripción de cómo controlar los aspectos relacionados con la estructura de XML y cómo derivar esquemas XML a partir de datos planos y jerárquicos.

3.1 Definición del esquema

Debido a que los documentos XML y los términos funcionales no son lo mismo, se deben tomar algunas decisiones con el fin de utilizar XML para representar los términos funcionales y adaptar los tipos de datos anteriores en una representación común. Un aspecto clave es cómo manejar la profundidad, la repetición y el orden; los elementos en XML deben tener orden y considerar la posibilidad de permitir repeticiones sobre diferentes niveles de profundidad.

Al aplicar estas distancias entre términos sobre una jerarquía, se debe garantizar su correcta correspondencia entre los diferentes elementos, ya que algunas partes pueden estar vacías, y determinar la manera en que se deben comparar. En otras palabras, la jerarquía puede tener diferente número de elementos, en el mismo nivel, por lo que es necesario garantizar que los cálculos de la distancia no sean afectados por la ausencia de características o por su orden. Esta dificultad requiere de la creación de un esquema general que permita a cada instancia, con sus características propias, ajustarse apropiadamente en un orden definido y sin perder ningún elemento o contenido. El esquema propuesto es un documento XML que contiene etiquetas de todos los elementos que pueden existir para un ejemplo dado.

```
...
 <ANATOMY>
   <DEF>ANATOMIC-FEATURES</DEF>
   <ORNAMENTATION>SMOOTH-ORNAMENTATION</ORNAMENTATION>
   <ECTOSOME>
     <DEF>VISIBLE-ECTOSOME</DEF>
     <SEPARABLE>YES</SEPARABLE>
   </ECTOSOME>
   <CORTEX>
     <DEF>CORTEX-CHARACTERISTICS</DEF>
     <CORTICAL-SPICULES>YES</CORTICAL-SPICULES>
     <THICK>3000</THICK>
   </CORTEX>
 </ANATOMY>
...
```

(a)

```
...
 <ANATOMY>
   <DEF>ANATOMIC-FEATURES</DEF>
   <ECTOSOME>
     <DEF>VISIBLE-ECTOSOME</DEF>
     <SEPARABLE>YES</SEPARABLE>
     <SPICULES>YES</SPICULES>
     <SKELETON>
       <DEF>SKELETON-CHARACTERISTICS</DEF>
       <ARRANGEMENT>PALISADE</ARRANGEMENT>
       <ASPECT>ORDERED</ASPECT>
     </SKELETON>
   </ECTOSOME>
   <CORTEX>
     <DEF>CORTEX-CHARACTERISTICS</DEF>
     <CORTICAL-SPICULES>YES</CORTICAL-SPICULES>
     <THICK>3000</THICK>
   </CORTEX>
 </ANATOMY>
...
```

(b)

Figura 3.1: Ejemplo de jerarquías utilizando diferentes elementos

Por ejemplo, la figura 3.1 muestra dos jerarquías donde <SPICULES> y <SKELETON> son elementos que existen en la jerarquía (b), pero no en (a); de la misma manera, el elemento <ORNAMENTATION> existe para la jerarquía (a), pero no para (b). Por lo tanto, es necesario crear

un esquema que contenga una estructura integrada para ambos casos; es decir, un esquema para las jerarquías de la figura 3.1 tendría, no sólo aquellos elementos comunes, sino también aquellos que están en alguna de las dos jerarquías como <SPICULES> , <SKELETON> y <ORNAMENTATION>. La figura 3.2 muestra todas las posibles etiquetas de un esquema utilizando las jerarquías (a) y (b) de la figura 3.1.

```
<ANATOMY>
  <DEF/>
  <ORNAMENTATION/>
  <ECTOSOME>
    <DEF/>
    <SEPARABLE/>
    <SPICULES/>
    <SKELETON>
      <DEF/>
      <ARRANGEMENT/>
      <ASPECT/>
    </SKELETON>
  </ECTOSOME>
  <CORTEX>
    <DEF/>
    <CORTICAL-SPICULES/>
    <THICK/>
  </CORTEX>
</ANATOMY>
```

Figura 3.2: Ejemplo de todas las posibles etiquetas utilizando las jerarquías de la figura 3.1.

El esquema permite, no solo tener una estructura general, sino también garantizar el formato, de manera que cada caso debe seguir esta estructura. Sin embargo, es necesario tener en cuenta que no todos los casos se pueden adaptar fácilmente a un ejemplo XML, por ejemplo, aquellos elementos que son hojas con contenido no pueden ser adaptados directamente cuando el esquema requiere un árbol. En esta situación el elemento es agregado independientemente de la estructura del esquema. Sin embargo, este hecho no afecta los cálculos de la distancia ya que se consideran como características diferentes.

A pesar de que existen diferencias entre una representación basada en términos y una representación en XML es posible convertir una representación en otra (como se presentó en la sección 2.5). En XML, cada término se representa como un elemento compuesto por una etiqueta y un valor; de esta manera, es posible convertir un término en un elemento XML añadiendo una etiqueta e insertando el término como un valor. También es posible convertir un documento XML en un término sin tener en cuenta la etiqueta y representado el valor como un término. Por ejemplo, las figuras 3.3 y 3.7 muestran cómo un documento XML puede representar dos tipos de jerarquía diferentes.

3.2 Derivación de esquemas XML jerárquicos a partir de datos planos

Los datos planos se refieren a problemas atributo-valor que son comunes en bases de datos, aprendizaje automático, minería de datos y otras áreas. En otras palabras, los datos son presentados en forma de una tabla de valores escalares. Sin embargo, en muchos casos, algún tipo de estructura puede ser inferida a partir de estos conjuntos de datos, ya sea porque originalmente fueron así, después de un proceso de aplanamiento, o porque algunas características pueden ser agrupadas de acuerdo con alguna razón. El hecho derivar una jerarquía a partir de datos planos es uno de los problemas del área del conocimiento de computación granular tratados en [1].

Para este caso se consideran tres posibles fuentes de estructuras de representación de datos planos:

1. **Igualdad de valor:** muchos conjuntos de datos son dados como datos planos, pero al hacer una revisión detallada es posible encontrar que algunos atributos están relacionados por los valores que toman. Por ejemplo, si dos variables X_1 y X_2 pueden tomar el valor $Este$, $Oeste$, $Norte$ y Sur, hay claramente una conexión entre ellas que puede ser explotada, especialmente a través del uso de igualdades. Por ejemplo, es posible definir una condición o regla usando X_1 y X_2 que solo tiene sentido si los tipos de datos son iguales; de la misma manera en que las repeticiones de una variable son permitidas en términos funcionales, como $f(a, X, X)$.

2. **Jerarquía inducida por nombre:** en muchos casos es posible encontrar simples estructuras en la jerarquía de atributos de acuerdo con sus nombres o su semántica. Por ejemplo *cap-shape*, *cap-surface* y *cap-color* son atributos que contienen sub-características de "*cap*". De esta manera, puede ser creada una jerarquía de un grupo de características.

3. **Jerarquía de similitud de atributos:** en otras ocasiones donde los nombres y los valores pueden ser diferentes, es posible establecer relaciones entre atributos los cuales pueden ser usados para inducir a una estructura. Un enfoque es conocido como un árbol de aglomeración de variables de Watanabe-Kraskov [32][24], el cual construye un dendrograma (un árbol de jerarquías) usando la métrica de similitud entre atributos.

A continuación se detallan estos tres aspectos utilzando algunos ejemplos.

3.2.1 Igualdad de valor

La igualdad de valor utiliza la relación existente entre los valores posibles de dos o más variables; para el caso en que existan dominios similares, estas variables pueden ser marcadas para ser tenidas en cuenta como un criterio adicional al momento de un proceso de clasificación o agrupamiento. La tabla 3.1 muestra el dominio de algunos atributos del conjunto de datos de *mushroom* (de UCI *machine learning repository* [16]), en el cual los atributos *stalk-color-above-ring* y *stalk-color-below-ring* están relacionados entre ellos; adicionalmente, es posible ver como *cap-color*, aunque no es exactamente igual, es muy similar son respecto a los otros dos atributos.

Atributo	Dominio
cap-color	brown, buff, cinnamon, gray, green, pink, purple, red, white, yellow
stalk-color-above-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow
stalk-color-below-ring	brown, buff, cinnamon, gray, orange, pink, red, white, yellow

Tabla 3.1: Ejemplo similaridad entre dominos de variables

La forma de relacionar estos atributos en XML depende de cómo las repeticiones y relaciones entre estas variables son tenidas cuenta por el algoritmo utilizado para calcular la distancia. Este algoritmo podría tener en cuenta solo los valores de las variables o tener en cuenta el par atributo-valor. En el primer caso no es requerido ningún tipo de transformación; para el segundo caso es necesario transformar el nombre de las etiquetas de las variables relacionadas, en un nombre común.

3.2.2 Jerarquía inducida por nombre

La figura 3.3 muestra un ejemplo de jerarquía la cual es inducida por los nombres de los atributos originales utilizando el conjunto de datos plano de *mushroom* de UCI [16], el cual originalmente no tiene una estructura (como se muestra en 3.3(a)). Después de agrupar por prefijos de nombres comunes se obtiene una jerarquía la cual es finalmente representada como

1. cap-shape
2. cap-surface
3. cap-color
4. bruises
5. odor
6. gill-attachment
7. gill-spacing
8. gill-size
9. gill-color
10. stalk-shape
11. stalk-root
12. stalk-surface-above-ring
13. stalk-surface-below-ring
14. stalk-color-above-ring
15. stalk-color-below-ring
16. veil-type
17. veil-color
18. ring-number
19. ring-type
20. spore-print-color
21. population
22. habitat

(a) Atributos originales

```
<Mushroom>
  <cap>
    <shape>CONVEX</shape>
    <surface>SMOOTH</surface>
    <color>WHITE</color>
  </cap>
  <bruises>BRUISES</bruises>
  <odor>ALMOND</odor>
  <gill>
    <attachment>FREE</attachment>
    <spacing>CROWDED</spacing>
    <size>NARROW</size>
    <color>WHITE</color>
  </gill>
  <stalk>
    <shape>TAPERING</shape>
    <root>BULBOUS</root>
    <surface>
      <above_ring>SMOOTH</above_ring>
      <below_ring>SMOOTH</below_ring>
    </surface>
    <color>
      <above_ring>WHITE</above_ring>
      <below_ring>WHITE</below_ring>
    </color>
  </stalk>
  <veil>
    <type>PARTIAL</type>
    <color>WHITE</color>
  </veil>
  <ring>
    <number>ONE</number>
    <type>PENDANT</type>
  </ring>
  <spore>
    <print>
      <color>BROWN</color>
    </print>
  </spore>
  <population>SEVERAL</population>
  <habitat>WOODS</habitat>
</Mushroom>
```

(b) Jerarquía inducida usando nombres comunes

Figura 3.3: Jeraquía inducida por nombres para el conjunto de datos de mushroom

un documento XML (mostrado en la figura 3.3(b)). Este documento XML puede ser procesado como un término funcional (como es presentado en 2.5); en concreto:

```
Mushroom(
    cap(CONVEX, SMOOTH, WHITE),
    BRUISES, ALMOND,
    gill(FREE, CROWDED, NARROW, WHITE),
    stalk(TAPERING, BULBOUS, surface(SMOOTH, SMOOTH), color(WHITE, WHITE)),
    veil(PARTIAL, WHITE),
    ring(ONE, PENDANT),
    spore(print(BROWN)),
    SEVERAL, WOODS)
```

3.2.3 Jerarquía de similitud de atributos

Este enfoque se basa en la idea de encontrar la similitud entre los atributos. Cuando los atributos son numéricos, normalmente se realiza a través de medidas de correlación. En el caso de atributos nominales, se pueden utilizar otras medidas de asociación como una prueba Chi-cuadrado; de esta manera se construye una matriz de similitud de atributos y, como se mencionó anteriormente, un dendrograma.

La tabla 3.2 es un ejemplo de asociación entre atributos utilizando una muestra del conjunto de datos *mushroom* con la prueba Chi-cuadrado de Weka [19] como medida de similitud. Posteriormente, la tabla 3.3 muestra la matriz de similitud resultante constrida a partir de los resultados de la tabla 3.2.

Con los datos de la matriz de similitud (mostrada en la tabla 3.3) se construye un dendrograma utilizando los 22 atrubutos del conjunto de datos anterior. En este dendrograma, en lugar de usar toda la jerarquía, sólo se utilizan los segmentos más grandes, debido a que algunas variables quedarían muy profundas y con muy baja ponderación. La figura 3.4 muestra el dendrograma resultante (construido por medio la utilidad 'DendroUPGMA' [17]) e identificando cuatro grupos como se muestra en la figura 3.5.

Atributo 1	Atributo 2	Prueba chi-cuadrado
cap-shape	cap-surface	83.49
cap-shape	cap-color	215.39
cap-shape	bruises	87.10
cap-shape	odor	327.91
cap-shape	gill-attachment	4.30
cap-shape	⋮	⋮
⋮	⋮	⋮
gill-attachment	cap-shape	4.30
gill-attachment	cap-surface	3.71
gill-attachment	cap-color	201.46
gill-attachment	bruises	3.33
gill-attachment	odor	358.79
gill-attachment	⋮	⋮
⋮	⋮	⋮

Tabla 3.2: Ejemplo de asociación entre variables utilizando la prueba Chi-cuadrado

	cap-shape	cap-surface	cap-color	bruises	odor	gill-attachment	...
cap-shape	0	83.49	215.39	87.10	327.91	4.30	...
cap-surface	83.49	0	167.81	4.04	135.83	3.71	...
cap-color	215.39	167.81	0	56.75	520.88	201.46	...
bruises	87.10	4.04	56.75	0	242.75	3.33	...
gill-attachment	327.91	135.83	520.88	242.75	0	358.79	...
odor	4.30	3.71	201.46	3.33	358.79	0	...
⋮	⋮	⋮	⋮	⋮	⋮	⋮	0

Tabla 3.3: Matriz de similitud entre atributos

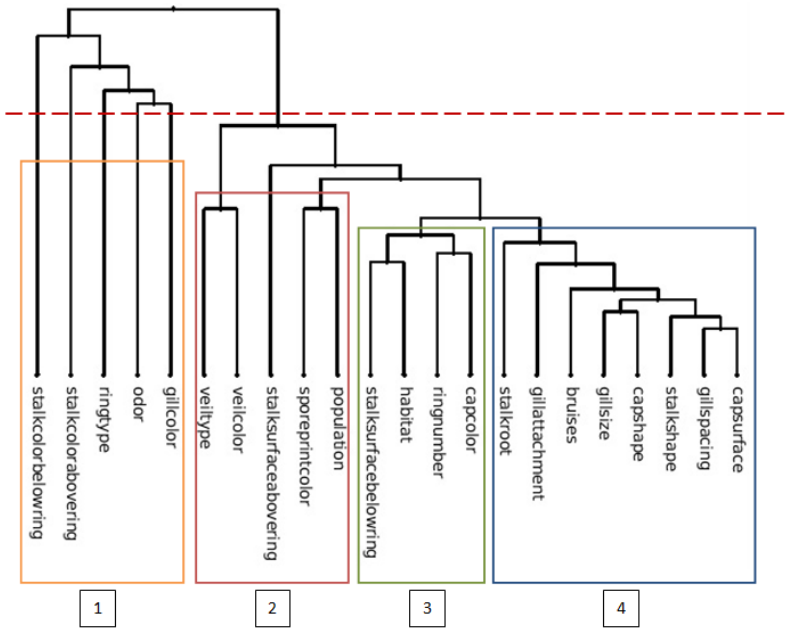


Figura 3.4: Jerarquia de similitud de atributos del conjunto de datos de *mushroom*

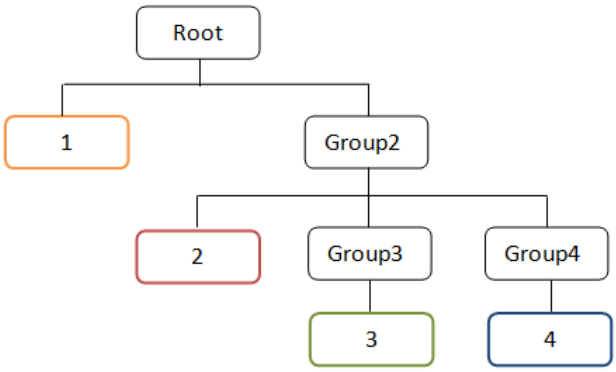


Figura 3.5: Jerarquia simplificada basada en la figura 3.4

A partir de esto, es colocada cada variable en un grupo que puede ser representado como un término funcional:

```
Mushroom(WHITE,WHITE, PENDAT, AMOND, WHITE,
   Group2(PARTIAL, WHITE, SMOOTH, BROWN, SEVERAL,
      Group3(SMOOTH, WOODS, ONE, WHITE),
      Group4(TAPERING, FREE, CROWDED, NARROW, BRUISES, BULBOUS, CONVES, SMOOTH)))
```

3.3 Derivación de esquemas XML jerárquicos a partir de datos jerárquicos

En este punto se trata de la transformación necesaria cuando los datos ya tienen originalmente una rica estructura jerárquica. Este tipo de derivaciones son más directas; consisten en transformaciones simples que deben garantizar una estructura bien formada para el documento XML; es necesario establecer si el orden, repeticiones y etiquetas son pertinentes para determinar las características y la ubicación correcta en la estructura XML. Un aspecto específico, como se mencionó anteriormente, es la forma de tratar las partes vacías de la jerarquía y la forma de compararlas con los términos que no estén vacíos. Este enfoque se basa en un esquema común para todos los ejemplos y la suposición de que la distancia entre una parte vacía y una no vacía es 1 (no está dado por el tamaño de una parte vacía).

La figura 3.6 muestra la estructura del ejemplo 220 del conjunto de datos 'sponge' del UCI repository [16]. La figura 3.7 muestra este mismo objeto representado como un término en LISP y su represntaciíon en un documento XML. Este es un conjunto de datos complejo que cuenta con una rica estructura, donde es posible resaltar que los métodos proposicionales clásicos no son aplicables.

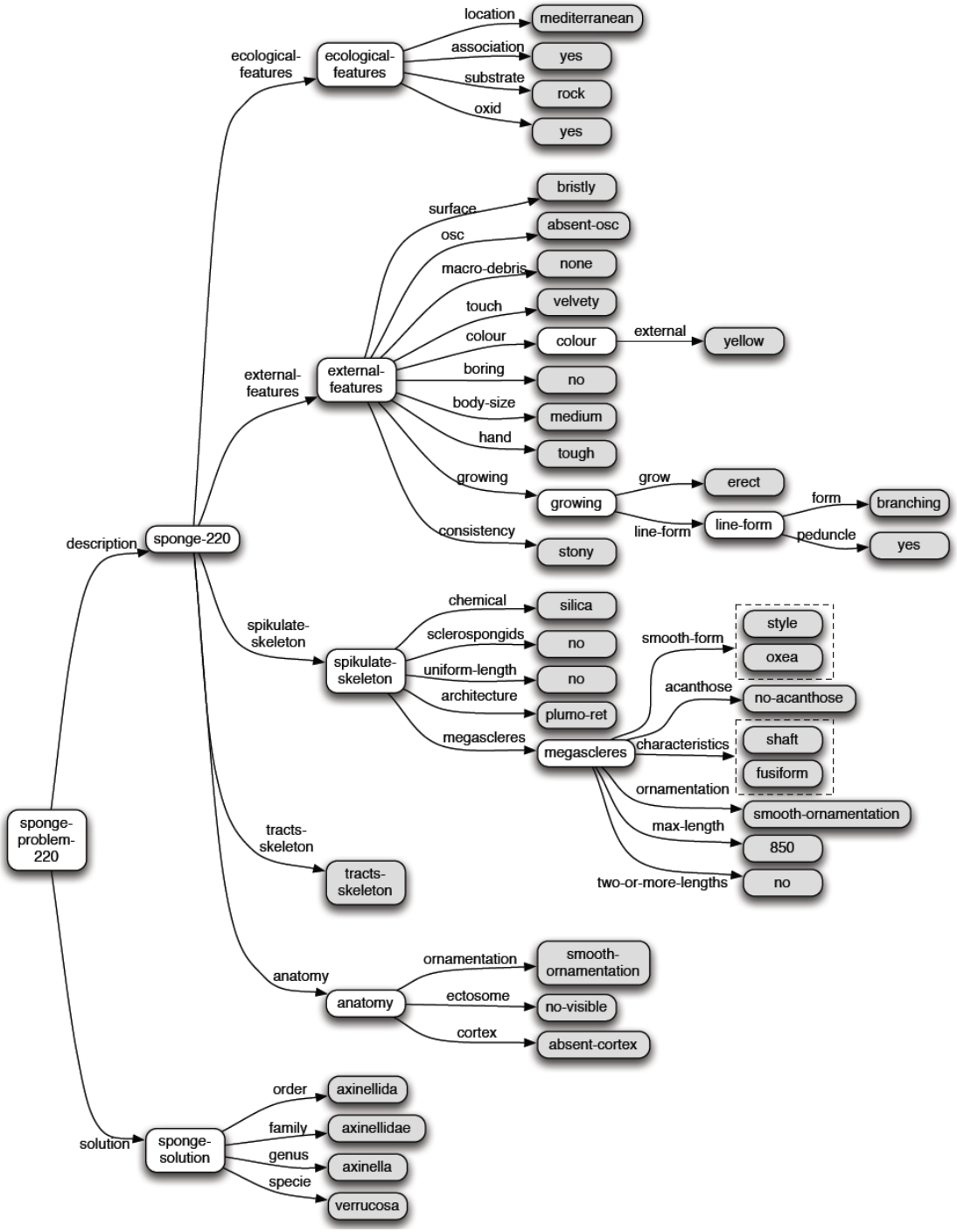


Figura 3.6: Jerarquía compleja (ejemplo 220) del conjunto de datos de sponge

```
(define-episode (sponge :ID SPONGE-220)
        (EXTERNAL-FEATURES
         (DEFINE (EXTERNAL-FEATURES)
                 (BODY-SIZE MEDIUM)
                 (CONSISTENCY STONY)
                 (HAND TOUGH)
                 (TOUCH VELVETY)
                 (SURFACE BRISTLY)
                 (GROWING
                  (DEFINE (GROWING)
                          (GROW ERECT)
                          (LINE-FORM
                           (DEFINE (LINE-FORM)
                                   (FORM BRANCHING)
                                   (PEDUNCLE YES)))))
                 (BORING NO)
                 (COLOUR (DEFINE (COLOUR) (EXTERNAL YELLOW)))
                 (CRIBLE NO)
                 (HOLLOW NO)
                 (BRUSH NO)
                 (BRIOZOA NO)
                 (MACRO-DEBRIS NONE)
                 (OSC (DEFINE (ABSENT-OSC)))
                 (GEMMULES NO)))
        (ECOLOGICAL-FEATURES
         (DEFINE (ECOLOGICAL-FEATURES)
                 (SUBSTRATE ROCK)
                 (OXID YES)
                 (LOCATION MEDITERRANEAN)
                 (ASSOCIATION YES)))
        (SPICULATE-SKELETON
         (DEFINE (SPICULATE-SKELETON)
                         (CHEMICAL SILICA)
                         (ARCHITECTURE PLUMO-RET)
                         (MEGASCLERES
                          (DEFINE (MEGASCLERES)
                                  (SMOOTH-FORM (DEFINE (SET) STYLE OXEA))
                                  (ACANTHOSE NO-ACANTHOSE)
                                  (CHARACTERISTICS
                                   (DEFINE (CHARACTERS) (SHAFT FUSIFORM)))
                                  (ORNAMENTATION SMOOTH-ORNAMENTATION)
                                  (MAX-LENGTH 850)
                                  (TWO-OR-MORE-LENGTH NO)))
                         (SCLEROSPONGIDS NO)
                         (UNIFORM-LENGTH NO)))
        ; (FIBROUS-SKELETON (DEFINE (NO-FIBROUS-SKELETON)))
        (TRACTS-SKELETON (DEFINE (TRACTS-SKELETON)))
        (ANATOMY
         (DEFINE (ANATOMIC-FEATURES)
                 (ORNAMENTATION SMOOTH-ORNAMENTATION)
                 (ECTOSOME NO-VISIBLE)
                 (CORTEX (DEFINE (ABSENT-CORTEX))))))
```

(a) Jerarquía en LISP

```
<SPONGES>
  <DEFINE-EPISODE>
    <EXTERNAL-FEATURES>
      <DEF>EXTERNAL-FEATURES</DEF>
      <BODY-SIZE>MEDIUM</BODY-SIZE>
      <CONSISTENCY>STONY</CONSISTENCY>
      <HAND>TOUGH</HAND>
      <TOUCH>VELVETY</TOUCH>
      <SURFACE>BRISTLY</SURFACE>
      <GROWING>
        <DEF>GROWING</DEF>
        <GROW>ERECT</GROW>
        <LINE-FORM>
          <DEF>LINE-FORM</DEF>
          <FORM>BRANCHING</FORM>
          <PEDUNCLE>YES</PEDUNCLE>
        </LINE-FORM>
      </GROWING>
      <BORING>NO</BORING>
      <COLOUR>
        <DEF>COLOUR</DEF>
        <EXTERNAL>YELLOW</EXTERNAL>
      </COLOUR>
      <CRIBLE>NO</CRIBLE>
      <HOLLOW>NO</HOLLOW>
      <BRUSH>NO</BRUSH>
      <BRIOZOA>NO</BRIOZOA>
      <MACRO-DEBRIS>NONE</MACRO-DEBRIS>
      <OSC>
        <DEF>ABSENT-OSC</DEF>
      </OSC>
      <GEMMULES>NO</GEMMULES>
    </EXTERNAL-FEATURES>
    <ECOLOGICAL-FEATURES>...</ECOLOGICAL-FEATURES>
    <SPICULATE-SKELETON>...</SPICULATE-SKELETON>
    <TRACTS-SKELETON>...</TRACTS-SKELETON>
    <ANATOMY>...</ANATOMY>
    <ORDER>AXINELLIDA</ORDER>
  </DEFINE-EPISODE>
</SPONGES>
```

(b) Jerarquía representada como un documento XML

Figura 3.7: Término en LISP y su representación en XML del conjunto de datos de sponge

4 Experimentos

En este capítulo se incluyen experimentos sobre un proceso de clasificación usando algunas distancias entre términos introducidas en la capítulo 2, con las transformaciones presentadas en la capítulo previo (capítulo 3).

Para el proceso de clasificación se utiliza el algoritmo k-NN con la variante de ponderación, usando la función de *atracción*. k-NN es un algoritmo simple y adecuado para analizar el efecto de las distancias y las transformaciones. Como se mencionó en la sección 2.3, este algoritmo clasifica una instancia en la clase más común de los k ejemplos más cercanos usando la distancia. El valor de k para los siguientes experimentos es calculado como $\sqrt{n}$ (donde n es el número de ejemplos), que es una manera comúnmente utilizada para definir el valor de k en este algoritmo. La función de *atracción*, definida como $\frac{1}{d^i}$ (donde d es la distancia e i es el *parámetro de atracción*), da mayor ponderación a los ejemplos más cercanos de k (en k-NN no ponderado, i es igual a 0); en otras palabras, mientras mayor sea i, menor importancia tiene k. En los experimentos se usan varios valores para i que varían de 0 a 3.

Se consideraron tres distancias: la distancia de Nienhuys-Cheng, la distancia de Estruch et al. y la distancia euclídea; las cuales son denotadas por d_N, d_E y d_U, respectivamente.

Para la evaluación experimental, se utilizaron tres conjuntos de datos del UCI *repository* [16]; dos de ellos, *mushroom* y *soybean*, son conjuntos de datos planos que son usados de diferentes maneras: inicialmente con su estructura plana y, porteriormente, jerarquizados utilizando dos métodos diferentes. El tercer conjunto de datos, *Demospongiae* (*sponge*), es un documento LISP el cual fue transformado en un documento XML, preservando la jerarquía y los valores entre los atributos.

Las tablas de resultados, para los conjuntos de datos de *mushroom* y *soybean*, incluyen los valores medios de los experimentos utilizando validación cruzada con 10 pliegues. Para la prueba de significancia entre los resultados de los métodos, se utiliza *t-Student* con una confianza de 95 %; si la diferencia de un método con respecto a otro es significativo, se incluyen su acrónimo (d_N, d_E, d_U) en la celda. En el conjunto de datos de *sponge* se utiliza 60 % de ejemplos de entrenamiento y 40 % para ejemplos de prueba.

4.1 Conjunto de datos *Mushroom*

Este conjunto de datos incluye descripciones de muestras hipotéticas correspondientes a 23 especies de setas; cada especie es identificada como comestible o venenosa. Este conjunto de datos cuenta con 8416 ejemplos, de los cuales se extrajo una muestra aleatoria de 1000 de ellos para el proceso de clasificación.

Primero se comparan las tres distancias usando un esquema plano (sin usar ningún tipo de jerarquía) y con diferentes valores para el parámetro i. Los resultados, presentados en la tabla 4.1 y la figura 4.1, muestran que las diferencias entre las tres distancias no son significativas.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	93.0	94.9	95.6	95.8
Distancia de Estruch et al. d_E	93.0	94.9	95.6	95.8
Distancia euclídea d_U	93.0	94.4	94.9	95.5

Tabla 4.1: Porcentajes de aciertos para el conjunto de datos de *mushroom* sin jerarquías.

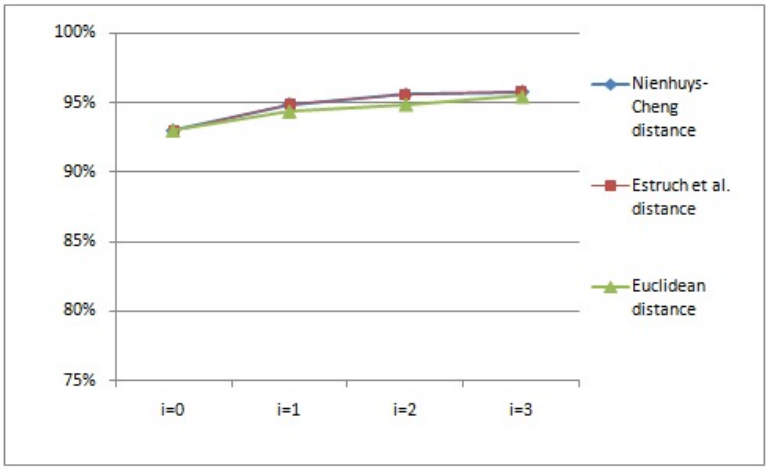


Figura 4.1: Porcentajes de aciertos para el conjunto de datos de *mushroom* sin jerarquías.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	99.8	99.8	99.9	99.9
Distancia de Estruch et al. d_E	99.8	99.8	99.8	99.8
Distancia euclídea d_U	93.0	94.4	94.9	95.5

Tabla 4.2: Porcentajes de aciertos para el conjunto de datos de *mushroom* con una jerarquía derivada del nombre de los atributos.

La tabla 4.2 y la figura 4.2 muestra los resultados usando una jerarquía inducida por los nombres de los atributos. Con esta estructura, es posible encontrar diferencias entre las distancias; el desempeño de las distancias de Estruch et al. y Nienhuys-Cheng es óptimo, mientras que los resultados de distancia euclídea no se ven afectados por la estructura y, al igual que la ejecución anterior, solo se incrementa el porcentaje de aciertos cuando el valor de i también es incrementado.

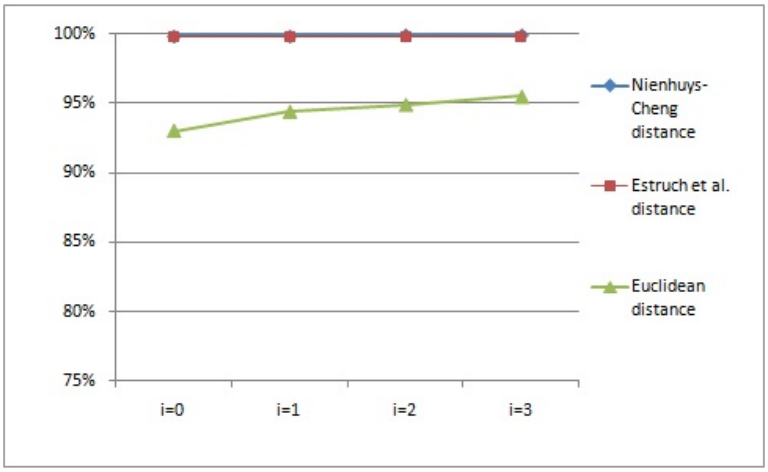


Figura 4.2: Porcentajes de aciertos para el conjunto de datos de *mushroom* con una jerarquía derivada del nombre de los atributos.

Finalmente, usando el otro método de agrupación, que considera la similitud entre atrubutos, basado en la métrica de Chi-cuadrado (ChiSquaredAttributeEval + Ranker en Weka [19]) y los grupos mostrados en la figura 3.4, se obtienen los resultados presentados en la tabla 4.3 y la figura 4.3. En estos resultados, las distancias entre términos aprovechan esta estructura y obtiene nuevamente mejores resultados que la distancia euclídea.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	99.5	99.8	100	100
Distancia de Estruch et al. d_E	99.5	100	100	100
Distancia euclídea d_U	93.0	94.4	94.9	95.5

Tabla 4.3: Porcentajes de aciertos para el conjunto de datos de *mushroom* de acuerdo con la similitud entre atributos.

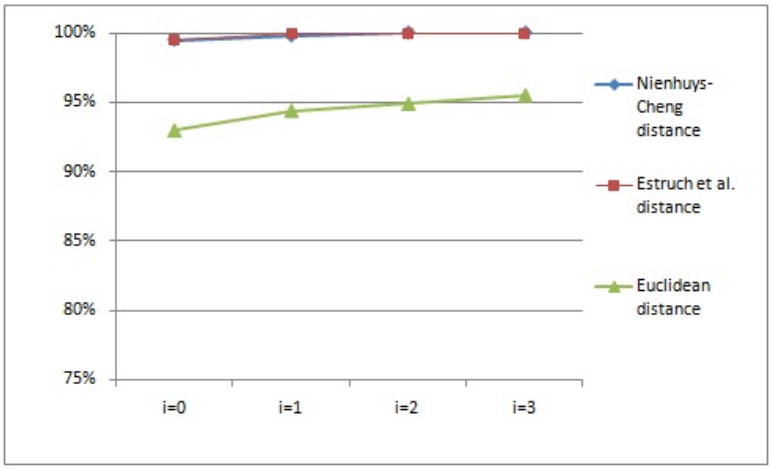


Figura 4.3: Porcentajes de aciertos para el conjunto de datos de *mushroom* de acuerdo con la similitud entre atributos.

4.2 Conjunto de datos *Soybean*

El conjunto de datos *soybean*, al igual que *mushroom*, es un conjunto de datos plano, compuesto de 307 instancias y cada instancia tiene 35 atributos que determinan 19 clases diferentes de enfermedades de la soja. Las ejecuciones realizadas con este conjunto de datos son iguales a las realizadas para *mushroom*: 1) sin utilizar ningún tipo de jerarquía, 2) creando una jerarquía a partir de los nombres y valores de los atributos y 3) creando una jerarquía de acuerdo con la similitud de los atributos.

La tabla 4.4 y la figura 4.4 muestra los resultados de las ejecuciones con las tres distancias usando la versión plana del conjunto de datos. En este caso, cuando $i = 0$ las tres distancias son iguales; sin embargo, al incrementar i las distancias entre términos superan un poco la distancia euclídea, aunque sus diferencias no son estadísticamente significativas.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	75.3	91.3	93.9	95.8
Distancia de Estruch et al. d_E	75.3	91.3	93.9	95.8
Distancia euclídea d_U	75.3	87.1	91.3	92.6

Tabla 4.4: Porcentajes de aciertos para el conjunto de datos de *soybean* sin jerarquías.

La tabla 4.5 y la figura 4.5 muestran el desempeño utilizando una jerarquía construida a partir de los nombres y valores de los atributos; el comportamiento de las tres distancias es similar y las diferencias no son significativas.

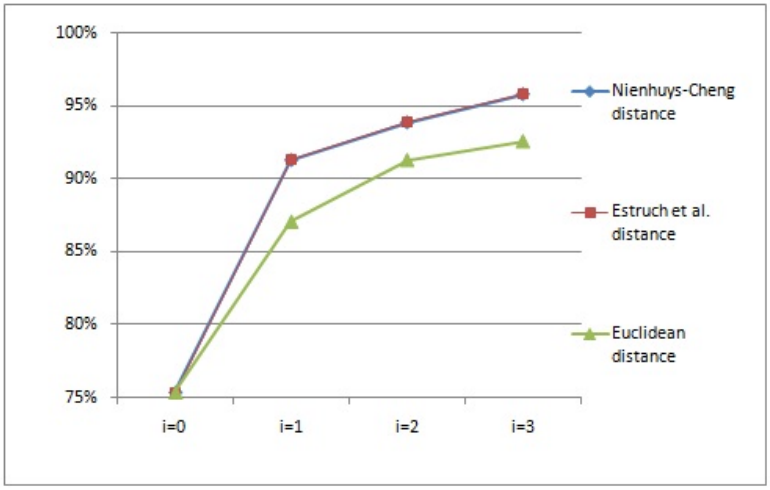


Figura 4.4: Porcentajes de aciertos para el conjunto de datos de *soybean* sin jerarquías.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	78.6	89.6	92.6	93.2
Distancia de Estruch et al. d_E	76.0	88.0	91.6	93.5
Distancia euclídea d_U	75.3	87.1	91.3	92.6

Tabla 4.5: Porcentajes de aciertos para el conjunto de datos de *soybean* con una jerarquía derivada del nombre y valor de los atributos.

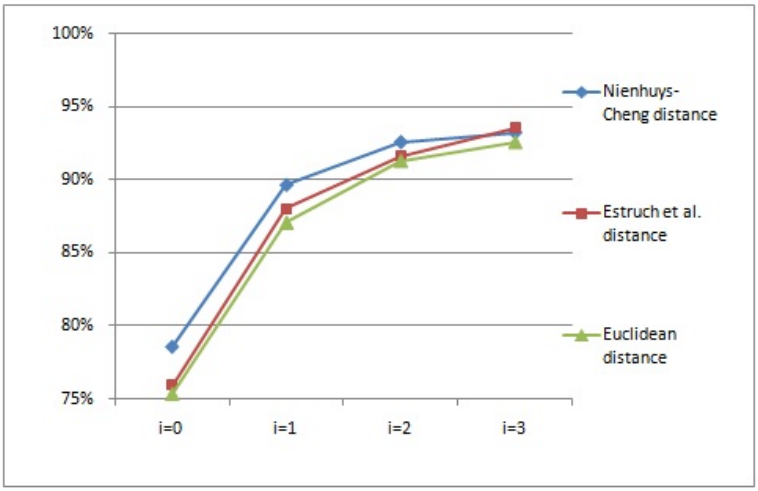


Figura 4.5: Porcentajes de aciertos para el conjunto de datos de *soybean* con una jerarquía derivada del nombre y valor de los atributos.

	$i = 0$ (%)	$i = 1$ (%)	$i = 2$ (%)	$i = 3$ (%)
Distancia de Nienhuys-Cheng d_N	86.1 d_U	99.0 d_U	99.4 d_U	99.4 d_U
Distancia de Estruch et al. d_E	86.1 d_U	99.0 d_U	99.4 d_U	99.4 d_U
Distancia euclídea d_U	75.3 $d_N d_E$	87.1 $d_N d_E$	91.3 $d_N d_E$	92.6 $d_N d_E$

Tabla 4.6: Porcentajes de aciertos para el conjunto de datos de *soybean* de acuerdo con la similitud entre atributos.

Finalmente, para este conjunto de datos, utilizando la métrica Chi-cuadrado para identificar la similitud entre atributos, se obtienen los resultados de la tabla 4.6 y la figura 4.6. Para esta situación, las diferencias son estadísticamente significativas; las distancias entre términos obtienen mejores resultados que la distancia euclídea, que siempre es igual y no es afectada por la estructura del conjunto de datos.

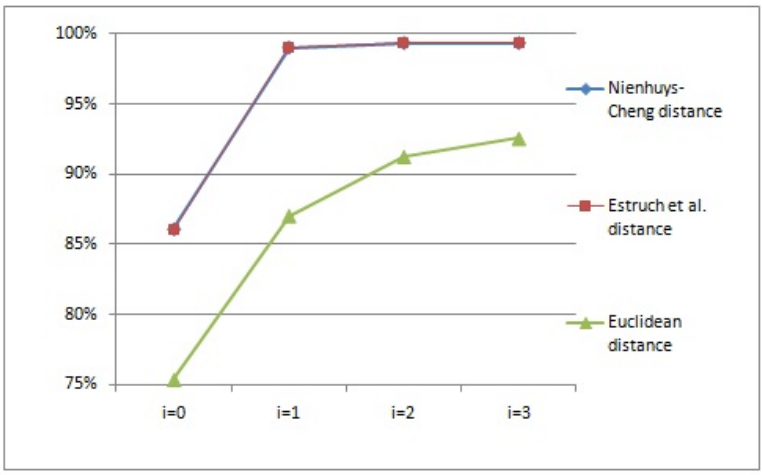


Figura 4.6: Porcentajes de aciertos para el conjunto de datos de *soybean* de acuerdo con la similitud entre atributos.

Utilizando la similitud entre atributos y las distancias entre términos se puede mejorar significativamente el desempeño de un proceso de clasificación aplicado sobre conjuntos de datos planos.

4.3 Conjunto de datos *Demospongiae*

El conjunto de datos *Demospongiae* es una estructura originalmente jerárquica con 503 instantacias de esponjas marinas; cada instancia es representada como un árbol usando términos en el lenguaje LISP. Cada árbol tiene entre 5 y 8 niveles de profundidad y el número de hojas varía entre 17 y 51 (ver la figura 3.6).

De este conjunto de datos, se extrae un documento XML bien formado que preserva la

estructura original. Para hacer esta transformación, cada línea en LISP es convertida en uno o varios elementos XML bien formados, por medio de la asignación de nombres y valores de cada elemento de cada una de las características descritas en LISP, acuerdo con su jerarquía y orden. Algunas transformaciones fueron definidas de la siguiente manera:

- Cada ejemplo en LISP, definido por la etiqueta `'define-episode'` y un identificador `'sponge :ID SPONGE-0'` fue transformado por un elemento XML: <DEFINE-EPISODE>, el cual adiciona un elemento hijo: <SPONGE_ID>SPONGE-0</SPONGE_ID>.
- Cada característica definida como `'EXTERNAL-FEATURES (DEFINE (EXTERNAL-FEATURES)'` es convertida en un elemento principal contenedor de varios elementos XML: <EXTERNAL-FEATURES>.
- Cada caracteristica simple como `'(BODY-SIZE SMALL)'` es convertida en un elemento <BODY-SIZE>SMALL</BODY-SIZE>.
- Adicionalmente, las coleciones como `'(SET) GREY WHITISH'` , es convertida en XML de la siguiente manera: <SET1>GREY</SET1> <SET2>WHITISH</SET2>.

La tabla 4.7 y la figura 4.7 muestran los resultados de la clasificación utilizando el documento XML generado a partir de las transformaciones anteriores. Con este conjunto de datos solo se aplican las distancias entre términos porque no es posible aplicar directamente la distancia euclídea; la distancia de Estruch et al. muestra mejores resultados que la distancia de Nienhuys-Cheng (cuando i es mayor que 0). Este conjunto de datos es útil para tener en cuenta las ventajas de las propiedades intrínsecas de las distancias entre términos; es un buen ejemplo para ilustrar las diferencias entre este tipo de distancias debido a que los datos son puramente semi-estructurados y se percibe el efecto de las repeticiones.

	$i=0$ (%)	$i=1$ (%)	$i=2$ (%)	$i=3$ (%)
Distancia de Nienhuys-Cheng d_N	60.9	71.3	74.3	73.8
Distancia de Estruch et al. d_E	60.9	73.3	78.7	77.7

Tabla 4.7: Porcentajes de aciertos para el conjunto de datos de *sponge* usando su jerarquía original.

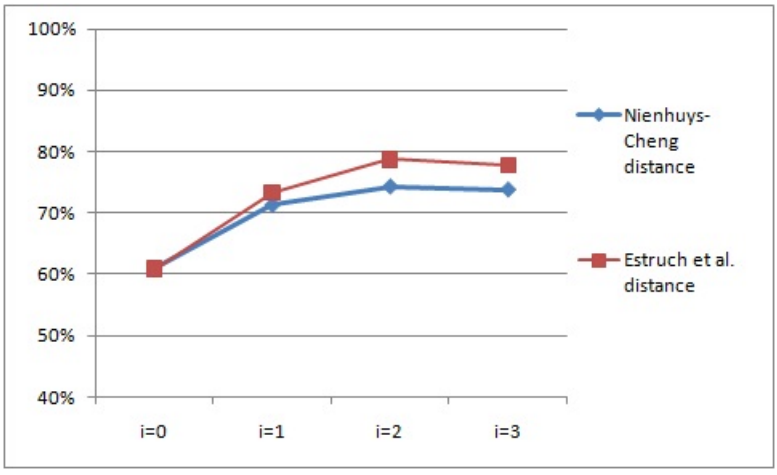


Figura 4.7: Porcentajes de aciertos para el conjunto de datos de *sponge* usando su jerarquía original.

5 Conclusiones y trabajos futuros

Las distancias entre términos, aunque inicialmente definidas en el área de programación inductiva, pueden ser usadas en una gama más amplia de aplicaciones. En este trabajo, por ejemplo, se utilizan diferentes tipos de transformaciones que buscan adaptar datos planos en diferentes grados de estructuras, aplicando las distancias entre términos. De igual manera, se utiliza la distancia entre términos sobre un conjunto de datos originalmente jerárquicos. Todas estas transformaciones son aplicadas sobre estructuras XML; de esta manera, cualquier conjunto de datos, plano o con algún nivel de jerarquía, puede ser transformado y, por medio de un proceso de clasificación (basado en distancias), utilizar las distancias entre términos.

5.1 Discusión

Con los resultados obtenidos en los experimentos realizados utilizando las diferentes transformaciones y las distancias entre términos (y comparándolas con la distancia euclídea, para el caso de conjuntos de datos planos), se puede identificar los siguientes tres aspectos:

- Los métodos para la construcción de jerarquías a partir de los nombres y los valores de atributos no siempre parecen dar buenos resultados. Sin embargo, estas asociaciones intuitivas entre atributos pueden modificar los resultados del proceso de clasificación y, en algunas ocasiones, de una manera sencilla y utilizando distancias entre términos, mejora los resultados obtenidos en la aplicación directa de este proceso sobre datos planos.
- El proceso basado en la similitud de atributos se caracteriza por agrupar las variables más parecidas entre sí y colocar más cerca de la raíz del árbol aquellos grupos que están alta-

mente relacionados; su principal beneficio se debe a la ponderación implícita que asumen las distancias entre términos donde, comúnmente, ofrecen mayor relevancia a los atributos que están cerca de la raíz del árbol. Sin embargo, la creación de estructuras altamente jerarquizadas puede generar que atributos con relevancia media y baja no sean considerados adecuadamente, debido a que su ponderación se hace muy pequeña; esta razón lleva a la construcción de una estructura con menos niveles de jerarquía. El uso de este proceso, basado en la similitud entre atributos para construir un dendrograma y a partir de este construir una jerarquía, mejora los resultados obtenidos sobre la aplicación de datos planos.

- Cuando los datos son originalmente jerárquicos no es posible aplicar directamente las distancias definidas para datos proposicionales y es relevante considerar las ventajas ofrecidas por las propiedades de distancias entre términos con el fin de mejorar los resultados de la clasificación.

Este trabajo ha mostrado una aplicación prometedora de las distancias entre términos para diferentes tipos de conjuntos de datos, lo que sugiere que su uso puede ser más amplio de lo que es ahora. Aunque existen distancias entre árboles que permiten la comparación de estructuras jerárquicas, como por ejemplo la distancia de edición, estas se basan principalmente en costos de operaciones de inserción y eliminación nodos y aristas, que dependen del número de transformaciones para convertir un árbol en otro; lo cual es un concepto que, en muchas ocasiones, pierde el sentido de comparación jerárquica, puesto que no considera el nivel de profundidad de las variables dentro de este tipo de estructuras.

5.2 Trabajos futuros

Existen algunos de aspectos que sería importante explorar y profundizar en trabajos futuros y que, seguramente, permitirían mejorar los procesos de aprendizaje por medio de la aplicación de distancias entre términos. A continuación se describen algunos de estos aspectos:

- **Manejo de conjuntos:** este trabajo se centra principalmente en la aplicación distancias sobre estructuras jerárquicas que son representadas por medio de XML. Sin embargo, el manejo de conjuntos, como tipos de datos anidados dentro de este tipo de estructuras jerárquicas, es un aspecto relevante para los procesos de clasificación. La utilización de

métricas sobre este tipo de dato, como por ejemplo distancias que tienen en cuenta la diferencia entre los elementos del conjunto, mejoraría los procesos de clasificación.

- **Manejo de valores desconocidos de las variables y ausencia de variables:** el desconocimiento del valor de una variable y la ausencia de ésta son aspectos que deben ser tratados de diferente manera; por ejemplo, intuitivamente la ausencia de una misma característica, entre dos objetos diferentes, representa mayor grado de similitud que el caso en que uno de estos objetos contiene la característica y el otro no; por otra parte, el desconocimiento del valor de una misma variable en dos instancias tiene una alta probabilidad de que efectivamente sean diferentes (esta probabilidad se incrementa, mientras mayor sea el dominio de los posibles valores de la variable en comparación); igualmente, pueden generarse múltiples combinaciones entre estos dos aspectos que seguramente, utilizando ponderaciones adecuadas, ayudarían a mejorar los procesos de aprendizaje por medio de la utilización de las distancias entre términos.
- **Distancias entre términos para agrupamiento:** adicionalmente, un trabajo futuro es la aplicación de distancias entre términos para tareas aprendizaje no supervisado; particularmente, tareas de agrupamiento basado en distancias.

5.3 Publicaciones

Fundamentado en el análisis y los resultados obtenidos de este proyecto de investigación, se desarrolló un artículo titulado *"Applying distances between terms to both flat and hierarchical data"* [2] en conjunto con los profesores José Hernández Orallo, César Ferri y María José Ramírez Quintana, miembros del Grupo DMIP (*Data Mining and Inductive Programming*). Este artículo fue aceptado por el comité del *4th International Workshop on Approaches and Applications of Inductive Programming*, presentado el 19 de julio de 2011 en Odense-Dinamarca (publicado en http://www.cogsys.wiai.uni-bamberg.de/aaip11/AAIP2011.pdf).

Bibliografía

[1] A. Bargiela and W. Pedrycz. *Granular computing: an introduction*. Springer, 2003.

[2] J.A. Bedoya-Puerta, C. Ferri, J. Hernández-Orallo, and MJ Ramırez-Quintana. Applying distances between terms to both flat and hierarchical data. In *Proceedings of AAIP 2011 4th International Workshop on Approaches and Applications of Inductive Programming*, page 1, 2011.

[3] H. Blockeel, L. De Raedt, and J. Ramon. Top-down induction of clustering trees. In *Proc. of the 15th International Conference on Machine Learning (ICML'98)*, pages 55–63. Morgan Kaufmann, 1998.

[4] H. Bunke. On a relation between graph edit distance and maximum common subgraph. *Pattern Recognition Letters*, 18(9):689–694, 1997.

[5] J. Cheney. Flux: functional updates for xml. In *Proc. 13th ACM SIGPLAN Intl. Conf. on Functional programming, ICFP*, pages 3–14. ACM, 2008.

[6] N. Cristianini and B. Scholkopf. Support vector machines and kernel methods: the new generation of learning machines. *Ai Magazine*, 23(3):31, 2002.

[7] T.G. Dietterich. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting and randomisation. *Machine Learning*, 40(2):130–157, 2000.

[8] V Estruch. Bridging the gap between distance and generalisation: Symbolic learning in metric spaces. *PhD thesis disseration, DSIC, Universitat Politecnica de Valencia, 2009.*

[9] V. Estruch, C. Ferri, J. Hernández-Orallo, and M.J. Ramırez-Quintana. A new context-sensitive and composable distance for first-order terms. technical report.

[10] V. Estruch, C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. An Integrated Distance for Atoms. *Functional and Logic Programming*, pages 150–164, 2010.

[11] V. Estruch, C. Ferri, J.Hernández-Orallo, and M.J. Ramírez-Quintana. Similarity functions for structured data. An application to decision trees. *Revista Iberoamericana de Inteligencia Artificial*, 10(29):109–121, 2006.

[12] C. Ferri. Multi-paradigm learning of declarative models. *PhD thesis disseration, DSIC, Universitat Politecnica de Valencia, 2003.*

[13] C. Ferri, J. Hernández-Orallo, and M.J. Ramírez-Quintana. Incremental learning of functional logic programs. In Herbert Kuchen and Kazunori Ueda, editors, *FLOPS*, volume 2024 of *Lecture Notes in Computer Science*, pages 233–247. Springer, 2001.

[14] P. Flener and D. Partridge. Inductive programming. *Automated Software Engineering*, 8(2):131–137, 2001.

[15] F. De Francesca, G. Gordano, R. Ortale, and A. Tagarelli. Distance-based clustering of XML documents. In *1st Intl. Workshop on Mining Graphs, Trees and Sequences (MGTS-2003)*, pages 75–78. L. De Raedt and T. Washio, editors, 2003.

[16] A. Frank and A. Asuncion. UCI machine learning repository, 2010.

[17] S. Garcia-Vallvé and Pere Puigbo. UPGMA dendroupgma: A dendrogram construction utility, 2009.

[18] R. Goldman, J. McHugh, and J. Widom. From semistructured data to xml: Migrating the lore data model and query language. In *Proceedings of the 2nd International Workshop on the Web and Databases (WebDB 99)*, pages 25–30. Citeseer, 1999.

[19] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I.H. Witten. The WEKA data mining software: an update. *ACM SIGKDD Explorations Newsletter*, 11(1):10–18, 2009.

[20] J. Hernández-Orallo and M.J. Ramírez-Quintana. Inverse narrowing for the induction of functional logic programs. In José Luis Freire-Nistal, Moreno Falaschi, and Manuel Vilares Ferro, editors, *APPIA-GULP-PRODE*, pages 379–392, 1998.

[21] J. Hernández-Orallo and M.J. Ramírez-Quintana. A strong complete schema for inductive functional logic programming. In Saso Dzeroski and Peter A. Flach, editors, *ILP*, volume 1634 of *Lecture Notes in Computer Science*, pages 116–127. Springer, 1999.

[22] J. Hernández Orallo, M.J. Ramírez Quintana, and C. Ferri Ramírez. Introducción a la minería de datos. *Editorial Pearson Educación SA, Madrid*, 2004.

[23] M. Hofmann, E. Kitzelmann, and U. Schmid. A unifying framework for analysis and evaluation of inductive programming systems. In *Second Conference on Artificial General Intelligence*, pages 55–60.

[24] A. Kraskov, H. Stogbauer, R.G. Andrzejak, and P. Grassberger. Hierarchical clustering based on mutual information. *Arxiv preprint q-bio/0311039*, 2003.

[25] N. Lachiche and P.A. Flach. A first-order representation for knowledge discovery and bayesian classification on relational data. In *Data Mining, decision Support, Meta-learning and ILP: Forum for Practical Problem Presentation and Prospective Solutions (DDMI-2000), Workshop of 4th International Conference on Principles of Data Mining and Knowledge Discovery (PKDD-2000)*, pages 49–60. Citeseer.

[26] T. M. Mitchell. *Machine Learning*. McGraw-Hill, 1997.

[27] S. Muggleton. Inductive Logic Programming. *New Generation Computing*, 8(4):295–318, 1991.

[28] S.H. Nienhuys-Cheng and R. de Wolf. *Foundations of Inductive Logic Programming*, volume 1228 of *Lecture Notes in Artificial Intelligence*. Springer, 1997.

[29] D. Partridge. The case for inductive programming. *Computer*, 30(1):36–41, 1997.

[30] G.D. Plotkin and Aarhus universitet. Datalogisk afdeling. *A structural approach to operational semantics*. Citeseer, 1981.

[31] J. Ramon and M. Bruynooghe. A framework for defining distances between first-order logic objects. In *Proc. of the 8th Int. Conference on Inductive Logic Programming (ILP'98)*, pages 271–280. Springer, 1998.

[32] S. Watanabe. *Knowing and guessing: A quantitative study of inference and information*. Wiley New York, 1969.

[33] G. Xing, Z. Xia, and J. Guo. Clustering xml documents based on structural similarity. In *Advances in Databases: Concepts, Systems and Applications*, volume 4443 of *LNCS*, pages 905–911. Springer, 2007.

[34] J. Y. Cai Y. Wang, D. J. Dewitt. X-diff: an effective change detection algorithm for xml documents. In *19th Intl. Conf.on Data Engineering*, pages 519–530, 2003.

Apéndices

A Detalle de los resultados

Este apéndice detalla los resultados presentados en el capítulo 4 para los conjuntos de datos planos (*mushroom* y *soybean*) y originalmente jerárquicos (*demospongiae*), utilizando el algoritmo de clasificación k-NN con diferentes valores para el *parámetro de atracción* i, aplicando las distancias entre términos y la distancia euclídea. Para los conjuntos de datos planos se utiliza validación cruzada con 10 pliegues; para el conjunto de datos originalmente jerárquico se utiliza 60 % de ejemplos de entrenamiento y 40 % para ejemplos de prueba.

A.1 Conjunto de datos *Mushroom*

Los resultados para este conjunto de datos son calculados utilizando la distancia de Nienhuys-Cheng, la distancia de Estruch et al. y la distancia euclídea sobre los tres tipos de estructuras: 1) plana (sin jerarquías), 2) jerarquía inducida por nombres y valores de los atributos y 3) jerarquía de acuerdo con la similitud de los atributos.

Estructura plana (sin jerarquías):

Distancia de Nienhuys-Cheng									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	99	99.00
1	100	58	58.00	60	60.00	60	60.00	62	62.00
2	100	94	94.00	96	96.00	99	99.00	100	100.00
3	100	100	100.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	93	93.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	87	87.00	95	95.00	100	100.00	100	100.00
8	100	98	98.00	98	98.00	98	98.00	98	98.00
9	100	100	100.00	100	100.00	99	99.00	99	99.00
Total/prom.	1000	930	93.00	949	94.90	956	95.60	958	95.80

Tabla A.1: Aciertos para el conjunto de datos *mushroom* sin jerarquías utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	99	99.00
1	100	58	58.00	60	60.00	60	60.00	62	62.00
2	100	94	94.00	96	96.00	99	99.00	100	100.00
3	100	100	100.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	93	93.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	87	87.00	95	95.00	100	100.00	100	100.00
8	100	98	98.00	98	98.00	98	98.00	98	98.00
9	100	100	100.00	100	100.00	99	99.00	99	99.00
Total/prom.	1000	930	93.00	949	94.90	956	95.60	958	95.80

Tabla A.2: Aciertos para el conjunto de datos *mushroom* sin jerarquías utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	58	58.00	60	60.00	60	60.00	60	60.00
2	100	94	94.00	94	94.00	96	96.00	98	98.00
3	100	100	100.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	93	93.00	98	98.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	87	87.00	94	94.00	95	95.00	99	99.00
8	100	98	98.00	98	98.00	98	98.00	98	98.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	930	93.00	944	94.40	949	94.90	955	95.50

Tabla A.3: Aciertos para el conjunto de datos *mushroom* sin jerarquías utilizando la distancia euclídea

Jerarquía inducida por nombres y valores de los atributos:

Distancia de Nienhuys-Cheng									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	100	100.00	100	100.00	100	100.00	100	100.00
2	100	99	99.00	99	99.00	100	100.00	100	100.00
3	100	99	99.00	99	99.00	99	99.00	99	99.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	100	100.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	100	100.00	100	100.00	100	100.00	100	100.00
8	100	100	100.00	100	100.00	100	100.00	100	100.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	998	99.80	998	99.80	999	99.90	999	99.90

Tabla A.4: Aciertos para el conjunto de datos *mushroom* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	100	100.00	100	100.00	100	100.00	100	100.00
2	100	99	99.00	99	99.00	100	100.00	100	100.00
3	100	99	99.00	99	99.00	98	98.00	98	98.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	100	100.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	100	100.00	100	100.00	100	100.00	100	100.00
8	100	100	100.00	100	100.00	100	100.00	100	100.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	998	99.80	998	99.80	998	99.80	998	99.80

Tabla A.5: Aciertos para el conjunto de datos *mushroom* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	58	58.00	60	60.00	60	60.00	60	60.00
2	100	94	94.00	94	94.00	96	96.00	98	98.00
3	100	100	100.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	93	93.00	98	98.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	87	87.00	94	94.00	95	95.00	99	99.00
8	100	98	98.00	98	98.00	98	98.00	98	98.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	930	93.00	944	94.40	949	94.90	955	95.50

Tabla A.6: Aciertos para el conjunto de datos *mushroom* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia euclídea

Jerarquía de similitud de los atributos:

Distancia de Nienhuys-Cheng									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	99	99.00	100	100.00	100	100.00	100	100.00
2	100	98	98.00	98	98.00	100	100.00	100	100.00
3	100	98	98.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	100	100.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	100	100.00	100	100.00	100	100.00	100	100.00
8	100	100	100.00	100	100.00	100	100.00	100	100.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	995	99.50	998	99.80	1000	100.00	1000	100.00

Tabla A.7: Aciertos para el conjunto de datos *mushroom* con jerarquía de similitud de los atributos utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	99	99.00	100	100.00	100	100.00	100	100.00
2	100	97	97.00	100	100.00	100	100.00	100	100.00
3	100	99	99.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	100	100.00	100	100.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	100	100.00	100	100.00	100	100.00	100	100.00
8	100	100	100.00	100	100.00	100	100.00	100	100.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	995	99.50	1000	100.00	1000	100.00	1000	100.00

Tabla A.8: Aciertos para el conjunto de datos *mushroom* con jerarquía de similitud de los atributos utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	100	100	100.00	100	100.00	100	100.00	100	100.00
1	100	58	58.00	60	60.00	60	60.00	60	60.00
2	100	94	94.00	94	94.00	96	96.00	98	98.00
3	100	100	100.00	100	100.00	100	100.00	100	100.00
4	100	100	100.00	100	100.00	100	100.00	100	100.00
5	100	93	93.00	98	98.00	100	100.00	100	100.00
6	100	100	100.00	100	100.00	100	100.00	100	100.00
7	100	87	87.00	94	94.00	95	95.00	99	99.00
8	100	98	98.00	98	98.00	98	98.00	98	98.00
9	100	100	100.00	100	100.00	100	100.00	100	100.00
Total/prom.	1000	930	93.00	944	94.40	949	94.90	955	95.50

Tabla A.9: Aciertos para el conjunto de datos *mushroom* con jerarquía de similitud de los atributos utilizando la distancia euclídea

A.2 Conjunto de datos *Soybean*

Los resultados del conjunto de datos *soybean*, al igual que *mushroom*, son calculados utilizando las distancias de Nienhuys-Cheng, la distancia de Estruch et al. y la distancia euclídea sobre sobre una estructura plana (sin jerarquías), una jerarquía inducida por nombres (y valores de los atributos) y una jerarquía de acuerdo con la similitud de los atributos.

Estructura plana (sin jerarquías):

Distancia de Nienhuys-Cheng									
No.	Elementos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
pliegue	del pliegue	Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	25	80.65	30	96.77	31	100.00	31	100.00
1	31	22	70.97	29	93.55	29	93.55	29	93.55
2	31	19	61.29	26	83.87	28	90.32	29	93.55
3	31	19	61.29	29	93.55	29	93.55	29	93.55
4	31	21	67.74	25	80.65	26	83.87	29	93.55
5	31	24	77.42	26	83.87	29	93.55	30	96.77
6	31	28	90.32	30	96.77	30	96.77	31	100.00
7	31	25	80.65	30	96.77	30	96.77	30	96.77
8	31	24	77.42	27	87.10	28	90.32	28	90.32
9	28	24	85.71	28	100.00	28	100.00	28	100.00
Total/prom.	307	231	75.35	280	91.29	288	93.87	294	95.81

Tabla A.10: Aciertos para el conjunto de datos *soybean* sin jerarquías utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No.	Elementos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
pliegue	del pliegue	Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	25	80.65	30	96.77	31	100.00	31	100.00
1	31	22	70.97	29	93.55	29	93.55	29	93.55
2	31	19	61.29	26	83.87	28	90.32	29	93.55
3	31	19	61.29	29	93.55	29	93.55	29	93.55
4	31	21	67.74	25	80.65	26	83.87	29	93.55
5	31	24	77.42	26	83.87	29	93.55	30	96.77
6	31	28	90.32	30	96.77	30	96.77	31	100.00
7	31	25	80.65	30	96.77	30	96.77	30	96.77
8	31	24	77.42	27	87.10	28	90.32	28	90.32
9	28	24	85.71	28	100.00	28	100.00	28	100.00
Total/prom.	307	231	75.35	280	91.29	288	93.87	294	95.81

Tabla A.11: Aciertos para el conjunto de datos *soybean* sin jerarquías utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	25	80.65	29	93.55	30	96.77	31	100.00
1	31	22	70.97	25	80.65	29	93.55	29	93.55
2	31	19	61.29	25	80.65	26	83.87	26	83.87
3	31	19	61.29	28	90.32	29	93.55	29	93.55
4	31	21	67.74	24	77.42	25	80.65	25	80.65
5	31	24	77.42	26	83.87	26	83.87	28	90.32
6	31	28	90.32	30	96.77	30	96.77	30	96.77
7	31	25	80.65	26	83.87	30	96.77	30	96.77
8	31	24	77.42	27	87.10	27	87.10	28	90.32
9	28	24	85.71	27	96.43	28	100.00	28	100.00
Total/prom.	307	231	75.35	267	87.06	280	91.29	284	92.58

Tabla A.12: Aciertos para el conjunto de datos *soybean* sin jerarquías utilizando la distancia euclídea

Jerarquía inducida por nombres y valores de los atributos:

Distancia de Nienhuys-Cheng									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	28	90.32	30	96.77	30	96.77	30	96.77
1	31	23	74.19	28	90.32	29	93.55	29	93.55
2	31	24	77.42	26	83.87	28	90.32	28	90.32
3	31	23	74.19	27	87.10	28	90.32	28	90.32
4	31	21	67.74	25	80.65	26	83.87	27	87.10
5	31	24	77.42	27	87.10	27	87.10	27	87.10
6	31	26	83.87	29	93.55	30	96.77	31	100.00
7	31	24	77.42	30	96.77	30	96.77	30	96.77
8	31	24	77.42	26	83.87	28	90.32	28	90.32
9	28	24	85.71	27	96.43	28	100.00	28	100.00
Total/prom.	307	241	78.57	275	89.64	284	92.58	286	93.23

Tabla A.13: Aciertos para el conjunto de datos *soybean* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	27	87.10	29	93.55	30	96.77	31	100.00
1	31	23	74.19	28	90.32	29	93.55	29	93.55
2	31	22	70.97	27	87.10	28	90.32	28	90.32
3	31	24	77.42	27	87.10	28	90.32	28	90.32
4	31	21	67.74	25	80.65	25	80.65	27	87.10
5	31	23	74.19	24	77.42	24	77.42	25	80.65
6	31	25	80.65	28	90.32	31	100.00	31	100.00
7	31	25	80.65	28	90.32	30	96.77	30	96.77
8	31	20	64.52	27	87.10	28	90.32	30	96.77
9	28	23	82.14	27	96.43	28	100.00	28	100.00
Total/prom.	307	233	75.96	270	88.03	281	91.61	287	93.55

Tabla A.14: Aciertos para el conjunto de datos *soybean* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	25	80.65	29	93.55	30	96.77	31	100.00
1	31	22	70.97	25	80.65	29	93.55	29	93.55
2	31	19	61.29	25	80.65	26	83.87	26	83.87
3	31	19	61.29	28	90.32	29	93.55	29	93.55
4	31	21	67.74	24	77.42	25	80.65	25	80.65
5	31	24	77.42	26	83.87	26	83.87	28	90.32
6	31	28	90.32	30	96.77	30	96.77	30	96.77
7	31	25	80.65	26	83.87	30	96.77	30	96.77
8	31	24	77.42	27	87.10	27	87.10	28	90.32
9	28	24	85.71	27	96.43	28	100.00	28	100.00
Total/prom.	307	231	75.35	267	87.06	280	91.29	284	92.58

Tabla A.15: Aciertos para el conjunto de datos *soybean* con jerarquía inducida por nombres y valores de los atributos utilizando la distancia euclídea

Jerarquía de similitud de los atributos:

Distancia de Nienhuys-Cheng									
No.	Elementos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
pliegue	del pliegue	Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	26	83.87	31	100.00	31	100.00	31	100.00
1	31	27	87.10	31	100.00	31	100.00	31	100.00
2	31	25	80.65	30	96.77	30	96.77	30	96.77
3	31	25	80.65	30	96.77	30	96.77	30	96.77
4	31	28	90.32	30	96.77	31	100.00	31	100.00
5	31	26	83.87	31	100.00	31	100.00	31	100.00
6	31	28	90.32	31	100.00	31	100.00	31	100.00
7	31	27	87.10	31	100.00	31	100.00	31	100.00
8	31	25	80.65	31	100.00	31	100.00	31	100.00
9	28	27	96.43	28	100.00	28	100.00	28	100.00
Total/prom.	307	264	86.09	304	99.03	305	99.35	305	99.35

Tabla A.16: Aciertos para el conjunto de datos *soybean* con jerarquía de similitud de los atributos utilizando la distancia de Nienhuys-Cheng

Distancia de Estruch et al.									
No.	Elementos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
pliegue	del pliegue	Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	26	83.87	31	100.00	31	100.00	31	100.00
1	31	27	87.10	31	100.00	31	100.00	31	100.00
2	31	25	80.65	30	96.77	30	96.77	30	96.77
3	31	25	80.65	30	96.77	30	96.77	30	96.77
4	31	28	90.32	30	96.77	31	100.00	31	100.00
5	31	26	83.87	31	100.00	31	100.00	31	100.00
6	31	28	90.32	31	100.00	31	100.00	31	100.00
7	31	27	87.10	31	100.00	31	100.00	31	100.00
8	31	25	80.65	31	100.00	31	100.00	31	100.00
9	28	27	96.43	28	100.00	28	100.00	28	100.00
Total/prom.	307	264	86.09	304	99.03	305	99.35	305	99.35

Tabla A.17: Aciertos para el conjunto de datos *soybean* con jerarquía de similitud de los atributos utilizando la distancia de Estruch et al.

Distancia euclídea									
No. pliegue	Elementos del pliegue	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
0	31	25	80.65	29	93.55	30	96.77	31	100.00
1	31	22	70.97	25	80.65	29	93.55	29	93.55
2	31	19	61.29	25	80.65	26	83.87	26	83.87
3	31	19	61.29	28	90.32	29	93.55	29	93.55
4	31	21	67.74	24	77.42	25	80.65	25	80.65
5	31	24	77.42	26	83.87	26	83.87	28	90.32
6	31	28	90.32	30	96.77	30	96.77	30	96.77
7	31	25	80.65	26	83.87	30	96.77	30	96.77
8	31	24	77.42	27	87.10	27	87.10	28	90.32
9	28	24	85.71	27	96.43	28	100.00	28	100.00
Total/prom.	307	231	75.35	267	87.06	280	91.29	284	92.58

Tabla A.18: Aciertos para el conjunto de datos *soybean* con jerarquía de similitud de los atributos utilizando la distancia euclídea

A.3 Conjunto de datos *Demospongiae*

Los resultados para este conjunto de datos (*demospongiae*) son calculados utilizando las distancias de Nienhuys-Cheng y la distancia de Estruch et al.

	Distancia de Nienhuys-Cheng							
	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
	No. ejemplos	%	No. ejemplos	%	No. ejemplos	%	No. ejemplos	%
Aciertos	123	60.89	144	71.29	150	74.26	149	73.76
Errores	79	39.11	58	28.71	52	25.74	53	26.24
Total	202		202		202		202	

Tabla A.19: Aciertos y errores para el conjunto de datos *demospongiae* utilizando la distancia de Nienhuys-Cheng

Distancia de Nienhuys-Cheng									
Clase	No. ejemplos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
ASTROPHORIDA	36	23	63.89	30	83.33	30	83.33	29	80.56
AXINELLIDA	28	20	71.43	24	85.71	24	85.71	24	85.71
DICTYOCERATIDA	20	20	100.00	20	100.00	20	100.00	20	100.00
HADROMERIDA	48	28	58.33	29	60.42	29	60.42	29	60.42
HALICHONDRIDA	16	1	6.25	8	50.00	9	56.25	9	56.25
HAPLOSCLERIDA	18	12	66.67	12	66.67	15	83.33	14	77.78
POECILOSCLERIDA	36	19	52.78	21	58.33	23	63.89	24	66.67
Total	202	123		144		150		149	

Tabla A.20: Aciertos por cada clase para el conjunto de datos *demospongiae* utilizando la distancia de Nienhuys-Cheng

	Distancia de Estruch et al.							
	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
	No. ejemplos	%	No. ejemplos	%	No. ejemplos	%	No. ejemplos	%
Aciertos	123	60.89	148	73.27	159	78.71	157	77.72
Errores	79	39.11	54	26.73	43	21.29	45	22.28
Total	202		202		202		202	

Tabla A.21: Aciertos y errores para el conjunto de datos *demospongiae* utilizando la distancia de Estruch et al.

Distancia de Estruch et al.									
Clase	No. ejemplos	$i = 0$		$i = 1$		$i = 2$		$i = 3$	
		Aciertos	%	Aciertos	%	Aciertos	%	Aciertos	%
ASTROPHORIDA	36	24	66.67	32	88.89	32	88.89	31	86.11
AXINELLIDA	28	20	71.43	23	82.14	23	82.14	23	82.14
DICTYOCERATIDA	20	20	100.00	20	100.00	20	100.00	20	100.00
HADROMERIDA	48	26	54.17	33	68.75	33	68.75	33	68.75
HALICHONDRIDA	16	0	0.00	4	25.00	8	50.00	8	50.00
HAPLOSCLERIDA	18	12	66.67	15	83.33	18	100.00	18	100.00
POECILOSCLERIDA	36	21	58.33	21	58.33	25	69.44	24	66.67
Total	202	123		148		159		157	

Tabla A.22: Aciertos por cada clase para el conjunto de datos *demospongiae* utilizando la distancia de Estruch et al.

B Descripción de la implementación de las distancias entre términos

En este apéndice se describe la implementación de las distancias de Nienhuys-Cheng y Estruch et al. utilizando XML como estructura de representación de datos.

B.1 Distancia de Nienhuys-Cheng

La distancia de Nienhuys-Cheng utiliza la profundidad de las ocurrencias de manera que las diferencias que están cerca de los símbolos raíz cuentan más que aquellas que no lo están. Dadas dos expresiones, $s = s_0(s_1, \ldots, s_n)$ y $t = t_0(t_1, \ldots, t_n)$, esta distancia está definida de la siguiente manera [9]:

$$d_N(s,t) = \begin{cases} 0, & \text{si } s = t \\ 1, & \text{si } \neg Compatible(s,t) \\ \frac{1}{2n}\sum_{i=1}^{n} d(s_i,t_i), & \text{otro caso} \end{cases}$$

Dos expresiones son incompatibles si los símbolos raíz o la aridad de las dos expresiones son diferentes.

Para implementar la distancia Nienhuys-Cheng entre dos ejemplos, representados en una estructura XML, es necesario recorrer (recursivamente) todos los elementos de un árbol e ir comparándolos con los elementos del segundo, aplicando la función de distancia.

Inicialmente se determina el número de hijos de cada árbol y puede ocurrir una de las siguientes tres situaciones: 1) ninguno de los dos elementos tiene hijos, 2) los dos elementos tienen diferente número de hijos, ó 3) los dos elementos tienen el mismo número de hijos. Para la primera situación, si el contenido de los dos elementos son iguales su distancia es 0; si por el contrario son diferentes, su distancia es 1; para el segundo caso, los elementos (o términos) son incompatibles y su distancia siempre es 1; en el tercer caso es necesario calcular recursivamente las distancias de cada elemento, sumarlas entre sí y finalmente multiplicarlas por el factor $\frac{1}{2n}$, donde n es el número de hijos del elemento.

```
        <Mushroom>
1           <bruises>BRUISES</bruises>
2           <odor>ALMOND</odor>
3           <population>SEVERAL</population>
4           <habitat>WOODS</habitat>
5           <cap>
5.1             <shape>CONVEX</shape>
5.2             <surface>SMOOTH</surface>
5.3             <color>WHITE</color>
            </cap>
6           <gill>
6.1             <attachment>FREE</attachment>
6.2             <spacing>CROWDED</spacing>
6.3             <size>NARROW</size>
6.4             <color>WHITE</color>
            </gill>
7           <stalk>
7.1             <shape>TAPERING</shape>
7.2             <root>BULBOUS</root>
7.3             <surface>
7.3.1               <above_ring>SMOOTH</above_ring>
7.3.2               <below_ring>SMOOTH</below_ring>
                </surface>
7.4             <color>
7.4.1               <above_ring>WHITE</above_ring>
7.4.2               <below_ring>WHITE</below_ring>
                </color>
            </stalk>
8           <veil>
8.1             <type>PARTIAL</type>
8.2             <color>WHITE</color>
            </veil>
9           <ring>
9.1             <number>ONE</number>
9.2             <type>PENDANT</type>
            </ring>
10          <spore>
10.1            <print>
10.1.1              <color>BROWN</color>
                </print>
            </spore>
        </Mushroom>
```

(a)

```
<Mushroom>
    <bruises>BRUISES</bruises>
    <odor>ALMOND</odor>
    <population>SEVERAL</population>
    <habitat>WOODS</habitat>
    <cap>
        <shape>CONVEX</shape>
        <surface>SMOOTH</surface>
        <color>YELLOW</color>
    </cap>
    <gill>
        <attachment>FREE</attachment>
        <spacing>CROWDED</spacing>
        <size>BROAD</size>
        <color>YELLOW</color>
    </gill>
    <stalk>
        <shape>TAPERING</shape>
        <root>BULBOUS</root>
        <surface>
            <above_ring>SMOOTH</above_ring>
            <below_ring>SMOOTH</below_ring>
        </surface>
        <color>
            <above_ring>WHITE</above_ring>
            <below_ring>WHITE</below_ring>
        </color>
    </stalk>
    <veil>
        <type>PARTIAL</type>
        <color>WHITE</color>
    </veil>
    <ring>
        <number>ONE</number>
        <type>PENDANT</type>
    </ring>
    <spore>
        <print>
            <color>YELLOW</color>
        </print>
    </spore>
</Mushroom>
```

(b)

Figura B.1: Ejemplo de dos jerarquías en XML para calcular distancias entre términos

La figura B.1 muestra dos ejemplos del conjunto de datos de *mushroom* representados en XML. Debido a que la distancia entre dos elementos iguales es 0, entonces se identifican únicamente los elementos que son diferentes; para el ejemplo de la figura B.1: *size* y tres elementos *color* pertenecientes a *cap*, *gill* y *spore*; luego se aplica la función de distancia sobre estos elementos (las distancias entre los elementos *color*, para los tres casos, son iguales a 1; por esta razón no se diferencian en este ejemplo; sin embargo, si el valor para alguno de estos elementos fuera una estructura, las distancias serían diferentes y deben calcularse cada una por separado):

$$d_N(color_1, color_2) = d_N(size_1, size_2) = 1$$

Después de calculada la distancia para *color* y *size*, es posible calcular las distancias para *cap* y *gill*:

$$d_N(cap_1, cap_2) = \frac{1}{2 \cdot 3}\big(d(color_1, color_2)\big) = \frac{1}{6} \cdot 1 = \frac{1}{6}$$

$$d_N(gill_1, gill_2) = \frac{1}{2 \cdot 4}\Big(\big(d(size_1, size_2)\big) + \big(d(color_1, color_2)\big)\Big) = \frac{1}{8} \cdot (1 + 1) = \frac{1}{4}$$

Para calcular la distancia de *spore* primero se debe calcular *print*:

$$d_N(print_1, print_2) = \frac{1}{2 \cdot 1}\big(d(color_1, color_2)\big) = \frac{1}{2} \cdot 1 = \frac{1}{2}$$

$$d_N(spore_1, spore_2) = \frac{1}{2 \cdot 1}\big(d(print_1, print_2)\big) = \frac{1}{2} \cdot \frac{1}{2} = \frac{1}{4}$$

Finalmente, utilizando las distancias de *cap*, *gill* y *spore* se calcula la distancia entre estas dos instancias:

$$d_N(gill_1, gill_2) = \frac{1}{2 \cdot 10}\Big(\big(d(cap_1, cap_2)\big) + \big(d(gill_1, gill_2)\big) + \big(d(spore_1, spore_2)\big)\Big)$$

$$= \frac{1}{20} \cdot \left(\frac{1}{6} + \frac{1}{4} + \frac{1}{4}\right) = \frac{1}{30}$$

B.2 Distancia de Estruch et al.

La distancia entre términos propuesta por Estruch et al. [9] tiene en cuenta los siguentes aspectos: el contexto de las diferencias, la complejidad sintáctica mediante el tamaño de una expresión y la ponderación asociada con las repeticiones utilizando relaciones de equivalencia. A continuación se describe la implementación de estos aspectos.

- **Diferencias sintácticas entre expresiones:**

  La diferencia sintáctica es un conjunto de ocurrencias incompatibles entre dos expresiones. Las ocurrencias son una secuencia de números positivos (separados por puntos), encabezada por el símbolo λ, que representan rutas de acceso en un árbol o un término (por ejemplo, sea $t = p(a, f(b))$, entonces el conjunto de ocurrencias $O(t) = \{\lambda, 1, 2, 2.1\}$).

  Para implementar la diferencia sintáctica, entre dos ejemplos, es necesario recorrer el documento XML y seleccionar las rutas en las cuales sus elementos son incompatibles y adicionarlos a una lista (de diferencias sintácticas). La incompatibilidad entre dos elementos se presenta cuando el número de hijos, etiquetas o valores en la misma ruta son diferentes; para el ejemplo de la figura B.1, la lista de diferencias sintácticas es la siguiente:

  $$O(Mushroom_1, Mushroom_2) = \{\lambda, 5.3, 6.3, 6.4, 10.1.1\}$$

- **Relaciones de equivalencia de clases:**

  Las diferencias repetidas entre los términos encontrados en la diferencia sintáctica son definidas como relaciones de equivalencia. La implementación de estas equivalencias de clases está definida por medio de la identificación de ocurrencias con repeticiones comunes (para las dos instancias) contenidas en la diferencia sintáctica; es decir, todos los elementos de cada ejemplo encontrados en la diferencia sintáctica, de acuerdo con su ruta, son comparados entre sí identificando sus repeticiones; posteriormente, estas repeticiones también son comparadas para los dos ejemplos y en caso de coincidir, sus rutas son asignadas a una misma clase.

  Inicialmente se construye una matriz cuadrada por cada ejemplo; la dimensión de esta matriz es igual al número de rutas existentes en la diferencia sintáctica, donde cada fila y columna representa uno de estos elementos para un ejemplo determinado. Luego, se

compara cada elemento con todos los demás: si tanto la etiqueta como el contenido del elemento son iguales, entonces la posición en la matriz correspondiente a estos elementos es marcada con 1, en caso contrario es marcada con 0. Cuando el elemento es comparado con él mismo (que corresponde a la diagonal de la matriz), este valor no será tenido en cuenta y por este motivo será marcado con el valor 0. Utilizando el ejemplo anterior (donde la diferencia sintáctica es: $O(Mushroom_1, Mushroom_2) = \{\lambda, 5.3, 6.3, 6.4, 10.1.1\}$, entonces se crean dos matrices como se muestra en la figura B.2 indicando cuáles elementos son iguales:

	5.3	6.3	6.4	10.1.1
5.3	0	0	1	0
6.3	0	0	0	0
6.4	1	0	0	0
10.1.1	0	0	0	0

(a)

	5.3	6.3	6.4	10.1.1
5.3	0	0	1	1
6.3	0	0	0	0
6.4	1	0	0	1
10.1.1	1	0	1	0

(b)

Figura B.2: Matriz de repeticiones para las jerarquias de la figura B.1

Para la matriz del ejemplo B.2b los elementos 5.3, 6.4 y 10.1.1 son iguales (en nombre de la etiqueta y valor). Por otra parte, para la matriz del ejemplo B.2a, únicamente los elementos 5.3 y 6.4 son iguales. Utilizando las matrices de los ejemplos anteriores, es posible construir una tercera matriz que contenga como resultado las repeticiones comunes. Para construir esta matriz resultante es necesario comparar cada celda y sólo serán marcadas con 1 aquellas que, en ambos casos, contengan el valor 1; es decir, las repeticiones son las mismas para ambos ejemplos. Utilizando las tablas de la figura B.2, únicamente serían marcadas las celdas 5.3 y 6.4.

	5.3	6.3	6.4	10.1.1
5.3	0	0	1	0
6.3	0	0	0	0
6.4	1	0	0	0
10.1.1	0	0	0	0

Tabla B.1: Matriz de repeticiones comúnes de la figura B.2

Utilizando la matriz resultante B.1 es posible asignar cada ruta dentro de una clase; la manera de hacerlo es colocando en una misma clase todas las rutas con repeticiones

comunes; luego las rutas de los elementos restantes de la diferencia sintáctica son asignadas, cada una, en una clase aparte. La tabla B.2 muestra la aisgnación de clases para cada ruta:

Ruta	Clase
5.3	1
6.4	1
6.3	2
10.1.1	3

Tabla B.2: Rutas por cada clase utilizando la matriz B.1

Esta tabla de equivalencias de clases B.2 es guardada en una estructura en memoria y, posteriormente, utilizada para calcular la ponderación de las ocurrencias.

- **Tamaño de una expresión**:

  El tamaño de una expresión $t = t_0(t_1, \ldots, t_n)$ está definido por la siguiente función [9]:

$$Size'(t) = \frac{1}{4} Size(t)$$

  donde:

$$Size(t_0(t_1, \ldots, t_n)) = \begin{cases} 1, \, n = 0 \\ 1 + \frac{\sum_{i=1}^{n} Size(t_i)}{2(n+1)}, \, n > 0 \end{cases}$$

  $Size$ es una función recursiva que se implementa obteniendo el número de hijos de un elemento; si el elemento no tiene hijos, su tamaño es 1; por el contrario, si el elemento tiene hijos, se calcula recursivamente el tamaño de todos los elementos hijos y se suman entre sí; luego, el valor obtenido divide por el factor $2(n + 1)$, donde n es el número de hijos y, finalmente, se le suma 1 a este valor. $Size'$ es calculado fácilmente dividiendo el valor de $Size$ entre 4.

  Los tamaños son calculados para todos los elementos de las dos instancias comparadas, relacionados en la lista de diferencias sintácticas y se almacenan en memoria para ser utilizados posteriormente. La tabla B.3 muestra los tamaños $Size'$ para los elementos de la tabla B.2.

Ruta	Clase	$size'$ ejemplo 1	$size'$ ejemplo 2
5.3	1	0.25	0.25
6.4	1	0.25	0.25
6.3	2	0.25	0.25
10.1.1	3	0.25	0.25

Tabla B.3: Tamaño de los elementos relacionados en la tabla B.2

- **Valor del contexto de ocurrencia**:

  El valor de contexto de ocurrencia de una expresión t, denotado por $C(o;t)$, es definido de la siguiente manera [9]:

$$C(o;t) = \begin{cases} 1, o = \lambda \\ 2^{Length(o)} \cdot \prod_{\forall o' \in Pre(o)} (Arity(t|_{o'}) + 1), \text{otro caso} \end{cases}$$

  donde la longitud de una ocurrencia, $Length(o)$, es el número de ítems en o. Por ejemplo, $Length(5.3) = 2$, $Length(10.1.1) = 3$.

  El valor del contexto de ocurrencia se refiere a las relación entre un subtérmino $t|_0$ y t en el sentido en que, para un valor alto de $C(o;t)$ corresponde a una posición profunda de $t|_0$ en t o la existencia de supertérminos de $t|_0$ con un gran número de argumentos [9].

  Para implementar el valor del contexto, por cada ocurrencia encontrada en la diferencia sintáctica, se recorre la rama del árbol ascendentemente calculando la aridad de cada padre y sumándole 1; luego, todas las aridades son multiplicadas entre sí; finalmente, se multiplica el valor anterior por $2^{Length(o)}$. Para aquellos casos en que la ocurrencia sea λ, de acuerdo con la definición, el valor de contexto será 1.

  La tabla B.4 muestra el valor de contexto calculado para los elementos de la tabla B.2:

Ruta	Clase	$size'$ ejemplo 1	$size'$ ejemplo 2	Valor de contexto
5.3	1	0.25	0.25	176
6.4	1	0.25	0.25	220
6.3	2	0.25	0.25	220
10.1.1	3	0.25	0.25	352

Tabla B.4: Valor de contexto de los elementos relacionados en la tabla B.2

- **Ponderación de las ocurrencias**:

  La función de ponderación w es una función que, dada dos expresiones s y t, se define de la siguiente manera [9]:

$$\forall o \in O_i^\star(s,t),\ w(o) = \frac{3f_i(o)+1}{4f_i(o)}$$

  Esta función asocia ponderaciones a las ocurrencias de manera que, mientras mayor sea el valor de contexto de una ocurrencia, menor es la ponderación asignada a esta; es decir, la diferencia sintáctica menos significativa que se refiere a la ocurrencia [9].

  Para la implementación de la ponderación, por cada ocurrencia relacionada en la diferencia sintáctica, primero se calcula el valor de $f_i(o)$ que corresponde al valor de posición de cada ocurrencia dentro de la clase obtenida en la relación de equivalencia. En el ejemplo anterior, $f_1(5.3) = 1$ y $f_1(6.4) = 2$, debido a que la equivalencia de clase 1 contiene dos ocurrencias; para la clase 2 y 3, que solo tienen una ocurrencia, entonces $f_2(6.3) = f_3(11.1.1) = 1$.

  Después de calculado el valor $f_i(o)$, se aplica la función de ponderación. La tabla B.5 muestra la ponderación calculada utilizando las relaciones de equivalencia obtenidas en la tabla B.2.

Ruta	Clase	$size'$ ejemplo 1	$size'$ ejemplo 2	Valor de contexto	Ponderación
5.3	1	0.25	0.25	176	1
6.4	1	0.25	0.25	220	0.875
6.3	2	0.25	0.25	220	1
10.1.1	3	0.25	0.25	352	1

Tabla B.5: Ponderación de las ocurrencias de los elementos relacionados en la tabla B.2

- **Distancia de Estruch et al.**:

  Dada dos expresiones s y t, la distancia de Estruch et al. se define de la siguiente manera:

$$d_E(s,t) = \sum_{o \in O^\star(s,t)} \frac{w(o)}{C(o)} \big(size'(s|_o) + size'(t|_o) \big)$$

  Finalmente, para calcular la distancia de Estruch et al. se suman los tamaños de todos los elemntos relacionados en la diferencia sintáctica, se multiplican por la ponderación y se dividen por el valor de contexto; para obtener la distancia, se suman estos valores entre sí. Para el ejemplo anterior (mostrado en la tabla B.5), la distancia es igual a 0.00852272.