Universidad de Jaén
Escuela Politécnica Superior (Jaén)

Proyecto Fin de Carrera:

INTERFAZ PARA GENERAR SEMÁNTICA EN CONJUNTOS DE ETIQUETAS NO BALANCEADOS

Alumno:
Carlos Alberto Sánchez Quintana

Tutores:
Dr. D. Luis Martínez López
Dr. Dª. Macarena Espinilla Estévez

Departamento de Informática

Área de Conocimiento: Lenguajes y Sistemas Informáticos

Julio 2009

Universidad de Jaén
Escuela Politécnica Superior de Jaén
Departamento de Informática

Dr. D. Luis Martínez López y Da. Macarena Espinilla Estévez, tutores del Proyecto Fin de Carrera titulado: Interfaz para generar semántica en conjuntos de etiquetas no balanceados que presenta D. Carlos Alberto Sánchez Quintana, autorizan su presentación para defensa y evaluación en la Escuela Politécnica Superior de Jaén.

Jaén, Julio 2009

El alumno:

D. Carlos Alberto Sánchez Quintana

Los Tutores:

Dr. D. Luis Martínez Lípez
Dª. Macarena Espinilla Estévez

Agradecimientos.

Quiero expresar a continuación mi gratitud a todas aquellas personas que me han ayudado a que llegara este momento. En primer lugar, estaré eternamente agradecido a mis padres y mi familia por su constante apoyo y cariño durante todos los años de mi vida. También quiero tener un recuerdo para mis amigos y compañeros de carrera por compartir un pedazo de su vida cnmigo y acompañarme en este viaje. Y por supuesto, reconocer la gran labor llevada a cabo por mis tutores de proyecto, Luis y Macarena, sobretodo por su paciencia y por no perder la confianza en mí.

Gracias a todos.

Índice General

Capítulo 1

PREFACIO

1.1.- Introducción al proyecto.

Cuando nos enfrentamos a cualquier tipo de problema, dependiendo de sus aspectos, podemos tratar con diferentes tipos de información. Normalmente, los problemas presentan aspectos cuantitativos que pueden ser representados mediante valores numéricos precisos. En otros casos, los problemas presentan aspectos cualitativos que son complejos de representar mediante un valor exacto y preciso. El enfoque lingüístico difuso trata con aspectos cualitativos que son representados mediante variables lingüísticas, proporcionando una importante herramienta para resolver problemas en diferentes áreas tales como recuperación de información [1], [2], [3], [4], [5], [6], [7], [8] administración de recursos humanos y evaluación de servicios, aplicaciones de seguridad [9], [10], [11], [12], [13], [14], [15], [16], operadores de agregación [17], [18], [19], [20], obtención de consenso [21], [22], [22] , etc …

Cuando un problema es resuelto usando información lingüística, implica la necesidad de una computación de palabras. Tres modelos computacionales pueden ser encontrados en la literatura especializada para abordar este proceso: (i) el modelo semántico [23], [24], (ii) el simbólico [25] y (iii) el modelo basado en las 2-tuplas lingüísticas [26]. El modelo basado en las 2-tuplas lingüísticas se muestra como mejor que los otros, debido

al hecho que es capaz de conseguir el proceso de computación con palabras de una manera precisa además de otras ventajas presentadas en [27].

La mayoría de los problemas de modelado de información con valores lingüísticos usan variables lingüísticas representadas por conjuntos de términos lingüísticos cuyos términos están uniforme y simétricamente distribuidos. Sin embargo, existen problemas que necesitan expresar sus variables con conjuntos de términos que no están ni uniforme ni simétricamente distribuidos [24], [28], [29], [30], [20]. Llamaremos a este tipo de conjuntos de términos lingüísticos "conjuntos de términos lingüísticos no balanceados". En algunos casos, la información lingüística no balanceada aparece como consecuencia de la naturaleza de las variables lingüísticas que participan en el problema como ocurre, por ejemplo, en los sistemas de calidad (figura 1.1); y en otros, aparece en problemas que tratan con escalas para valorar preferencias donde los expertos necesitan valorar en un lado un número de términos del dominio de referencia mayor que en otro (figura 1.2).

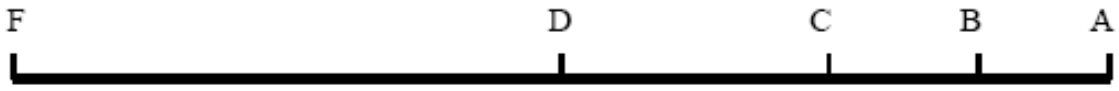

Figura 1.1: Sistema de calidad

Figura 1.2: Escala con más valores en el lado derecho del término medio

Los autores Luis Martínez, Francisco Herrera y Enrique Herrera-Viedma en el artículo "*A Fuzzy Linguistic Methodology To Deal With Unbalanced Linguistic Term Sets"* han desarrollado una metodología para representar, manejar y lograr el proceso de computación con palabras con conjuntos de términos lingüísticos no balanceados sin pérdida de información. Primero, se define un modelo de representación lingüística no balanceada, el cual asigna semánticas a los términos lingüísticos. En ese momento, se esboza un proceso para asignar semántica a los términos lingüísticos asociados a un

conjunto de términos lingüísticos no balanceado. Entonces, esas ideas son formalizadas mediante un algoritmo de representación semántica que representa cada término mediante una función de pertenencia paramétrica que es asignada usando la estructura de una jerarquía lingüística. Segundo, presentan un modelo computacional para conjuntos de términos no balanceados basado en el modelo lingüístico difuso de las 2-tuplas para llevar a cabo el proceso de computación de palabras sin pérdida de información.

En este proyecto se pretende llevar a cabo el desarrollo de una interfaz software para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información implementando la metodología presentada en el artículo citado anteriormente.

La aplicación desarrollada trabaja a nivel local y no hace uso de ningún sistema de base de datos. Su cometido es asignar semántica a los términos lingüísticos asociados a un conjunto de términos lingüísticos no balanceado que es definido en base a una serie de parámetros de entrada. El resultado obtenido puede ser exportado en un archivo XML para ser utilizado en otras aplicaciones o soportes.

La presente memoria se estructura de la siguiente manera:

En el capítulo 2 vamos a revisar los conceptos básicos y herramientas del modelado lingüístico para el manejo de información lingüística. Así como las nociones de lógica difusa y teoría de conjuntos difusos y el enfoque lingüístico difuso. Además introduciremos varios modelos computacionales para trabajar con información lingüística.

En el capítulo 3 abordaremos los procesos de la computación con palabras que surge como consecuencia del uso del enfoque lingüístico centrándonos en el modelo de representación de información lingüística basado en la 2-tuplas, definido como una mejora de los modelos clásicos para el modelado de la información lingüística.

En el capítulo 4 se presenta el desarrollo de una metodología para representar, manejar y lograr el proceso de computación con palabras con conjuntos de términos lingüísticos

no balanceados sin perdida de información. Se define un modelo de representación lingüística no balanceada el cual asigna semánticas a los términos lingüísticos. Después, se esboza un proceso para asignar semántica a los términos lingüísticos asociados a un conjunto de términos lingüísticos no balanceado. Entonces, esas ideas son formalizadas mediante un algoritmo de representación semántica que representa cada término mediante una función de pertenencia con parámetros que es asignada usando la estructura de la jerarquía lingüística. Dicho algoritmo de representación es la base de la aplicación software objeto de este proyecto, por lo que nos centraremos especialmente en este capítulo.

Por último se expondrán una serie de conclusiones y se adjuntará un manual de usuario de la aplicación. Finalmente, para aquellos lectores que estén interesados en el tema, se incluirá una amplia bibliografía.

1.2.- Propósito del proyecto.

El propósito de este proyecto es la implementación una interfaz software para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información. Para ello implementaremos la metodología presentada en el artículo "*A Fuzzy Linguistic Methodology To Deal With Unbalanced Linguistic Term Sets*" para representar, administrar y lograr el proceso de computación con palabras en conjuntos de términos lingüísticos no balanceados. El resultado obtenido se exportará en formato XML para poder ser utilizado por otras aplicaciones o soportes.

1.3.- Objetivos del proyecto.

1. Búsqueda y revisión bibliográfica.

2. Estudio y análisis de la metodología presentada en el artículo "*A Fuzzy Linguistic Methodology To Deal With Unbalanced Linguistic Term Sets"* para representar, administrar y lograr el proceso de computación con palabras en conjuntos de términos lingüísticos no balanceados.

3. Implementación del algoritmo de representación semántica para conjuntos de términos no balanceados propuesto en el artículo citado anteriormente.

4. Diseñar una interfaz sencilla y amigable para el usuario que le muestre gráficamente la representación semántica del conjunto no balanceado de entrada.

5. Exportar la representación semántica obtenida en formato xml.

Capítulo 2

MODELADO LINGÜÍSTICO DIFUSO DE LA INFORMACIÓN.

En este capítulo vamos a revisar el concepto de información lingüística y la utilización del modelado lingüístico difuso para el manejo de dicha información, que nos van a proporcionar una mayor flexibilidad en el tratamiento de la información.

2.1.- Introducción.

La Lógica Difusa se plantea como alternativa a la lógica tradicional, con el objetivo de introducir grados de incertidumbre en las sentencias que califica [31]. Hay numerosas situaciones en las que la lógica tradicional funciona perfectamente. Por ejemplo, supongamos que partimos de las calificaciones obtenidas en una clase y queremos agrupar a los aprobados (aquellos que hayan obtenido una calificación igual o superior a 5). El proceso de razonamiento que se seguiría mediante la lógica tradicional sería ir comparando cada calificación con 5 hasta obtener cuáles están aprobados y cuáles no:

Es cierto que 7 >= 5? SI: Aprobado
Es cierto que 4 >= 5? NO: No aprobado
Es cierto que 5 >= 5? SI: Aprobado

Sin embargo, el inconveniente de esta lógica es que en la vida real no nos encontramos frecuentemente con criterios de clasificación tan tajantes como en el ejemplo. En efecto, hay numerosas situaciones en las que la información no puede ser evaluada cuantitativamente de forma precisa, pero puede que sí sea posible hacerlo cualitativamente, y en estos casos hemos de hacer uso de un enfoque lingüístico. Por ejemplo, cuando intentamos calificar algún fenómeno relacionado con percepciones humanas, a menudo usamos palabras o descripciones en lenguaje natural, en lugar de valores numéricos. Supongamos que dado un conjunto de personas, las intentamos agrupar según su altura. Las personas no son sólo altas o bajas sino que la mayoría pertenecen a grupos de altura intermedia. La gente suele ser más bien alta o de altura media. Casi nunca las calificamos con rotundidad, porque el lenguaje que usamos nos permite introducir modificadores que añaden imprecisión: un poco, mucho, algo...

Como la lógica tradicional es bivaluada (sólo admite dos valores: o el elemento pertenece al conjunto o no pertenece), se ve maniatada para agrupar según su altura al anterior conjunto de personas, puesto que su solución sería definir un umbral de pertenencia (por ejemplo, un valor que todo el mundo considera que, de ser alcanzado o superado, la persona en cuestión puede llamarse alta). Si dicho umbral es 1.80, todas las personas que midan 1.80 o más serán altas, mientras que el resto serán bajas. Según esta manera de pensar, alguien que mida 1.79 será tratado igual que otro que mida 1.60, ya que ambos han merecido el calificativo de personas bajas.

Si dispusiéramos de una herramienta para caracterizar las alturas de forma que las transiciones entre las que son altas y las que no lo son fueran suaves, estaríamos reproduciendo la realidad mucho más fielmente. En la realidad hay unos puntos de cruce donde las personas dejan de ser altas para ser consideradas medianas, de forma que el concepto de alto decrece linealmente con la altura. Asignando una función lineal para caracterizar el concepto alto en lugar de definir un sólo umbral de separación estamos dando mucha más información acerca de los elementos. Esta función, como veremos, se llamará función de pertenencia. En este sentido, el uso de la Teoría de Conjuntos Difusos ha dado muy buenos resultados para el tratamiento de información de forma cualitativa [32]. El modelado lingüístico difuso es una herramienta que

permite representar aspectos cualitativos y que está basada en el concepto de variables lingüísticas, es decir, variables cuyos valores no son números, sino palabras o sentencias expresadas en lenguaje natural o artificial [32]. Cada valor lingüístico se caracteriza por un valor sintáctico o etiqueta y un valor semántico o significado. La etiqueta es una palabra o sentencia perteneciente a un conjunto de términos lingüísticos y el significado es un subconjunto difuso en un universo de discurso.

Se ha demostrado que es una herramienta muy útil en numerosos problemas, como por ejemplo en la toma de decisiones [39, 40, 41], evaluación de la calidad informativa de documentos Web [33], modelos de recuperación de información [34, 35, 36], diagnósticos clínicos [37], análisis político [38], etc.

En este capítulo nos disponemos a realizar una introducción al modelado difuso de la información lingüística. Vamos a introducir los conceptos básicos y herramientas del modelado lingüístico para el manejo de información lingüística, revisar las nociones de lógica difusa, teoría de conjuntos difusos y el enfoque lingüístico difuso.

2.2.- Conceptos básicos de información Lingüística.

Vamos a comenzar presentando una revisión de los conceptos básicos de la Teoría de Conjuntos Difusos que van a ser utilizados en el modelado lingüístico utilizado en esta memoria. El interés de la Teoría de Conjuntos Difusos se centra esencialmente en modelar aquellos problemas donde los enfoques clásicos de la Teoría de Conjuntos y la Teoría de la Probabilidad resultan insuficientes o no operativos. Por ello, generaliza la noción clásica de conjunto e introduce el concepto de ambigüedad, de manera que los conjuntos difusos nos proporcionan una nueva forma de representar la imprecisión e incertidumbre presentes en determinados problemas.

2.2.1.- Conjuntos Difusos y Funciones de Pertenencia.

La noción de conjunto refleja la tendencia a organizar, generalizar y clasificar el conocimiento sobre los objetos del mundo real. El encapsulamiento de los objetos es una colección cuyos miembros comparten una serie de características o propiedades que implican la noción de conjunto. Los conjuntos introducen una noción de dicotomía, que en esencia es una clasificación binaria: o se acepta o se rechaza la pertenencia de un objeto a una categoría determinada. Habitualmente la decisión de aceptar se denota como 1 y la de rechazar como 0. Esta decisión de aceptar o rechazar se expresa mediante una función característica, según las propiedades que posean los objetos del conjunto.

La Lógica Difusa se fundamenta en el concepto de conjunto difuso [42] que suaviza el requerimiento anterior y admite valores intermedios en la función característica, que se denomina función de pertenencia. Esto permite una interpretación más realista de la información, puesto que la mayoría de las categorías que describen los objetos del mundo real, no tienen unos límites claros y bien definidos.

Un conjunto difuso puede definirse como una colección de objetos con valores de pertenencia entre 0 (exclusión total) y 1 (pertenencia total). Los valores de pertenencia expresan los grados con los que cada objeto es compatible con las propiedades o características distintivas de la colección. Formalmente podemos definir un conjunto difuso como sigue.

Definición 2.1: Un conjunto difuso A sobre un dominio o universo de discurso U está caracterizado por una función de pertenencia que asocia a cada elemento del conjunto el grado con que pertenece a dicho conjunto, asignándole un valor en el intervalo [0,1]:

$$\mu_A : U \rightarrow [0, 1]$$

Así, un conjunto difuso A sobre U puede representarse como un conjunto de pares ordenados de un elemento perteneciente a U y su grado de pertenencia, $A = \{(x, \mu_A(x)) / x \in U, \mu_A(x) \in [0, 1]\}$. Por ejemplo, consideremos el concepto persona alta, en un contexto donde la estatura oscila entre 1 y 2 m. Como es de suponer, alguien que mida 1,30 m. no se puede considerar como persona alta por lo que su grado de pertenencia al

conjunto de personas altas será de 0. Por el contrario, una persona que mida 1,90m. sí la consideramos alta por lo que su grado de pertenencia al conjunto es de 1.

La representación de una función de pertenencia pueden adoptar distintas formas, cumpliendo propiedades específicas, pero es el contexto de la aplicación lo que determina la representación más adecuada en cada caso. Puesto que las valoraciones lingüísticas dadas por los usuarios son únicamente aproximaciones, algunos autores consideran que las funciones de pertenencia paramétricas son suficientemente buenas para capturar la imprecisión de tales valoraciones lingüísticas. La representación paramétrica es obtenida a partir de una 4-tupla (a, b, α, β), ver figura 2.1, donde a y b indican el intervalo en que el valor de pertenencia es 1, con α y β indicando los límites izquierdo y derecho del dominio de definición de la función de pertenencia trapezoidal. Un caso particular de este tipo de representación son las valoraciones lingüísticas cuyas funciones de pertenencia son triangulares, es decir, a = b, por lo que se representan por medio de una 3-tupla (α, a, β). La siguiente figura muestra la descripción y la representación gráfica de un ejemplo de función de pertenencia trapezoidal.

$$\mu_A(x, a, b, \alpha, \beta) = \begin{cases} 0, & x < \alpha \\ \dfrac{x-\alpha}{a-\alpha}, & \alpha \leq x \leq a \\ 1, & a \leq x \leq b \\ \dfrac{\beta-x}{\beta-b}, & b \leq x \leq \beta \\ 0, & x > \beta \end{cases}$$

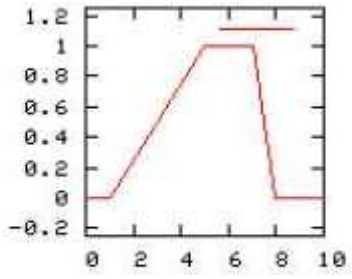

Figura 2.1: Ejemplo de función de pertenencia

2.2.2.- Definiciones Básicas

Definición 2.2: Se define el soporte de un conjunto difuso A en el universo U, como el conjunto formado por todos los elementos de U cuyo grado de pertenencia a A sea mayor que 0:

$$supp(A) = \{x \in U / \mu_A(x) > 0\}$$

Definición 2.3: La altura de un conjunto difuso A se define como el mayor grado de pertenencia de todos los elementos de dicho conjunto:

$$h(A) = max\{\mu_A(x) / x \in U\}$$

Definición 2.4: El α-corte de un conjunto difuso A es el conjunto formado por todos los elementos del universo U cuyos grados de pertenencia en A son mayores o iguales que el valor de corte $\alpha \in [0, 1]$:

$$A_\alpha = \{x \in U / \mu_A(x) \geq \alpha\}$$

Definición 2.5: Se denomina conjunto de niveles de un conjunto difuso A, al conjunto de grados de pertenencia de sus elementos:

$$L(_A) = \{a / \mu_A(x) = a, x \in U\}$$

2.2.3.-. Operaciones con Conjuntos Difusos.

Al igual que en la lógica tradicional, las operaciones lógicas que se pueden establecer entre conjuntos difusos son la intersección, la unión y el complemento. Al igual que el resultado de operar dos conjuntos clásicos es un nuevo conjunto clásico, las mismas operaciones con conjuntos difusos nos darán como resultado otros conjuntos también difusos.

Hay muchas formas de definir estas operaciones. Cualquier operación que cumpla las propiedades de una t-norma puede ser usada para hacer la intersección, de igual manera que cualquier operación que cumpla las propiedades de una t-conorma puede ser empleada para la unión. La tabla siguiente muestra las propiedades que deben cumplir las dos familias de funciones y algunos ejemplos.

	Propiedades	Ejemplos
T-Normas **T: [0,1] x [0,1] → [0,1]** $\mu_{A\cap B}(x)=T[\mu_A(x), \mu_B(x)]$	**Conmutativa:** T(a,b)=T(b,a) **Asociativa:** T(a,T(b,c))=T(T(a,b),c) **Monotonía:** T(a,b) >= T(c,d) si a>=c, y b>=d **Condiciones frontera:** T(a,1)=a	**Intersección estándar** T(a,b)=min(a,b) **Producto algebraico** T(a,b)=a · b **Intersección drástica** $T(a,b) = \begin{cases} a, \text{ si } b=1 \\ b, \text{ si } a=1 \\ 0, \text{ en otro caso} \end{cases}$
T-Normas **S: [0,1] x [0,1] → [0,1]** $\mu_{A\cup B}(x)=S[\mu_A(x), \mu_B(x)]$	**Conmutativa:** S(a,b)=S(b,a) **Asociativa:** S(a,S(b,c))=S(S(a,b),c) **Monotonía:** S(a,b) >= S(c,d) si a>=c, y b>=d **Condiciones frontera:** S(a,0)=a	**Unión estándar** S(a,b)=max(a,b) **Suma algebraica** S(a,b)= a + b - a · b **Unión drástica** $S(a,b) = \begin{cases} a, \text{ si } b=0 \\ b, \text{ si } a=0 \\ 1, \text{ en otro caso} \end{cases}$

Tabla 2.1: Propiedades de las funciones

Las operaciones se definen de la siguiente manera:

- Intersección: $A \cap B = \{(x, \mu_{A\cap B}) / \mu_{A\cap B}(x) = T[\mu_A(x), \mu_B(x)]\}$.
- Unión: $A \cup B = \{(x, \mu_{A\cup B}) / \mu_{A\cup B}(x) = S[\mu_A(x), \mu_B(x)]\}$.
- Complemento: $\mu_{\neg A}(x) = 1 - \mu_A(x)$.

En la figura siguiente podemos ver una representación gráfica de dichas operaciones.

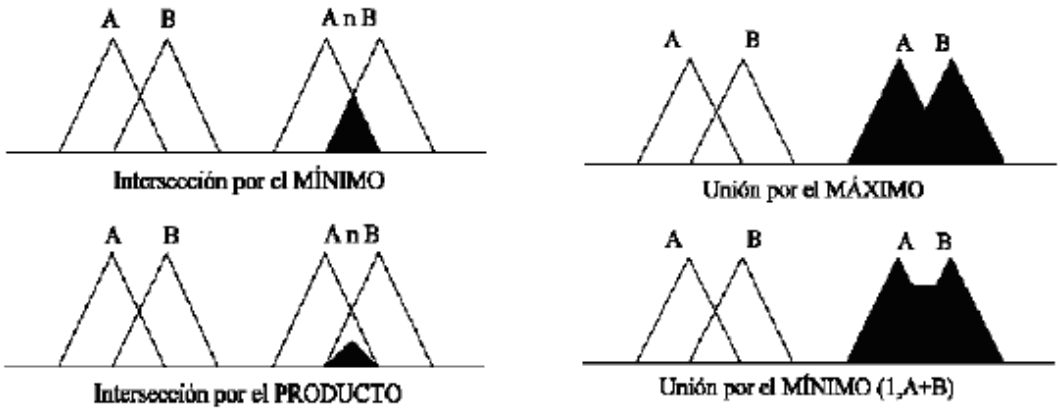

Figura 2.2: Intersección y Unión en conjuntos difusos.

2.2.4.-. Modelado Lingüístico Difuso.

La información que manejamos en el mundo real puede tener diferentes rangos de valoración y los valores pueden tener distinta naturaleza. En ocasiones, puede que no sea fácil valorar la información de forma precisa mediante un valor cuantitativo, sin embargo puede que sí sea factible hacerlo de forma cualitativa. En este caso, adoptar un enfoque lingüístico suele ofrecer mejores resultados que si aplicamos uno numérico. Por ejemplo, cuando evaluamos determinados aspectos relacionados con la percepción subjetiva (diseño, gusto, diversión, etc.), solemos utilizar palabras en lenguaje natural en lugar de valores numéricos (bonito, feo, dulce, salado, mucha, poca, etc.). Esto hecho se puede deber a diversas causas:

- Hay situaciones en las que la información, por su propia naturaleza, no puede ser cuantificada y por tanto únicamente puede ser valorada mediante el uso de términos lingüísticos, como sucede cuando realizamos una valoración sobre un libro que hayamos leído, que solemos usar términos como bueno, regular o malo.

- En otros casos, trabajar con información precisa de forma cuantitativa no es posible, o bien porque no están disponibles los elementos necesarios para llevar a cabo una medición exacta de esa información, o bien porque el coste es demasiado alto y nos basta con la aplicación de un valor aproximado. Por ejemplo, cuando evaluamos la velocidad de una motocicleta, en lugar de usar valores numéricos, solemos usar términos tales como rápida, muy rápida o lenta.

El modelado lingüístico difuso es, pues, un enfoque aproximado basado en la Teoría de Conjuntos Difusos. Este modelo representa los aspectos cualitativos como valores lingüísticos mediante el uso variables lingüísticas [32]. Una variable lingüística se caracteriza por un valor sintáctico o etiqueta que es una palabra o frase perteneciente a un conjunto de términos lingüísticos, y por un valor semántico o significado de dicha etiqueta que viene dado por un subconjunto difuso en un universo de discurso. Formalmente se define de la siguiente manera.

Definición 2.6: [32] Una variable lingüística está caracterizada por una 5-tupla (H, T(H),U,G,M), donde:

- H es el nombre de la variable;

- T(H) (o sólo T) simboliza el conjunto de términos lingüísticos de H, es decir, el conjunto de nombres de valores lingüísticos de H, donde cada valor es una variable difusa denotada genéricamente como X que toma valores en el universo de discurso;

- U el universo de discurso que está asociado con una variable base denominada u;

- G es una regla sintáctica (que normalmente toma forma de gramática) para generar los nombre de los valores de H;

- M es una regla semántica para asociar significado a cada elemento de H, que será un subconjunto difuso de U.

Por ejemplo, consideremos la variable lingüística H = velocidad, con U = [0, 125] y la variable base u ∈ U. El conjunto de términos asociados con la velocidad podría ser H(L) = {baja, media, alta} donde cada término en H(velocidad) es el nombre de un valor lingüístico de velocidad. El significado M(X) de una etiqueta H ∈ H(velocidad) se define como la restricción H(u) sobre la variable base u impuesta según el nombre de H. Por lo tanto M(X) es un conjunto difuso de U cuya función de pertenencia H(u) representa la semántica del nombre H. En la figura siguiente podemos ver una representación gráfica del ejemplo.

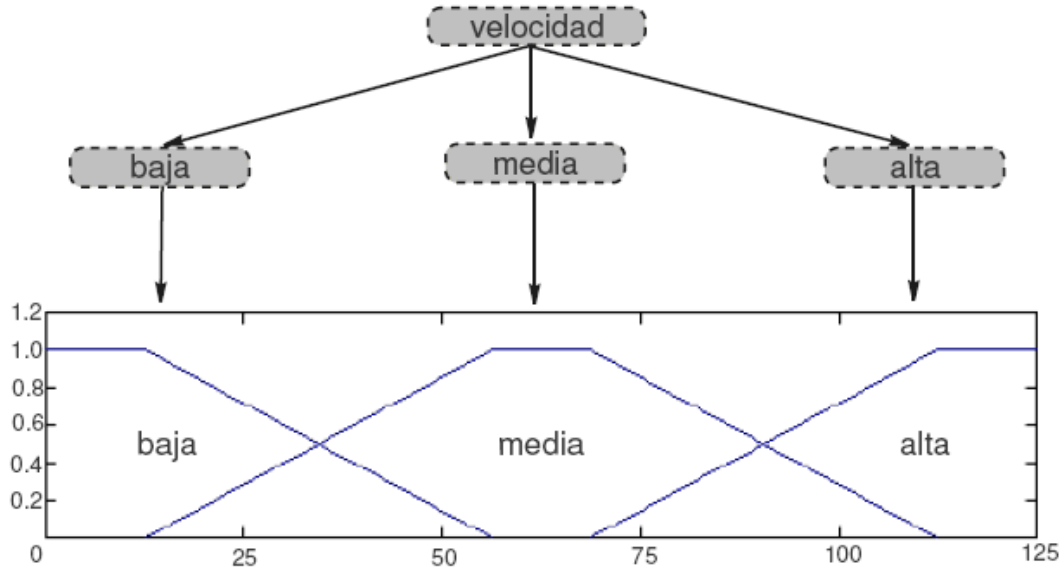

Figura 2.3: Ejemplo de una variable lingüística.

2.2.5.- Aplicación del Enfoque Lingüístico Difuso.

En cualquier ámbito en el que deseemos aplicar un enfoque lingüístico para la resolución de algún problema, debemos tomar dos decisiones:

- Modelo de representación: elección del conjunto de términos lingüísticos tanto su sintaxis como su semántica.

- Modelo computacional: definir el modelo computacional seleccionando los correspondientes operadores de comparación y de agregación.

Un aspecto importante que es necesario analizar con el fin de establecer la descripción de una variable lingüística es la granularidad de la incertidumbre [43], es decir, la cardinalidad del conjunto de términos lingüísticos usado para expresar y representar la información. La cardinalidad debe ser suficientemente baja como para no imponer una precisión excesiva en la información que se quiera expresar y suficientemente alta como para conseguir una discriminación de las valoraciones en un número limitado de grados. Habitualmente la cardinalidad usada en los modelos lingüísticos suele ser un valor impar, como 7 o 9, no superando las 11 o 13 etiquetas. El término medio representa una valoración de aproximadamente 0.5, y el resto de términos se sitúan simétricamente alrededor de este punto medio [43]. Estos valores clásicos de cardinalidad están basados en la línea de observación de Miller sobre la capacidad humana [44], en la que se indica que se pueden manejar razonablemente y recordar alrededor de 7 o 9 términos.

Una vez establecida la cardinalidad del conjunto de términos lingüísticos, hay que definir dicho conjunto, es decir, cuáles van a ser las etiquetas lingüísticas y su semántica asociada. Para asignar la semántica se pueden utilizar diferentes métodos:

1. Enfoque Basado en una Gramática Libre de Contexto.

Una posibilidad para generar el conjunto de términos lingüísticos consiste en utilizar una gramática libre de contexto *G*, donde el conjunto de términos pertenece al lenguaje generado por *G* [45, 46, 47, 48]. Una gramática generadora, *G*, es una 4-tupla (V_N, V_T, *I, P*), siendo V_N el conjunto de símbolos no terminales, V_T el conjunto de símbolos terminales, *I* el símbolo inicial y *P* el conjunto de reglas de producción. La elección de estos cuatro elementos determinará la cardinalidad y forma del conjunto de términos

lingüísticos. El lenguaje generado debería ser lo suficientemente grande para que pueda describir cualquier posible situación del problema.

De acuerdo con las observaciones de Miller [44], el lenguaje generado no tiene que ser infinito, sino más bien fácilmente comprensible.

Por ejemplo, entre los símbolos terminales y no terminales de *G* podemos encontrar términos primarios (por ejemplo: *alto, medio, bajo*), modificadores (por ejemplo: *no, mucho, muy, más o menos*), relaciones (por ejemplo: *mayor que, menor que*) y conectivos (por ejemplo: *y, o, pero*). Construyendo *I* como cualquier término primario, el conjunto de términos lingüísticos $T(H) = \{$*muy alto, alto, alto o medio, etc*$\}$ se genera usando *P*.

2. Enfoque Basado en Términos Primarios con una Estructura Ordenada.

Una alternativa para reducir la complejidad de definir una gramática consiste en dar directamente un conjunto de términos distribuidos sobre una escala con un orden total definido [52, 53, 54]. Por ejemplo, consideremos el siguiente conjunto de siete etiquetas $T(H) = \{N, MB, B, M, A, MA, P\}$.

$s_0 = N = Nada$
$s1 = MB = Muy\ Bajo$
$s_2 = B = Bajo$
$s_3 = M = Medio$
$s_4 = A = Alto$
$s_5 = MA = Muy\ Alto$
$s_6 = P = Perfecto$

donde $s_i < s_j$ si y sólo si $i < j$.

En estos casos, es necesario que el conjunto de términos lingüísticos satisfagan las siguientes condiciones adicionales [55].

1. Existe un operador de negación. Por ejemplo, $Neg(s_i) = s_j$, $j = g - i$ (g+1 es la cardinalidad de $T(H)$).

2. Tiene un operador de maximización: máx(s_i, s_j) = s_i *si* $s_i \geq s_j$.

3. Tiene un operador de minimización: mín(s_i, s_j) = *si si* $s_i \leq s_j$.

Capítulo 3

COMPUTACIÓN CON PALABRAS.

El uso del enfoque lingüístico implica la necesidad de realizar procesos de operar con palabras, denominados en inglés *Computing with Words* (CW). En este capítulo abordaremos los procesos de computación con palabras que surge como consecuencia del uso del enfoque lingüístico y nos centráremos en el modelo de representación de información lingüística basado en la 2-tuplas, definido como una mejora de los modelos computacionales clásicos.

3.1.- Introducción.

Desde los años 70 el concepto de CW ha sido ampliamente utilizado en diferentes áreas de investigación. CW es una metodología que permite realizar proceso de computación y razonamiento utilizando palabras pertenecientes a un lenguaje en lugar de números.

Dichos procesos se han llevado a cabo utilizando distintos modelos:

- *Modelo Basado en el Principio de Extensión* [25]. En él las operaciones se realizan utilizando la aritmética difusa [54] sobre los números difusos que soportan la semántica de las etiquetas lingüísticas. Por tanto, los resultados obtenidos son números difusos y para expresarlos mediante etiquetas lingüísticas hay que llevar a cabo procesos de aproximación lingüística.

- *Modelo Simbólico* [56]. Las operaciones se realizan sobre los índices de las etiquetas lingüísticas que son valores discretos. Debido a la naturaleza discreta de estos valores para obtener un resultado final en el conjunto de etiquetas hay que realizar operaciones de aproximación.

Los modelos mencionados están basados en el uso del enfoque lingüístico y presentan una seria limitación como es la pérdida de precisión al realizar las operaciones sobre las etiquetas a la hora de resolver problemas definidos en contextos complejos como los mostrados en [57]. Para solventar estos problemas se desarrolló un modelo de representación de información lingüística basado en la 2-tupla [58, 59] con un conjunto de operadores que permiten operar con precisión en conjuntos de etiquetas simétricamente distribuidas.

- *Modelo Basado en la Representación 2-tupla Lingüística* [58, 59]. Las operaciones se realizan sobre información lingüística expresada mediante 2-tuplas, permitiendo trabajar en un dominio de expresión lingüístico, pero tratándolo como un universo continuo. Lo cuál, es una importante ventaja sobre los modelos anteriores, ya que no hay que realizar operaciones de aproximación para expresar los resultados, ganándose precisión en los mismos.

A continuación, revisamos el modelo de representación basado en las 2-tuplas y su modelo de representación y computacional, ya que es la base del modelado para conjuntos de términos no balanceados, objeto de este proyecto.

3.2.- Modelado Lingüístico Difuso Basado en 2-tuplas.

El modelado lingüístico difuso basado en 2-tuplas [58, 59] es un tipo de modelado lingüístico difuso que nos permite reducir la pérdida de información que habitualmente se produce en los modelados lingüísticos difusos clásicos. Esta pérdida de información, que provoca una falta de precisión en los resultados, se debe al propio modelo de representación puesto que opera con valores discretos sobre un universo de discurso continuo. La principal ventaja del modelo computacional lingüístico basado en 2-tuplas, es que permite realizar procesos de computación con palabras de forma sencilla y además, sin pérdida de información, puesto que utiliza un modelo continuo de representación de la información.

Resumiendo lo anterior podemos decir que las diferentes ventajas de este modelo para representar la información lingüística con respecto a los modelos clásicos, son las siguientes.

1) El dominio lingüístico puede ser tratado como continuo, mientras que en los modelos clásicos son tratados como discreto.

2) El modelo computacional lingüístico basado en 2-tuplas lleva a cabo un proceso de "computación con palabras" sencillo y sin pérdida de información.

3) El resultado del proceso de "computación con palabras" está siempre expresado en el dominio lingüístico inicial.

Debido a estas ventajas, usaremos este modelo de representación lingüística para construir funciones de transformación entre los diferentes conjuntos de términos lingüísticos sin pérdida de información.

Para definirlo, tenemos que establecer el modelo de representación y el modelo computacional de las 2-tuplas para representar y agregar la información lingüística respectivamente.

3.2.1.- Modelo de Representación lingüística Basada en 2-tuplas.

Consideremos que S = $\{s_0, ..., s_g\}$ es un conjunto de términos lingüísticos con cardinalidad impar, donde el término intermedio representa una valoración de aproximadamente 0.5 y con el resto de términos del conjunto distribuidos simétricamente alrededor de ese punto intermedio. Asumimos que la semántica asociada con cada una de las etiquetas viene dada por medio de funciones de pertenencia triangulares, representadas por funciones triangulares (α, a, β) y consideramos todos los términos distribuidos sobre una escala sobre la que hay establecida una relación de orden total, es decir, $s_i \leq s_j \Leftrightarrow i \geq j$. En este contexto lingüístico difuso, si mediante un método simbólico de agregación de información lingüística [17, 39] obtenemos un valor $\beta \in [0, g]$, y $\beta \notin \{0, ..., g\}$, podemos usar una función de aproximación para expresar el resultado obtenido como un valor de S.

Definición 3.1: [58] Sea β el resultado de una agregación de los índices de un conjunto de etiquetas valoradas sobre un conjunto de términos lingüísticos S, es decir, el resultado de una operación de agregación simbólica, $\beta \in [0, g]$. Dados i = round(β) y $\alpha = \beta - i$ dos valores, tales que, $i \in [0, g]$ y $\alpha \in [-0.5, 0.5)$ entonces α es lo que denominamos **Traslación Simbólica**, que expresa la diferencia de información entre la información expresada por β y la etiqueta lingüística s_i más cercana a S.

El enfoque lingüístico difuso basado en 2-tuplas es desarrollado a partir del concepto de traslación simbólica, representando la información lingüística por medio de 2-tuplas (s_i, α_i), $s_i \in S$ y $\alpha_i \in [-0.5, 0.5)$:

s_i representa la etiqueta lingüística, y α_i es un valor numérico que expresa la traslación de β al índice de la etiqueta más cercana, i, en el conjunto de términos lingüísticos ($s_i \in S$).

Este modelo define un conjunto de funciones de transformación entre valores numéricos y 2-tuplas.

Definición 3.2: Sea $s_i \in S$ un término lingüístico, su representación mediante una 2-tupla se obtiene mediante la función θ:

$$\theta : [0, g] \longrightarrow \mathcal{S} \times [-0.5, 0.5)$$

$$\theta(s_i) = (s_i, 0) / s_i \in \mathcal{S}$$

Definición 3.3: [58] Siendo $S = \{s_0, ..., s_g\}$ un conjunto de términos lingüísticos y $\beta \in [0, g]$ un valor que representa el resultado de una operación de agregación simbólica, la 2-tupla que expresa la información equivalente a β se obtiene mediante la siguiente función:

$$\Delta : [0, g] \longrightarrow \mathcal{S} \times [-0.5, 0.5)$$

$$\Delta(\beta) = (s_i, \alpha), \; with \begin{cases} s_i & i = round(\beta) \\ \alpha = \beta - i & \alpha \in [-0.5, 0.5) \end{cases}$$

donde round(·) es el típico operador de redondeo, s_i es la etiqueta cuyo índice es el más cercano a β y α es el valor de la traslación simbólica.

Ejemplo 3.1: Representación 2-tupla.

Supongamos que trabajamos con el siguiente conjunto de términos lingüísticos S = {s0, s1, s2, s3, s4, s5, s6} y que como resultado de una operación de agregación simbólica se obtiene el valor β = 2.8. La representación de este valor mediante una 2-tupla lingüística, sería:

$$\Delta(\beta) = (s_3, -0.2)$$

Definición 3.4: [58] Sea S = {s0, ..., sg} un conjunto de términos lingüísticos y (si, α) una 2-tupla. Se define la función Δ^{-1} , tal que aplicada sobre una 2-tupla (si, α_i) devuelve su valor numérico $\beta \in$ [0, g].

$$\Delta^{-1} : \mathcal{S} \times [-0.5, 0.5) \longrightarrow [0, g]$$

$$\Delta^{-1}(s_i, \alpha) = i + \alpha = \beta$$

3.2.2.- Modelo Computacional Lingüístico de las 2-tuplas.

A continuación presentamos el modelo computacional que nos permite operar sobre la representación lingüística de las 2-tuplas, basándonos en los operadores de comparación, negación y agregación de 2-tuplas:

1. Operador de comparación de 2-tuplas. La comparación de información lingüística representada por medio de 2-tuplas se realiza de acuerdo a un orden lexicográfico normal. Consideremos dos 2-tuplas (s_k, α_1) y (s_l, α_2) que representan cálculos de información:

- si k < l entonces (s_k , α_1) es menor que (s_l , α_2).
- si k = l entonces
  - a) si $\alpha_1 = \alpha_2$ entonces (s_k , α_1) y (s_l , α_2) representan la misma información,
  - b) si $\alpha_1 < \alpha_2$ entonces (s_k , α_1)es menor que (s_l , α_2),
  - c) si $\alpha_1 > \alpha_2$ entonces (s_k , α_1) es mayor que (s_l , α_2)

2. Operador de negación de 2-tuplas. El operador de negación sobre una 2-tupla se define como:

$$Neg((s_i, \alpha)) = \Delta(g - (\Delta^{-1}(s_i, \alpha))).$$

siendo g + 1 la cardinalidad del conjunto de etiquetas S.

3. Operador de agregación de 2-tuplas. La agregación de información consiste en obtener un valor que resuma un conjunto de valores, por lo que el resultado de la agregación de un conjunto de 2-tuplas debe ser una 2-tupla. A lo largo de la literatura podemos encontrar numerosos operadores de agregación que nos permiten combinar la información de acuerdo a distintos criterios. Cualquiera de estos operadores ya existentes puede ser fácilmente extendido para trabajar con 2-tuplas, usando funciones Δ y Δ^{-1} que transforman valores numéricos en 2-tuplas y viceversa sin pérdida de información. Algunos ejemplos de estos operadores son los siguientes:

Definición 3.5: Media aritmética. Siendo x = {(r_1, α_1), . . . , (r_n, α_n)} un conjunto de 2-tuplas lingüísticas, la 2-tupla que simboliza la media aritmética, $\overline{x^e}$, se calcula de la siguiente forma:

$$\overline{x}^{e}[(r_1,\alpha_1),\ldots,(r_n,\alpha_n)] = \Delta(\sum_{i=1}^{n}\frac{1}{n}\Delta^{-1}(r_i,\alpha_i)) = \Delta(\frac{1}{n}\sum_{i=1}^{n}\beta_i).$$

Definición 3.6: Operador de media ponderada. Siendo x = {(r1, α 1), . . . , (rn, α n)} un conjunto de 2-tuplas lingüísticas y W = {w1, ...,wn} un vector numérico con sus pesos asociados, la 2-tupla que simboliza la media ponderada, $\overline{x^{\omega}}$, es:

$$\overline{x}^{w}[(r_1,\alpha_1),\ldots,(r_n,\alpha_n)] = \Delta(\frac{\sum_{i=1}^{n}\Delta^{-1}(r_i,\alpha_i)\cdot w_i}{\sum_{i=1}^{n}w_i}) = \Delta(\frac{\sum_{i=1}^{n}\beta_i\cdot w_i}{\sum_{i=1}^{n}w_i}).$$

Definición 3.7: Operador de media ponderada lingüística. Siendo x = {(r1, α 1), . . . , (rn, α n)} un conjunto de 2-tuplas y W = {(w1, α_1^w), ..., (wn, α_n^w)} sus pesos asociados representados mediante 2-tuplas lingüísticas, la 2-tupla que representa la media ponderada lingüística, $\overline{x_l^w}$, se calcula de la siguiente manera:

$$\overline{x}_l^w[((r_1,\alpha_1),(w_1,\alpha_1^w))\ldots((r_n,\alpha_n),(w_n,\alpha_n^w))] = \Delta(\frac{\sum_{i=1}^{n}\beta_i\cdot\beta_{W_i}}{\sum_{i=1}^{n}\beta_{W_i}}),$$

con $\beta_i = \Delta^{-1}(r_i,\alpha_i)$ y $\beta_{W_i} = \Delta^{-1}(w_i,\alpha_i^w)$.

Capítulo 4

INFORMACIÓN LINGÜÍSTICA NO BALANCEADA.

Según hemos estado viendo, ante cualquier problema que hace uso de información lingüística, el primer objetivo que hay que satisfacer es la elección de los términos lingüísticos con sus correspondientes semánticas, para así establecer el conjunto de etiquetas que se va a usar. A lo largo de la literatura podemos encontrar varias posibilidades distintas para la elección de los términos lingüísticos y sus semánticas:

1. Por un lado, podemos asumir que todos los términos del conjunto de etiquetas son igualmente informativos, es decir, están distribuidos simétricamente tal y como sucede en los modelados lingüísticos difusos que hemos estado viendo hasta ahora.

2. Por otro lado, podemos asumir que no todos los términos del conjunto de etiquetas son igualmente informativos, es decir, las etiquetas no están distribuidas simétricamente. En este caso, necesitamos un enfoque lingüístico difuso no balanceado [60, 61] para gestionar los conjuntos de términos lingüísticos con distintos niveles de discriminación a ambos lados del término medio.

En algunos casos, la información lingüística no balanceada aparece como consecuencia de la naturaleza de las variables lingüísticas que participan en el problema, y en otros, aparece en problemas que tratan con escalas para valorar preferencias donde los expertos necesitan valorar conjuntos con un número de términos mayor que en otro.

En este capítulo se presenta el desarrollo de una metodología para representar, manejar y lograr el proceso de computación con palabras con conjuntos de términos lingüísticos no balanceados sin perdida de información. Se define un modelo de representación lingüística no balanceada el cual asigna semánticas a los términos lingüísticos. Después, se esboza un proceso para asignar semántica a los términos lingüísticos asociados a un conjunto de términos lingüísticos no balanceado. Entonces, esas ideas son formalizadas mediante un algoritmo de representación semántica que representa cada término mediante una función de pertenencia con parámetros que es asignada usando la estructura de **jerarquía lingüística** que introduciremos antes de presentar el algoritmo. Dicho algoritmo de representación es el utilizado para diseñar la interfaz software objeto de este proyecto.

Comenzaremos revisando la estructura de la jerarquía lingüística y estableciendo las ideas básicas para representar conjuntos de términos no balanceados para a continuación presentar el modelo de representación lingüística no balanceada

4.1- Jerarquías lingüísticas.

4.1.1.- Definición.

Una Jerarquía lingüística es un conjunto de niveles, donde cada nivel es un conjunto de términos lingüísticos con una granularidad diferente del resto de niveles de la jerarquía [62]. A cada uno de los niveles de una jerarquía lingüística los vamos a denotar como l(t, n(t)), siendo t un número que indica el nivel de la jerarquía y n(t) la granularidad del conjunto de términos lingüísticos del nivel t.

Normalmente, las jerarquías lingüísticas trabajan con términos lingüísticos cuyas funciones de pertenencia son de forma triangular, simétricas y uniformemente distribuidas en el intervalo [0,1]. Además, los conjuntos de términos lingüísticos tienen una granularidad impar, con la etiqueta central indicando un valor de indiferencia.

Los niveles de una jerarquía lingüística están ordenados en función de su granularidad, es decir, que para dos niveles consecutivos t y t + 1, n(t + 1) > n(t). Por lo tanto, cada nivel t + 1 proporciona un refinamiento lingüístico con respecto al nivel anterior t.

Vamos a definir una jerarquía lingüística, LH, como la unión de todos los niveles t que la conforman:

$$LH = \bigcup_{t} l(t, n(t))$$

4.1.2.- Construcción de una jerarquía lingüística.

Para la construcción de LH debemos tener en mente que el orden jerárquico nos viene dado por el incremento de granularidad de los conjuntos de términos lingüísticos de cada nivel.

Partiendo de que $S_{n(t)} = \{ s_0^{n(t)}, \ldots, s_{n(t-1)}^{n(t)} \}$ sea el conjunto de términos lingüísticos definido para el nivel t con n(t) términos, la construcción de una jerarquía lingüística debe satisfacer las siguientes reglas básicas [63]:

1. Preservar todos los puntos modales previos de las funciones de pertenencia de cada uno de los términos lingüísticos de cada nivel con respecto a los del nivel siguiente.

2. Hacer que las transacciones entre dos niveles consecutivos sean suaves. El propósito es construir un nuevo conjunto de términos lingüísticos, $S_{n(t+1)}$, de forma que añadiremos un nuevo término lingüístico entre cada pareja de términos pertenecientes al

conjunto de términos del nivel anterior t. Para realizar esta inserción de nuevos términos, reduciremos el soporte de las etiquetas lingüísticas para dejar hueco entre ellas para la nueva etiqueta.

De forma genérica, podemos establecer que el conjunto de términos lingüísticos de nivel t+1, $S_{n(t+1)}$, puede ser obtenido a partir del nivel anterior t, $S_{n(t)}$, de la siguiente manera:

$$l(t, n(t)) \rightarrow l(t + 1, 2 \cdot n(t) - 1)$$

En la tabla 4.1 mostramos la granularidad necesaria en cada conjunto de términos lingüísticos de nivel t, dependiendo del valor n(t) definido en el primer nivel (para valores de 3 y 7 respectivamente).

	Nivel 1	Nivel 2	Nivel 3
l(t,n(t))	l(1,3)	l(2,5)	l(3,9)
l(t,n(t))	l(1,7)	l(2,13)	

Tabla 4.1: Granularidad en distintos niveles de una jerarquía

En la figura 4.1 se muestra un ejemplo gráfico de jerarquías lingüísticas. Se representa una jerarquía compuesta de 3 niveles, de 3, 5 y 9 etiquetas cada uno de ellos.

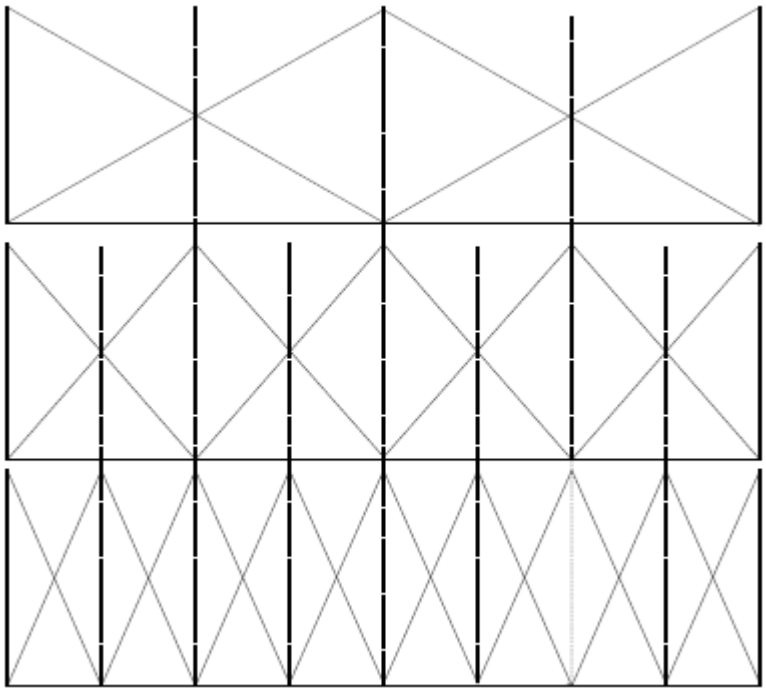

Figura 4.1: Jerarquía lingüística de 3, 5 y 9 niveles

4.1.3.- Funciones de transformación entre niveles de una jerarquía lingüística.

Las jerarquías lingüísticas son útiles para representar información lingüística multi-granular y por tanto permiten trabajar con información lingüística sin pérdida de información. Para conseguirlo, fue definida una familia de funciones de transformación entre etiquetas de diferentes niveles.

Definición 4.1: Sea LH = $\bigcup_t$ l(t, n(t)) una jerarquía lingüística cuyos conjuntos de términos lingüísticos son denotados como $S_{n(t)} = \{s_0^{n(t)}, \ldots, s_{n(t-1)}^{n(t)}\}$. La función de transformación de una etiqueta lingüística (representada mediante una 2-tupla) de un nivel t a una etiqueta de un nivel consecutivo t + c, con c ∈ {−1, 1}, se define como:

$$TF^t_{t+c} : l(t, n(t)) \longrightarrow l(t+c, n(t+c))$$

$$TF^t_{t+c}(s_i^{n(t)}, \alpha^{n(t)}) = \Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t+c)-1)}{n(t)-1})$$

Esta función de transformación fue generalizada para transformar términos lingüísticos entre cualquier nivel dentro de la jerarquía lingüística.

Definición 4.2: LH = $\bigcup_t$ l(t, n(t)) una jerarquía lingüística cuyos conjuntos de términos lingüísticos son denotados como $S_{n(t)} = \{ s_0^{n(t)}, \ldots, s_{n(t-1)}^{n(t)} \}$. La función de transformación recursiva entre una etiqueta lingüística (representada mediante una 2-tupla) perteneciente a un nivel t y una etiqueta perteneciente al nivel t´ = t + a, con a ∈ Z, se define como:

$$TF^t_{t'} : l(t, n(t)) \longrightarrow l(t', n(t'))$$

Si $|a| > 1$ entonces

$$TF^t_{t'}(s_i^{n(t)}, \alpha^{n(t)}) = TF_{t'}^{t+\frac{t-t'}{|t-t'|}}(TF^t_{t+\frac{t-t'}{|t-t'|}}(s_i^{n(t)}, \alpha^{n(t)}))$$

Si $|a| = 1$ entonces

$$TF^t_{t'}(s_i^{n(t)}, \alpha^{n(t)}) = TF^t_{t+\frac{t-t'}{|t-t'|}}(s_i^{n(t)}, \alpha^{n(t)})$$

Esta función recursiva se puede definir no recursivamente, de la siguiente manera:

$$TF_{t'}^{t} : l(t, n(t)) \longrightarrow l(t', n(t'))$$

$$TF_{t'}^{t}(s_i^{n(t)}, \alpha^{n(t)}) = \Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t') - 1)}{n(t) - 1})$$

Proposición 4.1: La función de transformación entre términos lingüísticos de niveles diferentes de una jerarquía lingüística es biyectiva;

$$TF_{t}^{t'}(TF_{t'}^{t}(s_i^{n(t)}, \alpha^{n(t)})) = (s_i^{n(t)}, \alpha^{n(t)})$$

Demostración:

$$TF_{t'}^{t}(s_i^{n(t)}, \alpha^{n(t)}) = \Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t') - 1)}{n(t) - 1}),$$

Por lo tanto,

$$TF_{t}^{t'}(\Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t') - 1)}{n(t) - 1})) =$$

$$= \Delta(\frac{\Delta^{-1}(\Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t') - 1)}{n(t) - 1})) \cdot (n(t) - 1)}{n(t') - 1}) =$$

$$= \Delta(\frac{\Delta^{-1}(s_i^{n(t)}, \alpha^{n(t)}) \cdot (n(t') - 1) \cdot (n(t) - 1)}{(n(t) - 1) \cdot (n(t') - 1)}) =$$

$$= (s_i^{n(t)}, \alpha^{n(t)})$$

Este resultado garantiza que la transformación entre niveles de una jerarquía lingüística se lleva a cabo sin pérdida de información.

Ejemplo 4.1: Aquí mostramos cómo actúan las funciones de transformación sobre la jerarquía lingüística, LH = $\bigcup_t$ l(1, 3), cuyos conjuntos de términos son

$$\begin{array}{ll} l(1,3) & \{s_0^3, s_1^3, s_2^3\} \\ l(2,5) & \{s_0^5, s_1^5, s_2^5, s_3^5, s_4^5\} \\ l(3,9) & \{s_0^9, s_1^9, s_2^9, s_3^9, s_4^9, s_5^9, s_6^9, s_7^9, s_8^9\} \end{array}$$

Las transformaciones entre términos de los diferentes niveles son llevadas a cabo como

$$TF_1^3(s_5^9, 0) = \Delta^{-1}(\frac{\Delta(s_5^9, 0) \cdot (3-1)}{9-1}) = \Delta^{-1}(1.25) = (s_1^3, .25)$$

$$TF_3^1(s_1^3, .25) = \Delta^{-1}(\frac{\Delta(s_1^3, .25) \cdot (8-1)}{3-1}) = \Delta^{-1}(5) = (s_5^9, .0)$$

$$TF_2^3(s_5^9, 0) = \Delta^{-1}(\frac{\Delta(s_5^9, 0) \cdot (5-1)}{9-1}) = \Delta^{-1}(2.5) = (s_5^5, -.5)$$

$$TF_1^2(s_3^5, -.5) = \Delta^{-1}(\frac{\Delta(s_3^5, -.5) \cdot (3-1)}{5-1}) = \Delta^{-1}(1.25) = (s_1^3, .25)$$

4.2.- Ideas básicas para la representación de información lingüística no balanceada.

El primer paso para manejar información lingüística no balanceada tal y como aparece en las figuras 4.2 y 4.3 usando el enfoque lingüístico difuso es obtener una representación semántica, debido al hecho de que nuestro objetivo es manejar y operar con esos términos sin pérdida de información. En esta sección, introduciremos las ideas básicas para representar, mediante funciones de pertenencia, la semántica de cada

término del conjunto de términos no balanceado usando la estructura jerarquía lingüística.

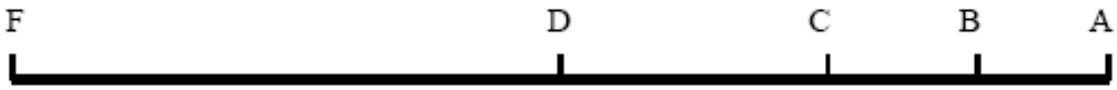

Figura 4.2: Sistema de calidad

Figura 4.3: Escala con más valores en el lado derecho del término medio

Para presentar las ideas básicas que utilizará el algoritmo de representación fijaremos una serie de conceptos y notación. Consideramos un conjunto de términos lingüísticos no balanceado, S, el cual tiene una etiqueta mínimo, una etiqueta máximo, una etiqueta central y el resto de etiquetas no están uniforme y simétricamente distribuidas alrededor de la etiqueta central, a ambos lados del conjunto. Para manejar este tipo de información proponemos dividir el conjunto de términos lingüísticos no balanceado S en tres subconjuntos, es decir, $S=S_L \cup S_C \cup S_R$:

- Conjunto del lado izquierdo, S_L, contiene todas las etiquetas menores que la etiqueta central.
- Conjunto central, S_C, simplemente contiene la etiqueta central.
- Conjunto del lado derecho, S_R, contiene todas las etiquetas mayores que la etiqueta central.

Por ejemplo los subconjuntos para el conjunto no balanceado de la figura 4.2 serían $S_L=\{F\}$, $S_C=\{D\}$ y $S_R=\{C, B, A\}$.

Se pretende representar las etiquetas de un conjunto de términos lingüísticos no balanceado S a través de los niveles de la jerarquía lingüística LH=$\bigcup_t l(t,n(t))$. Para hacer esto, analizamos cómo representar los tres subconjuntos S_L, S_C y S_R. Distinguimos las siguientes dos posibilidades:

A. Representación usando un nivel de la jerarquía lingüística.

Para representar los términos de $S_R(S_L)$ observamos si la siguiente condición es satisfecha:

$$\exists t \in LH, \frac{n(t)-1}{2} = \#(S_R), ó, \frac{n(t)-1}{2} = \#(S_L)$$

Siendo #(S_R), #(S_L) la cardinalidad de S_R y S_L respectivamente.

Cuando la condición mostrada anteriormente es satisfecha, es decir, existe un nivel en LH cuya granularidad del subconjunto es la misma que la del subconjunto lateral de S. Entonces el procedimiento básico de representación de las etiquetas del subconjunto lateral $S_R(S_L)$ es el siguiente:

1) Asignar las etiquetas desde $S_R^{n(t)}$ ($S_L^{n(t)}$) a $S_R(S_L)$, es decir, $S_R \leftarrow S_R^{n(t)}$ ($S_L \leftarrow S_L^{n(t)}$).

2) El subconjunto central $S_C=\{s_C\}$ es asignado dependiendo del conjunto lateral representado S_L o S_R. Cuando estamos tratando con el conjunto lateral S_R las semánticas asignadas, s_C, estarán por debajo de la etiqueta central $s_C^{n(t)} \in S^{n(t)}$, es decir, $\underline{s_C} \leftarrow \underline{s_C^{n(t)}}$ mientras que si estamos tratando con el conjunto lateral S_L las semánticas asignadas a, s_C, estarán por encima, es decir, $\overline{s_C} \leftarrow \overline{s_C^{n(t)}}$.

B. Representación usando dos niveles.

Si la condición del apartado anterior no se cumple la representación de $S_R(S_L)$ depende de la distribución de S. En tal caso, describimos la distribución de S mediante un conjunto de cinco valores:

$$\{((\#S_L), \text{density}\, S_L), \#(S_C), (\#(S_R), \text{density}\, S_R)\}$$

Siendo $\text{density}\, S_L$ y $\text{density}\, S_R$ variables simbólicas expresadas en el conjunto {mitad, extremo}, el cual indica si la mayor granularidad del conjunto lateral derecha(izquierda) de S está concentrada cerca de la etiqueta central o cerca de la etiqueta máximo(mínimo). Esta descripción para el conjunto de términos de la figura 4.2 sería S={F,D,C,B,A}, es {(1,extremo), 1, (3, extremo)}. Asumiendo esta descripción de S, el proceso para representar el conjunto lateral $S_R(S_L)$ es:

a) Seleccionar los niveles de jerárquicos para asignar las semánticas.
b) Proceso de representación del conjunto lateral.
c) Representación del conjunto central.

Observación 4.1: Para simplificar la explicación nos centraremos en S_R, aunque el proceso es análogo para S_L.

1) Selección de los niveles jerárquicos para asignar las semánticas:

Dado que la ecuación del apartado anterior no es satisfecha, entonces buscamos dos niveles t y t+1 en LH, tales que,

$$\frac{n(t)-1}{2} < \#(S_R) < \frac{n(t+1)-1}{2}$$

Entonces los términos de S_R estarán representados mediante los subconjuntos laterales derechos del nivel t y t+1, llamados conjuntos asignables y denotados como $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$, respectivamente.

Observación 4.2: proponemos un modelo de construcción semántica usando dos niveles de la jerarquía lingüística, aunque el número de niveles para modelar la semántica de un conjunto de términos lingüísticos podría ser mayor. Esto implica, sin embargo, una mayor complejidad en el modelo de construcción semántica, y el resultado podría no apreciar una notable mejora.

Los conjuntos de etiquetas asignables contienen las semánticas que pueden ser asignadas a los términos de S_R y ellos variarán a lo largo del proceso de representación, debido al hecho de que la misma etiqueta de los conjuntos asignables no puede ser asignada dos veces. Inicialmente, esos conjuntos asignables están compuesto de $AS_R^{n(t)} = S_R^{n(t)} = \{ s_{\frac{n(t)-1}{2}+1}^{n(t)}, \ldots, s_{n(t)-1}^{n(t)} \}$, y $S_R^{n(t+1)} = S_R^{n(t+1)} = \{ s_{\frac{n(t+1)-1}{2}+1}^{n(t+1)}, \ldots, s_{n(t)-1}^{n(t)} \}$. En el proceso de representación las cardinalidades de $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$ decrementarán después de cada asignación semántica y sólo podremos asegurar que $AS_R^{n(t)} \subseteq S_R^{n(t)}$ y $AS_R^{n(t+1)} \subseteq S_R^{n(t+1)}$.

Ejemplo 4.2: Por ejemplo, si usamos la LH mostrada en la figura 4.1 siguiente para representar las etiquetas del conjunto de términos, S={F,D,C,B,A} mostrada al principio de esta sección, entonces los conjuntos asignables para el conjunto lateral derecho S_R={C,B,A} son los siguientes:

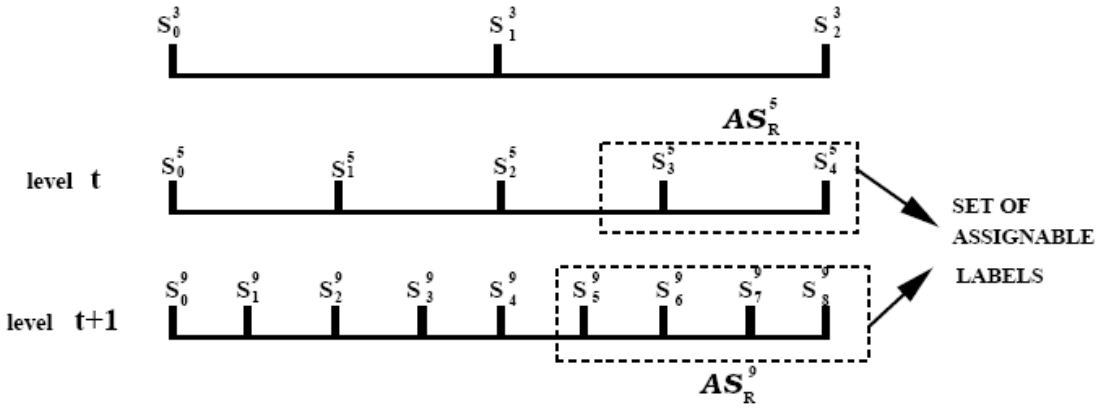

Figura 4.4: Conjunto de etiquetas asignables

Una vez que los conjuntos asignables iniciales han sido seleccionados es necesario decidir cómo usar los conjuntos $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$ para representar las etiquetas de S_R. Esta decisión depende de la distribución de S_R, es decir, del valor de la variable density S_R.

La idea consiste en representar el lado de S_R con el más alto nivel de granularidad desde $AS_R^{n(t+1)}$ y el lado de S_R con el más bajo nivel de granularidad desde $AS_R^{n(t)}$. Por consiguiente, distinguimos en S_R dos subconjuntos de etiquetas, $S_R = S_{RC} \cup S_{RE}$, siendo S_{RC} el subconjunto que contiene las etiquetas cercanas a la etiqueta central de S y S_{RE} el subconjunto que contiene las etiquetas cercanas a la máxima etiqueta de S, entonces, la regla de decisión para representar las etiquetas de S_R desde los conjuntos asignables $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$ se muestran en la siguiente tabla.

SI *(densidad$_{S_{RE}}$ = "Extrema")*

ENTONCES

S_{RE} es representado en $AS_R^{n(t+1)}$ $S_{RE} \subset AS_R^{n(t+1)}$

S_{RC} es representado en $AS_R^{n(t)}$ $S_{RC} \subset AS_R^{n(t)}$

SINO

S_{RE} es representado en $AS_R^{n(t)}$ $S_{RE} \subset AS_R^{n(t)}$

S_{RC} es representado en $AS_R^{n(t+1)}$ $S_{RC} \subset AS_R^{n(t+1)}$

Tabla 4.2: Regla de decisión para representar SR desde los conjuntos asignables

2) Proceso de representación de un conjunto lateral: el proceso de representación asignará semánticas a todas las etiquetas del conjunto lateral S_R mediante un proceso iterativo usando ambos conjuntos asignables de etiquetas $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$. Para controlar la actualización de los conjuntos asignables después de cada asignación semántica durante este proceso una regla representación, R^{Rep}, será definida. Para definirla tenemos en cuenta como es construida LH, es decir, cada etiqueta del nivel t, $s_j^{n(t)} \in AS_R^{n(t)}$, $s_j^{n(t)} \neq s_C^{n(t)}$, tiene asociadas dos etiquetas del nivel t+1, $s_{2\bullet j}^{n(t+1)} \in AS_R^{n(t+1)}$ y $s_{2\bullet j-1}^{n(t+1)} \in AS_R^{n(t+1)}$.

El proceso de representación comienza usando $AS_R^{n(t+1)}$ para representar las etiquetas de S_R situadas en el lado con la mayor densidad y una vez que la primera etiqueta ha sido representada la regla de representación prepara los conjuntos asignables para las siguientes etiquetas y guarda la asignaciones semánticas. Esta regla de representación actúa como sigue:

R^{Rep} : cuando una etiqueta $s_i^R \in S_R$ es representada mediante una etiqueta $s_k^{n(t+1)} \in AS_R^{n(t+1)}$, k=2*j o k=2*j -1, entonces $s_k^{n(t+1)}$ es eliminada de $AS_R^{n(t+1)}$ y su etiqueta asociada $s_j^{n(t)} \in AS_R^{n(t)}$ es también eliminada si no lo ha sido ya.

En ese momento, el proceso iterativo para representar S_R consiste en la asignación de semántica a sus términos desde el conjunto asignable, $AS_R^{n(t+1)}$, y la aplicación de la regla de representación R^{Rep} hasta que el número de etiquetas no representadas de S_R coincida con el número de etiquetas asignables en $AS_R^{n(t)}$. En este momento, las etiquetas no representadas son asignadas directamente desde $AS_R^{n(t)}$.

Ejemplo 4.3: Asumiendo el caso mostrado en el ejemplo 1 con S={F,D,C,B,A}, $densidad_R$ ={extremo} y los conjuntos asignables mostrados en la figura 4.4:

S_R={C,B,A} S_{RE}={B,A} S_{RC}={C}

$AS_R^5 = \{ s_3^5,\ s_4^5 \}$

$AS_R^9 = \{ s_5^9,\ s_6^9,\ s_7^9,\ s_8^9 \}$

$S_{RE} \subset AS_R^9$ y $S_{RC} \subset AS_R^5$

Las asociaciones entre las etiquetas de ambos niveles son:

- La etiqueta $s_4^5 \in AS_R^5$ es asociada con las etiquetas $s_7^9,\ s_8^9 \in AS_R^9$.
- La etiqueta $s_3^5 \in AS_R^5$ es asociada con las etiquetas $s_5^9,\ s_6^9 \in AS_R^9$.

El proceso de representación para este ejemplo comienza usando la etiqueta $s_8^9 \in AS_R^9$ para representar las valoraciones lingüísticas $A \in S_{RE}$ entonces la regla de representación elimina la etiqueta s_8^9 de AS_R^9 y s_4^5 desde AS_R^5.

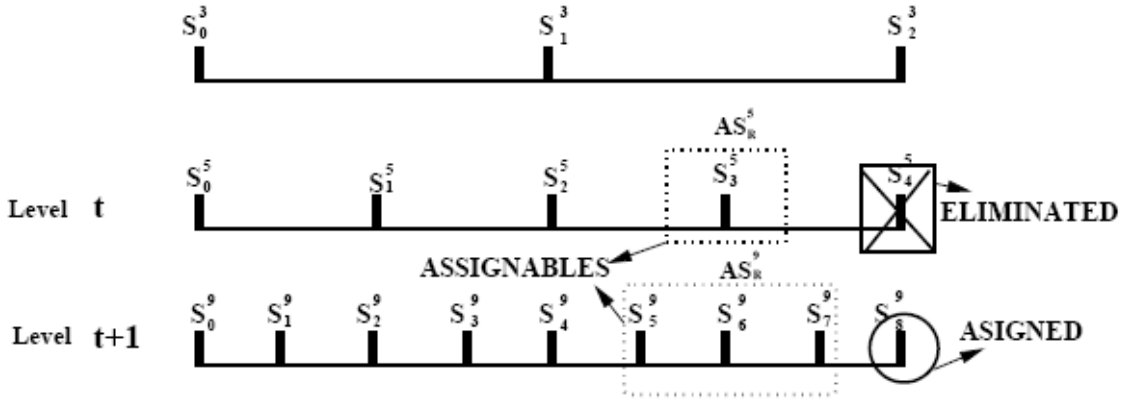

Figura 4.5: R^{Rep} si Densidad S_R ="extrema" (primera asignación)

El proceso va representando los términos lingüísticos $B \in S_{RE}$ con la etiqueta $s_7^9 \in AS_R^9$, y la regla no elimina ninguna etiqueta de AS_R^5 porque s_4^5 ha sido ya eliminada.

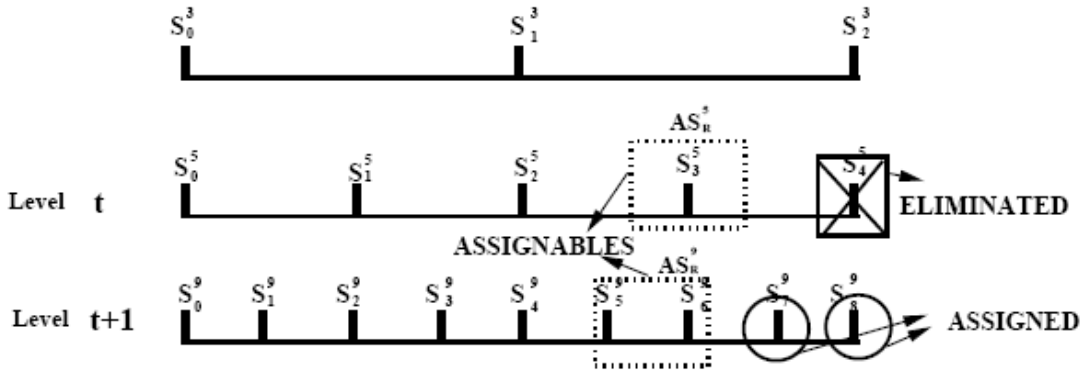

Figura 4.6: R^{Rep} si Densidad S_R =”extrema” (primera asignación)

Si seguimos con el proceso iterativo, la última etiqueta representada del conjunto del lado derecho es la valoración lingüística C, entonces tal término es representado en AS_R^5 mediante una sola etiqueta, s_3^5. El proceso de representación para S_R ya ha finalizado y los términos lingüísticos no balanceados A, B, C están representados mediante las etiquetas asociadas a LH, s_8^9, s_7^9 y s_3^5, respectivamente.

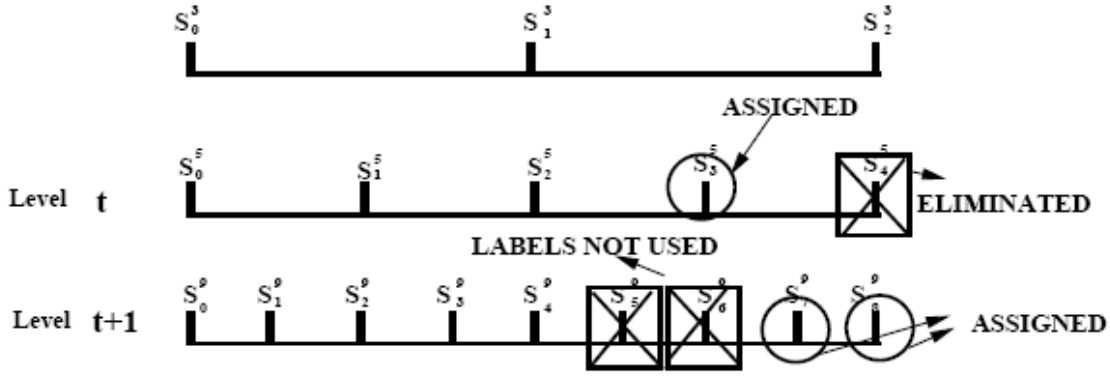

Figura 4.7: Representación de las etiquetas A, B y C de la figura 1

Ejemplo 4.4: Suponiendo el mismo caso que en el ejemplo 4.3 pero con un conjunto de cinco términos lingüísticos no balanceado con densidad$_R$ ={mitad}, entonces:

S_R={C,B,A} S_{RE}={A} S_{RC}={C, B}

$AS_R^5 = \{ s_3^5,\ s_4^5 \}$

$AS_R^9 = \{ s_5^9,\ s_6^9,\ s_7^9,\ s_8^9 \}$

$S_{RE} \subset AS_R^5$ y $S_{RC} \subset AS_R^9$

La regla de representación actuaría como se muestra en la figura siguiente.

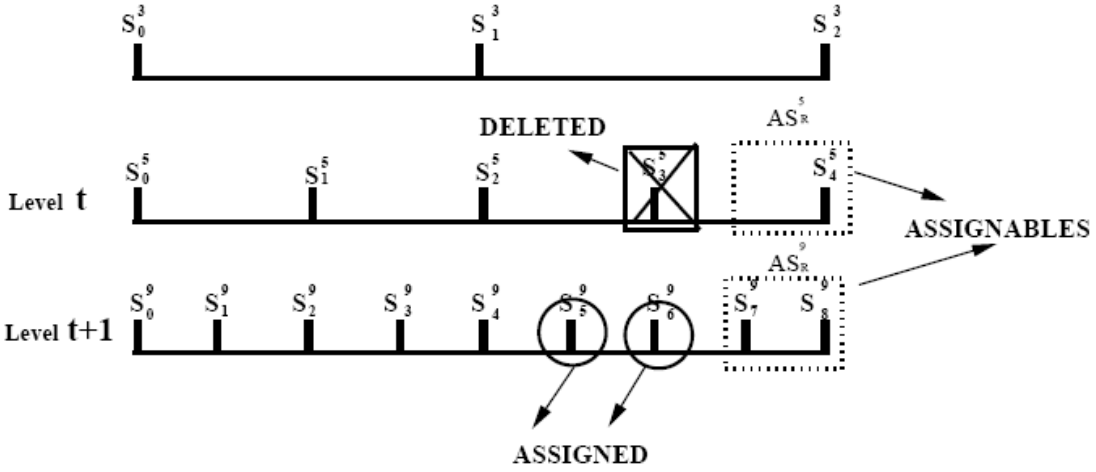

Figura 4.8: R^{Rep} si Densidad S_R ="central" (después de dos asignaciones)

3) Mejorando el proceso de representación: El proceso de representación iterativo necesita varias rondas para representar las etiquetas de S_{RE} y S_{RC} de los conjuntos asignables $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$ según la regla de decisión (Tabla 4.2).

   Está claro que este proceso puede ser mejorado y simplificado si podemos calcular a priori el número de etiquetas de $AS_R^{n(t)}$ y $AS_R^{n(t+1)}$, nombradas como lab_t y lab_{t+1}, respectivamente las cuales serán usadas para representar las etiquetas de S_{RE} y S_{RC}. En tal caso, todas las etiquetas de S_R pueden ser representadas en una única ronda porque conocemos cuantas etiquetas y cuales

de ellas representarán las etiquetas de S_{RE} y S_{RC} desde los conjuntos asignables iniciales.

Está visto que $lab_t + lab_{t+1} = \#(S_R)$. La siguiente proposición nos permite una manera de calcular ambos valores.

Proposición 4.2: El número de etiquetas utilizadas de $AS_R^{n(t)}$, lab_t, para representar las etiquetas de S_{RE} se calcula como: $lab_t = (n(t+1)-1)/2 - \#(S_R)$.

Demostración: Por un lado, conocemos que LH esta construida de tal manera que una etiqueta de un conjunto lateral en el nivel t tiene asociada dos etiquetas del nivel t+1. Entonces lab_t etiquetas del nivel t tienen asociadas $(2*lab_t)$ etiquetas del nivel t+1. Por otro lado, siguiendo la regla de representación R^{Rep} sabemos que cuando dos etiquetas del nivel t+1 son usadas en el proceso de representación entonces sus etiquetas asociadas del nivel t son eliminadas. Seguidamente, tenemos $lab_{t+1} = (n(t+1)-1)/2 - (2*lab_t)$, y como $lab_{t+1} = \#(S_R) - lab_t$ entonces satisface $(n(t+1)-1)/2 - (2*lab_t) = \#(S_R) - lab_t$, y consecuentemente $lab_t = (n(t+1)-1)/2 - \#(S_R)$.

Ejemplo 4.5: Usando el caso del ejemplo 2 con S={F,D,C,B,A}, $densidad_R$ ={extremo} y los conjuntos asignables mostrados en la figura 4.4:

S_R={C,B,A} S_{RE}={B,A} S_{RC}={C}

$AS_R^5 = \{ s_3^5, s_4^5 \}$

$AS_R^9 = \{ s_5^9, s_6^9, s_7^9, s_8^9 \}$

$S_{RE} \subset AS_R^9$ y $S_{RC} \subset AS_R^5$

Podemos encontrar a priori la cardinalidad y etiquetas de S_{RE} = {A,B} y S_{RC}={C}porque, $lab_2 = (9-1)/2 - 3 = 1$, y $lab_3 = 3 - 1 = 2$. Por ello sabemos que {A,B} será representado con semánticas de AS_R^9 y {C} de AS_R^5 según la regla de decisión.

4) Representado el conjunto central: Finalmente, tenemos que establecer la representación asociada con la etiqueta central de S, es decir, la representación de $S_C = \{s_C\}$. En nuestro caso, dado que estamos representando S_R, establecemos la representación de $\underline{s_C}$. La representación de $\underline{s_C}$ depende del valor de la variable densidad S_R. Entonces, se representará según la tabla 4.3.

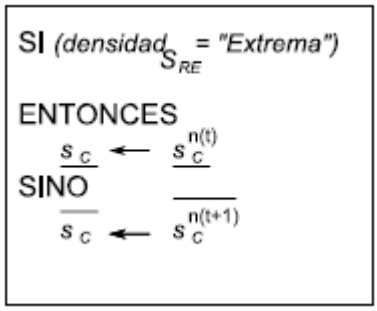

Tabla 4.3: Representación de $\underline{s_C}$

4.3.- Modelo de representación lingüística no balanceado.

En esta sección formalizamos las ideas introducidas en la anterior sección. Desarrollamos un algoritmo de representación semántica para conjuntos de términos no balanceados que proporciona una semántica a los términos lingüísticos asociados a un conjunto de términos lingüísticos no balanceado. Primero, definimos varias funciones de representación que controlan las asignaciones semánticas de cada término lingüístico de según varios parámetros. Después introducimos algunos pasos necesarios adicionales que cubre algunos vacíos en la actual representación para garantizar que la representación del conjunto de términos no balanceado soportará el proceso de computación con palabras sin pérdida de información. Y finalmente, presentamos el algoritmo de representación semántica formal que asigna las semánticas al conjunto de términos lingüísticos no balanceado.

4.3.1.- Funciones de representación.

Según las ideas básicas del proceso de representación, las semánticas asignadas a cada término dependen de la densidad del conjunto lateral, densidad $S_R \in$ {extremo, centro} y del nivel de LH usado para asignar las semánticas, t o t+1. Por lo tanto, podemos deducir que necesitamos diferentes funciones de representación de acuerdo con los parámetros mencionados más arriba. Aquí presentamos cuatro diferentes funciones de representación que cubren las diferentes posibilidades y sus papeles en relación al valor de sus parámetros.

1) *Función de representación de S_R en el nivel t+1 de LH*: $asignar_{t+1}^{R}$(densidad): Esta función lleva a cabo la representación de términos lingüísticos no balanceados en el conjunto lateral derecho S_R desde el conjunto asignable $AS_R^{n(t+1)}$ del nivel t+1 de LH. Esto actúa dependiendo de el valor del parámetro **densidad** perteneciente a {centro, extremo}:

   a. Si densidad=centro entonces lab_{t+1} etiquetas contenidas en $S_{RC} \in S_R$ son representadas mediante las lab_{t+1} etiquetas más pequeñas contenidas en $AS_R^{n(t+1)}$ siguiendo la regla de representación R^{Rep} y empezando por la etiqueta que sigue a la etiqueta central, esto es $s_{C+1}^{n(t+1)}$.

   b. Si densidad=extremo entonces lab_{t+1} etiquetas contenidas en $S_{RE} \in S_R$ son representadas mediante las lab_{t+1} etiquetas más grandes contenidas en $AS_R^{n(t+1)}$ siguiendo la regla de representación R^{Rep} y empezando por la etiqueta más alta, esto es $s_{n(t+1)-1}^{n(t+1)}$.

2) *Función de representación de S_R en el nivel t de una LH*: $asignar_t^R$(densidad): Esta función lleva a cabo la representación de etiquetas lingüísticas no balanceadas de S_R en el subconjunto de etiquetas asignable $AS_R^{n(t)}$ del nivel t de LH. De igual manera, actúa dependiendo del valor del parámetro **densidad** $\in$ {centro, extremo}:

a. Si densidad=centro entonces las lab_t etiquetas contenidas en $S_{RE} \in S_R$ son representadas mediante las lab_t etiquetas más altas contenidas en $AS_R^{n(t)}$ empezando por la etiqueta $s_{C+1+\delta}^{n(t)}$, siendo δ=round(lab_{t+1}/2).

b. Si densidad=extremo entonces las lab_t etiquetas contenidas en $S_{RC} \in S_R$ son representadas mediante las lab_t etiquetas más pequeñas contenidas en $AS_R^{n(t)}$ empezando por la etiqueta $s_{n(t)-1-\delta}^{n(t)}$.

3) *Función de representación de S_L en el nivel t+1 de LH*: $asignar_{t+1}^{L}$ (densidad): Esta función lleva a cabo la representación de etiquetas lingüísticas no balanceadas de S_L en el subconjunto de etiquetas asignables $AS_L^{n(t+1)}$ del nivel t+1 de LH. Esto actúa dependiendo del valor del parámetro **densidad** ∈ {centro, extremo}:

a. Si densidad=centro entonces las lab_{t+1} etiquetas contenidas en $S_{LC} \in S_L$ son representadas mediante las lab_{t+1} etiquetas más grandes contenidas en $AS_L^{n(t+1)}$ siguiendo la regla de representación R^{Rep} y empezando por la etiqueta anterior a la etiqueta central, esto es $s_{C-1}^{n(t+1)}$.
b. Si densidad=extremo entonces las lab_{t+1} etiquetas contenidas en $S_{LE} \in S_L$ son representadas mediante las lab_{t+1} etiquetas más pequeñas contenidas en $AS_L^{n(t+1)}$ siguiendo la regla de representación R^{Rep} y empezando por la etiqueta más pequeña, esto es $s_0^{n(t+1)}$.

4) *Función de representación de S_L en el nivel t de una LH*: $asignar_t^L$ (densidad): Esta función lleva a cabo la representación de etiquetas lingüísticas no balanceadas de S_L en el subconjunto de etiquetas asignables $AS_L^{n(t)}$ del nivel t de LH. También actúa dependiendo del valor del parámetro **densidad** ∈ {centro, extremo}:

a. Si densidad=centro entonces las lab_t etiquetas contenidas en $S_{LE} \in S_L$ son representadas mediante las lab_t etiquetas más pequeñas contenidas en $AS_L^{n(t)}$ empezando por la etiqueta $s_{C-1-\delta}^{n(t)}$, siendo δ=round(lab_{t+1}/2).

b. Si densidad=extremo entonces las lab_t etiquetas contenidas en $S_{LC} \in S_L$ son representadas mediante las lab_t etiquetas más pequeñas contenidas en $AS_L^{n(t)}$ empezando por la etiqueta $s_{\delta}^{n(t)}$.

4.3.2.- Establecimiento de puentes entre niveles.

En [65] estuvimos estudiando varias condiciones que deben ser satisfechas por la semánticas del conjunto de términos lingüísticos S a fin de garantizar que el proceso de computación de palabras usando el modelo computacional de las 2-tuplas lingüísticas es llevado a cabo de una manera precisa. Tales condiciones son:

1) S es una partición difusa. De acuerdo con Ruspini [44], una familia finita $\{s_0, \ldots, s_g\}$de subconjuntos difusos en el universo X (en nuestro caso X=[0,1]) es denominado una partición difusa si $\sum_{i=0}^{g} \mu_{si}(x) = 1, \ \forall x \in X$.
2) Las funciones de pertenencia de sus términos son triangulares, esto es, $s_i=(a_i, b_i, c_i)$. Entonces $\exists! x / \mu_{si}(x) = 1, s_i \in S$.

Nuestro objetivo es representar los conjuntos de términos no balanceados tal que podamos operar con ellos de una manera precisa, pero siguiendo la idea básica expuesta para representar un conjunto de términos lingüístico no balanceado, S, y las anteriores funciones de representación. La semántica obtenida para los términos de S satisface la segunda condición, pero no la primera. Por lo tanto, proponemos algunos pasos

adicionales que nuestro algoritmo de representación debe llevar a cabo para representar S como una partición difusa.

Podemos observar en el ejemplo 4.3 donde se usa la propuesta inicial para representar la información no balanceada del conjunto lateral derecho S_R =(C, B, A) mostrado en la figura 4.2. Las etiquetas de LH implicadas en su representación son $\{s_3^5\}$ y $\{s_7^9, s_8^9\}$, respectivamente (gráficamente mirar la figura 4.9).

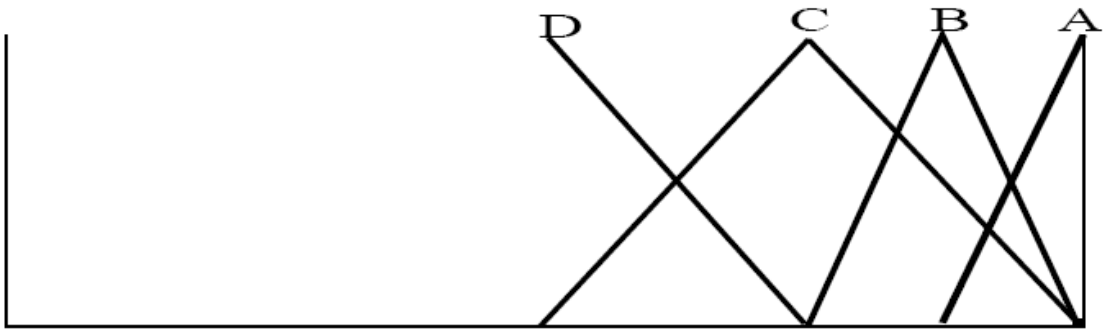

Figura 4.9: Representación inicial de S_R

En tal situación la semántica asociada con S_R no puede formar una partición difusa a causa de que la representación del lado bajo de la etiqueta C. Podemos ver que la etiqueta C representa el salto entre los niveles t y t+1, denotándolo como s_{jump}, y siempre que ocurra este salto aparece el mismo problema en lo que se refiere a la partición difusa. Por lo tanto en tales saltos tenemos que tender un puente entre términos no balanceados, de una manera similar a la etiqueta central s_C, para obtener una partición difusa.

La representación dependerá de la densidad del conjunto lateral (mirar tabla 4.4).

SI *(densidad*$_{S_R}$ *= "Extrema")*

ENTONCES

$$\overline{s_{jump}} \leftarrow \overline{s_i^{n(t)}}, \quad \underline{s_{jump}} \leftarrow \underline{s_k^{n(t+1)}}, k = 2 * i$$

SINO

$$\underline{s_{jump}} \leftarrow \underline{s_i^{n(t)}}, \quad \overline{s_{jump}} \leftarrow \overline{s_k^{n(t+1)}}, k = 2 * i$$

Tabla 4.4: Puenteo de etiquetas entre niveles

Por lo tanto, para representar S_R mostrado en la figura 10 como una partición difusa puentearemos el salto representando C con la siguiente semántica: $\underline{C} = \underline{s_{jump}} \leftarrow \underline{s_6^9}$, $\overline{C} = \overline{s_{jump}} \leftarrow \overline{s_3^5}$. La figura 4.10 muestra la nueva representación para S_R.

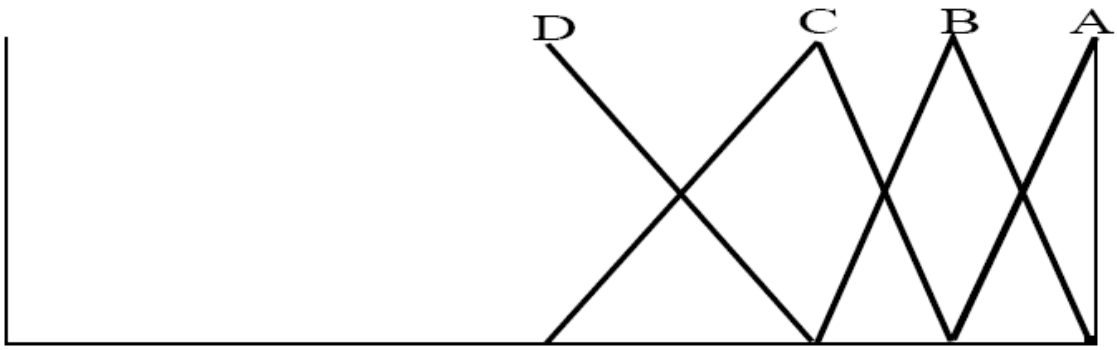

Figura 4.10: Representación de S_R como partición lógica

4.3.3.- Salida: Semántica e información adicional.

El algoritmo de representación proporciona la semántica para el conjunto de términos lingüísticos no balanceado y la siguiente información adicional, a fin de controlar y manejar el modelado de información lingüística en un conjunto de términos lingüísticos no balanceado S:

a. Una representación semántica jerárquica, LH(S): Para un conjunto de términos lingüísticos no balanceado S = {s_i, i=0, …., g} obtenemos su representación en la LH, esto es LH(S)=$\{ s_{I(i)}^{G(i)}, i = 0,...., g\}$, tal que $\forall s_i \in S \; \exists l(t, n(t)) \in LH$ que contiene una etiqueta $s_k^{n(t)} \in S^{n(t)}$, de tal manera que I(i)=k y G(i)=n(t), siendo I y G funciones que asignan a cada etiqueta no balanceada $s_i \in$ S el índice de la etiqueta que la representa en LH y la granularidad del conjunto de etiquetas de LH en la cuál está representada, respectivamente. Esta representación será generada por las funciones de representación.

b. Un punto de unión, Brid: definimos una función booleana Brid: S $\rightarrow$ {true, false} para aquellas $s_i \in$ S que son considerados s_{jump} , esto es, etiquetas cuyas representaciones semánticas son conseguidas de dos niveles de LH (incluyendo la etiqueta central s_C).

c. Ordenación de conjuntos: Los cinco subconjuntos del conjunto de términos lingüísticos no balanceado S: S_{LE}, S_{LC}, S_C, S_{RC}, S_{ER} son ordenados en orden incremental.

d. Conjunto de niveles de LH, T_{LH}: Contiene aquellos niveles usados en la representación de S: T_{LH} = { T_{LE}, T_{LC}, T_C, T_{RE}}, donde t_{LE} es el nivel de LH usado para representar S_{LE}, t_{LC} es el nivel de LH usado para representar S_{LC}, y sucesivamente. Debemos puntualizar que si $\exists l(t,(n(t)),(n(t)\text{-}1)/2 = \#(S_R)$, entonces t_{RC}= t_{RE}=t y S_{RC}= SER= S_R. De igual manera ocurre para S_L.

4.3.4.- Algoritmo.

Usando las ideas iniciales, las funciones de representación y el proceso de puenteo, presentamos el algoritmo de representación semántica para conjuntos de términos lingüísticos no balanceados que representa los términos no balanceados mediante las funciones de pertenencia triangulares usando las jerarquías lingüísticas.

Observación 4.3: Aquellos pasos del algoritmo señalados con (**) han sido incluidos para llevar a cabo el proceso de puenteo. El término Brid asignado verdadero corresponde a las etiquetas asignadas en la misma línea.

INPUTS: $\{(\#(\mathcal{S}_L), density_{\mathcal{S}_L}), \#(\mathcal{S}_C), (\#(\mathcal{S}_R), density_{S_R})\}$,
$\mathcal{S} = \{s_0, ..., s_n\}$,
$LH = \bigcup_t l(t, n(t))$ **BEGIN**

IF $\exists l(t, n(t)), \frac{n(t)-1}{2} = \#(\mathcal{S}_R)$ **THEN**
To represent $\mathcal{S}_R$ by means of $S_R^{n(t)}$: $\mathcal{S}_R \leftarrow S_R^{n(t)}$
To represent $\underline{s_C} \in \mathcal{S}$ as $\underline{s_C^{n(t)}}$: $\underline{s_C} \leftarrow \underline{s_C^{n(t)}}$
$t_{RE} = t_{RC} = t$
ELSE
To look for t and $t+1$, such that:
$\frac{n(t)-1}{2} < \#(\mathcal{S}_R) < \frac{n(t+1)-1}{2}$
$lab_t = \frac{n(t+1)}{2} - \#(\mathcal{S}_R)$
$lab_{t+1} = \#(\mathcal{S}_R) - lab_t$
$assign_{t+1}^R(density_{\mathcal{S}_R})$
$assign_t^R(density_{\mathcal{S}_R})$
IF $density_{\mathcal{S}_R}$= "extreme"
THEN
$\underline{s_{C+lab_t}} \leftarrow \underline{s_{n(t+1)-1-lab_{t+1}}^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$\underline{s_C} \leftarrow \underline{s_C^{n(t)}}$; $Brid \leftarrow "True"$ $(**)$
$t_{RE} = t,\ t_{RC} = t+1$
ELSE
$\overline{s_{C+1+lab_{t+1}}} \leftarrow \overline{s_{C+1+lab_{t+1}}^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$\underline{s_C} \leftarrow \underline{s_C^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$t_{RE} = t+1,\ t_{RC} = t$
END-IF
END-IF

IF $\exists l(t, n(t)), \frac{n(t)-1}{2} = \#(\mathcal{S}_L)$
THEN
To represent $\mathcal{S}_L$ by means of $S_L^{n(t)}$: $\mathcal{S}_L \leftarrow S_L^{n(t)}$
To represent $\overline{s_C} \in \mathcal{S}$ as $\overline{s_C^{n(t)}}$: $\overline{s_C} \leftarrow \overline{s_C^{n(t)}}$
$t_{LE} = t_{LC} = t$
ELSE
Look for t and $t+1$, such that:
$\frac{n(t)-1}{2} < \#(\mathcal{S}_L) < \frac{n(t+1)-1}{2}$
$lab_t = \frac{n(t+1)}{2} - \#(\mathcal{S}_L)$
$lab_{t+1} = \#(\mathcal{S}_L) - lab_t$
$assign_{t+1}^L(density_{\mathcal{S}_L})$
$assign_t^L(density_{\mathcal{S}_L})$
IF $density_{\mathcal{S}_L}$= "extreme"
THEN
$\overline{s_{C-lab_t}} \leftarrow \overline{s_{lab_{t+1}+1}^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$\overline{s_C} \leftarrow \overline{s_C^{n(t)}}$; $Brid \leftarrow "True"$ $(**)$
$t_{LE} = t+1,\ t_{LC} = t$
ELSE
$\underline{s_{C-lab_{t+1}-1}} \leftarrow \underline{s_{C-lab_{t+1}-1}^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$\overline{s_C} \leftarrow \overline{s_C^{n(t+1)}}$; $Brid \leftarrow "True"$ $(**)$
$t_{LE} = t,\ t_{LC} = t+1$
END-IF
END-IF
END

OUTPUTS:
$S = \{s_0 = (a_0, b_0, c_0), ..., s_n = (a_n, b_n, c_n)\}$
$LH(\mathcal{S}) = \{s_{I(i)}^{G(i)},\ i = 0, ..., g\}$
T_{LH}
$Brid$

4.3.5.- Usando el algoritmo de representación.

En esta sección aplicamos el algoritmo de representación a un conjunto de términos no balanceado. Usaremos la LH mostrada en la figura 4.1. Vamos a suponer que queremos manejar información lingüística evaluada sobre el conjunto de términos lingüísticos no balanceado mostrado en la figura 4.3, esto es, S= {N, L, M, AH, H, QH, VH, AT, T}. Entonces, la descripción de S en el algoritmo es {(2, extremo),1,(6,extremo)}, siendo S_L= {N, L}, S_C={M}, y S_R={AH, H, QH, VH, AT, T}. Entonces, la representación de S de acuerdo con el algoritmo de representación es la siguiente:

1. Representación de S_R: Como en LH la condición $\exists l(t,(n(t)),(n(t)-1)/2 = \#(S_R)$ no es satisfecha, tenemos que buscar dos niveles t y t+1 tales que $(n(t)-1)/2 < \#(S_R) < (n(t+1)-1)/2$. Los niveles que satisfacen esta condición son t=3 y t+1=4 porque sus respectivas cardinalidades en LH son n(t)=9 y n(t+1)=17. Por lo tanto, lab_t=2 y lab_{t+1}=4. Como $densidad_{S_R}$ = extremo la función de representación $asignar_{t+1}^{R}$ y $asignar_t^R$ representa 4 etiquetas S_{RE}={T, AT, VH, QH} en el nivel 4 y 2 etiquetas S_{RC}={H,AH} en el nivel 3 respectivamente. Aplicando ambas funciones obtenemos la siguiente representación de S_{RE} y S_{RC} en LH (ver figura 4.11).

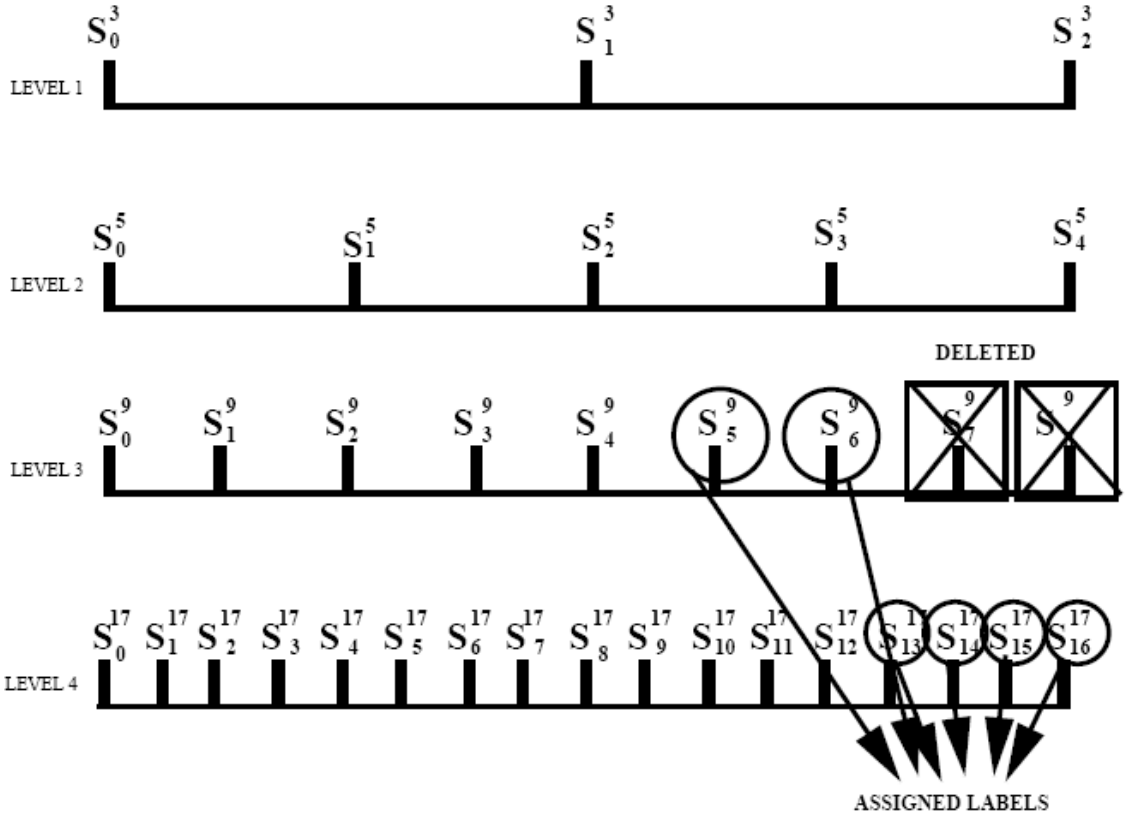

Figura 4.11: Etiquetas usadas para representar S_R

$S_{RE}=\{T \leftarrow s_{16}^{17}, AT \leftarrow s_{15}^{17}, VH \leftarrow s_{14}^{17}, QH \leftarrow s_{13}^{17}\}$, $S_{RC}=\{H \leftarrow s_{6}^{9}, AH \leftarrow s_{5}^{9}\}$.

La actual representación no es una partición difusa porque no puenteamos el s_{jump} todavía. La etiqueta H representa s_{jump} entre el nivel 3 y 4 como puede verse en la figura 4.12.

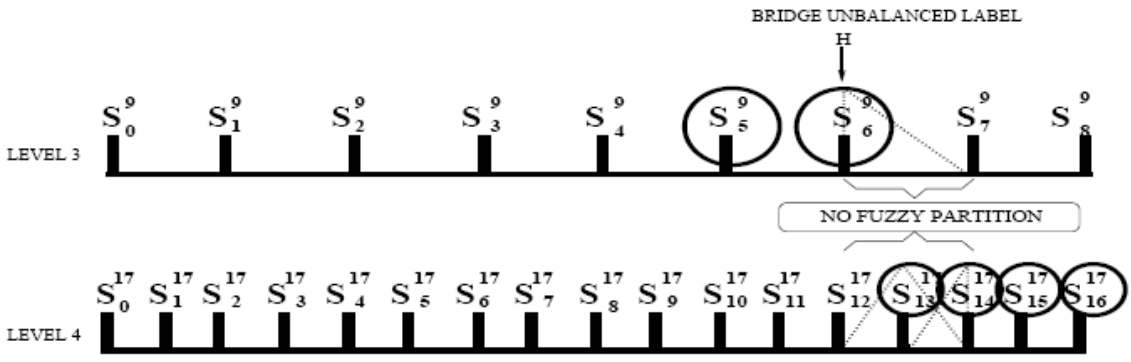

Figura 4.12: Partición no difusa

Tenemos que puentear la etiqueta s_{jump}. De acuerdo con la tabla 4.4 la representación debe ser: $\overline{H} \leftarrow \overline{s_6^9}$ y $\underline{H} \leftarrow \underline{s_{12}^{17}}$ (ver figura 4.13).

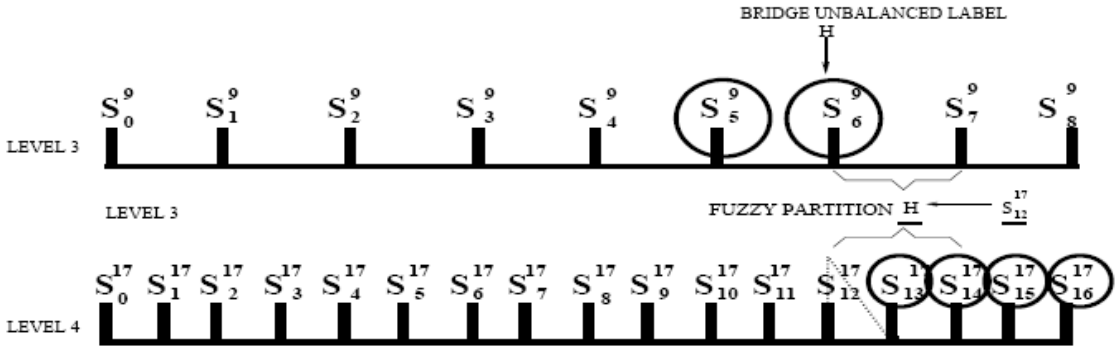

Figura 4.13: Partición difusa

2) Representación de $\overline{S_C}$: Como $densidad_{S_R}$ =extremo, entonces siguiendo el algoritmo el lado bajo de la etiqueta central $\underline{M}$ es representada en el nivel 3 de LH mediante $\underline{s_4^9}$ como muestra la figura 4.14.

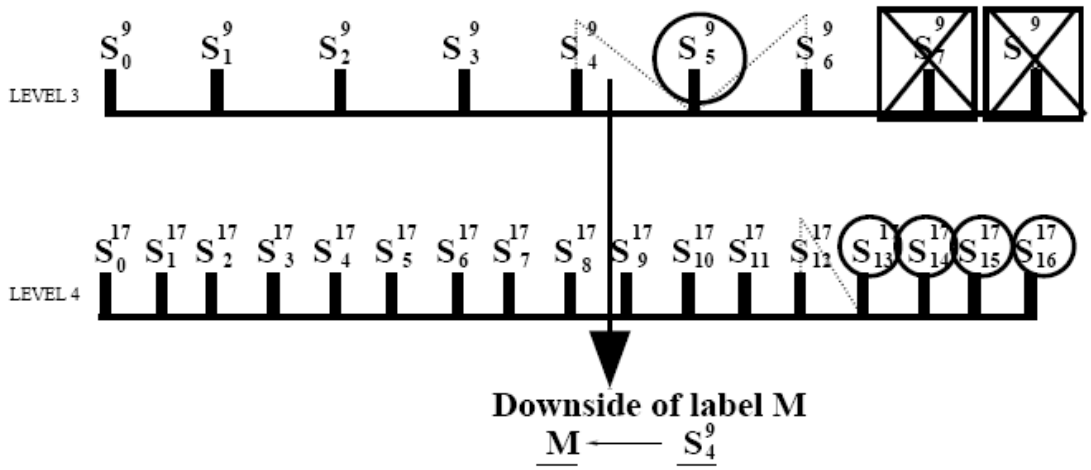

Figura 4.14: Representación de $\underline{M}$

3) Representación de S_L: En este caso, en LH la condición $\exists t \in LH \mid (n(t)-1)/2 = \#(S_L)$ es satisfecha con t=2 porque n(t)=5. Entonces la representación de S_L se obtiene del nivel 2 de LH de la siguiente manera $\{L \leftarrow s_1^5, N \leftarrow s_0^5\}$.

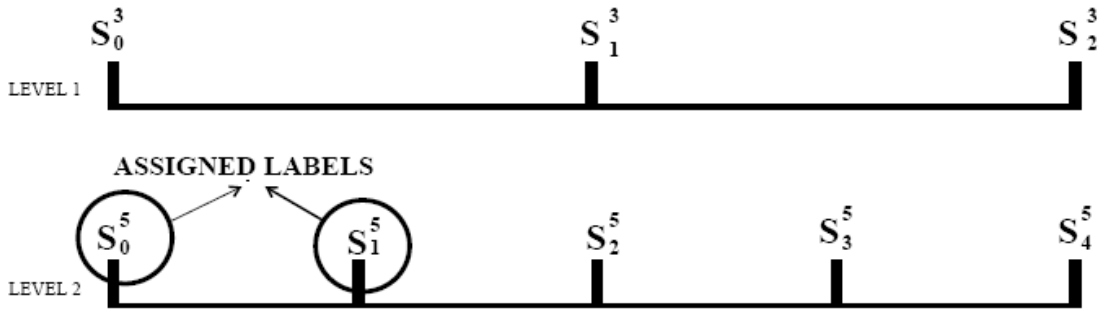

Figura 4.15: Etiquetas usadas para representar S_L

4) Representación de $\overline{S_C}$: El lado superior de la etiqueta central $\overline{M}$ es representada en el nivel 2 de LH mediante $\overline{s_2^5}$ como muestra la figura 4.16.

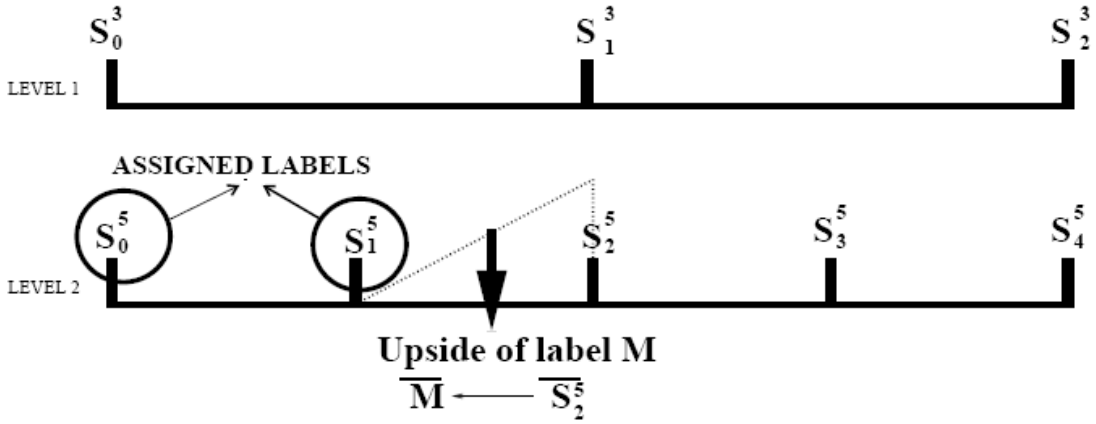

Figura 4.16: Representación de $\overline{M}$

Consecuentemente, al final del algoritmo de representación S={N, L, M, AH, H, QH, VH, AT, T} la semánticas obtenidas son:

- $S_L : \{N \leftarrow \underline{s_0^5}, L \leftarrow s_1^5\}$.
- $S_C : \{M \leftarrow s_2^5 \cup \underline{s_4^9}\}$, y

- $S_R : \{AH \leftarrow s_5^9, H \leftarrow \overline{s_6^9} \cup \underline{s_{12}^{17}}, QH \leftarrow s_{13}^{17}, VH \leftarrow s_{14}^{17}, AT \leftarrow s_{15}^{17}, T \leftarrow s_{16}^{17}\}$.

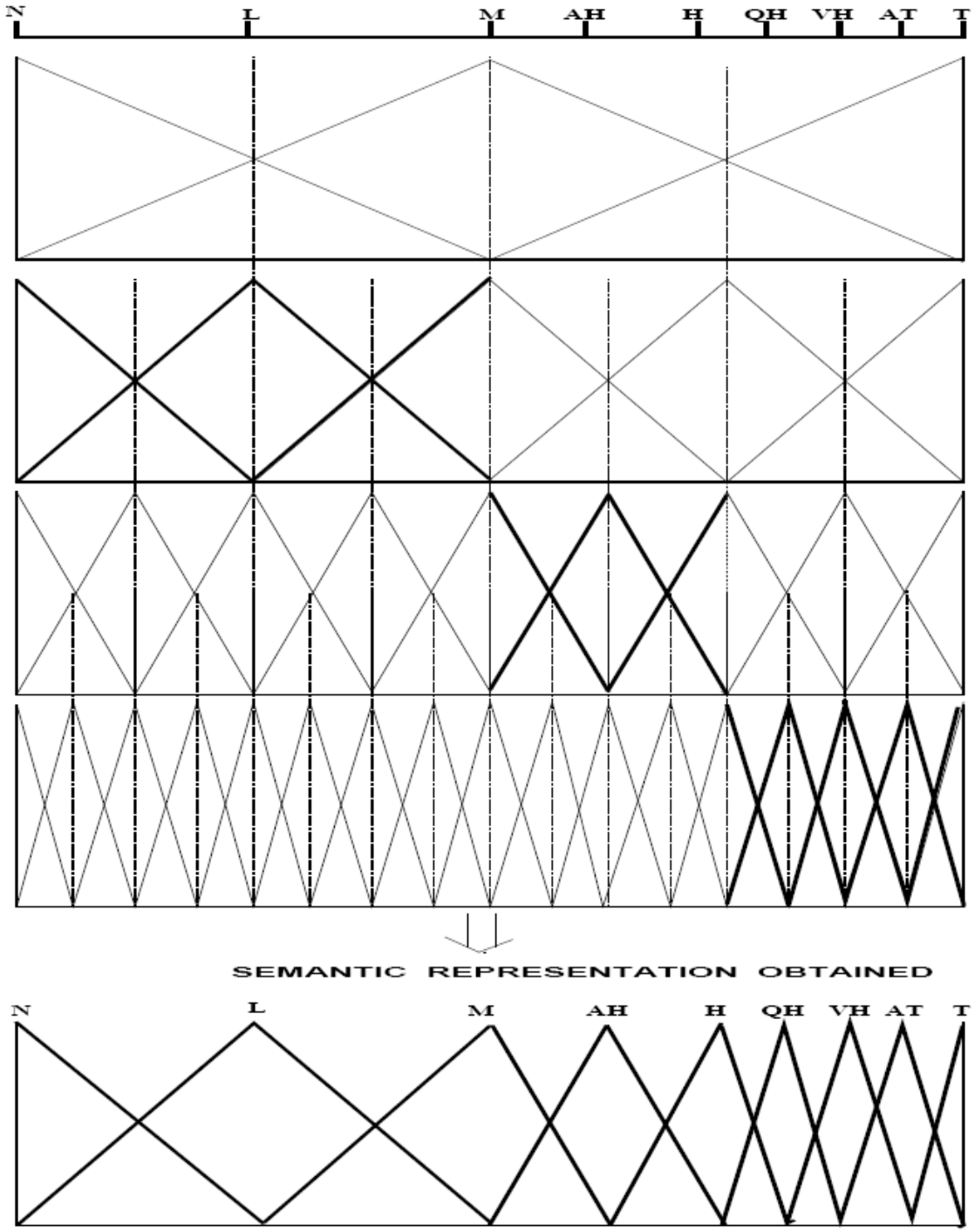

Figura 4.17: Semántica de S={N, L, M, AH, H, QH, VH, AT, T} en LH.

Para controlar tal representación de S en el proceso de computación de palabras el algoritmo proporciona la siguiente información:

1) LH y Brid(S), los cuáles son dados en la tabla VI

$\mathcal{S}$	$LH(\mathcal{S})$	$Brid(\mathcal{S})$
$s_0 = N$	$s_{I(0)}^{G(0)} = s_0^5$	Falso
$s_1 = L$	$s_{I(1)}^{G(1)} = s_1^5$	Falso
$s_2 = M$	$s_{I(2)}^{G(2)} = s_2^5$ **o** s_4^9	Verdad
$s_3 = AH$	$s_{I(3)}^{G(3)} = s_5^9$	Falso
$s_4 = H$	$s_{I(4)}^{G(4)} = s_6^9$ **o** s_{12}^{17}	Verdad
$s_5 = QH$	$s_{I(5)}^{G(5)} = s_{13}^{17}$	Falso
$s_6 = VH$	$s_{I(6)}^{G(6)} = s_{14}^{17}$	Falso
$s_7 = AT$	$s_{I(7)}^{G(7)} = s_{15}^{17}$	Falso
$s_8 = T$	$s_{I(8)}^{G(8)} = s_{16}^{17}$	Falso

Tabla 4.5: LH(S) y Brid(S)

Observación 4.4: Como puede observarse en la tabla 4.5 cuando Brid(S)=Trae existen dos posibles representaciones en LH. En tal caso, tenemos que usar una de ellas con el fin de facilitar y simplificar el proceso de CW. Para hacer esto, proponemos el uso de evaluaciones definidas en el bajo nivel, esto es, $s_{I(2)}^{G(2)} = s_2^5$ y $s_{I(4)}^{G(4)} = s_6^9$.

2) Los siguientes cinco subconjuntos de etiquetas lingüísticas no balanceadas están ordenados en orden creciente:

- $S_{LE} = \{S_{LC} = S_L = \{N, L\},$
- $S_C = \{M\},$
- $S_{RC} = \{AH, H\},$
- $S_{RE} = \{QH, VH, AT, T\}.$

3) El conjunto de niveles de LH usado en la representación de S, $\{t_{LE}, t_{LC}, t_{RC}, t_{RE}\} = \{2,2,3,4\}$.

Capítulo 5

DESARROLLO DEL SISTEMA.

Una vez que se ha presentado el desarrollo de la metodología para representar, administrar y lograr el proceso de computación con palabras con conjuntos de términos lingüísticos no balanceados sin perdida de información y esas ideas han sido formalizadas mediante un algoritmo de representación que ha sido analizado con detalle que conformaba la primera pare del proyecto, en este capítulo se va a detallar el desarrollo de la *aplicación software para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información* implementando la metodología citada anteriormente.

La aplicación desarrollada es bastante sencilla en cuanto a su funcionamiento ya que trabaja a nivel local y no hace uso de ningún sistema de base de datos. Su cometido es asignar semántica a los términos lingüísticos asociados a un conjunto de términos lingüísticos no balanceado que es definido en base a una serie de parámetros de entrada. El resultado obtenido puede ser exportado en un archivo XML para ser utilizado en otras aplicaciones o soportes.

Por lo tanto, esta segunda parte es un proyecto de desarrollo software y, como tal, para su desarrollo deben seguirse las actividades de la Ingeniería del Software. No existe una definición única y estándar para la Ingeniería del Software pero, las dos que se presentan a continuación pueden resultar perfectamente válidas para este cometido:

Ingeniería del Software es la construcción de software de calidad con un presupuesto limitado y un plazo de entrega en contextos de cambio continuo.

Ingeniería del Software es el establecimiento y uso de principios y métodos firmes de ingeniería para obtener software económico que sea fiable y funcione de manera eficiente en máquinas reales.

Las actividades que conforman la Ingeniería del Software son las siguientes:

- Especificación de Requerimientos: se obtienen el propósito del sistema y las propiedades y restricciones del mismo.

- Análisis del Sistema: se obtiene un modelo del sistema correcto, completo, consistente, claro y verificable.

- Diseño del Sistema: se definen los objetivos del proyecto y las estrategias a seguir para conseguirlos.

- Implementación: se traduce el modelo a código fuente.

5.1. Especificación de Requerimientos

El primer paso en la Ingeniería del Software debe ser determinar el propósito último del proyecto, las propiedades que debe satisfacer y las restricciones a las que está sometido.

Este es, sin duda, una paso de vital importancia dentro del desarrollo de un proyecto software ya que, sin conocer el propósito del proyecto y todas las limitaciones de diversa índole a las que debe hacer frente, difícilmente se podrá realizar una aplicación software que cumpla dicho propósito.

En un proyecto de ámbito comercial para una empresa real, para determinar el propósito del mismo se recurre a una serie de estudios como pueden ser entrevistas con los clientes, encuestas con posibles usuarios, estudios de la situación actual del sistema o estudios de viabilidad. En nuestro caso no nos encontramos ante un proyecto comercial sino ante uno académico por lo que el propósito es conocido desde el mismo momento e la concepción del mismo:

El desarrollo de una interfaz software para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información mediante la implementación de la metodología presentada anteriormente. El resultado obtenido se exportará en formato xml para poder ser utilizado por otras aplicaciones o soportes.

Habiendo determinado el propósito último del proyecto, el siguiente paso consiste n especificar los requerimientos del mismo. Los requerimientos de un proyecto software son el conjunto de propiedades o restricciones definidas con total precisión, que dicho proyecto software debe satisfacer. Existen dos tipos bien diferenciados de tales requerimientos:

- Requerimientos funcionales: aquellos que se refieren específicamente al funcionamiento de la aplicación o sistema.

- Requerimientos no funcionales: aquellos no referidos al funcionamiento estricto sino a otros factores externos.

En los dos apartados siguientes se pasarán a definir cuáles son estos requerimientos (tanto funcionales como no funcionales) para el proyecto del que se ocupa esta memoria.

Sin embargo, estas definiciones sólo serán previas ya que en la actividad de análisis del sistema, donde se crearán los casos de uso y sus escenarios, se descubrirán nuevas necesidades que no son observables en esta primera actividad y que permitirán refinar completamente estos requerimientos.

5.1.1. Requerimientos funcionales

Los requerimientos funcionales de un sistema software son aquellos que se encargan de describir las funcionalidades que el sistema debe proporcionar a los usuarios del mismo para cumplir sus expectativas.

Normalmente, estos requerimientos se obtendrían de la interacción con el cliente mediante diversas entrevistas y/o encuestas. En nuestro caso, al tratarse de un proyecto académico, nos encontramos ante la situación de la no existencia de cliente alguno por lo que la información sobre las funcionalidades que debe disponer la aplicación se ha obtenido mediante el análisis del ejemplo de aplicación de la metodología a desarrollar que se presenta en el capítulo 5 ya que reproduce fielmente el escenario de la aplicación real a desarrollar.

Tras el análisis realizado se ha llegado a la conclusión de que las funcionalidades que el usuario potencial espera de la aplicación a desarrollar son las siguientes:

1. Permitir al usuario **introducir los parámetros que definen el conjunto de etiquetas no balanceado** de entrada para el cuál se desea obtener una semántica. Estos parámetros son el número de etiquetas de ambos subconjuntos laterales así como la densidad de cada uno de ellos.

2. **Visualizar gráficamente el conjunto de términos lingüísticos no balanceado de entrada** con la distribución de etiquetas seleccionada a ambos lados de la etiqueta central.

3. **Visualizar gráficamente la jerarquía lingüística utilizada** para representar la semántica del conjunto no balanceado de entrada resaltando las etiquetas utilizadas para obtenerla.

4. **Visualizar gráficamente la semántica obtenida** para el conjunto de entrada tras la aplicación del algoritmo.

5. Visualizar una tabla en la que aparezca para **cada etiqueta el nivel o niveles de la jerarquía lingüística utilizados para su representación semántica**, así como el **valor de Brid** que indica si esa etiqueta supone o no un salto entre dos niveles de la jerarquía.

6. Visualizar una tabla en la que aparezcan los **cinco subconjuntos del conjunto de términos lingüísticos no balanceado** S: S_{LE}, S_{LC}, S_C, S_{RC}, S_{ER} ordenados en orden incremental, además del **conjunto de niveles de la jerarquía usados** para representarlos.

7. **Exportar la semántica obtenida en formato xml** con una estructura definida a priori de manera que pueda ser usada por otras aplicaciones o soportes.

5.1.2. Requerimientos no funcionales

Los requerimientos no funcionales son aquellos que restringen los requerimientos funcionales. Son tan importantes como los propios requerimientos funcionales y pueden incluso a llegar a ser críticos para la aceptación del sistema. Estos requerimientos normalmente especifican propiedades del sistema o del producto en si (plataforma, velocidad, rendimiento...) y del diseño de la interfaz gráfica con el usuario además de todas las restricciones impuestas por la organización (políticas de empresa, estándares, legalidad vigente...).

Al no ser este un proyecto comercial para ninguna organización o empresa real, no debemos someternos a restricciones organizacionales. Por lo tanto, los requerimientos no funcionales que se deben obtener y analizar son los referentes a las necesidades hardware y software de los equipos informáticos para que estos proporcionen al usuario

las funcionalidades requeridas de forma eficiente y los referentes a la interfaz gráfica entre la aplicación y el usuario.

5.1.2.1.- Requerimientos de Equipo Informático

Al hablar de los requerimientos del equipo informático y debido a que la aplicación desarrollada trabaja a nivel local en el equipo las necesidades de equipo informático del son muy simples. Distinguiremos entre requerimientos hardware y requerimientos software:

Requerimientos hardware:

- Velocidad: el equipo debe ser lo suficientemente rápido como para ejecutar la aplicación en el menor tiempo posible y con la mayor fiabilidad. Cualquier microprocesador actual es capaz de cumplir con esta labor. RECOMENDABLE: Intel Dual Core.

- Memoria: el equipo debe disponer de la suficiente memoria RAM libre para cargar la máquina virtual de java que permitirá ejecutar la aplicación. RECOMENDABLE: 1GB DE RAM.

- Almacenamiento: Puesto que la aplicación no hace uso de ningún sistema de gestión de base de datos (ya que el resultado se exporta en un solo archivo xml) para almacenar información este aspecto es secundario por lo que no se requiere de una gran capacidad de almacenamiento. RECOMENDABLE: 1 DISCO DURO SATA II 120 GB.

- Tarjeta gráfica: las tarjetas gráficas de las que disponen los equipos informáticos actuales son de gran potencia por lo que es inútil establecer ningún requerimiento en este aspecto, aunque si es necesario destacar que debido a que la aplicación utiliza la librería Java 3D, la tarjeta gráfica debe contener el chip Creator 3D el cual hace uso (como Java 3D) del API de OpenGL. Creator 3D.

Requerimientos software:

- Sistema operativo: Windows 2000 SP4 o Windows XP SP2.
- Máquina virtual Java JRE 1.4.2_05 o superior.
- Intérprete de lenguaje xml como por ejemplo Internet Explorer 6.0 o Mozilla Firefox.

5.1.2.2.- Requerimientos de la Interfaz

Los requerimientos de la interfaz gráfica entre la aplicación y el usuario están íntimamente ligados a la usabilidad y sus principios. La usabilidad se puede definir de varias formas :

- Usabilidad se define coloquialmente como facilidad de uso, ya sea de una página web, una aplicación informática o cualquier otro sistema que interactúe con un usuario.

- Usabilidad se refiere a la capacidad de un software de ser comprendido, aprendido, usado y ser atractivo para el usuario, en condiciones específicas de uso.

- Usabilidad es la efectividad, eficiencia y satisfacción con la que un producto permite alcanzar objetivos específicos a usuarios específicos en un contexto de uso específico.

A partir de estas tres definiciones se pueden obtener los principios básicos de la usabilidad, los cuales se asociarán a los requerimientos no funcionales que deberá cumplir la interfaz gráfica:

- Facilidad de aprendizaje: se refiere a la facilidad con la que nuevos usuarios pueden tener una interacción efectiva. Depende de los siguiente factores:

    - Predecibilidad: una vez conocida la aplicación, se debe saber en cada momento a que estado se pasará en función de la tarea que se realice.

    - Síntesis: los cambios de estado tras una acción deben ser fácilmente captados.

    - Generalización: las tareas semejantes se resuelven de modo parecido.

    - Familiaridad: el aspecto de la interfaz tiene que resultar conocido y familiar para el usuario.

    - Consistencia: siempre se han de seguir una misma serie de pasos para realizar una tarea determinada.

- Flexibilidad: relativa a la variedad de posibilidades con las que el usuario y el sistema pueden intercambiar información. También abarca la posibilidad de diálogo, la multiplicidad de vías para realizar la tarea, similitud con tareas anteriores y la optimización entre el usuario y el sistema.

- Robustez: es el nivel de apoyo al usuario que facilita el cumplimiento de sus objetivos o, también, la capacidad del sistema para tolerar fallos. Esta relacionada con los siguiente factores:

    - Observación: el usuario debe poder observar el estado del sistema sin que esta observación repercuta de forma negativa en él.

    - Recuperación de información: la aplicación debe poder deshacer alguna operación y permitir volver a un estado anterior.

- Tiempo de respuesta: es el tiempo necesario para que el sistema pueda mostrar los cambios realizados por el usuario.

5.2.- Análisis de Sistema

Una vez conocido el propósito del proyecto software, las propiedades que debe cumplir y las restricciones a las que debe someterse, llega el momento de analizar el sistema y crear un modelo del mismo que sea correcto, completo, consistente, claro y verificable.

Para conseguir esto se crearán y definirán casos de uso en base a los requerimientos previamente obtenidos y se describirán ciertos escenarios de acción de dichos casos de uso.

5.2.1. Casos de Uso

Un caso de uso representa una clase de funcionalidad dada por el sistema como un flujo de eventos. También se puede definir como la representación de una situación o tarea de interacción de un usuario con la aplicación.

Los casos de uso describen como se realiza una tarea de manera exacta y constan de los siguientes elementos:

- Nombre único e unívoco.
- Actores participantes.
- Condiciones de entrada.
- Flujo de eventos.

- Condiciones de salida.

- Requerimientos especiales.

Por lo tanto, es necesario determinar cuales son los actores participantes en cada no de los casos de uso. Un actor modela una entidad externa que se comunica con el sistema, es decir, es un tipo de usuario del sistema. Un actor, al igual que un caso de uso, debe tener un nombre único y puede tener una descripción asociada.

- Usuario: se trata del usuario tipo de la aplicación, es decir el que la va a usar para obtener la semántica de un conjunto no balanceado de términos lingüísticos definido mediante unos parámetros de entrada.

- Soporte XML: se trata del archivo en el que se exporta la semántica obtenida para el conjunto de entrada, así como la jerarquía lingüística utilizada para ello.

Una vez definidos cuales van a ser los actores del sistema, es el momento de crear los distintos casos de uso. A la hora de realizar esta acción es importante que cada uno de los requerimientos funcionales ya definidos aparezca en al menos uno de los casos de uso aunque, por otra parte, puede haber casos de uso nuevos, en los que no aparezca ninguno de los requerimientos, ya que estamos en una fase de refinamiento del sistema donde queremos construir un modelo detallado del mismo.

Un paso previo a la creación y descripción de los distintos casos de uso es la obtención de los diversos diagramas de casos de uso de nuestro sistema.

El primero es un diagrama frontera, es decir, un diagrama que describe completamente la funcionalidad de un sistema:

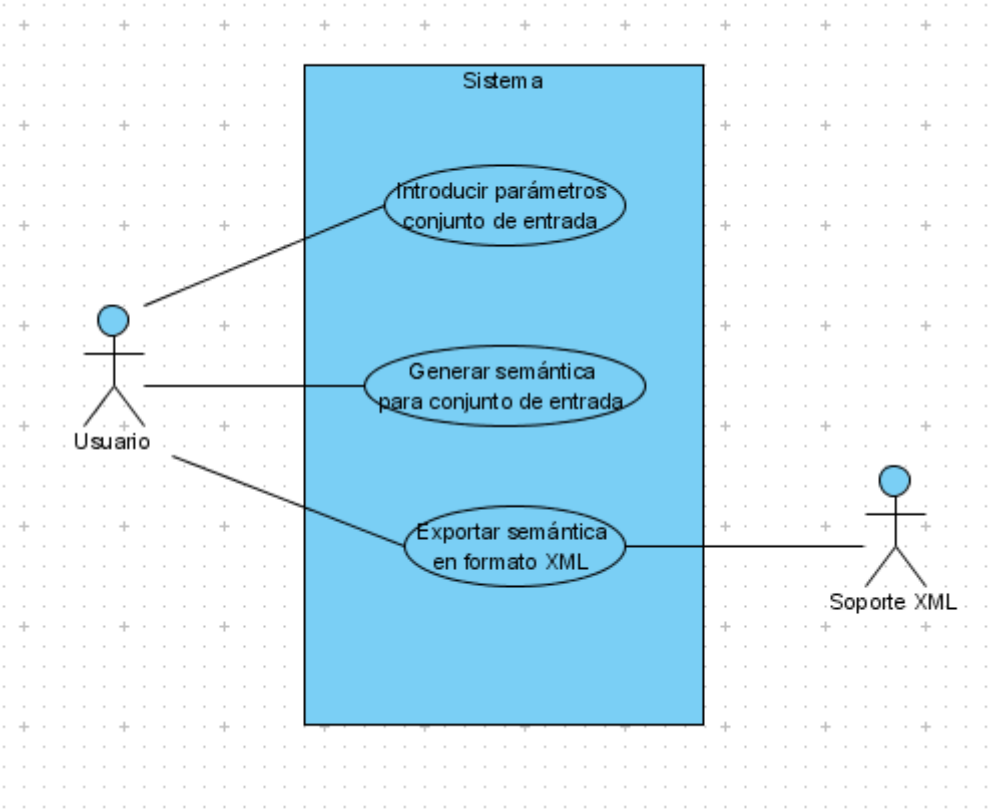

Figura 5.1: Diagrama frontera del sistema

A continuación, se describen detalladamente cada uno de los casos de uso mostrados en el diagrama frontera.

Caso de Uso 1: Introducir parámetros del conjunto de entrada.

Actores participantes: Usuario.

Condiciones de entrada: El Usuario está delante de la aplicación y posee los parámetros del conjunto de entrada que quiere introducir.

Flujo de eventos:

1. El usuario inicia la aplicación.
2. La aplicación muestra unos campos de texto dónde introducir los parámetros del conjunto de entrada no balanceado.

3. El usuario introduce los cuatro parámetros que define el conjunto de entrada no balanceado.
4. El sistema comprueba que los parámetros del conjunto de entrada son correctos (E-1, E-2, E-3).
5. Si el conjunto es correcto, el sistema muestra la representación gráfica del mismo.

Condiciones de salida: El Usuario puede ver la representación gráfica del conjunto de entrada definido por los parámetros introducidos.

Excepciones:

E-1: El conjunto de términos introducido tiene el mismo número de etiquetas a ambos lados de la etiqueta central, es decir es balanceado. El usuario puede volver a introducir el número de etiquetas a ambos lados o salir del caso de uso.

E-2: La cardinalidad del conjunto de etiqueta es par. El usuario puede volver a introducir el número de etiquetas a ambos lados o salir del caso de uso.

E-3: Hay más de 8 términos en uno o ambos lados de la etiqueta central. El usuario puede volver a introducir el número de etiquetas a ambos lados o salir del caso de uso.

Caso de Uso 2: Generar semántica para el conjunto de entrada.

Actores participantes: Usuario.

Condiciones de entrada: El Usuario está delante de la aplicación y quiere obtener la semántica asociada al conjunto no balanceado cuyos parámetros ha introducido y que el sistema le está mostrando gráficamente.

Flujo de eventos:

1. El sistema habilita la opción *Aplicar algoritmo*.

2. El usuario elije la opción *Aplicar Algoritmo.*
3. El sistema aplica el algoritmo de representación y muestra en pantalla la siguiente información:

- **Representación gráfica de la jerarquía lingüística utilizada** para representar la semántica del conjunto no balanceado de entrada resaltando las etiquetas utilizadas para obtenerla.

- **Representación gráfica la semántica obtenida** para el conjunto de entrada tras la aplicación del algoritmo.

- Una tabla en la que aparecen para **cada etiqueta el nivel o niveles de la jerarquía lingüística utilizados para su representación semántica**, así como el **valor de Brid** que indica si esa etiqueta supone o no un salto entre dos niveles de la jerarquía.

- Una tabla en la que aparecen los **cinco subconjuntos del conjunto de términos lingüísticos no balanceado** S: S_{LE}, S_{LC}, S_C, S_{RC}, S_{ER} ordenados en orden incremental, además del **conjunto de niveles de la jerarquía usados** para representarlos.

Condiciones de salida: El Usuario puede ver semántica obtenida para el conjunto de entrada no balanceado.

Caso de Uso 3: Exportar semántica en formato XML.

Actores participantes: Usuario y soporte XML.

Condiciones de entrada: El Usuario está delante de la aplicación y desea exportar a un archivo XML la semántica obtenida por el sistema y que está siendo mostrada al usuario gráficamente.

Flujo de eventos:

1. El sistema habilita la opción *Exportar*.
2. El usuario elije la opción *Exportar.*
3. Si no se producen excepciones (E-1), el soporte XML crea y abre un archivo XML que contiene la semántica obtenida para el conjunto de entrada así como la jerarquía lingüística utilizada.

Condiciones de salida: El Usuario obtiene un archivo XML que puede ser guardado y utilizado para otras aplicaciones.

Excepciones:
E-1: El soporte XML no puede crear el archivo XML debido a que el directorio de creación está protegido contra escritura. Finaliza el caso de uso.

Como podemos comprobar, cada uno de los requerimientos funcionales definidos previamente han aparecido en al menos uno de los casos de uso.

5.2.2. Escenarios.

Un caso de uso es una representación abstracta, una abstracción, de una funcionalidad del sistema a realizar. La representación concreta de un caso de uso se realiza mediante la creación de uno o más escenarios que muestren todas las interacciones posibles entre El sistema y sus usuarios.

Un escenario esta formado por los siguientes elementos:

- Un nombre único y unívoco.
- Una descripción.
- Los actores participantes.

- El flujo de eventos.

Como se ha indicado, para cada caso de uso puede haber varios escenarios. A continuación se muestran los escenarios utilizados para demostrar las funcionalidades del sistema:

Escenario 1: Crear conjunto no balanceado de entrada con 2 etiquetas en el lado izquierdo, densidad central y 6 en el lado derecho, densidad extremo.

Nombre: CrearConj4cent6extr.

Descripción: El usuario va a introducir un conjunto de entrada cuyos parámetros son:

- Lado izquierdo: 2 etiquetas y densidad central.
- Lado Derecho: 6 etiquetas y densidad extremo.

Actores: Antonio.

Flujo de eventos:

1. Antonio inicia la aplicación.
2. La aplicación muestra unos campos de texto dónde introducir los parámetros del conjunto de entrada no balanceado.
3. Antonio introduce los datos 2, centro,6, extremo y selecciona *Refrescar Conjunto NB*.
4. El sistema comprueba que los parámetros del conjunto de entrada son correctos y muestra la representación gráfica del conjunto introducido por Antonio.

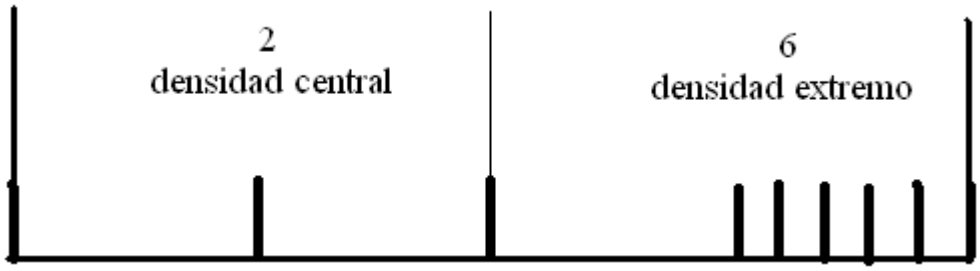

Figura 5.2: Conjunto de entrada no balanceado

Escenario 2: Obtener la semántica para el conjunto no balanceado de entrada creado previamente con 2 etiquetas en el lado izquierdo, densidad central y 6 en el lado derecho, densidad extremo.

Nombre: ObtenerSemanticaConj4cent6extr.

Descripción: El usuario quiere obtener la semántica para el conjunto no balanceado de entrada introducido previamente de parámetros:

- Lado izquierdo: 2 etiquetas y densidad central.
- Lado Derecho: 6 etiquetas y densidad extremo.

Actores: Antonio.

Flujo de eventos:

1. Antonio elije la opción *Aplicar Algoritmo.*
2. El sistema aplica el algoritmo de representación y muestra en pantalla la siguiente información:

- **Representación gráfica de la jerarquía lingüística utilizada formada por 4 niveles de 3, 5, 9 y 17 etiquetas respectivamente.**

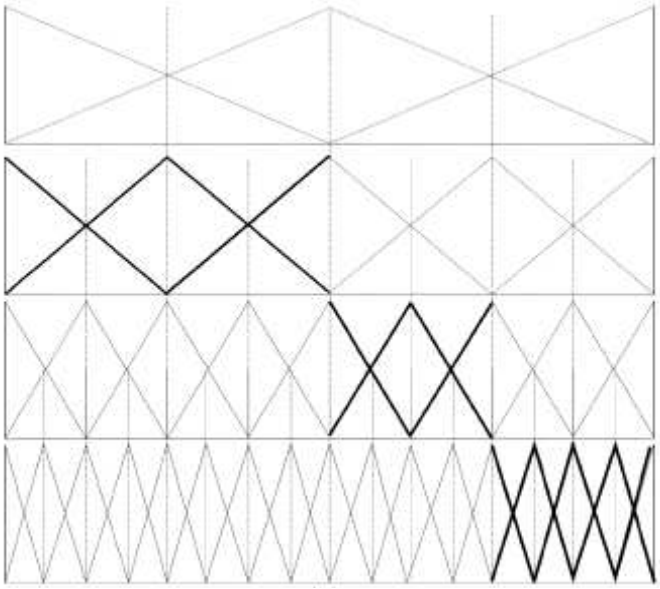

Figura 5.3: Jerarquía lingüística

- **Representación gráfica la semántica obtenida** para el conjunto de entrada tras la aplicación del algoritmo.

Figura 5.4: Semántica obtenida

- Una tabla en la que aparecen para **cada etiqueta el nivel o niveles de la jerarquía lingüística utilizados para su representación semántica**, así como el **valor de Brid** que indica si esa etiqueta supone o no un salto entre dos niveles de la jerarquía.

$\mathcal{S}$	$LH(\mathcal{S})$	$Brid(\mathcal{S})$
$s_0 = N$	$s_{I(0)}^{G(0)} = s_0^5$	False
$s_1 = L$	$s_{I(1)}^{G(1)} = s_1^5$	False
$s_2 = M$	$s_{I(2)}^{G(2)} = s_2^5$ **or** s_4^9	True
$s_3 = AH$	$s_{I(3)}^{G(3)} = s_5^9$	False
$s_4 = H$	$s_{I(4)}^{G(4)} = s_6^9$ **or** s_{12}^{17}	True
$s_5 = QH$	$s_{I(5)}^{G(5)} = s_{13}^{17}$	False
$s_6 = VH$	$s_{I(6)}^{G(6)} = s_{14}^{17}$	False
$s_7 = AT$	$s_{I(7)}^{G(7)} = s_{15}^{17}$	False
$s_8 = T$	$s_{I(8)}^{G(8)} = s_{16}^{17}$	False

Figura 5.5: LH(S) y Brid.

- Una tabla en la que aparecen los **cinco subconjuntos del conjunto de términos lingüísticos no balanceado** S: S_{LE}, S_{LC}, S_C, S_{RC}, S_{ER} ordenados en orden incremental, además del **conjunto de niveles de la jerarquía usados** para representarlos.

Escenario 3: Exportar semántica en formato XML.

Nombre: ExportarSemanticaConj4cent6extr.

Descripción: El usuario quiere exportar a un archivo XML la semántica obtenida para el conjunto no balanceado de entrada introducido previamente de parámetros:

- Lado izquierdo: 4 etiquetas y densidad central.
- Lado Derecho: 6 etiquetas y densidad extremo.

Actores: Antonio y Soporte XML.

Flujo de eventos:

2. Antonio elije la opción *Exportar.*
3. El soporte XML crea y abre un archivo XML que contiene la semántica obtenida para el conjunto de entrada así como la jerarquía lingüística utilizada.

5.3. Diseño del Sistema.

Sin duda, realizar de manera adecuada cada una de las actividades que conlleva la Ingeniería del Software es indispensable para la realización de un proyecto software de calidad. Por lo tanto, no se puede decir que ninguna de estas actividades sea más importante que otra. Sin embargo, si podemos decir que la actividad de diseño es la más delicada y la más laboriosa de llevar a cabo.

Es delicada porque si no se lleva a cabo correctamente se hace imposible el codificar, de manera correcta, en la actividad de implementación el modelo obtenido en el análisis del sistema, lo que puede repercutir en el desperdicio de todo el esfuerzo realizado durante las primeras actividades de la Ingeniería del Software.

Y es laboriosa porque las estrategias a seguir para conseguir que esta traducción entre modelo y código se lleve a cabo correctamente son muy diversas y bastante complejas. Se puede decir, por tanto, que el diseño del sistema es la actividad de la Ingeniería del Software en la que se identifican los objetivos finales del sistema y se plantean las diversas estrategias para alcanzarlos en la actividad de implementación.

Sin embargo, el sistema no se suele diseñar de una sola vez sino que hay que diferenciar entre el diseño y estructura de los datos que se van a manejar y el diseño de la interfaz entre la aplicación y el usuario. Estas dos fases del diseño no se realizan de forma consecutiva una detrás de la otra sino que lo normal es realizarlas de manera concurrente y finalizarlas a la vez.

5.3.1. Diseño de los Datos.

La intención de esta fase del diseño software es determinar la estructura que poseen cada uno de los elementos de información del sistema, es decir, la estructura de los datos sobre los que va a trabajar.

Debido a que la complejidad de nuestra aplicación no está en el tratamiento de la información no es necesario ningún sistema de base de datos que almacene y modele información en forma de tablas. Sin embargo, una de las funcionalidades de la

aplicación consiste en la exportación (y por lo tanto almacenamiento) de la semántica obtenida para el conjunto de etiquetas no balanceado de entrada, así como la jerarquía usada para representarla. Para almacenar esta información utilizaremos la tecnología XML, debido principalmente a:

- XML tiene la ventaja de la portabilidad y compatibilidad con plataformas y aplicaciones, cosa deseada en nuestro caso.
- XML es ideal para almacenar pocos datos, como es el caso de nuestra aplicación.

A continuación vamos a conocer un poco esta tecnología que hemos elegido para almacenar la información generada por la aplicación.

XML.

XML representa una manera distinta de hacer las cosas, más avanzada, cuya principal novedad consiste en permitir compartir los datos con los que se trabaja a todos los niveles, por todas las aplicaciones y soportes. Así pues, el XML juega un papel importantísimo en este mundo actual, que tiende a la globalización y la compatibilidad entre los sistemas, ya que es la tecnología que permitirá compartir la información de una manera segura, fiable, fácil. Además, XML permite al programador y los soportes dedicar sus esfuerzos a las tareas importantes cuando trabaja con los datos, ya que algunas tareas tediosas como la validación de estos o el recorrido de las estructuras corre a cargo del lenguaje y está especificado por el estándar, de modo que el programador no tiene que preocuparse por ello.

El XML se creó para que cumpliera varios objetivos:

- Que fuera idéntico a la hora de servir, recibir y procesar la información que el HTML, para aprovechar toda la tecnología implantada para este último.
- Que fuera formal y conciso desde el punto de vista de los datos y la manera de guardarlos.

- Que fuera extensible, para que lo puedan utilizar en todos los campos del conocimiento.
- Que fuese fácil de leer y editar.
- Que fuese fácil de implantar, programar y aplicar a los distintos sistemas.

El XML se puede usar para infinidad de trabajos y aporta muchas ventajas en amplios escenarios:

- Comunicación de datos. Si la información se transfiere en XML, cualquier aplicación podría escribir un documento de texto plano con los datos que estaba manejando en formato XML y otra aplicación recibir esta información y trabajar con ella.
- Migración de datos. Si tenemos que mover los datos de una base de datos a otra sería muy sencillo si las dos trabajasen en formato XML.

La sintaxis que utilizar XML en sus archivos es extremadamente sencilla:

Dicen que el XML es un 10% del SGML y de verdad lo es, porque en realidad las normas que tiene son muy simples. Se escribe en un documento de texto ASCII, igual que el HTML y en la cabecera del documento se tiene que poner el texto

<?xml version="1.0"?>

En el resto del documento se deben escribir etiquetas como las de HTML, las etiquetas que nosotros queramos, por eso el lenguaje se llama XML, lenguaje de etiquetas extendido. Las etiquetas se escriben anidadas, unas dentro de otras.

<ETIQ1>...<ETIQ2>...</ETIQ2>...</ETIQ1>

Cualquier etiqueta puede tener atributos. Le podemos poner los atributos que queramos.

<ETIQ atributo1="valor1" atributo2="valor2"...>

Los comentarios de XML se escriben igual que los de HTML.

<!-- Comentario -->

Y esto es todo lo que es el lenguaje XML en si, aunque tenemos que tener en cuenta que el XML tiene muchos otros lenguajes y tecnologías trabajando alrededor de él. Sin embargo, no cabe duda que la sintaxis XML es realmente reducida y sencilla.

Para definir qué etiquetas y atributos debemos utilizar al escribir en XML tenemos que fijarnos en la manera de guardar la información de una forma estructurada y ordenada.

Por ejemplo, si deseamos guardar la información relacionada con una película en un documento XML podríamos utilizar un esquema con las siguientes etiquetas.

```
<?xml version="1.0"?>
<PELICULA nombre="El Padrino" año=1985>
<PERSONAL>
</DIRECTOR nombre="Georgie Lucas">
</INTERPRETE nombre="Marlon Brando" interpreta-a="Don Corleone">
</INTERPRETE nombre="Al Pacino" interpreta-a="Michael Corleone">
</PERSONAL>
</ARGUMENTO descripción="Pelicula de mafias sicilianas en Estados Unidos">
</PELICULA>
```

Como se puede ver, hemos utilizado las etiquetas que hemos querido para poner este ejemplo y las hemos anidado de manera que la etiqueta más grande es la PELICULA y dentro de ella tenemos el PERSONAL y el ARGUMENTO. A su vez, dentro de PERSONAL tenemos tanto al DIRECTOR como a los actores (INTERPRETE).

Acabamos de introducir la tecnología XML y su sintaxis, usada por la aplicación para almacenar la semántica obtenida para el conjunto no balanceado de entrada y la jerarquía utilizada para representarla. A continuación mostramos la estructura que tendrá el archivo XML que utiliza la aplicación para tal fin. Para comprender la estructura de una forma más clara presentamos el archivo generado por la aplicación para un conjunto de etiquetas no balanceado con las siguientes características:

- Número de etiquetas del lado izquierdo: 1.
- Densidad del lado izquierdo: Extremo.
- Número de etiquetas del lado derecho: 3.
- Densidad del lado izquierdo: Extremo.

La estructura del archivo XML generado en este caso es la siguiente:

```
<escala>
      <LH>
            <nivel>
              <t>0</t>
              <granularidad>3</granularidad>
                 </nivel>
            <nivel>
              <t>1</t>
              <granularidad>5</granularidad>
            </nivel>
            <nivel>
              <t>2</t>
              <granularidad>9</granularidad>
            </nivel>
    </LH>
     <Termino>
       <idTermino>S0</idTermino>
       <idS>0</idS>
       <a>0.0</a>
       <b>0.0</b>
       <c>0.5</c>
       <puente>false</puente>
       <idTerminoParteBaja>0</idTerminoParteBaja>
       <GranularidadParteBaja>3</GranularidadParteBaja>
       <idTerminoParteAlta>0</idTerminoParteAlta>
       <GranularidadParteAlta>3</GranularidadParteAlta>
     </Termino>
    <Termino>
        <idTermino>S1</idTermino>
        <idS>1</idS>
        <a>0.5</a>
        <b>0.0</b>
        <c>0.75</c>
        <puente>true</puente>
        <idTerminoParteBaja>1</idTerminoParteBaja>
        <GranularidadParteBaja>3</GranularidadParteBaja>
        <idTerminoParteAlta>2</idTerminoParteAlta>
        <GranularidadParteAlta>5</GranularidadParteAlta>
      </Termino>
```

```
    <Termino>
      <idTermino>S2</idTermino>
      <idS>2</idS>
      <a>0.75</a>
      <b>0.5</b>
      <c>0.875</c>
      <puente>true</puente>
      <idTerminoParteBaja>3</idTerminoParteBaja>
      <GranularidadParteBaja>5</GranularidadParteBaja>
      <idTerminoParteAlta>6</idTerminoParteAlta>
      <GranularidadParteAlta>9</GranularidadParteAlta>
    </Termino>
    <Termino>
       <idTermino>S3</idTermino>
       <idS>3</idS>
       <a>0.875</a>
       <b>0.75</b>
       <c>1.0</c>
       <puente>false</puente>
       <idTerminoParteBaja>7</idTerminoParteBaja>
       <GranularidadParteBaja>9</GranularidadParteBaja>
       <idTerminoParteAlta>7</idTerminoParteAlta>
       <GranularidadParteAlta>9</GranularidadParteAlta>
     </Termino>
    <Termino>
      <idTermino>S4</idTermino>
      <idS>4</idS>
      <a>1.0</a>
      <b>0.875</b>
      <c>1.0</c>
      <puente>false</puente>
      <idTerminoParteBaja>8</idTerminoParteBaja>
      <GranularidadParteBaja>9</GranularidadParteBaja>
      <idTerminoParteAlta>8</idTerminoParteAlta>
      <GranularidadParteAlta>9</GranularidadParteAlta>
    </Termino>
</escala>
```

5.3.2. Diseño de la Interfaz

En esta fase del diseño del sistema software se define cual va a ser la apariencia visual de la aplicación, es decir, se define la interfaz visual entre el usuario y la aplicación. Sin duda, realizar un buen diseño de la interfaz resulta primordial ya que esta debe

presentarse atractiva al usuario de la aplicación pero a la vez le debe de resultar fácil de entender y trabajar sobre ella.

Aunque existe una guía de estilo estándar para desarrollar interfaces para aplicaciones de escritorio de Windows XP/Linux vamos a definir nuestra propia guia de estilo y procurar que, en base a ella, la interfaz resultante consiga unas cotas dignas de atractivo visual, familiaridad y facilidad de uso.

En este apartado vamos a definir nuestra guía de estilo, a describir y analizar las metáforas empleadas y por último mostraremos unos prototipos del sistema:

5.3.2.1.-Guía de Estilo.

Antes de ponerse a diseñar una interfaz de usuario, se debe definir el estilo de la misma. Esto es de vital importancia cuando el diseño va a ser compartido entre varios diseñadores, ya que ayuda a mantener la coherencia interna de la interfaz.

Sin embargo, en contra de lo que pueda parecer en un principio, también es de mucha utilidad definir una guía de estilo cuando solo hay un diseñador encargado de la interfaz. Esto se debe a varias razones:

A veces es posible que mantener la coherencia y consistencia de una interfaz, si esta es muy grande o muy ambiciosa, sea algo complicado incluso si solo hay un diseñador si no tiene una base.

El diseñador primitivo puede, por las más diversas razones, abandonar el diseño y es de utilidad para sus sustitutos contar con una guía de estilo predefinida para no tener que empezar de cero otra vez. Lo mismo puede aplicarse si no es el diseñador original el que se encarga del mantenimiento o la actualización de la interfaz.

Quedando demostrada la utilidad del uso de guías de estilo podemos pasar a definir las reglas, normas y recomendaciones que contendrá la guía de estilo de nuestra interfaz:

Fuentes:

Tipo: Arial, Times New Roman Tamaño: 12px. Para los encabezados de gráficos y tablas se usa letra mayúscula.

Colores:

El fondo de la aplicación tendrá un color gris. El fondo de las gráficas será de color blanco, del mismo modo que el fondo de las tablas.

En las gráficas se usarán el color negro para los ejes y azul para las etiquetas.

Botones:

Los botones tendrán un fondo azul claro y la fuente Arial tamaño 12px. Estarán situados en la parte superior de la pantalla.

5.3.2.2.- Metáforas.

Una metáfora es el empleo de un objeto con un significado o dentro de un contexto diferente al habitual. Al diseñar una interfaz gráfica, la utilización de metáforas resulta muy útil ya que permiten al usuario, por comparación con otro objeto o concepto, comprender de una manera más intuitiva las diversas tareas que la interfaz permite desarrollar.

Al igual que pasa en el ámbito de la literatura, para que una metáfora cumpla con su cometido, el desarrollador de la aplicación y el usuario final de ésta deben tener una base cultural similar. Es muy posible que el uso de un icono de manera metafórica sea entendido de una manera por el usuario occidental y de otra, bien distinta por un usuario oriental. Hay que intentar por lo tanto, que las metáforas empleadas sean lo más universales posibles para que, así sean comprendidas a la perfección por la mayor parte de los usuarios potenciales.

Las aplicaciones de escritorio de Windows suelen seguir la Guía de Estilo XP y utilizan una serie de metáforas con las que el usuario está plenamente familiarizado (por ejemplo, una lupa con un signo '+' en su interior establece que la función del icono es,

inequívocamente, la de realizar un aumento de zoom). En el mundo de las aplicaciones Web también existen una cantidad de metáforas de amplia difusión como puede ser, por ejemplo, el celebre carrito de la compra que emplean casi todos los comercios online.

Pero las metáforas no sólo dependen del tipo de aplicación (escritorio o Web) sino también del ámbito de la misma. Por ejemplo, el carrito de la compra es una metáfora conocida por todos pero, si nuestra aplicación no va a vender nada al usuario no resulta conveniente utilizarla ya que puede confundir.

En nuestra aplicación no hemos creído conveniente la realización de ninguna metáfora, pues la interfaz es muy sencilla e intuitiva.

5.3.2.3.- Prototipos.

En este apartado vamos a definir la estructura de nuestra interfaz con el usuario. Mediante la elaboración de prototipos se pretende esbozar lo que será la interfaz de usuario de nuestra aplicación. Dichos prototipos no expresan el diseño final, simplemente una idea de lo que será nuestro sistema. Estos prototipos serán susceptibles de cambio durante el proceso de implementación.

Los prototipos de papel son una forma de crear una imagen palpable de lo que será una futura aplicación o sitio web. Su creación y manipulación es rápida y elástica. Además permite a los usuarios imaginarse lo que será la futura aplicación en funcionamiento sin interferencias de tipo:

- *Técnico*: plataformas, utilización de elementos multimedia como Flash…

- *Gráfico*: colores, tipografía que en muchas ocasiones desvían el centro de la discusión hacia asuntos cuya importancia es muy relativa en el proceso inicial. Nos podemos encontrar debatiendo sobre el tono de verde de una pantalla (aspecto secundario) frente a lo principal: secciones, datos necesarios, utilización futura del

producto. Como símil en la "vida real" sería como discutir acerca de la pintura de un coche, cuando ni siquiera sabemos cómo va a ser.

Es decir, se centra el debate en la esencia y concepto de la futura aplicación: la funcionalidad, el contenido y el flujo de pantallas. Permite el desarrollo de maquetas rápido y sin esfuerzo en escribir código (HTML, Visual Basic...).

Los prototipos en papel nos ayudarán a ver que los contenidos de las páginas son los apropiados sin la necesidad de redactar contenidos finales (basta con transmitir el concepto de lo que será cada contenido mediante texto simulado), también servirán como una primera aproximación a la maquetación final de las pantallas que conformen la aplicación (sin necesidad de presentar una maquetación exacta de lo que será cada pantalla, ni colores e iconos).

Los participantes en el trabajo con prototipos en papel serán:

- *El experto*: es la persona que dirige la sesión y deberá tener conocimientos de usabilidad y diseño centrado en el usuario.

- *Usuarios*: representa a los usuarios finales de la aplicación y son quienes dirán si el funcionamiento es correcto. Además, ayudarán a encontrar y corregir la ubicación de los contenidos.

Las ventajas que nos aportan los prototipos en papel son las siguientes:

- *Fáciles de realizar*: No hay necesidad de escribir/saber código.

- *Económicos*: Permite la detección temprana de errores de concepto y problemas en cuanto a funcionalidades a un mínimo coste.

- *Independiente de tecnología y dispositivos*: No hay necesidad de herramientas específicas. Podemos realizar y corregir un prototipo de papel en cualquier parte con lápiz y papel.

- *Espontaneidad en críticas por parte de los usuarios envueltos en el proceso de diseño*: Ven claramente que no hay un diseño y permite una crítica más abierta sin centrarse en aspecto visual o detalles puramente técnicos.

Un principio general es que, los elementos que posee la interfaz tienen que estar colocados de tal modo que, a la hora de realizar una tarea, el usuario deba recorrer la interfaz en la misma dirección que lee un texto, es decir, de izquierda a derecha y de arriba abajo. En países orientales no se realizaría de la misma forma.

Debido a la sencillez de la aplicación desarrollada ésta sólo se compone de una sola pantalla donde se van mostrando al usuario los resultados obtenidos. Dicho esto, pasamos a presentar la pantalla de la aplicación.

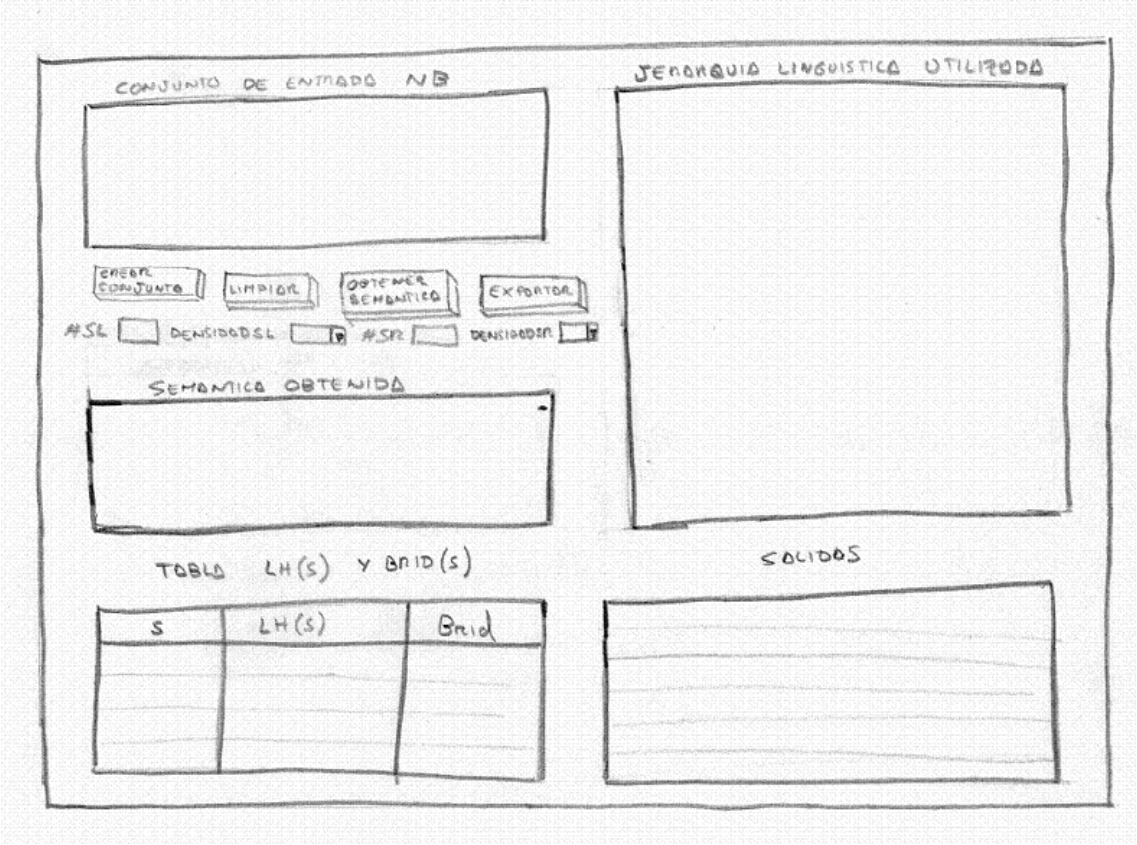

Figura 5.6: Pantalla principal de la aplicación

5.3.2.4.- Caminos de navegación.

Hasta este momento tenemos un diseño visual de la interfaz estática, es decir, cada pantalla diseñada individualmente, pero no tenemos una idea de si en el conjunto de la interacción, la acción va a transcurrir de forma comprensible para el usuario. Para ello vamos a diseñar la interfaz en movimiento y comprobar que es usable.

Para estudiar los caminos de interacción se empleará una herramienta llamada *storyboard*, que consiste en mostrar, a modo de secuencia, las distintas pantallas por las que se va pasando al realizar el usuario una determinada acción sobre la aplicación. En nuestro caso la aplicación sólo está compuesta por una única pantalla, de esta forma lo que mostraremos será los pasos en los que se muestran los resultados al usuario conforme avanza la interacción con el sistema. Mediante flechas se ayuda a entender que es lo que ha desencadenado el paso de una pantalla a otra.

La técnica del *Storyboarding* nos resultará altamente útil para describir los escenarios de situaciones concretas que ayuden a entender partes del sistema. Con los *storyboards* se consigue dotar al escenario descrito en lenguaje natural de la componente gráfica que facilita la comprensión y el detalle. Además, podemos analizar las pantallas de este escenario para ver si los pasos que se dan son los correctos y si la acción se entiende bien, de forma que podría ser necesario añadir más pantallas intermedias o, por el contrario, podría convenir fusionar algunas pantallas. Los *storyboard* nos facilitan esta tarea ya que nos permiten ver juntas todas las pantallas asociadas a una tarea o escenario.

A continuación presentamos el *storyboard* diseñado que muestra el proceso generado en por la aplicación para la introducción de un conjunto no balanceado y la obtención de la semántica asociada:

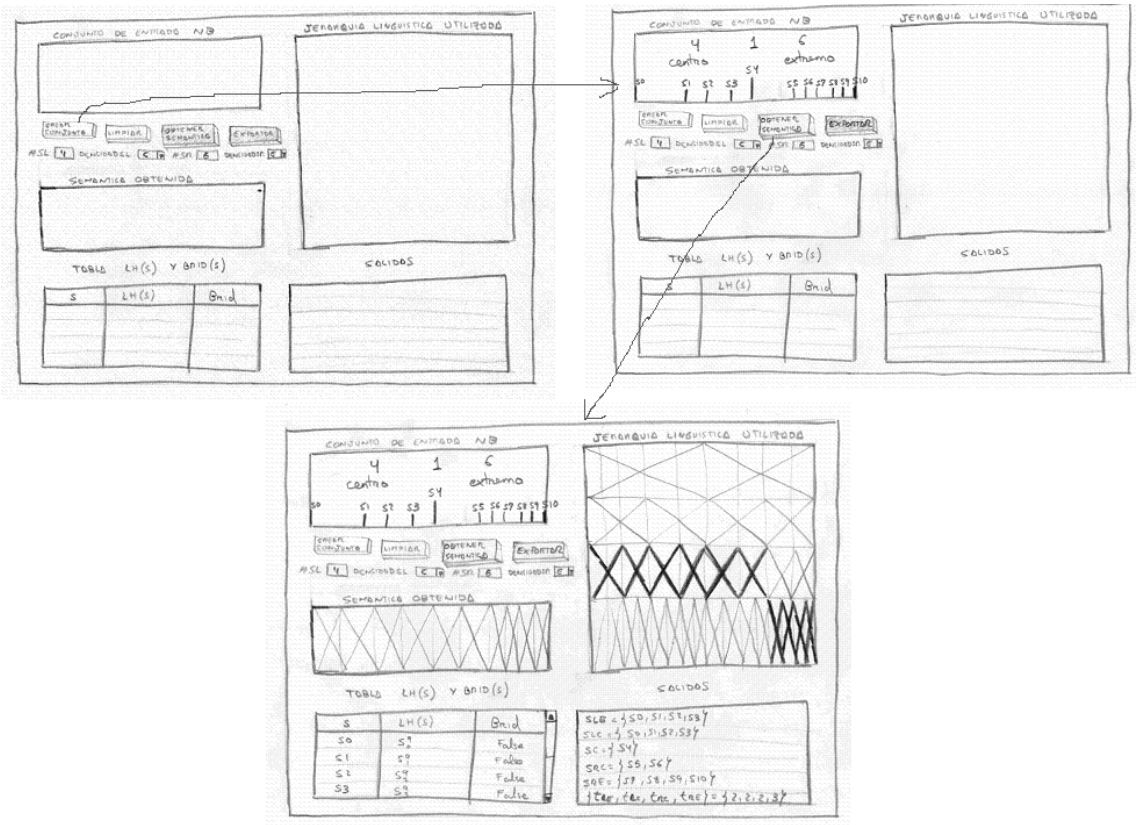

Figura 5.7: Storyboard del sistema

5.4.- Implementación

La implementación es la actividad final de la Ingeniería del Software, aquella en la que el modelo obtenido en las actividades anteriores se debe transformar en código fuente. Para ello se debe ser cuidadoso en la elección del lenguaje de programación empleado para la codificación y de la herramienta utilizada para generarla.

5.4.1. Tipo de arquitectura de la aplicación.

En nuestro caso, vamos a desarrollar una sencilla aplicación de escritorio. El funcionamiento de las arquitecturas de este tipo es sencilla: la aplicación se encuentra instalada en el ordenador a la que los usuarios acceden de forma local, sin necesidad de disponer de una conexión a Internet ni acceder a un servidor local o externo.

5.4.2. Lenguaje de programación utilizado.

La implementación de esta aplicación se ha realizado usando Java. Java es un lenguaje de programación orientado a objetos desarrollado por Sun Microsystems a principios de los años 90. El lenguaje en sí mismo toma mucha de su sintaxis de C y C++, pero tiene un modelo de objetos más simple y elimina herramientas de bajo nivel, que suelen inducir a muchos errores, como la manipulación directa de punteros o memoria.

Las aplicaciones Java están típicamente compiladas en un bytecode, aunque la compilación en código máquina nativo también es posible. En el tiempo de ejecución, el bytecode es normalmente interpretado o compilado a código nativo para la ejecución, aunque la ejecución directa por hardware del bytecode por un procesador Java también es posible.

La implementación original y de referencia del compilador, la máquina virtual y las librerías de clases de Java fueron desarrollados por Sun Microsystems en 1995. Desde entonces, Sun ha controlado las especificaciones, el desarrollo y evolución del lenguaje a través del Java Community Process, si bien otros han desarrollado también implementaciones alternativas de estas tecnologías de Sun, algunas incluso bajo licencias de software libre.

Entre noviembre de 2006 y mayo de 2007, Sun Microsystems liberó la mayor parte de sus tecnologías Java bajo la licencia GNU GPL, de acuerdo con las especificaciones del Java Community Process, de tal forma que prácticamente todo el Java de Sun es ahora software libre (aunque la biblioteca de clases de Sun que se requiere para ejecutar los programas Java todavía no es software libre).

5.4.3. Herramienta de desarrollo.

El entorno de desarrollo elegido ha sido Netbeans 4.1. NetBeans se refiere a una plataforma para el desarrollo de aplicaciones de escritorio usando Java y a un entorno de desarrollo integrado (IDE) desarrollado usando la Plataforma NetBeans.

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos.

Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

NetBeans es un proyecto de código abierto de gran éxito con una gran base de usuarios, una comunidad en constante crecimiento, y con cerca de 100 socios en todo el mundo. Sun MicroSystems fundó el proyecto de código abierto NetBeans en junio 2000 y continúa siendo el patrocinador principal de los proyectos.

5.4.4. Instalación de máquina virtual de java y funcionamiento.

La instalación de la máquina virtual viene documentada paso a paso en el Anexo I.

Por su parte, en el Anexo II se encuentra disponible el manual de usuario.

Capítulo 6

CONCLUSIONES.

A continuación revisamos cuáles han sido las principales conclusiones durante el desarrollo de esta memoria.

El propósito de este proyecto era la implementación una interfaz software para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información. Para ello se ha implementado la metodología presentada en el artículo "*A Fuzzy Linguistic Methodology To Deal With Unbalanced Linguistic Term Sets"* para representar, administrar y lograr el proceso de computación con palabras en conjuntos de términos lingüísticos no balanceados.

Como punto más importante de esta memoria cabe destacar que se ha presentado el desarrollo de una metodología para tratar con información lingüística no balanceada, es decir, información lingüística expresada en conjuntos de términos lingüísticos cuyas etiquetas no están distribuidas uniforme y simétricamente a ambos lados de la etiqueta central. Esta metodología está basada en el concepto de jerarquía lingüística y en el modelo de representación lingüística de las 2-tuplas. Esta metodología está compuesta por un algoritmo de representación y una aproximación computacional para información lingüística no balanceada. Hemos implementado el citado algoritmo para generar semántica en conjuntos de etiquetas no balanceados sin pérdida de información a través de una interfaz software. El resultado obtenido es exportado en formato XML para poder ser utilizado por otras aplicaciones o soportes.

La metodología presentada en esta memoria es muy útil para modelar diferentes problemas de la vida real tratando con términos lingüísticos expresados en conjuntos de términos lingüísticos, tales como evaluación de procesos, toma de decisiones, etc.

BIBLIOGRAFÍA.

[1] M. Roubens, "Fuzzy sets and decision analisis," Fuzzy Sets Syst., vol. 90, pp 199.199-206, 1997.

[2] M. Delgado, J.L. Verdegay, and M.A. Vila "Linguistic decisión making models," Int. J. Intell. Syst., vol. 7, pp 479-492, 1992.

[3] G. Bordogna, G. Passi, and M. Fedrizzi, "A linguistic modelling of consensus in group decision making based on OWA operators," IEEE Trans. Syst., Man, Cybern. A, vol. 27, pp. 126-132, Jan. 1997.

[4] S.M. Chen, "A new method for tool steel materials selection under fuzzy environment," Fuzzy Sets Syst., vol. 92, pp. 265-274, 1997.

[5] F. Herrera López, C. Mendaña, and M.A. Rodríguez, "A linguistic decisión model for personnel management solved with a linguistic biobjective genetic algorithm," Fuzzy Sets Syst., vol. 118, no. 1, pp. 47-64, 2001.

[6] O. Cordón, F. Herrera and I. Zwir, "Linguistic modelling by hierarchical systems of linguistic rules," Dept. Computer Sciences and A.I. Granada Univ., Spain.

[7] "Hierarchical knowledge bases for fuzzy rule-based systems," in Proc. IPMU2000 Conf., Madrid, Spain, 2000, pp. 1770-1777.

[8] S. Zadrozny and J. Kacprzyk, "Computing with words for text processing: An approach to the text categorization," *Inf. Sci.*, vol. 176, no. 4, pp. 415–437, 2006.

[9] J. M. Benítez, J. C. Martín, and C. Román, "Using fuzzy number for measuring quality of service in the hotel industry," *Tourism Manag.*, vol. 28, no. 2, pp. 544–555, 2007.

[10] Y.-L. Kuo and C.-H. Yeh, "Evaluating passenger services of asia-pacific international airports," *Transp. Res. E, Logist. Transp. Rev.*, vol. 39, no. 1, pp. 35–48, 2003.

[11] S. L. Chang, R. C. Wang, and S. Y. Wang, "Applying fuzzy linguistic quantifier to select supply chain partners at different phases of product life cycle," *Int. J. Prod. Econ.*, vol. 100, no. 2, pp. 348–359, 2006.

[12] S. L. Chang, R. C. Wang, and S. Y. Wang, "Applying a direct multigranularity linguistic and strategy-oriented aggregation approach on the assessment of supply performance," *Eur. J. Oper. Res.*, vol. 177, no. 2, pp. 1013–1025, 2007.

[13] C.-T. Chen, C.-T. Lin, and S.-F. Huang, "A fuzzy approach for supplier evaluation and selection in supply chain management," *Int. J. Prod. Econ.*, vol. 102, no. 2, pp. 289–301, 2006.

[14] C. H. Cheng and Y. Lin, "Evaluating the best main battle tank using fuzzy decision theory with linguistic criteria evaluation," *Eur. J. Oper. Res.*, vol. 142, pp. 174–186, 2002.

[15] L. Martínez, "Sensory evaluation based on linguistic decision analysis," *Int. J. Approx. Reason.*, vol. 44, no. 2, pp. 148–164, 2007.

[16] L. Martínez, J. L. Da Ruan, and J. B. Yang, "Dealing with heterogeneous information in engineering evaluation processes," *Inf. Sci.*, vol. 177, no. 7, pp. 1533–1542, 2007.

[17] F. Herrera and E. Herrera-Viedma, "Aggregation operators for linguistic weighted information," *IEEE Trans. Syst., Man Cybern. A,* [18] J. I. Peláez and J. M. Doña, "Lama:A linguistic aggregation of majority additive operator," *Int. J. Intell. Syst.*, vol. 18, no. 7, pp. 809–820, 2003.

[19] V. Torra, "Negation function based semantics for ordered linguistic labels," *Int. J. Intell. Syst.*, vol. 11, pp. 975–988, 1996.

[20] V. Torra, "Aggregation of linguistic labels when semantics is based on antonyms," *Int. J. Intell. Syst.*, vol. 16, pp. 513–524, 2001.

[21] D. Ben-Arieh and C. Zhifeng, "Linguistic labels aggregation and consensus measure for autocratic decision-making using group recommendations," *IEEE Trans. Syst., Man, Cybern. A, Syst. Humans*, vol. 36, no. 3, pp. 558–568, 2006.

[22] G. Bordogna, M. Fedrizzi, and G. Pasi, "A linguistic modeling of consensus in group decision making based on OWA operators," *IEEE Trans. Syst., Man. Cybern. A: Syst. Humans*, vol. 27, pp. 126–132, 1997.

[23] E. Herrera-Viedma, L. Martínez, F. Mata, and F. Chiclana, "A consensus support system model for group decision-making problems with multi-granular linguistic preference relations," *IEEE Trans. Fuzzy Syst.*, vol. 13, no. 5, pp. 644–658, 2005.

[24] P. P. Bonissone and K. S. Decker*, Selecting Uncertainty Calculi and Granularity: An Experiment in Trading-Off Precision and Complexity*, ser. Uncertainty in Artificial Intelligence, L. H. Kanal and J. F. Lemmer, Eds., ed. Amsterdam, The Netherlands: North-Holland, 1986.

[25] R. Degani and G. Bortolan, "The problem of linguistic approximation in clinical decision making," *Int. J. Approx. Reason.*, vol. 2, pp. 143–162, 1988.

[26] F. Herrera, E. Herrera-Viedma, and J. L. Verdegay, "Direct approach

processes in group decision making using linguistic OWA operators," *Fuzzy Sets Syst.*, vol. 79, pp. 175–190, 1996.

[26] F. Herrera and L. Martínez, "A 2-tuple fuzzy linguistic representation model for computing with words," *IEEE Trans. Fuzzy Syst.*, vol. 8, no. 6, pp. 746–752, 2000.

[27] F. Herrera and L. Martínez, "The 2-tuple linguistic computational model. Advantages of its linguistic description, accuracy and consistency," *Int. J. Uncertain., Fuzz. Knowl.-Based Syst.*, vol. 9, pp. 33–49, 2001, Suppl.

[28] J. Liu, J. B. Yang, J. Wang, H. S. Sii, and Y. M. Wang, "Fuzzy rulebased evidential reasoning approach for safety analysis," *Int. J. General Syst.*, vol. 33, no. 2–3, pp. 183–204, 2004.

[29] H. S. Sii and J. Wang, "A subjective design for safety framework for offshore engineering products," in *Workshops on Reliability and Risk Based Inspection Planning and ESRA Technical Committee on Offshore Safety*, Zurich, Switzerland, 2000.

[30] V. Torra, "Knowledge based validation: Synthesis of diagnoses through synthesis of relations," *Fuzzy Sets Syst.*, vol. 113, no. 2, pp. 167–176, 2000.

[31] L.A. Zadeh, J. Kacprzyk. Fuzzy Logic for the Management of Uncertainty. John Wiley, New York, 1992.

[32] L.A. Zadeh. The concept of a linguistic variable and its applications to approximate reasoning. Part I. Information Sciences, 8, 199-249, 1975. Part II, Information Sciences, 8, 301-357, 1975. Part III, Information Sciences, 9, 43-80, 1975.

[33] E. Herrera-Viedma, E. Peis. Evaluating the informative quality of documents in SGML-format using fuzzy linguistic techniques based on computing with words. Information Processing & Management, 39(2), 195-213, 2003.

[34] G. Bordogna, G. Pasi. An Ordinal Information Retrieval Model. International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems, 9(1), 63-75, 2001.

[35] E. Herrera-Viedma. Modeling the retrieval process of an information retrieval system using an ordinal fuzzy linguistic approach. J. of the American Society for Information Science and Technology, 52(6), 460-475, 2001.

[36] E. Herrera-Viedma. An information retrieval system with ordinal linguistic weighted queries based on two weighting elements. Int. J. of Uncertainty, Fuzziness and Knowledge-Based Systems, 9, 77-88, 2001.

[37] R. Degani, G. Bortolan. The Problem of Linguistic Approximation in Clinical Decision Making. Int. J. of Approximate Reasoning, 2, 143-162, 1988.

[38] B. Arfi. Fuzzy decision making in politics: A linguistic fuzzy-set approach (LFSA). Political Analysis, 13 (1), 23-56, 2005.

[39] F. Herrera, E. Herrera-Viedma, J.L. Verdegay. Direct approach processes in group decision making using linguistic OWA operators. Fuzzy Sets and Systems, 79, 175-190, 1996.

[40] V. Torra. Aggregation of Linguistic Labels when Semantics is based on Antonyms. International Journal of Intelligent Systems, 16, 513-524, 2001.

[41] R.R. Yager. On Ordered Weighted Averaging Aggregation Operators in Multicriteria Decision Making. IEEE Trans. on Systems and Cybernetics, 18 (1), 183-190, 1988.

[42] L.A. Zadeh. Fuzzy Sets. Information and Control, 8(3), 338-353, 1965.

[43] P.P. Bonissone, K.S. Decker. Uncertainty in artificial intelligence. Ch. Selecting Uncertainty Calculi and Granularity: an experiment in trading-off precision an complexity, L.H. Kanal and J.F. Lemmer. Eds. North-Holland, 217-247, 1986.

[44] G.A. Miller. The magical number seven or minus two: some limits on our capacity of processing information. Psychological Rev. 63, 81-97, 1956.

[45] P.P. Bonissone. *A fuzzy sets based linguistic approach: Theory and applications*, pages 329-339. Approximate Reasoning in Decision Analysis, North-Holland, 1982.

[46] G. Bordogna and G. Pasi. A fuzzy linguistic approach generalizing Boolean information retrieval: a model and its evaluation. *Journal of the American Society for Information Science*, 44(2):70-82, 1993.

[47] L.A. Zadeh. The concept of a linguistic variable and its applications to approximate reasoning. *Information Sciences, Part I, II, III*, 8,8,9:199-249,301-357,43-80, 1975.

[48] L.A. Zadeh. A computational approach to fuzzy quantifiers in natural languages. *Computer & Mathematics with Applications*, 9(1):149-184, 1983.

[49] F. Herrera and E. Herrera-Viedma. Linguistic decision analysis: steps for solving decision problems under linguistic information. *Fuzzy Sets and Systems*, 115(1):67-82, 2000.

[50] R.R. Yager. An approach to ordinal decision making. *International Journal of Approximate Reasoning*, 12(3-4):237-261, 1995.

[51] R.R. Yager. Aggregation operators and fuzzy system modelling. *Fuzzy Sets and Systems*, 67(2):129-145, 1993.

[52] A. Anzaldúa-Morales. *La evaluación sensorial de los alimentos en la teoría y la práctica*. Acribia, Zaragoza. España, 1994.

[53] J. Dombi. *Fuzzy logic and soft computing*, chapter A general framework for the utility-based and outranking methods, pages 202{208. In: B.B. Bouchon (Ed.). World Scientific, London, 1995.

[54] D. Dubois and H. Prade. *Fuzzy sets and systems: theory and applications*. Academic Press, New York, 1980.

[55] D. Dubois and H. Prade. Rough fuzzy-sets and fuzzy rough sets. *International Journal of General Systems*, 13(2-3):191{209, 1990.

[56] M. Delgado, J.L. Verdegay, and M.A. Vila. Linguistic decision making models. *International Journal of Intelligent Systems*, 7(5):479-492, 1992.

[57] M. Delgado, F. Herrera, E. Herrera-Viedma, and L. Martínez. Combining numerical and linguistic information in group decision making. *Information Sciences*, 107(1-4):177-194, 1998.

[58] F. Herrera and L. Martínez. The 2-tuple linguistic computational model. Advantages of its linguistic description, accuracy and consistency. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 9(Suppl.):33-49, 2001.

[59] F. Herrera and L. Martínez. A 2-tuple fuzzy linguistic representation model for computing with words. *IEEE Transactions on Fuzzy Systems*, 8(6):746-752, 2000.

[60] F. Herrera, E. Herrera-Viedma, L. Mart´ınez. An Information Retrieval System with Unbalanced Linguistic Information Based on the Linguistic 2-tuple Model. 8th International Conference on Information Processing and Management of Uncertainty in Knowledge-Bases Systems (IPMU'2002).Annecy (France), 23-29.

[61] F. Herrera, E. Herrera-Viedma, L. Martínez, P.J. Sánchez. A Methodology for Generating the Semantics of Unbalanced Linguistic Term Sets. 9th International Conference on Fuzzy Theory and Technology, Florida, 151-154, 2003.

[62] O. Cordón, F. Herrera, I. Zwir. Linguistic modelling by hierarchical systems of linguistic rules. IEEE Transactions on Fuzzy Systems, 10 (1), 2-20, 2001.

[63] F. Herrera, L. Martínez. A model based on linguistic 2-tuples for dealing with multigranularity hierarchical linguistic contexts in multiexpert decision-making. IEEE Transactions on Systems, Man and Cybernetics. Part B: Cybernetics, 31(2), 227-234, 2001.

[64] F. Herrera and L. Martínez, "An approach for combining linguistic and numerical information based on 2-tuple fuzzy representation model in decision-making," *Int. J. Uncertain., Fuzz. Knowl.-Based Syst.*, vol. 8, no. 5, pp. 539–562, 2000.

[65] E. H. Ruspini, "A new approach to clustering," *Inf. Contr.*, vol. 15, pp. 22–32, 1969.