RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

REPORTE TÉCNICO
Reconocimiento de Patrones

Estudio sobre algoritmos de agrupamiento jerárquico con funciones de semejanza no simétricas

Lic. Dustin L. Espinosa,
Dr. C. José Ruiz Shulcloper

RT_028 **junio 2010**

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

RNPS No. 2142
ISSN 2072-6287
Versión Digital

SERIE AZUL

CENATAV
Centro de Aplicaciones de Tecnologías de Avanzada
MINISTERIO DE LA INDUSTRIA BÁSICA

REPORTE TÉCNICO
Reconocimiento de Patrones

Estudio sobre algoritmos de agrupamiento jerárquico con funciones de semejanza no simétricas

Lic. Dustin L. Espinosa,
Dr. C. José Ruiz Shulcloper

RT_028 **junio 2010**

7ma. No. 21812 e/218 y 222,
Rpto. Siboney, Playa;
Ciudad de La Habana.
Cuba. C.P. 12200
www.cenatav.co.cu

Índice

Estudio sobre algoritmos de agrupamiento jerárquico con funciones de semejanza no simétricas

Lic. Dustin L. Espinosa, Dr. C. José Ruiz Shulcloper

Centro de Aplicaciones de Tecnología de Avanzada, 7a #21812 e/218 y 222, Siboney, Playa, Ciudad de La Habana, Cuba
despinosa@cenatav.co.cu

RT_ 028 CENATAV
Fecha del camera ready: 18 de febrero de 2010

Resumen: El presente trabajo es un estudio de los principales métodos para obtener una jerarquía de particiones cuando la función de semejanza entre los objetos es no simétrica. Los dos primeros epígrafes son introductorios y exponen los conceptos y definiciones principales de la materia. El estado del arte de los algoritmos que obtienen una jerarquía de particiones, asumiendo que la función de semejanza es no simétrica, es descrito en el tercer epígrafe. Primero se presentan los enfoques que simetrizan la función de similitud, señalando cada caso los principales aspectos a tener en cuenta para su aplicación. Posteriormente se analizan los algoritmos que trabajan directamente con la función no simétrica. En el cuarto epígrafe se muestran resultados experimentales de los principales algoritmos. Finalmente se ofrecen las conclusiones del trabajo, y posibles líneas de investigación futura.

Palabras clave: agrupamiento jerárquico, particiones, similitud no simétrica, simetrización

Abstract: This work is a study of the main methods for obtaining a hierarchy of partitions when the function of similarity between objects is not symmetric. The first two sections are introductory and presents the concepts and definitions in this field. The state of the art algorithms that derive a hierarchy of partitions, assuming the similarity function is not symmetrical, is described in the third section. It first presents the approaches that symmetrize the similarity function, pointing out in each case the main issues to consider for its application. Later, the algorithms that work directly with the non symmetric similarity function are analyzed. The experimental results of the main algorithms are shown in the fourth section. Last section concludes the work and mention authoress's future research.

Keywords: Hierarchical Clustering, Partitioning, Asymmetric Similarity, Symmetrization

1. Introducción

Con el desarrollo de la humanidad aumenta la cantidad de información que es generada cada día. Organizar y extraer la información útil, resulta vital en muchas disciplinas. Por esto, herramientas que brindan una organización en clases o grupos, donde elementos de una misma clase se *parecen* más que elementos de clases distintas, permitiendo descubrir relaciones *ocultas* entre los objetos, ganan mayor importancia. Muchas de estas herramientas nos ofrecen esta organización con diferentes niveles de abstracción, es decir, cada nivel nos brinda una visión más específica o más general de la información que los otros, lo que permite relacionar de forma jerárquica estas clases obtenidas y facilita el análisis desde varios puntos de vista.

Los algoritmos que obtienen este tipo de estructuración, se pueden dividir en dos clases, algoritmos de agrupamiento jerárquico y algoritmos de particiones anidadas. Ambos tipos de algoritmo producen una secuencia de estructuraciones, donde cada nivel de esta secuencia es más *fino*(vista más específica) o más *grueso*(vista más general) que los niveles anteriores. En la literatura es común no distinguir los algoritmos jerárquicos de los algoritmos de particiones anidadas, pero existen diferencias significativas a la hora de obtener la secuencia de estructuraciones que provocan esta distinción. Los jerárquicos generan cada nivel basados en un *criterio de agrupamiento*(propiedad), que define las relaciones que existen entre los elementos de cada clase, así como el tipo de estructuración. Cada nivel de la jerarquía se obtiene utilizando el mismo criterio de agrupamiento, y por lo general para obtener un nivel individual, es necesario hallar todos los anteriores. Así mismo, en cada nivel se obtiene el mismo tipo de estructuración, por lo general una partición, donde los objetos pueden pertenecer a una sola clase. Single-Link, Complete-Link, Average-Link, algoritmo de Ward, son ejemplos de los algoritmos de agrupamiento jerárquico más utilizados. Por su parte, los algoritmos de particiones anidadas siempre generan particiones, pueden aplicar más de un criterio de agrupamiento para obtener la secuencia y cada nivel se puede obtener de forma independiente. Nested Partition Method es un ejemplo de este tipo de algoritmo.

Estos algoritmos han encontrado aplicación en muchas áreas de la ciencia. En la biología se han utilizado para el estudio de taxonomías [1] y el estudio del genoma humano [2][3]. También se han aplicado en problemas de optimización, con la idea de acelerar la búsqueda del óptimo en aquellas regiones con mayores opciones de contener éste [4]. Además, encontramos uso de estos métodos en ramas muy diversas como las geo-ciencias [5], resúmenes de vídeos [6][7], y reconocimiento de voz [8]. Otras de las aplicaciones donde se han utilizado son el conteo de personas en imágenes de vídeo [9], segmentación de imágenes [10][11], selección de variables [12] y en problemas de regionalización [13].

Un punto esencial en el desarrollo de algoritmos de agrupamiento, es el concepto de semejanza (similitud) entre dos objetos, pues los criterios de agrupamiento se basan en esta semejanza para formar las clases. Generalmente, esta semejanza es calculada a través de una función, que calcula el grado de similitud entre dos objetos. En la mayoría de los casos esta función es simétrica, no obstante, desde los años 70 en el campo de psicología, varios investigadores crearon modelos que establecían que las funciones de semejanza podían ser no simétricas [14][15]. Los juicios de semejanza son extensiones de declaraciones del tipo "*i es como j*", las cuáles son direccionales y por lo general no iguales al inverso "*j es como i*" [14]. La no simetría se justifica por la importancia que tienen las características más distintivas y las más prominentes, ya que los humanos tienden a seleccionar dichas características para realizar las generalizaciones o particularizaciones. Por ejemplo, es más común decir *el retrato se parece a la persona* que *la persona se parece al retrato*, así como la expresión *el hijo se parece al padre* es más aceptada que el opuesto [16]. Las funciones de similitud no simétricas se han desarrollado en varios campos como la geo-semántica [17], el procesamiento de vídeos [18][19], la minería de datos [20][21], recuperación de texturas [22], química [23], análisis de secuencias de datos biológicos [24].

En la literatura encontramos muchas definiciones de semejanza, sin llegarse a un consenso por los autores sobre qué es una función de semejanza. Muchas de estas definiciones

incluyen restricciones como la simetría, que como vimos es violada frecuentemente. Un esfuerzo por brindar una definición general de función de semejanza, lo encontramos en [25].

Abordaremos en el presente trabajo los principales enfoques para obtener una jerarquía de particiones, cuando la función de semejanza es no simétrica. Para atacar este problema hay dos enfoques principales reportados en la literatura. El primero de éstos consiste en simetrizar, es decir obtener una función simétrica a partir de la original, siendo los procedimientos más usados el mínimo, el máximo y la media. La idea consiste en aplicar un algoritmo de agrupamiento jerárquico para funciones de similitud simétricas, utilizando esta nueva función simétrica.

En el segundo enfoque se trabaja directamente con la función no simétrica. Los algoritmos de este tipo representan la semejanza a través de un grafo dirigido, que permite modelar la dirección de la semejanza de forma natural, y todo el trabajo se realiza a partir de este grafo.

De estos dos enfoques el más utilizado es el primero. Esto se debe fundamentalmente a la existencia en la literatura de una gran variedad de algoritmos para el caso simétrico [26][27][28]. No obstante, debemos responder algunas preguntas antes de aplicar el enfoque de simetrización. ¿Qué función de simetrización es la que mejor se ajusta al algoritmo de agrupamiento que queremos usar? Para algoritmos basados en propiedades simples como máxima similitud o mínima similitud, tales como Single-Link o el Complete-Link, la respuesta pudiera ser simple, en otros casos no tanto. Otro punto es que al simetrizar obtenemos un valor del grado de semejanza entre A y B, lo que elimina la dirección y la posibilidad de establecer un referente en la comparación, en otras palabras representamos dos valores con uno. ¿Es admisible que el algoritmo de agrupamiento que se aplique, no haga uso de toda la información que aportaba la función no simétrica? Por último, la simetrización no sólo provoca esta pérdida de información, también provoca un cambio en la similitud entre dos objetos. Por ejemplo sea la S función de similitud no simétrica original, S^* la obtenida al simetrizar, y sean A y B tales que $S(A,B) \neq S(B,A)$, entonces se cumple una de dos:

1. $S^*(A,B) < S(A,B) \vee S^*(A,B) < S(B,A)$, se disminuye la semejanza real en al menos un sentido.
2. $S^*(A,B) > S(A,B) \vee S^*(A,B) > S(B,A)$, se aumenta la semejanza real en al menos un sentido.

lo que nos lleva a preguntarnos lo siguiente: ¿Qué implicaciones tiene este cambio en el algoritmo de agrupamiento? ¿Es admisible?

Los algoritmos del segundo enfoque han sido menos explorados. En los últimos años han aparecido varios trabajos reportados en la literatura [20][29]. Teniendo en cuenta las dificultades que presenta la simetrización y los positivos resultados obtenidos recientemente en el segundo enfoque, creemos que en un futuro este enfoque debe predominar.

El presente trabajo está organizado en cuatro epígrafes. En el segundo epígrafe expondremos cómo se representan los objetos, definimos los conceptos de función de semejanza, simétrica, asimétrica, y no simétrica, y por último veremos los conceptos de partición, jerarquía de particiones. El estado del arte de los algoritmos que obtienen una jerarquía de particiones, asumiendo que la función de semejanza es no simétrica, es descrito en el tercer epígrafe. En el cuarto se muestran resultados experimentales de los principales algoritmos. Finalmente se ofrecen las conclusiones del trabajo, y posibles líneas de investigación futura.

2. Conceptos y definiciones

2.1. Representación de los objetos

En este trabajo consideramos que los objetos bajo estudio pertenecen a un Universo U (ej. el universo de todas las noticias publicadas por la AFP en el año 2008, el universo de todos los animales, etc.). Generalmente estos objetos son entidades o conceptos que los humanos creamos, vemos o entendemos (ej. un documento, un tipo de suelo, una enfermedad, o una imagen).

Cuando los especialistas se enfrentan a un problema de reconocimiento de patrones, realizan una modelación matemática del mismo, de la cual se obtiene una representación de los objetos bajo estudio. Existen dos formas fundamentales de hacer esto, la primera, basada en teoría de gramáticas formales, cada objeto es representado como una secuencia de símbolos de un cierto alfabeto, y es típica del enfoque sintáctico-estructural. En la segunda, que será la que asumiremos en este trabajo, se describen los objetos a partir de un conjunto de variables (rasgos, características) no necesariamente numéricas, y es la utilizada en el enfoque estadístico y en el lógico-combinatorio.

En muchos casos todos los rasgos toman valores reales, y se pueden ver como puntos en $\mathbb{R}^m$ (siendo $\mathbb{R}$ el conjunto de los números reales), lo que permite utilizar la gran variedad de herramientas matemáticas definidas sobre espacios reales. En otras ocasiones todas las variables son nominales, y en ocasiones la modelación matemática nos lleva a tener rasgos numéricos y no numéricos juntos, lo que nos obliga a quedarnos con un simple producto cartesiano del que no podemos asumir ninguna estructura interna.

Definición 2.1. *Entendemos por una representación de un objeto $o \in U$ al m-uplo $I(o) = (r_1(o), r_2(o), \ldots, r_m(o))$, donde cada r_i representa un rasgo, $r_i : U \to R_i$, R_i es el conjunto de valores admisibles de la variable r_i y $r_i(o)$ representa el valor que toma la variable r_i en el objeto o. De esta forma una representación de un objeto o es un elemento del producto cartesiano $E = R_1 \times R_2 \times \ldots \times R_m$.*

Obsérvese que dos objetos pueden compartir la misma representación en un espacio de representación determinado, y un objeto puede tener varias representaciones en espacios diferentes. En vistas a simplificar la notación generalmente denotamos $I(o)$ como o.

Como dijimos anteriormente, asumimos que los conjuntos de valores admisibles R_i pueden ser numéricos y no numéricos. Además, suponemos que en R_i puede existir el símbolo $*$, que denota ausencia de información, es decir, la descripción de un objeto puede estar incompleta, en el sentido que al menos existe una variable para la cual no se conoce el valor que ella toma en este objeto.

Todos los algoritmos son concebidos sobre el espacio de representaciones. A partir de aquí asumiremos que $\Omega \subseteq E$ es el conjunto de representaciones de objetos (patrones) de interés.

2.2. Funciones de semejanza

Una vez que se tienen representados los objetos, debe ser definida la manera en que se va a medir la semejanza o parecido de cada uno de ellos con el resto. Usualmente esta semejanza

se calcula por medio de una función (medida) de distancia. Una medida de distancia como la Euclidiana puede utilizarse para reflejar disimilitud entre dos patrones, mientras que las medidas de similitud caracterizan la semejanza conceptual entre patrones[26].

Definición 2.2. *Sea Ω una muestra de objetos, $R = \{r_1, r_2, \ldots, r_m\}$ conjunto de rasgos que describen los elementos de Ω, L un conjunto totalmente ordenado con un elemento mínimo m y un elemento máximo M. Decimos que una función $\Gamma : \Omega \times \Omega \to L$ es una función de similitud (semejanza) ssi Γ satisface las siguientes propiedades:*

1. $\forall o_i \in \Omega$, $\Gamma(o_i, o_i) = M$.
2. $\forall T = \{r_{i_1}, r_{i_2}, \ldots, r_{i_p}\}$, $T \subset R$, $T \neq \emptyset$, $\Omega_T = R_{i_1} \times R_{i_2} \times \ldots \times R_{i_p}$ *se cumple que:* $\Gamma : \Omega_T \times \Omega_T \to L$ *es una función de similitud.*

Además si:

- $\forall o_i, o_q \in \Omega$, $\Gamma(o_i, o_q) = \Gamma(o_q, o_i)$ *se dice que* Γ *es una función de similitud simétrica.*
- $\exists o_i, o_q \in \Omega : \Gamma(o_i, o_q) \neq \Gamma(o_q, o_i)$ *se dice que* Γ *es una función de similitud no simétrica.*
- $\forall o_i, o_q \in \Omega$, $\Gamma(o_i, o_q) \neq \Gamma(o_q, o_i)$ *se dice que* Γ *es una función de similitud asimétrica.*

Una función de disimilitud es el opuesto de una función de similitud. Por lo tanto, una distancia es un caso particular de disimilitud[25]. La función de similitud debe escogerse tras un cuidadoso análisis en la modelación matemática del problema, y debe estar en correspondencia con la representación utilizada. No podemos utilizar la distancia Euclidiana cuando los datos son nominales, y en muchos trabajos se ha visto la ineficacia de codificar los datos con miras a utilizar algunas herramientas matemáticas. Por ejemplo, si tenemos el rasgo temperatura ={*normal*, *fiebre_ leve*,*fiebre_ alta*} y lo codificamos *normal* = 1, *fiebre_ leve* = 2, *fiebre_ alta* = 3, que médico aceptaría que *normal* + *fiebre_ leve* = *fiebre_ alta* (ni siquiera está claro que esta suma tenga sentido).

2.2.1. Ejemplos de funciones de semejanza no simétricas

La función de semejanza MDSM[1] es un ejemplo de medida no simétrica que podemos encontrar en el campo de la geo-semántica [17]. Aquí los objetos están descritos por tres rasgos: partes, funciones, atributos, todos de tipo conjunto. La función de similitud global Γ es la suma pesada de las similitudes por partes (Γ_p), funciones(Γ_f) y atributos(Γ_a) respectivamente.

$$\Gamma(o_i, o_q) = w_p\Gamma_p(o_i, o_q) + w_f\Gamma_f(o_i, o_q) + w_a\Gamma_a(o_i, o_q) \tag{1}$$

Para cada rasgo $t \in \{p, f, a\}$ se utiliza la función $\Gamma_t(o_i, o_q)$ (Ecuación 2), basada en el modelo de *coincidencias por características* de Tversky [14]. En $\Gamma_t(o_i, o_q)$, T_i es el valor que toma el rasgo t en o_i ($t(o_i)$), $\alpha(o_i, o_q)$ es una función simétrica que toma valores en el intervalo $[0, 0{,}5]$ y calcula la importancia de las características distintivas de dos objetos, tomando en cuenta que las diferencias tienen una menor importancia en una evaluación de la semejanza. $\Gamma_t(o_i, o_q) \neq \Gamma_t(o_q, o_i) \iff \alpha(o_i, o_q) \neq 0{,}5$.

$$\Gamma_t(o_i, o_q) = \frac{|T_i \cap T_q|}{|T_i \cap T_q| + \alpha(o_i, o_q)|T_i \setminus T_q| + (1 - \alpha(o_i, o_q))|T_q \setminus T_i|} \tag{2}$$

[1] MDSM:Matching-Distance Similarity Measure

En el área del procesamiento de vídeo encontramos una función de disimilitud no simétrica cuyo objetivo es calcular la disimilitud entre dos objetos (de arbitraria forma) en secuencias de vídeos[18]. Cada objeto se representa como una colección de $VOPs$[2], un VOP, por cada imagen de la secuencia. Un VOP es un vector de rasgos numéricos, que describe la forma y la textura del objeto en un instante particular de la secuencia de vídeo.

$$d(o_i, o_j) = \frac{1}{N} \sum_{v \in V_i} \arg\ \min_x \{d(v, v_j) \,/\, v_j \in V_j\}$$

donde N es el tamaño de la secuencia de vídeo, V_t es el conjunto de VOP de un objeto o_t y d representa la distancia euclidiana. Otras medidas no simétricas en la misma área las podemos encontrar en [6][19].

La minería de datos es otro campo donde se han aplicado las funciones no simétricas. En [30] encontramos la definición de dos funciones para medir la (di)similitud de dos palabras. La primera se denomina *Similitud Lógica Difusa*[3] (3), utilizada en recuperación de información [31], y la segunda es la medida probabilística *Divergencia de Kullback-Leibler*[4] (4).

Similitud Lógica Difusa:

$$s_{ij} = \frac{|L_{1_i} \cap L_{1_j}|}{|L_{1_i}|} \tag{3}$$

donde L_{1_t} es el conjunto de los documentos indexados por la palabra t. En [32] se dice que esta medida puede ser interpretada como el grado con el que el tema i es un subconjunto del tema j.

Divergencia de Kullback-Leibler:

$$d_{ij} = \sum_{t \in W} p(t|i) \log \left(\frac{p(t|i)}{p(t|j)} \right) \tag{4}$$

donde W es el conjunto de todas las palabras bajo estudio y $p(a|b)$ representa la probabilidad de que la palabra a aparezca junto a la palabra b. d_{ij} obtiene valores pequeños si las palabras que aparecen junto a la palabra i, también aparecen junto a la palabra j.

La minería de datos es quizas donde más ejemplos de este tipo de funciones existen [20][21]. En muchas otras áreas se han aplicado las funciones de (di)similitud no simétricas. Algunas aplicaciones las podemos encontrar en recuperación de texturas [22], química[23], análisis de secuencias de datos biológicos [24].

2.3. Jerarquía de particiones

Como dijimos en la introducción existen dos formas de obtener una secuencia de estructuraciones, a través de un algoritmo jerárquico o utilizando un algoritmo de particiones anidadas. La diferencia entre estos métodos está en que los jerárquicos aplican un único

[2] Video Object Plane
[3] Fuzzy Logic Similarity
[4] Kullback Leibler Divergence

criterio de agrupamiento, y por lo general para obtener un nivel de la secuencia es necesario obtener todos los niveles anteriores. Los algoritmos de particiones anidadas pueden emplear más de un criterio de agrupamiento y cada nivel de la jerarquía se obtiene de forma independiente. En este trabajo estamos interesados en los métodos jerárquicos y este epígrafe introduce las definiciones y conceptos más relevantes.

Definición 2.3. *Sea el conjunto* Ω*, una partición,* Φ*, de* Ω*, divide* Ω *en los subconjuntos* $\{C_1, C_2, \ldots, C_k\}$ *tal que:*

- $C_i \cap C_q = \emptyset \quad i, q = \overline{1,k},\ i \neq q$
- $C_1 \cup C_2 \cup \ldots \cup C_k = \Omega$ *[28]*

Definición 2.4. *Una partición* Φ *está anidada en la partición* Ψ *si cada elemento de* Φ *es un subconjunto de un elemento de* Ψ*.*

Definición 2.5. *Una jerarquía de particiones anidadas sobre el conjunto* Ω *es una secuencia finita de particiones* $P_1, P_2, \ldots, P_n$ *sobre* Ω *tal que cada partición* P_{i+1} *está anidada en* P_i*.*

Un algoritmo de agrupamiento jerárquico es un algoritmo que construye una secuencia de particiones anidadas sobre el conjunto Ω. Los algoritmos de agrupamiento jerárquico se dividen en aglomerativos y divisivos. Un algoritmo aglomerativo empieza ubicando los n ($n = |\Omega|$) objetos en n grupos unitarios. Luego el criterio de agrupamiento indica la forma en que se utiliza la función de (di)similitud para mezclar dos o más de estos grupos triviales, anidando la partición trivial en una segunda partición. Este proceso se repite para formar una secuencia de particiones anidadas, donde el número de grupos disminuye a medida que aumenta la secuencia, hasta quedar una partición con un solo grupo que contiene todos los elementos. Un algoritmo divisivo realiza el proceso inverso[28].

Cuando nos enfrentamos a un problema de agrupamiento jerárquico, partimos del hecho que no existe un algoritmo capaz de lidiar con la gran diversidad de estructuras presentes en las bases de datos multidimensionales [26]. Por lo que necesitamos escoger el algoritmo que mejor se adecue a nuestro problema particular. Existe una basta literatura al respecto cuando la función de semejanza es simétrica [27][26][28]. Sin embargo, para el caso no simétrico, esto no ha sido tratado en la literatura. Esto motiva en muchos casos a que en presencia de una función de similitud no simétrica, se aplique algún tipo de simetrización para obtener una simétrica y poder utilizar entonces, un algoritmo para funciones simétricas.

Tabla 1. Todos los posibles casos que ocurren al simetrizar, cuando $\Gamma_{iq} < \Gamma_{qi}$. $L = [0, 1]$

Caso	Relación entre S y Γ	Descripción + Ejemplo
1	$(S_{iq} < \Gamma_{iq}) \wedge (S_{iq} < \Gamma_{qi})$	Se disminuye la semejanza original tanto de o_i a o_q como de o_q a o_i. ej. $S_{iq} = \text{Prom}(\Gamma_{iq}, \Gamma_{qi}) \cdot \text{Min}(\Gamma_{iq}, \Gamma_{qi})$
2	$(S_{iq} = \Gamma_{iq}) \wedge (S_{iq} < \Gamma_{qi})$	Se disminuye la semejanza original de o_q a o_i. ej. $S_{iq} = \text{Min}(\Gamma_{iq}, \Gamma_{qi})$
3	$(S_{iq} > \Gamma_{iq}) \wedge (S_{iq} < \Gamma_{qi})$	Se aumenta la semejanza original de o_i a o_q y se disminuye la semejanza original de o_q a o_i. ej. $S_{iq} = \text{Prom}(\Gamma_{iq}, \Gamma_{qi})$
4	$(S_{iq} > \Gamma_{iq}) \wedge (S_{iq} = \Gamma_{qi})$	Se aumenta la semejanza original de o_i a o_q. ej. $S_{iq} = \text{Max}(\Gamma_{iq}, \Gamma_{qi})$
5	$(S_{iq} > \Gamma_{iq}) \wedge (S_{iq} > \Gamma_{qi})$	Se aumenta la semejanza original tanto de o_i a o_q como de o_q a o_i. ej. $S_{iq} = \Gamma_{iq} + \Gamma_{qi}$

Las principales consideraciones a tener en cuenta al simetrizar, es la pérdida de información y el falseo de la similitud. La pérdida de información viene dada por el hecho que, con la función original se puede establecer una dirección y un referente en la comparación, lo que permite establecer una diferencia entre la similitud "*de* o_i *a* o_q" y la similitud "*de* o_q *a* o_i". Con la función simétrica obtenemos valor de la similitud "*entre* o_i *y* o_q", lo que elimina la posibilidad de establecer una dirección y por consiguiente un referente, imposibilitando la diferenciación entre la similitud "*de* o_i *a* o_q" y la similitud en el sentido opuesto. Por otro lado, al tener que utilizar un sólo valor para expresar la similitud en ambos sentidos, necesariamente se falsea (se obtiene un valor diferente a la similitud original) la similitud en al menos un sentido.

Tabla 2. Ejemplo de matriz de similitud no simétrica

objetos	o_1	o_2	o_3	o_4
o_1	1	0.71	0.62	0.4
o_2	0.33	1	0.35	0.44
o_3	0.43	0.18	1	0.59
o_4	0.4	0.44	0.6	1

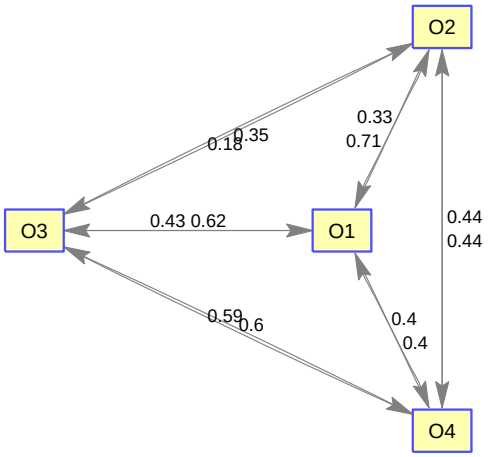
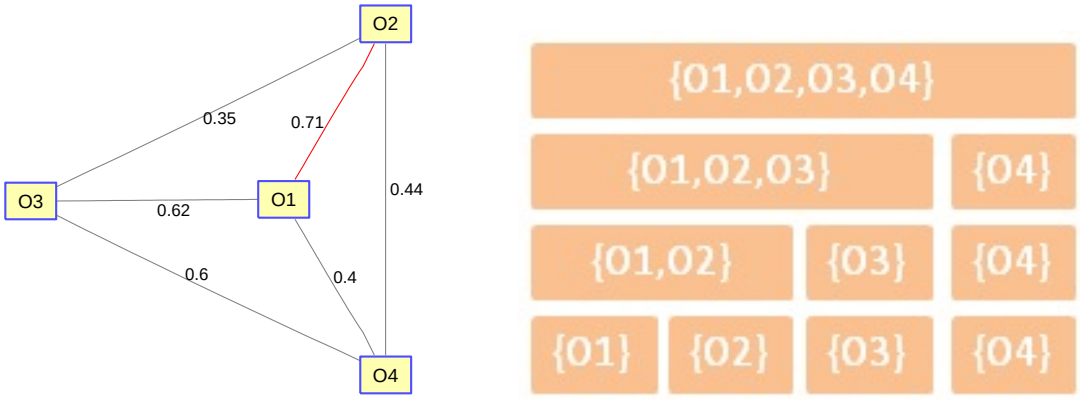


Fig. 1. Grafo dirigido formado a partir de las similitudes en la tabla 2

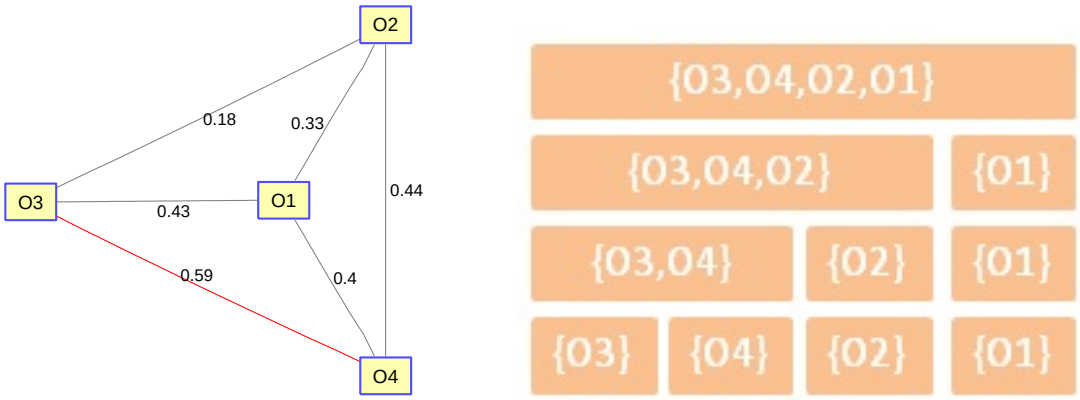
En la tabla 1 se muestran las posibles modificaciones, con respecto a la semejanza, que ocurren al simetrizar, para cada par de objetos o_i, o_q, cuando $\Gamma_{iq} < \Gamma_{qi}$. En todos los ejemplos se asume como L al intervalo cerrado $[0, 1]$, además para simplificar la notación, de aquí en adelante utilizaremos Γ_{iq} en vez de $\Gamma(o_i, o_q)$. En la tabla 2 se muestra una pequeña matriz no simétrica, que representa la similitud de 4 objetos. La figura 1 muestra esta matriz representada en un grafo dirigido. A su vez las figuras 2, 3 y 4 muestran los grafos que se forman al simterizar utilizando máximo, mínimo y promedio respectivamente y la jerarquía obtenida al aplicar el algoritmo Single-Link en cada caso. Cuando analizamos este pequeño ejemplo, se puede observar que ninguna jerarquía es igual. Esto muestra que Single-link hace una interpretación diferente de la similitud, para cada simetrización. Buscar cuál es la mejor simetrización para un algoritmo determinado, depende del problema particular que se quiera resolver. Por ejemplo, si para nuestro problema son más relevantes las características

comunes de los objetos, entonces para un algoritmo como Single-Link, que está basado en máximas similitudes, parece más adecuado con la *esencia* del problema simetrizar vía máximo. Si por el contrario buscamos separar los objetos más disimilares, parece el mínimo el más adecuado. Pero cuando queremos algo más *complejo*, no parece que exista una simetrización más adecuada, véase en el ejemplo, que la partición $\{\{o_1, o_2\}, \{o_3, o_4\}\}$ nunca es obtenida, y sin embargo ambos grupos parecen tener importancia, pues son los grupos que primero se forman en las jerarquías de las figuras 2, 3 y 4 ($\{o_1, o_2\}$ para máximo, $\{o_3, o_4\}$ para los otros casos).



(a) Grafo **(b)** Jerarquía

Fig. 2. Grafo que se forma al simetrizar con máximo, y jerarquía obtenida con Single-Link

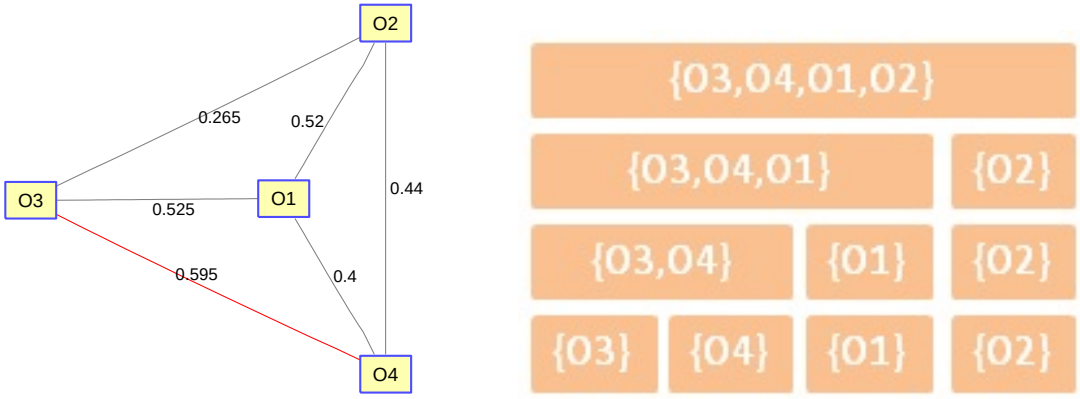


(a) Grafo **(b)** Jerarquía

Fig. 3. Grafo que se forma al simetrizar con mínimo, y jerarquía obtenida con Single-Link

Las implicaciones de la pérdida y falseo de información a la hora de agrupar, dependen del problema en particular con el que se esté tratando. Sería una tarea monumental realizar un estudio exhaustivo al respecto, que abarque cada algoritmo de agrupamiento jerárquico para funciones de semejanza simétricas, reportado en la literatura. Cuando con simples simetrizaciones no podemos obtener resultados satisfactorios, la opción más recomendable sería aplicar un algoritmo que trabaje directamente con la función no simétrica. Buscar

un método de simetrización más complejo, puede tener igual o mayor complejidad que la opción anterior, con lo que se pierde la única ventaja que puede tener simetrizar, que es exactamente facilitar el trabajo.



(a) Grafo **(b)** Jerarquía

Fig. 4. Grafo que se forma al simetrizar con promedio, y jerarquía obtenida con Single-Link

3. Métodos que obtienen una jerarquía de particiones, cuando la función de semejanza es no simétrica

En este epígrafe mostramos los principales algoritmos desarrollados para obtener una jerarquía de particiones, a partir de un conjunto de objetos, teniendo en cuenta que la función de (di)similitud entre estos objetos es no simétrica. Al final de cada algoritmo exponemos nuestras consideraciones del mismo, así como su costo computacional.

3.1. MIN y MAX. Extensiones de Hubert

El primer intento de atacar el problema de obtener una jerarquía de particiones, teniendo en cuenta el caso de la no simetría de la función de (di)similitud entre dos objetos, fue en el trabajo de Hubert [33]. Hubert propone en su trabajo una extensión a los algoritmos jerárquicos para el caso simétrico presentados por Johnson en [34].

Los algoritmos de Johnson trabajan con una función de disimilitud $d : \Omega^2 \to \mathbb{R}$, de la cual asumen las siguientes propiedades:

- $d(o_i, o_j) \geq 0$, positividad
- $d(o_i, o_j) = 0 \Leftrightarrow o_i = o_j$, nulidad
- $d(o_i, o_j) = d(o_j, o_i)$, simetría.

Estos algoritmos construyen la jerarquía de forma aglomerativa. Como se explicó en el capítulo anterior los algoritmos aglomerativos comienzan colocando a cada objeto como un grupo unitario, y a partir de ahí en cada paso se busca el par de grupos *menos disimilar*[5]

[5] Cuando tratamos con una función de **disimilitud** buscamos el valor mínimo y cuando tratamos con una función de **similitud** buscamos el máximo valor.

utilizando una función de disimilitud entre grupos (criterio de agrupamiento). Estos grupos se mezclan formando un nuevo grupo, el proceso termina cuando todos los objetos están en el mismo grupo.

Los algoritmos de Johnson, denominados MIN y MAX, fueron concebidos para tratar el caso simétrico y se diferencian en la forma de definir la función de disimilitud entre grupos:

MIN $D_{MIN}(C_p, C_q) = \min_{\substack{o_i \in C_p \\ o_j \in C_q}} \{d(o_i, o_j)\}$

MAX $D_{MAX}(C_p, C_q) = \max_{\substack{o_i \in C_p \\ o_j \in C_q}} \{d(o_i, o_j)\}$

siendo d la función de disimilitud comentada arriba.

Algoritmo 1 MIN

Entrada: Ω conjunto de objetos, d función de disimilitud.

1. Por cada objeto crear un grupo unitario.
2. **mientras** todos los objetos no estén en un mismo grupo. **hacer**
3. Buscar el par de grupos C_p, C_q donde alcanza el mínimo $D_{MIN}(C_i, C_j)$.
4. Mezclar los grupos C_p y C_q en un solo grupo.
5. **fin mientras**

Algoritmo 2 MAX

Entrada: Ω conjunto de objetos, d función de disimilitud.

1. Por cada objeto crear un grupo unitario.
2. **mientras** todos los objetos no estén en un mismo grupo. **hacer**
3. Buscar el par de grupos C_p, C_q donde alcanza el mínimo $D_{MAX}(C_i, C_j)$.
4. Mezclar los grupos C_p y C_q en un solo grupo.
5. **fin mientras**

Hubert propone una vía bastante intuitiva para tratar el caso de la no simetría, simplemente simetrizar. En el caso del algoritmo MIN, (ver **Alg.** 1), plantea reemplazar la función D_{MIN} por:

$$D'_{MIN}(C_p, C_q) = \min_{\substack{o_i \in C_p \\ o_j \in C_q}} \{\min\{d(o_i, o_j), d(o_j, o_i)\}\}$$

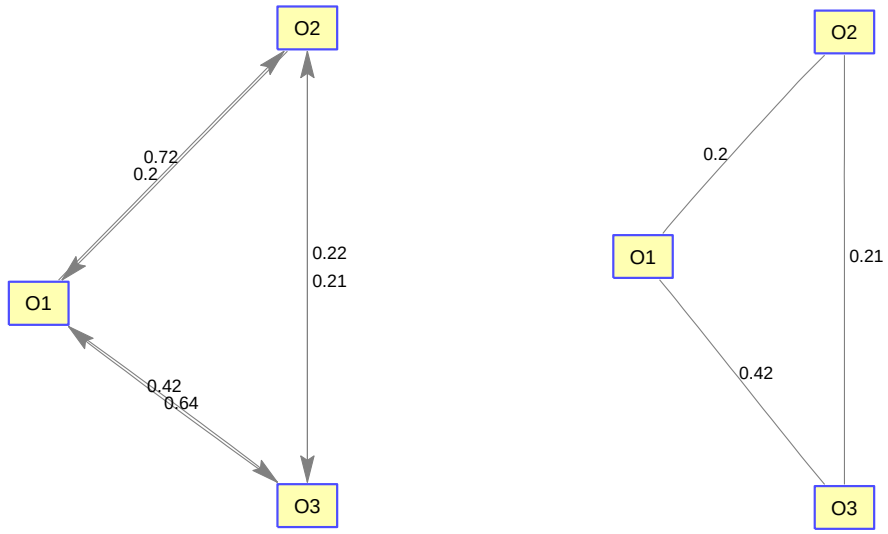
y para el caso del algoritmo MAX, (ver **Alg.** 2), plantea reemplazar D_{MAX} por una de las dos variantes de abajo:

$$D'_{MAX}(C_p, C_q) = \max_{\substack{o_i \in C_p \\ o_j \in C_q}} \{\max\{d(o_i, o_j), d(o_j, o_i)\}\}$$

$$D''_{MAX}(C_p, C_q) = \max_{\substack{o_i \in C_p \\ o_j \in C_q}} \{\min\{d(o_i, o_j), d(o_j, o_i)\}\}$$

Con estas extensiones el autor nos muestra la vía más sencilla para utilizar métodos que fueron creados para funciones simétricas. Cómo se explicó en el capítulo anterior, simetrizar trae consigo pérdida de información. Si esta pérdida de información es *aceptable*, depende

del problema en cuestión. En particular al simetrizar con mínimo se pierde completamente la información de un sentido. En el ejemplo que se muestra en la figura 5, el MIN sólo tiene la información que se ve en la figura 5b, y por consiguiente agrupa primero o_1 con o_2. Si en el problema en el que se trabaja, el par (o_2, o_3) es más similar que el par (o_1, o_2) (ej. porque la centésima de diferencia entre los mínimos no es más importante que las 5 décimas de diferencia entre los máximos, etc.), el MIN siempre fallará.



(a) grafo dirigido **(b)** grafo no dirigido

Fig. 5. Ejemplo de grafo dirigido formado a partir de las disimilitudes de 3 objetos, y el grafo no dirigido que se forma al simetrizar con mínimo

Computacionalmente el hecho de que la función de disimilitud sea no simétrica, nos añade un costo adicional, pues originalmente en los métodos presentados por Johnson era necesario comparar los objetos en un sólo sentido, pero ahora necesitamos comparar cada par de objetos en ambos sentidos.

3.2. Fórmula de actualización extendida

Lance y Williams propusieron una vía para generalizar el procedimiento de los algoritmos de agrupamiento jerárquicos aglomerativos (AJA), para el caso simétrico [35]. Esta propuesta fue extendida por Yadohisa [36], para incluir el caso cuando la función de (di)similitud es no simétrica, y además establecer un esquema general para simetrizar. Ambas propuestas fueron concebidas para trabajar con disimilitudes.

Los AJA crean en cada paso un nuevo grupo producto de la unión de los dos grupos menos disimilares. La idea de Lance y Williams fue crear una fórmula general para actualizar, en cada paso, la disimilitud entre este nuevo grupo y el resto de los grupos.

Con vistas a simplificar la notación, denotaremos la disimilitud entre dos grupos C_i y C_j como D_{ij}. Denotaremos C_{ij} al grupo formado por la unión de los grupos C_i y C_j. Además llamaremos n_i a la cantidad de elementos del grupo C_i, es decir $n_i = |C_i|$.

Sea el grupo C_{ij}, formado al mezclar los grupos C_i y C_j, Lance y Williams [35] definen la función de disimilitud entre el grupo C_{ij} y otro grupo C_k como:

$$D_{(ij)k} = \alpha_i D_{ik} + \alpha_j D_{jk} + \beta D_{ij} + \gamma |D_{ik} - D_{jk}|$$

donde α_i, α_j, β y γ son constantes ó funciones basadas en n_i, n_j ó n_k, definidas por el usuario. Esta fórmula abarca muchos algoritmos aglomerativos para funciones simétricas, en el sentido de que, en dependencia de los parámetros escogidos, es posible definir muchos de éstos. Algunos de los más usados, que pueden definirse a través de esta fórmula son: Single-Link [26] ($\alpha_i = \alpha_j = \frac{1}{2}, \gamma = -\frac{1}{2}, \beta = 0$), Average-Link [28] ($\alpha_i = \frac{n_i}{n_i+n_j}, \alpha_j = \frac{n_j}{n_i+n_j}, \gamma = \beta = 0$), y el algoritmo de Ward [37] ($\alpha_i = \frac{n_i+n_k}{n_i+n_j+n_k}, \alpha_j = \frac{n_j+n_k}{n_i+n_j+n_k}, \beta = \frac{-n_k}{n_i+n_j+n_k}, \gamma = 0$).

Algoritmo 3 Esquema de Lance y Williams

Entrada: Ω conjunto de objetos, d función de disimilitud, α_i, α_j, β, γ.

1. Por cada objeto crear un grupo unitario.
2. Calcular las disimilitudes entre cada par de estos grupos unitarios, basándose en d.
3. **mientras** todos los objetos no estén en un mismo grupo **hacer**
4. Obtener el par de grupos C_i, C_j menos disimilares.
5. Hacer $C_{ij} = C_i \cup C_j$
6. Actualizar la disimilitud $D_{(ij)k}$, teniendo en cuenta los parámetros α_i, α_j, β, γ.
7. **fin mientras**

La extensión de Yadohisa[36] para calcular la disimilitud de C_{ij} a C_k en vista de la no simetría es:

$$D_{(ij)k} = \alpha_i^{'} f^{'}(D_{ik}, D_{ki}) + \alpha_j^{'} f^{'}(D_{jk}, D_{kj}) + \beta^{'} g^{'}(D_{ij}, D_{ij}) +$$

$$\gamma^{'} |f^{'}(D_{ik}, D_{ki}) - f^{'}(D_{jk}, D_{kj})|$$

En el otro sentido, la disimilitud de C_k a C_{ij} es:

$$D_{k(ij)} = \alpha_i^{''} f^{''}(D_{ik}, D_{ki}) + \alpha_j^{''} f^{''}(D_{jk}, D_{kj}) + \beta^{''} g^{''}(D_{ij}, D_{ij}) +$$

$$\gamma^{''} |f^{''}(D_{ik}, D_{ki}) - f^{''}(D_{jk}, D_{kj})|$$

y como en el caso simétrico $\alpha_i^{'}$, $\alpha_j^{'}$, $\alpha_i^{''}$, $\alpha_j^{''}$, $\beta^{'}$, $\beta^{''}$, son constantes o funciones basadas en n_i, n_j o n_k, y $f^{'}$, $f^{''}$, $g^{'}$, $g^{''}$ son funciones definidas por el usuario.

La propuesta de Yadohisa es cambiar en cada lugar que aparece $d(o_i, o_j)$ en la fórmula de Williams por una función de las dos disimilitudes $d(o_i, o_j)$ y $d(o_j, o_i)$ (ej. $f^{'}(x, y) = x, g^{'} = max(x, y)$), que sea la encargada de manejar la no simetría. Obsérvese que si se escogen $f^{'} = f^{''}$ y $g^{'} = g^{''}$, ambas funciones simétricas (ej. máximo, mínimo, promedio), no es necesario actualizar la disimilitud en ambos sentidos.

Este algoritmo (ver alg. 4) generaliza el procedimiento de todos los algoritmos que abarca el algoritmo de Williams, para el trabajo con funciones no simétricas, mediante una correcta selección de parámetros. Así mismo, las propuestas de Hubert son un caso particular de esta de Yadohisa. Por ejemplo $D_{MAX}^{'}$ se puede definir con los parámetros $\alpha_i = \alpha_j = \frac{1}{2}$, $\gamma = \frac{1}{2}$, y $f^1 = f^2 = g^1 = g^2 =$ máx. Este esquema propuesto por Yadohisa,

permite mayor libertad a la hora de escoger la forma de simetrizar la función de disimilitud original. La función de disimilitud entre grupos resultante D_{pq}, puede ser no simétrica, no obstante, como en cada paso se busca el mínimo de todas las disimilitudes entre los grupos, implícitamente sólo se toma en cuenta el mínimo en entre D_{pq} y D_{qp}.

Algoritmo 4 Esquema de Yadohisa

Entrada: Ω conjunto de objetos, d función de disimilitud, $\alpha_i^{'}$, $\alpha_j^{'}$, $\alpha_i^{''}$, $\alpha_j^{''}$, $\beta^{'}$, $\beta^{''}$, $f^{'}$, $f^{''}$, $g^{'}$, $g^{''}$.

1. Por cada objeto crear un grupo unitario.
2. Calcular las disimilitudes entre cada par de estos grupos unitarios, en ambos sentidos, basándose en d.
3. **mientras** todos los objetos no estén en un mismo grupo **hacer**
4. Obtener el par de grupos C_i, C_j donde D_{pq} alcanza el mínimo.
5. Hacer $C_{ij} = C_i \cup C_j$
6. Actualizar la disimilitud de C_{ij} con el resto de los grupos, basándose en D_{pq}.
7. **fin mientras**

Para el esquema de Williams, las implementaciones más eficientes reportadas en la literatura, tienen un orden en el rango $[O(n^2), O(n^3)]$ [38][39]. El mejor tiempo reportado, es en un algoritmo de Day y Edelsbrunner [38] donde se obtiene un orden de $O(n^2 \log_n)$. En este trabajo los autores hacen uso intensivo de la estructura de datos, cola con prioridad, para guardar los vecinos más cercanos de cada grupo, y acelerar, en cada paso, la actualización de la disimilitud del nuevo grupo que se crea con el resto. Esta implementación es fácilmente adaptable al esquema de Yadohisa[6], pues la simetría sólo se utiliza para no calcular en ambos sentidos la disimilitud. Precisamente este cálculo de la disimilitud en ambos sentidos, sería el cómputo extra en una adaptación de la implementación de Day y Edelsbrunner para el esquema de Yadohisa.

3.3. Esquema general para algoritmos de Agrupamiento Jerárquico Aglomerativos Asimétricos (AJAA)

Otra extensión a la fórmula de Lance y Williams, pero manteniéndose en los AJA que trabajan con funciones de disimilitud simétricas, fue propuesta por Jambu [40]. Lo nuevo que propone Jambu es incorporar en la fórmula de actualización de la disimilitud entre C_{ij} y C_k, la denominada distancia de combinación de C_i, C_j y C_k (ver abajo).

Definición 3.1. *La distancia de combinación de un grupo C_p, denotada por h_p, se define como:*

1. *Si $n_p = 1$, entonces $h_p = 0$*
2. *Si $n_p > 1$, entonces C_p fue formado al unir los grupos C_u y C_v, entonces $h_p = D(C_u, C_v)$*

La distancia de combinación de C_p, nos da una medida de cuán disimilares son los elementos de C_p, y los algoritmos que la usan como *Suma de Cuadrados*[7] [40] y *Algoritmo de la Disimilitud media*[8] [41], buscan darle peso a este factor [42].

[6] Asumiendo que cada función definida por el usuario (f^1, g^1) sea de orden constante
[7] Sum of Squares
[8] Mean Dissimilarity Algorithm

Sea el grupo C_{ij}, formado al mezclar los grupos C_i y C_j, Jambu define la función de disimilitud entre el grupo C_{ij} y otro grupo C_k como:

$$D_{(ij)k} = \alpha_i D_{ik} + \alpha_j D_{jk} + \beta D_{ij} + \gamma |D_{ik} - D_{jk}| + \delta_i h_i + \delta_j h_j + \varepsilon h_k$$

donde al igual que en la de Lance y Williams α_i, α_j, β, γ, δ_i, δ_j, y ε son constantes ó funciones basadas en n_i, n_j ó n_k, definidas por el usuario. El aporte de Jambu, es que logra generalizar el procedimiento de los algoritmos basados en la distancia de combinación, y los AJA definidos por Lance y Williams, en un esquema único, ya que los primeros no podían ser definidos a través del esquema de Lance y Williams.

En [43], Takeuchi propone una modificación del esquema de Jambu, para el trabajo con funciones no simétricas. Al igual que en el esquema de Yadohisa actualiza la disimilitud en ambos sentidos. La fórmula para actualizar la disimilitud entre el grupo C_{ij} y otro grupo C_k, teniendo en cuenta la no simetría y la distancia de combinación queda:

$$D_{(ij)k} = \alpha_i^{'} f^{'}(D_{ik}, D_{ki}) + \alpha_j^{'} f^{'}(D_{jk}, D_{kj}) + \beta^{'} g^{'}(D_{ij}, D_{ij}) +$$
$$\gamma^{'} |f^{'}(D_{ik}, D_{ki}) - f^{'}(D_{jk}, D_{kj})| + \delta_i^{'} h_i + \delta_j^{'} h_j + \varepsilon^{'} h_k$$

En el otro sentido, la disimilitud de C_k a C_{ij} es:

$$D_{k(ij)} = \alpha_i^{''} f^{''}(D_{ik}, D_{ki}) + \alpha_j^{''} f^{''}(D_{jk}, D_{kj}) + \beta^{''} g^{''}(D_{ij}, D_{ij}) +$$
$$\gamma^{''} |f^{''}(D_{ik}, D_{ki}) - f^{''}(D_{jk}, D_{kj})| + \delta_i^{''} h_i + \delta_j^{''} h_j + \varepsilon^{''} h_k$$

Esta fórmula se denominó Fórmula de Actualización Asimétrica, AUF[9] por sus siglas en inglés. Similar a las anteriores $\alpha_i^{'}$, $\alpha_j^{'}$, $\alpha_i^{''}$, $\alpha_j^{''}$, $\beta^{'}$, $\beta^{''}$, $\gamma^{'}$, $\gamma^{''}$, $\delta_i^{'}$, $\delta_j^{'}$, $\varepsilon^{'}$, $\delta_i^{''}$, $\delta_j^{''}$, $\varepsilon^{''}$, son constantes o funciones basadas en n_i, n_j o n_k, y $f^{'}$, $g^{'}$, $f^{''}$, $g^{''}$ son funciones definidas por el usuario.

El esquema que propone Takeuchi está compuesto por dos fases principales. Primero se seleccionan los grupos a mezclar, y posteriormente se actualiza la disimilitud entre este nuevo grupo y el resto. En el esquema se incluye una función W, que es un criterio de simetrización (ej. mín, máx, promedio). Teniendo calculada la disimilitud entre todos los pares de grupos en ambos sentidos, D_{ij} y D_{ji}, se busca el par de grupos menos disimilares basándose en $W(D_{ij}, D_{ji})$. Esto es más flexible que la propuesta de Yadohisa, donde el criterio W siempre es el mínimo.

El objetivo de Takeuchi, es generalizar la simetrización en el procedimiento de los algoritmos jerárquicos aglomerativos, cuando la función de disimilitud es no simétrica. Esta fórmula abarca más algoritmos que el de Yadohisa, al incluir la distancia de combinación. Un ejemplo de esto último es el algoritmo *Suma de Cuadrados* [40], cuyos parámetros serían: $\alpha_i^{'} = \alpha_i^{''} = \frac{n_i}{n_{ij}+n_k}$, $\alpha_j^{'} = \alpha_j^{''} = \frac{n_j}{n_{ij}+n_k}$, $\beta^{'} = \beta^{''} = \frac{n_{ij}}{n_{ij}+n_k}$, $\gamma^{'} = \gamma^{''} = 0$, $\delta_i^{'} = \delta_i^{''} = \frac{-n_i}{n_{ij}+n_k}$, $\delta_j^{'} = \delta_j^{''} = \frac{-n_j}{n_{ij}+n_k}$, $\varepsilon^{'} = \varepsilon^{''} = \frac{-n_{ij}}{n_{ij}+n_k}$, $f^{'}(x,y) = x$, $f^{''}(x,y) = y$, $g^{'}(x,y) = g^{''}(x,y) = W$, siendo W una función como máx, mín o promedio.

El ejemplo más sencillo de los algoritmos vistos anteriormente es el Single-Link, donde $\alpha_i^1 = \alpha_i^2 = \alpha_j^1 = \alpha_j^2 = \frac{1}{2}$, $\gamma^1 = \gamma^2 = \frac{-1}{2}$, f^1, f^2, g^1, g^2 y W como en la suma de cuadrados, todos los demás parámetros como 0. La definición de muchos algoritmos representados por AJAA los podemos encontrar en [43].

[9] Asymmetric Updating Formula

Algoritmo 5 Esquema de Takeuchi, AJAA

Entrada: Ω conjunto de objetos, d función de disimilitud, $\alpha_i^{'}$, $\alpha_j^{'}$, $\alpha_i^{''}$, $\alpha_j^{''}$, $\beta^{'}$, $\beta^{''}$, $\gamma^{'}$, $\gamma^{''}$, $\delta_i^{'}$, $\delta_j^{'}$, $\varepsilon^{'}$, $\delta_i^{''}$, $\delta_j^{''}$, $\varepsilon^{''}$, $f^{'}$, $g^{'}$, $f^{''}$, $g^{''}$, W.

1. Por cada objeto crear un grupo unitario.
2. Calcular las disimilitudes entre cada par de estos grupos unitarios, en ambos sentidos, basándose en d.
3. **mientras** todos los objetos no estén en un mismo grupo **hacer**
4. Obtener el par de grupos C_i, C_j donde $W(D_{pq}, D_{qp})$ alcanza el mínimo.
5. Hacer $C_{ij} = C_i \cup C_j$
6. Actualizar $D_{(ij)k}$ y $D_{k(ij)}$ basándose en AUF.
7. **fin mientras**

3.4. CLASSIC

CLASSIC es un algoritmo de agrupamiento jerárquico, desarrollado por Ozawa [44]. Fue el primer enfoque que se apartaba de la simetrización. Se basa en el popular concepto del vecino más cercano, el cual se extiende, como veremos más adelante, para el trabajo con funciones de semejanza no simétricas.

Este método toma como entrada una función de semejanza $\Gamma : \Omega^2 \to [0, 1]$, y asume que cumple las siguientes propiedades:

- $\Gamma(o_i, o_i) = 1$
- $\Gamma(o_i, o_j) < 1$, $i \neq j$

Ozawa considera que, cuando construimos una jerarquía de particiones, el valor de la función de semejanza, entre dos elementos o_i y o_j, no es lo importante, sino cuántos objetos hay más similares a o_i que o_j, es decir, el orden que ocupa o_j en el conjunto Ω con respecto a su parecido con o_i. Basándose en su planteamiento, Ozawa construye, a partir de la función de semejanza, una matriz $O_{n\times n}$ que llama matriz de orden, la cual es la base de su algoritmo. $O_{n\times n}$ se construye de la siguiente forma: Por cada $o_i \in \Omega$, se ordenan todos los elementos o_j (o_i incluido), con respecto a su semejanza con o_i ($\Gamma(o_i, o_j)$), y a cada posición $[i, j]$ de O se le asigna un número que representa el orden que tiene o_j en cuánto a la similitud con o_i. Como o_i siempre es el más similar a si mismo $O_{i,i} = 0$, y si hay dos elementos o_{j_1}, o_{j_2} igualmente similares a o_i, es decir $\Gamma(o_i, o_{j_1}) = \Gamma(o_i, o_{j_2})$ se asigna el mismo número en las posiciones correspondientes, $O_{i,j_1} = O_{i,j_2}$. Por ejemplo, asumamos que el orden con respecto a o_i es $(o_i, o_{m_1}, \ldots, o_{m_k}, o_{m_{k+1}}, \ldots, o_{m_{n-1}})$, que $\Gamma(o_i, o_{m_k}) = \Gamma(o_i, o_{m_{k+1}})$ y que todos las demás similitudes son distintas, entonces la fila i-ésima de O quedaría $O_{i,i} = 0, O_{i,m_1} = 1, \ldots, O_{i,m_k} = k, O_{i,m_{k+1}} = k, \ldots, O_{i,m_n} = n - 2$.

Dado el conjunto Ω y una matriz de orden $O_{n\times n}$, se define la siguiente relación:

$$R = o_i R o_j \iff O_{i,j} = O_{j,i} \leq 1$$

R relaciona dos elementos que son mutuamente sus vecinos más cercanos, pues $O_{i,j} = 1$ implica que o_j es el elemento más similar a o_i. Esta relación es reflexiva y simétrica, pero no es transitiva, por lo que no es una relación de equivalencia. Al no ser R una relación de equivalencia, no es posible obtener una partición del conjunto Ω. Ozawa no utiliza esta relación como criterio de agrupamiento, pero constituye la base para construir la relación de equivalencia denominada Relación de los Vecinos más Cercanos (Nearest Neighbours

Relation), que constituirá el criterio de agrupamiento del algoritmo CLASSIC. Primero veamos algunas operaciones con relaciones, necesarias para introducir la definición de la Relación de los Vecinos más Cercanos.

Dadas R_1 y R_2 dos relaciones binarias sobre Ω, la composición de R_1 y R_2 denotada por R_1R_2 se define como:

$$\forall o_i, o_j \in \Omega,\ R_1R_2 = o_i(R_1R_2)o_j \Longleftrightarrow \exists o_q \in \Omega \,/\, o_iR_1o_q \wedge o_qR_2o_j$$

$$\left(\text{Si } R_1 = R_2 = R, R_1R_2 = R^2\right).$$

Dadas R_1 y R_2 dos relaciones binarias sobre Ω, la unión de R_1 y R_2 denotada por $R_1 \cup R_2$ se define como:

$$\forall o_i, o_j \in \Omega,\ R_1 \cup R_2 = o_i(R_1 \cup R_2)o_j \Longleftrightarrow o_iR_1o_j \vee o_iR_2o_j,$$

Dadas R_1 y R_2 dos relaciones binarias sobre Ω:

$$R_1 \subseteq R_2 \Longleftrightarrow \forall o_i, o_j \in \Omega,\ o_iR_1o_j \Rightarrow o_iR_2o_j$$

La clausura transitiva de una relación R_1 se define como:

$$\hat{R_1} = \bigcup_{h=1}^{\alpha} R_1^h = R_1 \cup R_1^2 \cup \ldots \cup R_1^{\alpha}$$

Retomando la primera relación definida, que denominamos R, Ozawa define la Relación de los Vecinos más Cercanos como la clausura transitiva de R:

$$\hat{R} = \bigcup_{h=1}^{n-1} R^h = R \cup R^2 \cup \ldots \cup R^{n-1}$$

con $n = |\Omega|$. $\hat{R}$ es transitiva pues $\forall o_i, o_j, o_p \in \Omega$ tal que $o_i\hat{R}o_j \wedge o_j\hat{R}o_p \Rightarrow \exists h_1, h_2 / o_iR^{h_1}o_j \wedge o_jR^{h_2}o_p$, con $1 \leq h_1, h_2 \leq n-1 \Rightarrow o_i(R^{h_1}R^{h_2})o_p$ y como $R^{h_1}R^{h_2} \subseteq \hat{R}$, entonces $o_i\hat{R}o_p$. Luego, $\hat{R}$ es una relación de equivalencia, y nos garantiza que: $\forall o_i \in \Omega$ se cumple que en la clase de equivalencia de o_i por $\hat{R}$ están todos los vecinos más cercanos a o_i y es el criterio de agrupamiento utilizado por Ozawa para construir la jerarquía de particiones.

Observemos un ejemplo de esta relación. Tomemos la siguiente matriz donde se encuentran las similitudes para $\Omega = \{o_1, o_2, o, o_4, o_5\}$.

$$\begin{bmatrix} 1 & 0{,}3 & 0{,}7 & 0{,}6 & 0{,}2 \\ 0{,}4 & 1 & 0{,}25 & 0{,}1 & 0{,}8 \\ 0{,}65 & 0{,}15 & 1 & 0{,}65 & 0{,}4 \\ 0{,}7 & 0{,}2 & 0{,}6 & 1 & 0{,}15 \\ 0{,}3 & 0{,}9 & 0{,}5 & 0{,}5 & 1 \end{bmatrix}$$

La matriz de orden nos quedaría:

$$\begin{bmatrix} 0\,3\,1\,2\,4 \\ 2\,0\,3\,4\,1 \\ 1\,3\,0\,1\,2 \\ 1\,3\,2\,0\,4 \\ 3\,1\,2\,2\,0 \end{bmatrix}$$

Entonces $\hat{R}$, particiona el conjunto Ω en los siguientes subconjuntos $\{o_1, o_3\}$, $\{o_4\}$ y $\{o_2, o_5\}$.

La idea del algoritmo CLASSIC, para construir una jerarquía de particiones anidadas, es partir de la matriz de orden $O_{n\times n}$ e ir construyendo una secuencia de matrices de orden, de las cuales se obtienen las particiones correspondientes, basadas en la relación $\hat{R}$. Esta secuencia de matrices de orden es obtenida aplicando el algoritmo RANKOR[44].

Algoritmo 6 RANKOR

Entrada: $O^k_{n\times n}$ matriz de orden.
Salida: $O^{k+1}_{n\times n}$ matriz de orden.

```
    Para i = 1 to n do hacer
 2.   Para j = 1 to n do hacer
        si O^k_{i,j} == O^k_{j,i} ≤ 1 then
 4.       O^{k+1}_{i,j} = 0
        else
 6.       O^{k+1}_{i,j} = O^k_{i,j}
        fin si
 8.     secmin = 2
        Para j = 1 to n do hacer
10.       si O^{k+1}_{i,j} == 1 then
            secmin = 1
12.       fin si
        fin Para
14.     Para j = 1 to n do hacer
          si O^{k+1}_{i,j} ≠ 0 then
16.         O^{k+1}_{i,j} = O^{k+1}_{i,j} − secmin + 1
          fin si
18.     fin Para
      fin Para
20. fin Para
    return O^{k+1}
```

CLASSIC a partir de la función de semejanza, obtiene una matriz de orden, desde donde hace todo el trabajo. Por esto, está mejor situado en problemas donde para cada elemento, el orden de las similitudes es más importante, que el valor de éstas. Este enfoque tiene como desventaja, que no es robusto ante el ruido, pues al desechar el valor de la similitud, y sólo tomar en cuenta la matriz de orden, puede darse el caso que, dos elementos extremos, o_p, o_q, que no tienen nada en común respecto a la función de semejanza original, entre ellos, ni con el resto de los objetos, pero $O_{pq} = O_{qp} = 1$, se agrupen en fases tempranas del algoritmo. Además, como se muestra en las matrices 3 y 4, puede llegarse a una matriz de orden, donde no existan pares relacionados por $\hat{R}$, lo que provoca que el algoritmo falle en encontrar la secuencia de particiones anidadas.

Algoritmo 7 CLASSIC

Entrada: Ω conjunto de objetos a agrupar, $\Gamma : \Omega^2 \to [0, 1]$ función de similitud.
Salida: Jerarquía de particiones anidadas.

1. Crear conjunto C donde se almacenará la jerarquía.
2. Obtener matriz de orden $O^0_{n\times n}$ a partir de Γ;
3. **Para** $i = 1$ to n do **hacer**
4. $\quad O^i_{n\times n} = RANKOR(O^2_{i-1})$
5. $\quad R_i =$ Hallar clases de equivalencia de $\hat{R}$ dados Ω y $O^i_{n\times n}$
6. $\quad C = C \cup \{R_i\}$
7. **fin Para**
8. **return** C

Este algoritmo produce menos niveles que los algoritmos jerárquicos tradicionales (ej. Single-Link, Complete-Link), pues en cada nivel es posible unir más de dos grupos, pero, para obtener un nivel cualquiera de la jerarquía, es necesario haber obtenido todos los niveles anteriores. También debemos notar que para obtener un nuevo nivel, pueden ser necesarias varias iteraciones, pues es posible que los nuevos vecinos que se forman al aplicar RANKOR, ya estuvieran en el mismo grupo.

Tabla 3. matriz de similitud, que provoca una matriz de orden, donde no hay pares relacionados por $\hat{R}$

$$\begin{bmatrix} 1 & 0{,}82 & 0{,}4 & 0{,}16 \\ 0{,}69 & 1 & 0{,}72 & 0{,}21 \\ 0{,}65 & 0{,}15 & 1 & 0{,}43 \\ 0{,}7 & 0{,}31 & 0{,}24 & 1 \end{bmatrix}$$

Tabla 4. matriz de orden, donde no hay pares relacionados por $\hat{R}$

$$\begin{bmatrix} 0 & 1 & 2 & 3 \\ 2 & 0 & 1 & 3 \\ 1 & 3 & 0 & 2 \\ 1 & 2 & 3 & 0 \end{bmatrix}$$

Desde el punto de vista computacional, el proceso completo es costoso. El paso más significativo en este sentido es obtener la nueva matriz de orden aplicando RANKOR. Aquí se recorre cada posición de la matriz O^{k+1}_n, primero inicializando los valores a partir de O_k y después asignándole el valor correspondiente a la iteración $k + 1$. Como la matriz es de $n \times n$, esto nos da un costo cuadrático para RANKOR y como se llama n veces nos da un costo computacional del orden n^3.

3.5. TCUAP

El algoritmo TCUAP[10][45] es un algoritmo de agrupamiento jerárquico, basado en grafos dirigidos, que fue concebido para agrupar documentos. El objetivo fundamental de los autores es atacar tres problemas, (1) funciones no simétricas, (2) obtener grupos de cualquier tamaño y forma, (3) casos extremos (outliers) [45].

[10] Text Clustering Using Asymmetric Proximity

TCUAP trabaja con una función de similitud no simétrica Γ. Los autores representan los objetos a través de un grafo dirigido $G(\Omega, A)$, utilizando un enfoque que denominaron *k-Máximos Similares*[11] (kHS por sus siglas en inglés). Esto es, Ω es el conjunto de objetos bajo estudio y $A = \{(o_i, o_j)/\,\Gamma(o_i, o_j) \geq \rho_i\}$. El valor ρ_i se define de forma independiente para cada objeto o_i de la siguiente manera:

$$\rho_i = \Gamma^i_{max}(1-k) + k\Gamma^i_{ave}$$

donde Γ^i_{max} es el máximo valor y Γ^i_{ave} es el promedio, de todas las similitudes que salen de o_i, $k \in (0, 1)$ es un parámetro definido por el usuario. Los autores de TCUAP no ofrecen una opción para definir este parámetro k, el cuál controla la posición de ρ_i en el intervalo $(\Gamma^i_{ave}, \Gamma^i_{max})$. A mayores valores de k, ρ_i se aproxima a Γ^i_{ave} (más suave es el umbral), y más objetos pueden ser considerados similares, y para valores pequeños de k, ρ_i crece, y se acerca a Γ^i_{max}.

En [45] se dice que el digrafo G por lo general es esparcido (de cada vértice salen pocas aristas), y que las componentes fuertemente conexas (ver definición abajo) de G son subconjuntos de alta intra-conectividad. Precisamente en estas componentes está basado el algoritmo.

Definición 3.2. *Sea el digrafo $G(V, A)$, una componente fuertemente conexa CC es un subconjunto de V, que cumple: $\forall v_i, v_j \in CC$, existe un camino de v_i a v_j y existe un camino de v_j a v_i, ambos caminos en G, y no existe componente fuertemente conexa CC_1, tal que $CC \subset CC_1$.*

Para el algoritmo TCUAP se define una función de similitud no simétrica entre grupos $\delta(C_i, C_j)$, que depende del tamaño de los grupos en cuestión y de la similitud entre los elementos de estos grupos.

$$\delta(C_i, C_j) = \frac{\sum_{o_r \in C_i} \sum_{o_s \in C_j} \Gamma(o_r, o_s)}{(n_i * n_j)^\alpha}$$

El parámetro α es definido por el usuario y sirve para ajustar la importancia del tamaño de los grupos. Obsérvese que con $\alpha = 1$, obtenemos la medida utilizada en el Average-Link o UPGMA[12] como también se le conoce.

La idea general del algoritmo es la siguiente, construimos el digrafo G y obtenemos las componentes fuertemente conexas. Luego construimos otro grafo dirigido G_1, donde los vértices representan estas componentes fuertemente conexas de G, y las aristas se crean igual que en G, pero utilizando como función de similitud a δ. Este proceso se repite hasta que se cumpla cierta condición de parada.

Este algoritmo, al igual que CLASSIC, construye menos niveles que los algoritmos aglomerativos clásicos (ej. Single-Link, Average-Link), pues en cada iteración se mezclan todos los objetos que están en una misma componente fuertemente conexa. Un problema que presenta este algoritmo es que, el hecho de que el umbral de semejanza se modifique dinámicamente en cada iteración y en dependencia de varios parámetros (k, Γ^i_{max}), provoca que a partir del segundo nivel, no resulta intuitivo encontrar una interpretación o caracterización de los agrupamientos con respecto a la función de similitud Γ.

[11] k-Highest Similarity
[12] Unweighted Pair Group Method Arithmetic Average

Algoritmo 8 TCUAP

Entrada: Ω conjunto de objetos a agrupar, Γ función de similitud, k parámetro para el cálculo del umbral, α parámetro para el cálculo de la similitud entre grupos, ϵ condición de parada.

1. P = calcular umbrales por cada objeto o_i
2. G = grafo inicial construido a partir de Ω, Γ y P
3. C = Componentes fuertemente conexas de G
4. **mientras** $\epsilon == false$ **hacer**
5. P = calcular umbrales, tomando como objetos los grupos de C, y δ como función de similitud
6. G_1 = construir grafo cuyos vértices son los grupos de C y las aristas se obtienen basandose en δ y P
7. C = Componentes fuertemente conexas de G
8. **fin mientras**

Desde punto el de vista computacional, TCUAP, puede llegar a tener un costo en tiempo cercano a $O(n^3)$. En cada iteración, para calcular los umbrales y construir el grafo, es necesario calcular la similitud entre cada par de grupos obtenido en el nivel anterior. Este cálculo siempre es cuadrático con respecto a la cantidad de grupos en el nivel en cuestión. Asumiendo que se obtengan h niveles el orden total es $O(hn^2)$. Si h es cercano a n, entonces en cada nivel aparecen muchas componentes conexas de tamaño 1, y el costo del algoritmo se hace aproximadamente $O(n^3)$.

3.6. IROCK

El IROCK[13] o ROCK Mejorado [20], como su nombre lo indica, es una modificación del algoritmo de agrupamiento jerárquico ROCK[14] [46]. Fue creado para aplicarlo en el agrupamiento de documentos con funciones de similitud no simétricas, aunque se uso no es exclusivo de esta área.

En el algoritmo ROCK se representan los objetos en un grafo $G(\Omega, A)$, donde Ω está compuesto por los objetos a analizar y $A = \{(o_i, o_j)/\Gamma(o_i, o_j) \geq \beta_0\}$. Γ es una función de similitud y β_0 es un umbral definido por el usuario. La idea fundamental de ROCK es que dos objetos que comparten muchos vecinos, por lo general deben estar en el mismo grupo.

ROCK utiliza una función $g(C_i, C_j)$ para determinar el mejor par de grupos mezclar en cada nivel.

$$g(C_i, C_j) = \frac{\sum_{o_r \in C_i} \sum_{o_s \in C_j} link(o_r, o_s)}{(n_i + n_j)^{f(\beta_0)} - n_i^{f(\beta_0)} - n_j^{f(\beta_0)}}$$

donde $link(o_r, o_s)$ es la cantidad de vecinos comunes entre dos elementos en G, el denominador es el valor esperado de vecinos comunes necesarios para normalizar esta medida de interconectividad y $f(\beta_0) = 1 + 2\frac{1-\beta_0}{1+\beta_0}$ es una forma de controlar el tamaño de la vecindad de estos grupos [46]. En cada paso, se une el par de grupos donde g alcanza el máximo. Este criterio enfatiza la inter-conectividad entre dos grupos, pero no toma en cuenta el peso de estas conexiones[20]. Por ejemplo, una conexión de un vértice cuyo valor fuera cercano a 1, tiene la misma importancia que otra conexión del mismo vértice con un valor cerca del

[13] Improved Rock
[14] RObust Clustering using linKs

umbral mínimo. Al tomar en cuenta el peso de las conexiones, podemos diferenciar estos casos.

La modificación propuesta en IROCK, es utilizar, en vez de la cantidad, el peso de las conexiones comunes. Se define el grafo dirigido con peso en las aristas $G'(V, A')$, con $A' = \{(o_i, o_j)/\Gamma'(o_i, o_j) \geq \beta_0\}$, siendo Γ' una función de similitud no simétrica. Llamemos $w_{ij} = \Gamma'(o_i, o_j)$, al peso de la arista (o_i, o_j). Sea S_{ij} el conjunto de vecinos comunes de o_i y o_j en G', los autores de IROCK definen la función $\varphi(o_i, o_j)$, para medir el peso de las conexiones comunes entre dos objetos en G' como:

$$\varphi(o_i, o_j) = \sum_{o_k \in S_{ij}} |w_{ik} - |w_{ik} - w_{jk}||$$

La idea es, que si el peso de las conexiones de o_j a S_{ij} es grande entonces, la similitud de d_i a d_j es grande. Obsérvese que si $w_{ik} > w_{jk}$ entonces sumamos w_{jk}, en caso contrario sumamos $2 * w_{ik} - w_{jk}$. En ROCK cada enlace común aportaba igual, asumiendo la misma importancia. Aquí el aporte del enlace viene dado por los valores que tomen w_{ik} y w_{jk}.

IROCK, al igual que ROCK, es un algoritmo aglomerativo. Basándose en φ, se define la función $g'(C_i, C_j)$, para escoger el mejor par de grupos para mezclar en cada iteración del algoritmo.

$$g'(C_i, C_j) = \frac{\sum_{o_r \in C_i} \sum_{o_s \in C_j} \varphi(o_r, o_s)}{(n_i + n_j)^{f(\beta_0)} - n_i^{f(\beta_0)} - n_j^{f(\beta_0)}}$$

donde el denominador es igual que en ROCK. El numerador de la fracción, nos da una medida del peso que tienen las conexiones comunes de ambos grupos. El par de grupos donde g' alcance el máximo es mezclado en cada iteración.

Algoritmo 9 IROCK

Entrada: Ω conjunto de objetos, Γ' función de similitud no simétrica, k cantidad de grupos.

1. C = grupos iniciales (Por cada objeto crear un grupo unitario).
2. M = calcular matriz similitud entre todo par de objetos.
3. O = calcular matriz de peso de las conexiones comunes para todo par de objetos según φ, basándose en M para la similitud.
4. W = calcular matriz de peso de las conexiones comunes para todo par de grupos según g'.
5. **mientras** $|C| \leq k$ **hacer**
6. $C_i, C_j = arg\ max(g'(C_r, C_s))$
7. Hacer $C_{ij} = C_i \cup C_j$
8. $Actualizar(C, C_{ij}, C_i, C_j)$
9. $Actualizar(W, C_{ij}, C_i, C_j)$
10. **fin mientras**

Este algoritmo trata con la función de similitud no simétrica en ambos sentidos, en el cálculo de φ. En φ, se intenta recoger la información relevante sobre los vecinos comunes de todo par de vértices. Esta información, es utilizada por la función de similitud entre grupos g' (que no es simétrica), y se sigue un proceso aglomerativo para construir una jerarquía de particiones. Termina cuando obtenemos una partición de Ω con un solo grupo. Como muchos de los algoritmos jerárquicos aglomerativos, en IROCK para obtener un nivel cualquiera de la jerarquía se deben obtener todos los niveles anteriores.

El algoritmo tiene un costo computacional elevado, tanto la fase inicial, como el proceso iterativo. Inicialmente se construye la matriz de similitud lo que es $O(n^2)$, se hallan las conexiones comunes entre todo par de vértices, y sus pesos, que es $O(rn^2)$ siendo r el promedio de grado de salida de los vértices en G'. Posteriormente se entra en el ciclo, que se efectúa n veces, ejecutando una búsqueda del máximo en la matriz W, y una actualización de la similitud del nuevo grupo formado, con el resto, lo que nos da un costo cúbico para el ciclo completo. Sumando todo tenemos $O(n^2) + O(rn^2) + O(n^3)$.

3.7. Agrupamiento jerárquico aglomerativo basado en el nivel de indiscernibilidad

Este método está basado en el concepto de nivel de indiscernibilidad. Sea el conjunto Ω, R una relación de equivalencia sobre Ω, decimos que $o_i, o_j \in \Omega$ son indiscernibles por R si están en la misma clase de equivalencia generada por R. Para una familia de relaciones de equivalencia $\mathbf{P}$, una relación de indiscernibilidad sobre $\mathbf{P}$, se define como la intersección de cada una de las relaciones $Q \in \mathbf{P}$ [29]. El nivel de indiscernibilidad representa el grado de acuerdo global, para clasificar dos objetos como indiscernibles [29].

El algoritmo trabaja con una función de disimilitud no simétrica d. Esta es utilizada para calcular el nivel de indiscernibilidad por cada par de objetos, y a partir de aquí se construye una jerarquía de particiones de forma aglomerativa.

El nivel de indiscernibilidad, es calculado basándose en clasificaciones binarias (ver def. 3.3). La idea es obtener para cada objeto o_i, a partir de la función de disimilitud d, una clasificación de Ω en dos conjuntos disjuntos, P_i y $\Omega - P_i$ (o_i es similar, o_i es disimilar). Y a partir de las clasificaciones binarias obtenidas, hallar el nivel de indiscernibilidad para cada par de objetos.

Definición 3.3. *Sea R_i una relación de equivalencia sobre Ω, definida por el objeto o_i, se define la clasificación binaria para o_i como:*

$$\Omega/R_i = \{P_i, \Omega - P_i\}$$

con $i = \overline{1, |\Omega|}$, y $P_i = \{o \in \Omega / \; o_i R_i o$ (o es indiscernible con o_i por R_i) $\}$.

Para construir R_i se utiliza la función de disimilitud d:

$$R_i = o_p R_i o_q \Longleftrightarrow d(o_i, o_p) \leq \beta_0 \wedge d(o_i, o_q) \leq \beta_0$$

$o_p, o_q \in \Omega$, y β_0 es un umbral para la disimilitud definido por el usuario.

$$\begin{bmatrix} 0{,}0 & 0{,}1 & 0{,}1 & 0{,}7 & 0{,}9 \\ 0{,}2 & 0{,}0 & 0{,}1 & 0{,}6 & 0{,}8 \\ 0{,}7 & 0{,}1 & 0{,}0 & 0{,}2 & 0{,}8 \\ 0{,}2 & 0{,}3 & 0{,}2 & 0{,}0 & 0{,}6 \\ 0{,}7 & 0{,}6 & 0{,}9 & 0{,}1 & 0{,}0 \end{bmatrix}$$

Por ejemplo, sea $\Omega = \{o_1, o_2, o_3, o_4, o_5\}$, cuya matriz de disimilitud no simétrica viene dada por la matriz de arriba [29]:

Entonces, para el umbral $\beta_0 = 0{,}5$, obtenemos las siguientes clasificaciones binarias:

1. $\Omega/R_1 = \{\{o_1, o_2, o_3\}\{o_4, o_5\}\}$
2. $\Omega/R_2 = \{\{o_1, o_2, o_3\}\{o_4, o_5\}\}$
3. $\Omega/R_3 = \{\{o_2, o_3, o_4\}\{o_1, o_5\}\}$
4. $\Omega/R_4 = \{\{o_1, o_2, o_3, o_4\}\{o_5\}\}$
5. $\Omega/R_5 = \{\{o_4, o_5\}\{o_1, o_2, o_3\}\}$

Aquí podemos ver que los objetos o_1 y o_2 son indiscernibles por Ω/R_1, Ω/R_2 Ω/R_4 y Ω/R_5 y discernibles solamente por Ω/R_3. Mientras los objetos o_3 y o_4 son indiscernibles por Ω/R_3 y Ω/R_4 y discernibles por Ω/R_1, Ω/R_2 y Ω/R_3.

Los autores de este método introducen una medida denominada nivel de indiscernibilidad $\gamma : \Omega^2 \longrightarrow [0, 1]$. La función $\gamma(o_i, o_j)$, cuantifica la proporción de clasificaciones binarias que concuerdan en clasificar a o_i y o_j como indiscernibles. Altos valores de γ implica que los objetos son clasificados como indiscernibles la mayoría de las veces.

$$\gamma(o_i, o_j) = \frac{\sum_{k=1}^{n} \delta_k^{indis}(o_i, o_j)}{\sum_{k=1}^{n} \delta_k^{indis}(o_i, o_j) + \sum_{k=1}^{n} \delta_k^{dis}(o_i, o_j)} \tag{5}$$

donde

$$\delta_k^{indis}(o_i, o_j) = \begin{cases} 1, \text{ if } o_i \in [o_k]R_k \wedge o_j \in [o_k]R_k \\ 0 \text{ e.o.c} \end{cases} \tag{6}$$

y

$$\delta_k^{dis}(o_i, o_j) = \begin{cases} 1, \text{ if } (o_i \in [o_k]R_k \wedge o_j \notin [o_k]R_k) \\ \quad \vee\ (o_i \notin [o_k]R_k \wedge o_j \in [o_k]R_k) \\ 0 \text{ e.o.c} \end{cases} \tag{7}$$

En la ecuación (6), $\delta_k^{indis}(o_i, o_j) = 1$ sólo se cumple cuando o_i y o_j son indiscernibles por Ω/R_k, y ambos lo son con o_k. Por su parte en la ecuación (7), $\delta_k^{dis}(o_i, o_j) = 1$ se cumple cuando o_i u o_j es discernible con o_k por Ω/R_k, y a la vez o_i y o_j son discernibles por Ω/R_k.

En la siguiente matriz se presentan los niveles de indiscernibilidad para cada par de objetos del ejemplo anterior.

$$\begin{bmatrix} 1{,}0 & 0{,}75 & 0{,}75 & 0{,}2 & 0{,}0 \\ 0{,}75 & 1{,}0 & 1{,}0 & 0{,}4 & 0{,}0 \\ 0{,}75 & 1{,}0 & 1{,}0 & 0{,}4 & 0{,}0 \\ 0{,}2 & 0{,}4 & 0{,}4 & 1{,}0 & 0{,}33 \\ 0{,}0 & 0{,}0 & 0{,}0 & 0{,}33 & 1{,}0 \end{bmatrix}$$

No resulta difícil notar que el nivel de indiscernibilidad es una función simétrica. La idea para obtener una jerarquía de particiones es aplicar un algoritmo del tipo linkage (Single-Link, Average-Link, Complete-Link), utilizando δ como función de similitud simétrica entre los objetos.

En este método no se toma en cuenta el valor de la función de disimilitud original d. La información que se extrae de d es binaria, en el sentido de que sólo interesa si es menor o igual que cierto umbral β_0, definido por el usuario. Esto trae como consecuencia que no es posible encontrar una caracterización de los agrupamientos obtenidos en cada nivel, sin importar el algoritmo Π que se escoja para obtener la jerarquía.

La complejidad total del método depende del cálculo del nivel de indescirnibilidad γ, por cada par de objetos, y del algoritmo jerárquico Π que se escoja. Para el cálculo de γ,

por cada par (o_i, o_j), se recorre el conjunto Ω completo calculando δ^{indis} y δ^{dis}. Para la primera parte nos queda entonces un costo $O(n^3)$. Debemos notar que como γ es una función simétrica, sólo la calculamos en un sentido. El costo final del algoritmo es $O(n^3) + O(\Pi)$.

Algoritmo 10 AJA basado en el nivel de indiscernibilidad

Entrada: Ω conjunto de objetos, d función de disimilitud no simétrica, β_0 umbral para la disimilitud, Π algoritmo jerárquico (ej. Single-Link)

1. $R =$ Por cada objeto obtener una clasificación binaria de Ω, utilizando d y β_0
2. $M =$ Calcular matriz del nivel de indescirnibilidad por cada par de objetos, basándose en las clasificaciones binarias R.
3. Utilizar M y Π para construir la jerarquía de particiones

4. Resultados experimentales

En esta sección, se evalúan los resultados de varios de los algoritmos presentados en el epígrafe anterior. Con este objetivo se utilizan dos colecciones de documentos, las cuales han sido utilizadas en otros trabajos [47]. Ambas fueron obtenidas de diferentes fuentes, en busca de diversidad en el experimento. Para estas bases de datos, expertos humanos han identificado los tópicos presentes. En la Tabla 5 se presentan las características generales de estas colecciones.

La colección AFP proviene de la conferencia TREC-5 [48], y contiene 695 artículos en castellano, publicados por la agencia AFP en el año 1994. Esta es la colección de menor cantidad de documentos, así como la de menor número de términos. Por su parte, TDT2 versión 4.0 [49] es una colección en inglés. TDT es empleada en competiciones internacionales de Detección y Seguimiento de Tópicos (Topic Detection and Tracking, en inglés). Consta de noticias publicadas de enero a junio en el año 1998 provenientes de seis fuentes diferentes: dos agencias de noticias (New York Times News Service, Associated Press WorldStream News Service), dos programas de radio (PRI The World, VOA English News Programs) y dos programas de televisión (CNN Headline News , ABC World News Tonight).

Tabla 5. Descripción de las colecciones de documentos

Colección	Fuente	Documentos	Términos	Tópicos
AFP	TREC-5	695	12575	25
TDT	TDT2	9824	55112	193

Como para ambas colecciones de documentos, contamos con los tópicos manuales, utilizamos para evaluar la calidad de los grupos obtenidos por los algoritmos, la medida F [50], ampliamente utilizada para este tipo de colecciones. Esta es una medida externa, es decir, mide qué cerca está el agrupamiento obtenido con las clases manuales o tópicos extraídos por expertos y que son suministrados en las colecciones. Para un grupo C_i obtenido por un algoritmo y un grupo manual C_j, se calculan los factores de precisión[15] $P(C_i, C_j)$ y recobrado[16] $R(C_i, C_j)$:

- $P(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_i|}$

[15] Precision es el término en inglés
[16] Recall es el término en inglés

- $R(C_i, C_j) = \frac{|C_i \cap C_j|}{|C_j|}$

la medida F combina estos factores para obtener una evaluación de la calidad de un agrupamiento.

$$F(C_i, C_j) = 2\frac{P(C_i, C_j)R(C_i, C_j)}{P(C_i, C_j) + R(C_i, C_j)}$$

Esta medida nos brinda el grado de emparejamiento entre cada grupo generado por el algoritmo y los clases construidas manualmente. Para obtener una medida global de una partición será necesario asociar cada grupo obtenido con la clase manual que maximice esta medida. Para ello se utiliza la siguiente función:

$$\sigma(i) = \arg\max_{C_j}\{F(C_i, C_j)\}$$

Así, la medida la medida $FMicro - promediada$ le otorga el mismo peso a cada documento, por lo que se considera un promedio por documento, es decir, un promedio sobre todos los pares documento/clase. La medida $Fmicro - promediada$ se calcula, de la siguiente forma:

$$microP = \frac{1}{N_{topicos}} \sum_{i=1}^{N_{topicos}} \frac{|C_i \cap C_j|}{|\sigma(i)|}$$

$$microR = \frac{1}{N_{topicos}} \sum_{i=1}^{N_{topicos}} \frac{|C_i \cap C_j|}{|C_i|}$$

$$microF_{(i,j)} = 2 \times \frac{microPmicroR}{microP + microR}$$

siendo $N_{topicos}$, la cantidad de clases manuales. Esta fue la medida escogida para evaluar la calidad de los agrupamientos.

En cuanto a la representación de los documentos, se utilizó el tradicional modelo vectorial, y el peso de los términos fue normalizado por la longitud. Luego, cada documento es un vector de p términos: $o_i = (w_{i_1}, w_{i_2}, \ldots, w_{i_p})$. No se consideraron como términos palabras con poco contenido semántico (stop words), como son las preposiciones, adverbios, conjunciones, artículos, etc. Para calcular la similitud de los documentos se utilizó una función de similitud no simétrica, diseñada por los creadores de TCUAP [45]. Sea T_{ij} el conjunto de términos comunes entre o_i y o_j:

$$\Gamma(o_i, o_j) = \frac{\sum_{k=1}^{|T_{ij}|} |w_{i_k} - |w_{i_k} - w_{j_k}||}{\sum_{k=1}^{p} w_{j_k}}$$

En esta medida, si los términos de o_j en T_{ij}, tienen un peso alto, entonces la similitud de o_i a o_j tiene un valor elevado.

Para los experimentos se seleccionaron Single-Link y Complete-Link, utilizando mínimo y máximo, como funciones de simetrización. También se corrieron el CLASSIC, TCUAP e IROCK. Tanto para TCUAP, como para IROCK, se seleccionaron los parámetros utilizados en los trabajos donde primero fueron publicados estos métodos. Así en TCUAP, $k = 0{,}03$ y $alpha = 2{,}5$, mientras que en IROCK, $\beta_0 = 0{,}3$.

En este experimento, se corrieron los algoritmos, y se evaluó la medida $FMicro - promediada$ en cada partición de la jerarquía. En las Figuras 6 y 7 se muestra los resultados de esta medida en las bases de datos AFP y TDT respectivamente. También en las tablas 6 y 7 se muestran los mejores resultados obtenidos por cada algoritmo y el tamaño de la partición donde se alcanzó este valor.

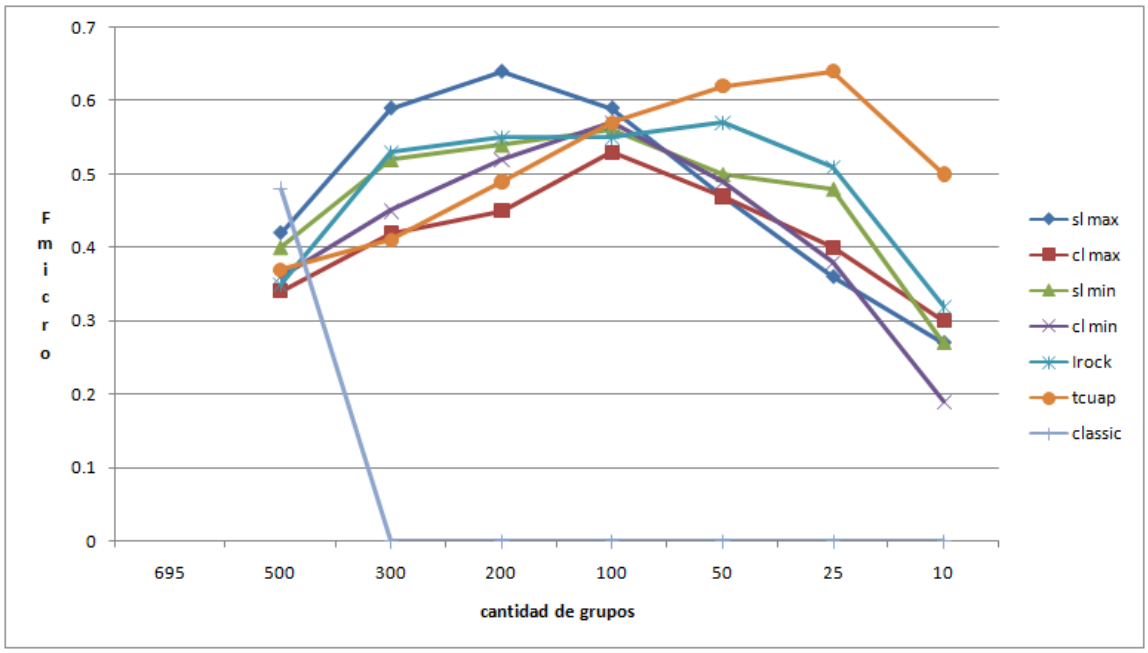


Fig. 6. Resultados en AFP con la medida *F Micro-promediada*

Como se observa en la Figura 6, CLASSIC obtiene resultados aceptables hasta llegar a 500 grupos. Aquí el método falla en encontrar la jerarquía, por el problema descrito en la subsección de 3.4, donde no existen pares de elementos que se relacionen por $\hat{R}$. Para esta base de datos, TCUAP es el que alcanza el mejor resultado, y muestra también la progresión más estable, monótona creciente en casi todo el gráfico. Además, su mejor resultado lo obtiene con una cantidad de grupos cercana a la cantidad de tópicos manuales. Por su parte IROCK, logra mostrar resultados comparables a los otros métodos, y sólo es superado por Single-Link-Max hasta los 100 grupos, a partir de donde IROCK supera a todos, excepto TCUAP. Por su parte, los métodos con simetrización mantienen aproximadamente el mismo comportamiento. Hasta llegar a los 100 grupos, mantienen un progresión monótona creciente, y después caen rápidamente. Aquí, tanto Single-Link como Complete-Link, obtuvieron los mejores resultados utilizando el mínimo. En esta base de datos IROCK y TCUAP, resultan ser los más estables, si se mira el gráfico completo. Esto es una propiedad deseable en casos, por ejemplo, donde existe interés en obtener una partición específica, pues la probabilidad de obtener una partición *mala* es más baja.

En la base TDT, el algoritmo CLASSIC muestra el mismo comportamiento que en AFP. Obtiene, hasta los 7000 grupos, los mejores resultados, comparado con el resto de los algoritmos. Pero, a partir de ese momento el algoritmo es incapaz de formar la jerarquía. Por otro lado, resulta interesante ver cómo los algoritmos TCUAP e IROCK, mantienen

un comportamiento muy similar, superando fácilmente al resto de los algoritmos, en todo el gráfico. Esto nos muestra que existen colecciones de documentos, donde los métodos que trabajan directamente con la función no simétrica, son capaces de recuperar grupos que se le escapan a métodos que utilizan la simetrización. Los otros métodos tienen un comportamiento bastante similar.

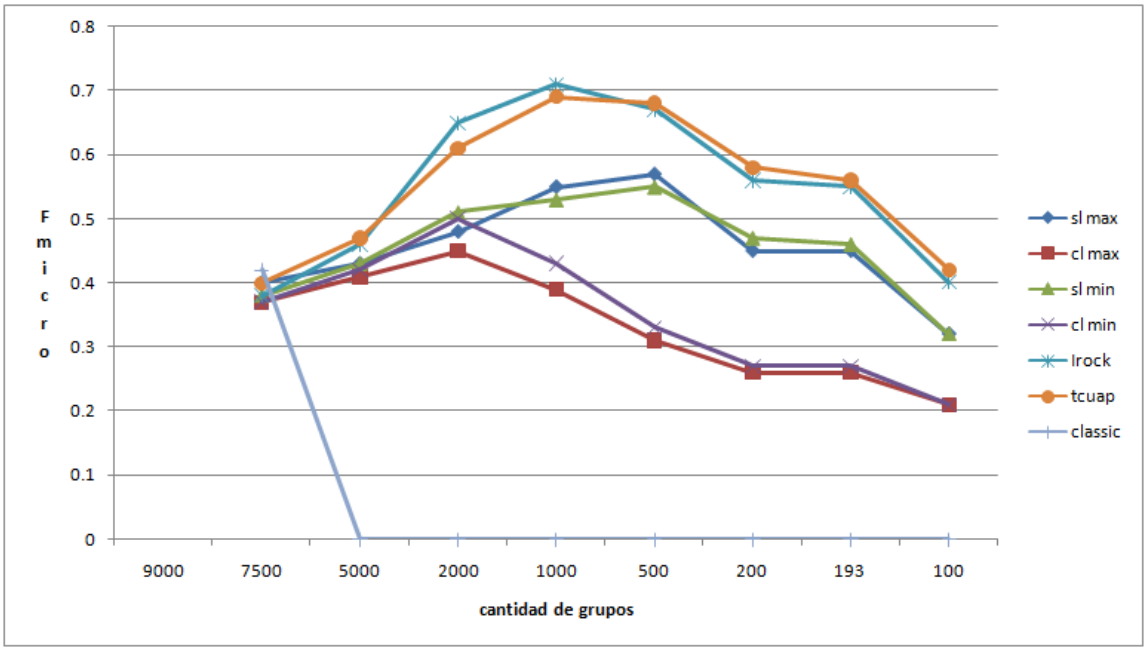


Fig. 7. Resultados en TDT con la medida *F Micro-promediada*

Tabla 6. *F Micro-promediada* para AFP

Algoritmo	Máximo Valor	Cantidad de grupos
SL_MAX	0.64	210
CL_MAX	0.53	100
SL_MIN	0.57	165
CL_MIN	0.58	118
IROCK	0.6	250
TCUAP	0.65	46
CLASSIC	0.48	500

Otro experimento que se realizó, fue en la obtención de taxonomías, utilizando la base de datos Zoo [51]. Esta base de datos contiene 101 objetos (animales) descritos en términos de 16 variables de las cuales 15 son booleanas y 1 es numérica. Aunque desde el punto de vista biológico estas variables no son suficientemente representativas para hacer un estudio taxonómico, nosotros agruparemos estos animales atendiendo a la similitud entre ellos calculada sólo en términos de estas variables. La descripción de la colección Zoo se encuentra en la Tabla 8.

Las variables que describen a estos objetos son: cubierto de pelos (r_1), cubierto de plumas (r_2), ovíparos (r_3), dar leche (r_4), volar (r_5), ser acuático (x_6), ser depredador (r_7), ser dentado (r_8), ser vertebrado (r_9), tener respiración pulmonar (r_{10}), ser venenoso (r_{11}),

tener aletas (r_{12}), cantidad de patas (r_{13}) tener cola (r_{14}), ser doméstico (r_{15}) y ser felino (r_{16}). Estos animales están agrupados en 7 clases: mamíferos, aves, reptiles, peces, insectos, anfibios y por último una séptima clase que pudiéramos llamarla misceláneas (Ver Tabla 9 para descripción de la distribución).

Tabla 7. *F Micro-promediada* para TDT

Algoritmo	**Máximo Valor**	**Cantidad de grupos**
SL_MAX	0.58	539
CL_MAX	0.46	2336
SL_MIN	0.55	521
CL_MIN	0.5	2088
IROCK	0.72	1218
TCUAP	0.7	750
CLASSIC	0.42	8293

Tabla 8. Descripción de las colección Zoo

Colección	**Animales**	**Variables**	**Clases**
Zoo	101	16	7

Tabla 9. Base de Datos Zoo

Clase	Distribución
1	41
2	20
3	5
4	13
5	4
6	8
7	10

Para calcular la semejanza de un animal a otro, diseñamos una medida de similitud, basada en el Modelo por Contraste de Tversky [14]. La idea es obtener una función de similitud, que aumente su valor a mayor cantidad de características comunes, y disminuya a medida que aumenten las características diferentes entre los objetos.

$$\Gamma(o_i, o_j) = \frac{1}{16}(T(o_i, o_j) - \frac{1}{4}D(o_i, o_j) - \frac{1}{2}D(o_j, o_i))$$

donde:

- $T(o_i, o_j) = \sum_{k=1}^{16} r_k(o_i) \times r_k(o_j)$
- $D(o_i, o_j) = \sum_{k=1}^{16} Dif(r_k(o_i), r_k(o_j))$
- $Dif(x, y) = \{1 \text{ si } x = 1 \wedge y = 0,\ 0 \text{ en otro caso}\}$

Γ nos da una medida de la semejanza de o_i a o_j. El mayor peso de la función lo tienen las características comunes, y después el peso de las características de o_j que o_i no tiene. Γ toma valores en $[-1, 1]$, por lo que la transformamos en una función de disimilitud que toma valores en $[0, 1]$ de la siguiente forma:

$$d(o_i, o_j) = (1 - \Gamma(o_i, o_j))/2$$

A partir de esta función y la base de datos, se vuelven a correr los algoritmos y se evalúa la medida *F Micro-promediada*. En la Figura 8, se muestran los resultados, y en la Tabla 10 se muestran los mejores valores obtenidos.

En esta base de datos los resultados son un poco diferentes. CLASSIC forma la primera partición con unos 50 grupos, obteniendo el mejor resultado nuevamente para esa cantidad de grupos. Se mantiene como el mejor hasta llegar a 25 donde vuelve a fallar, como en las colecciones anteriores. Por su parte TCUAP, no muestra los mismos resultados que con los documentos, y es superado ampliamente por los demás algoritmos, a partir de la partición con cardinalidad 50. IROCK por su parte vuelve a ser competitivo, mostrando que puede ser utilizado con objetos diversos, manteniendo un buen rendimiento. Algo a tener en cuenta es que la función de disimilitud utilizada, no genera mucha asimetría, en el sentido que existen muchos pares o_i, o_j, que cumplen que $d(o_i, o_j) = d(o_j, o_i)$, por este motivo la simetrización, no provoca una pérdida de información significativa, y tanto Single-Link como Completed-Link obtienen buenos resultados en sus dos variantes de simetrización.

Tabla 10. *F Micro-promediada* para Zoo

Algoritmo	Máximo Valor	Cantidad de grupos
SL_MAX	0.82	16
CL_MAX	0.8	14
SL_MIN	0.82	12
CL_MIN	0.83	12
IROCK	0.81	7
TCUAP	0.63	15
CLASSIC	0.74	34

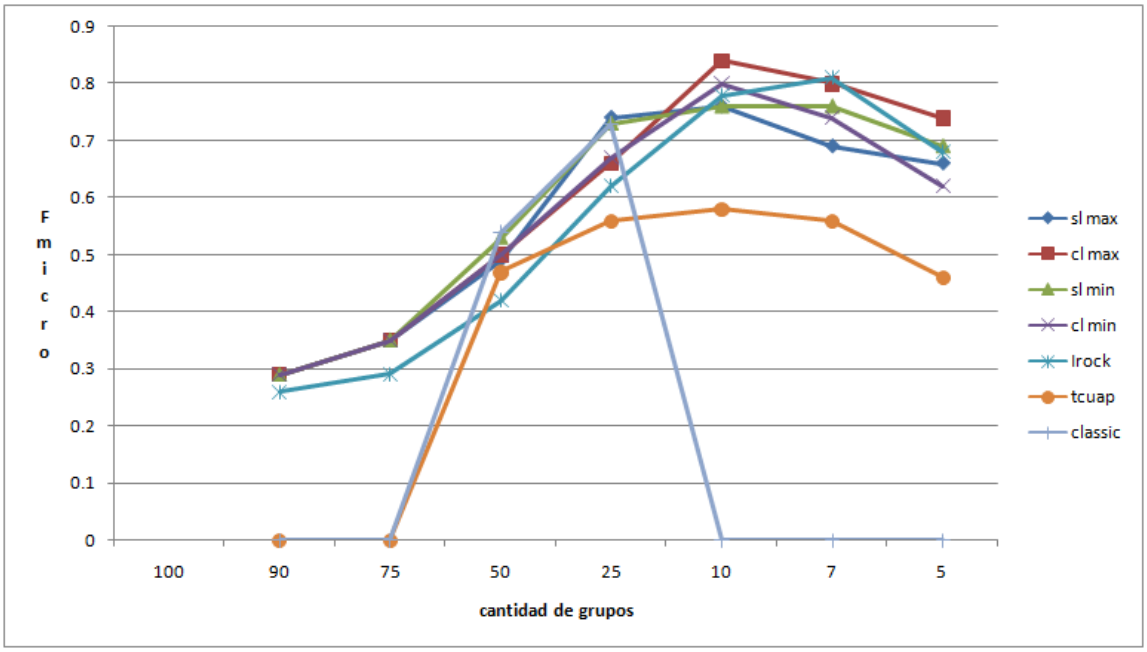


Fig. 8. Resultados en Zoo con la medida *F Micro-promediada*

5. Conclusiones

En la década del 70 aparecieron en el campo de la psicología, los primeros trabajos donde se justificaba la no simetría en las funciones de semejanza. Posteriormente han aparecido aplicaciones de las funciones de semejanza no simétricas en diversas ramas como la geosemántica, el procesamiento de vídeos, y la minería de datos. Por su parte el agrupamiento jerárquico ha demostrado ser una herramienta muy poderosa en todas estas ramas, pero la mayoría de los algoritmos desarrollados son para funciones de semejanza simétricas. En los últimos años se han desarrollado esfuerzos importantes en resolver el problema de obtener una jerarquía de particiones cuando la función de (di) similitud es no simétrica. No obstante, comparado con el caso simétrico, es poco lo que se han explorado estos métodos.

Muchas veces la solución que se da, es la de obtener una función simétrica a partir de la original, y aplicar un algoritmo que trabaje con funciones simétricas. La simetrización puede representar en ocasiones (depende del problema particular que se quiera solucionar), una simplificación del problema: se descarta la información que proporciona la no simetría, y se tergiversa la realidad. Si después de la modelación del problema, se llega a la conclusión de que una función no simétrica, es la que mejor refleja la (di) similitud entre el conjunto de objetos, es necesario la aplicación de métodos que exploten esto, con el propósito de lograr los mejores resultados.

En la evaluación de los resultados de un algoritmo de agrupamiento basándose solamente en la información del propio agrupamiento (índices de evaluación internos), generalmente se utilizan conceptos como: *conectividad*, *separación* y *compacidad*, los cuales fueron concebidos para funciones simétricas. Una redefinición de éstos, en términos de la no simetría, se hace necesaria para obtener medidas de calidad eficaces. Por ejemplo, la *separación*, ¿depende de las aristas de salida, de las aristas de entrada, o de una combinación de ambas (ej. max)?, poco existe en la literatura al respecto para el caso no simétrico.

Otro punto importante a destacar es la complejidad computacional de este tipo de métodos. La no simetría, duplica el cómputo a la hora de calcular las (di)similitudes. Debemos notar que el algoritmo IROCK, que fue el de mejores resultados en la experimentación, es también el más costoso de los algoritmos presentados en este trabajo.

La mayoría de los algoritmos no proveen la forma de obtener niveles individuales de la jerarquía de forma independiente. Por lo general para obtener un nivel, es necesario obtener todos los niveles anteriores. Así mismo, no resulta fácil interpretar estas particiones en términos de la función de (di)similitud. Otro punto a notar es el hecho de que siempre se aplica el mismo criterio a la hora de formar los agrupamientos, asumiendo que las características de las estructuraciones que se buscan en cada nivel, son las mismas. Estas cuestiones han sido abordadas en el caso simétrico, pero nada se ha realizado en este sentido cuando la función de (di)similitud es no simétrica.

La creación de un nuevo algoritmo, que elimine los problemas existentes, para obtener una secuencia de particiones cuando la función de similitud es no simétrica, es el objetivo de los autores. Este algoritmo debe hacer uso de la no simetría, dándole algún sentido a la información proporcionada por la función en ambos sentidos. También se deben considerar problemas de eficiencia, así como la interpretación de las particiones en término de la función de (di)similitud. Otros punto importante a considerar, es la aplicación de dis-

tintos criterios de agrupamiento, en dependencia del nivel, permitiendo obtener cada nivel independientemente.

Referencias bibliográficas

1. R. Varshavsky, D. Horn, and M. Linial, "Global considerations in hierarchical clustering reveal meaningful patterns in data," *PLoS ONE*, vol. 3, p. e2247, 05 2008.
2. K. Yeung, M. Medvedovic, and R. Bumgarner, "Clustering gene-expression data with repeated measurements," *Genome Biology*, vol. 4, no. 5, p. R34, 2003.
3. M. C.V. Nascimento, F. M.B. Toledoa, and A. C.P.L.F. de Carvalho, "Investigation of a new grasp-based clustering algorithm applied to biological data," *Operations Research and Data Mining in Biological Systems*, vol. 37, pp. 1381–1388, 2009.
4. A. K. Hartmann, A. Mann, and W. Radenbach, "Solution-space structure of (some) optimization problems," *Journal of Physics: Conference Series*, vol. 95, p. 012011 (10pp), 2008.
5. R. Ramachandran, J. Rushing, X. Li, C. Kamath, H. Conover, and S. Graves, "Bird's-eye view of data mining in geosciences," *Geological Society of America Special Papers*, vol. 397, pp. 235–247, 2006.
6. S. M. Iacob, R. L. Lagendijk, and M. E. Jacob, "Asymmetric similarity measures for video summarisation," in *State-of-the-Art in Content-Based Image and Video Retrieval [Dagstuhl Seminar, 5-10 December 1999]*, (Deventer, The Netherlands, The Netherlands), pp. 255–278, Kluwer, B.V., 2001.
7. D.-K. Liu and R.-H. Hwang, "P2broadcast: a hierarchical clustering live video streaming system for p2p networks: Research articles," *Int. J. Commun. Syst.*, vol. 19, no. 6, pp. 619–637, 2006.
8. A. Nowak, A. Wakulicz-Deja, and S. Bachliński, "Optimization of speech recognition by clustering of phones," *Fundamenta Informaticae*, vol. 72, pp. 283–293, 2006.
9. D. Biliotti, G. Antonini, and J. P. Thiran, "Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences," in *WACV-MOTION '05: Proceedings of the IEEE Workshop on Motion and Video Computing (WACV/MOTION'05) - Volume 2*, (Washington, DC, USA), pp. 50–57, IEEE Computer Society, 2005.
10. X. Qing, Y. Jie, and D. Siyi, "Unsupervised multiscale image segmentation using wavelet domain hidden markov tree," in *PRICAI 2004: Trends in Artificial Intelligence*, pp. 797–804, 2004.
11. J. Smolka and M. Skublewska-Paszkowska, *Computer Recognition Systems 2*, ch. Comparison of Hierarchical Cluster Analysis Methods Applied to Image Segmentation by Watershed Merging, pp. 84–91. Springer Berlin / Heidelberg, 2008.
12. A. Honkela, J. Seppä, and E. Alhoniemi, "Agglomerative independent variable group analysis," *Neurocomput.*, vol. 71, no. 7-9, pp. 1311–1320, 2008.
13. A. R. Ramachandra and V. Srinivas, "Regionalization of watershedsnext term by previous termhybrid-cluster analysis," *Journal of Hydrology*, vol. 318, pp. 37–56, 2006.
14. A. Tversky, "Features of similarity," *Psychological review*, vol. volume 84, pp. 327–352, 1977.
15. C. L. Krumhansl, "Concerning the applicability of geometric models to similarity data: The interrelationship between similarity and spatial density," *Psychological Review*, vol. 85, pp. 445–463, 1978.
16. A. Tversky and I. Gati, "Studies of similarity," *Cognition and categorization, Hillsdale, NJ Lawrence Erlbaum Associates*, vol. 4, pp. 79–98, 1978.
17. A. Rodríguez and E. Max, "Comparing geospatial entity classes: An asymmetric and context-dependent similarity measure," *International Journal of Geographical Information Science*, vol. 18, pp. 219–256, 2004.
18. B. Erol and F. Kossentini, "A robust distance measure for the retrieval of video objects," in *SSIAI '02: Proceedings of the Fifth IEEE Southwest Symposium on Image Analysis and Interpretation*, (Washington, DC, USA), pp. 40–45, IEEE Computer Society, 2002.
19. S.-C. Cheung, *Efficient video similarity measurement and search.* PhD thesis, University of California, Berkeley, 2002. Chair-Zakhor, Avideh.
20. S. Song and C. Li, "Improved rock for text clustering using asymmetric proximity," in *SOFSEM*, pp. 501–510, 2006.
21. D. Ienco and R. Meo, "Towards the automatic construction of conceptual taxonomies," in *DaWaK '08: Proceedings of the 10th international conference on Data Warehousing and Knowledge Discovery*, (Berlin, Heidelberg), pp. 327–336, Springer-Verlag, 2008.

22. M. N. Do and M. Vetterli, "Wavelet-based texture retrieval using generalized gaussian density and kullback-leibler distance," *IEEE Trans. Image Processing*, vol. 11, pp. 146–158, 2002.
23. V. Monev, "Introduction to similarity searching in chemistry. institute of organic chemistry," in *Bulgarian Academy of Sciences, Sofia 1113, Bulgaria. Match-Communications in Mathematical and in Computer Chemistry 51*, pp. 7–38, 2004.
24. W. Pentney, "Spectral clustering of biological sequence data," in *AAAI*, pp. 845–850, 2005.
25. J. Ruiz-Shulcloper, "Pattern recognition with mixed and incomplete data," *Pattern Recognition and Image Analysis*, vol. 18, pp. 563–576, 2008.
26. A. K. Jain, M. N. Murty, and P. J. Flynn, "Data clustering: a review," *ACM Comput. Surv.*, vol. 31, no. 3, pp. 264–323, 1999.
27. R. Xu and D. Wunsch, *Clustering*. Wiley-IEEE Press, 2009.
28. A. K. Jain and R. C. Dubes, *Algorithms for clustering data*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1988.
29. S. Hirano and S. Tsumoto, "Hierarchical clustering of non-euclidean relational data using indiscernibility-level," in *RSKT*, pp. 332–339, 2008.
30. A. Muñoz and M. Martin-Merino, "New asymmetric iterative scaling models for the generation of textual word maps," in *6th International Conference on the Statistical Analysis of Textual Data*, vol. 2, 2002.
31. M. Rorvig, "Images of similarity: A visual exploration of optimal similarity metrics and scaling properties of trec topic-document sets," *Journal of the American Society for Information Science*, vol. 50, pp. 639–651, 1999.
32. B. Kosko, *Neural networks and fuzzy systems: dynamical systems approach to machine intelligence*. Upper Saddle River, NJ, USA: Prentice-Hall, Inc., 1992.
33. L. Hubert, "Min and max hierarchical clustering using asymmetric similarity measures," *Psychometrika*, vol. 38, pp. 63–72, 1973.
34. S. C. Johnson, "Hierarchical clustering schemes," *Psychometrika*, vol. 32, pp. 241–254, 1967.
35. G. N. Lance and W. T. Williams, "A General Theory of Classificatory Sorting Strategies: 1. Hierarchical Systems," *The Computer Journal*, vol. 9, no. 4, pp. 373–380, 1967.
36. T. Saito and H. Yadohisa, *Data analysis of asymmetric structures: advanced approaches in computational statistics*. Marcel Dekker, 2005.
37. J. H. Ward Jr., "Hierarchical grouping to optimize an objective function," *Journal of the American Statistical Association*, vol. 58, pp. 236–244, 1963.
38. W. H. E. Day and H. Edelsbrunner, "Efficient algorithms for agglomerative hierarchical clustering methods," *Journal of Classification*, vol. 1, pp. 7–24, 1984.
39. M. R. Anderberg, *Cluster analysis for applications*. Academic Press, Inc, 1973.
40. M. Jambu and M. O. Lebeaux, *Cluster Analysis and Data Analysis*. North Holland, Amsterdam & New York, 1983.
41. J. Podani, "New combinatorial clustering methods," *Vegetatio*, vol. 81, pp. 61–77, 1989.
42. P. Arabie, L. J. Hubert, and G. de Soete, *Clustering and Classification*. World Scientific, 1996, 1996.
43. A. Takeuchi, T. Saito, and H. Yadohisa, "Asymmetric agglomerative hierarchical clustering algorithms and their evaluations," *J. Classif.*, vol. 24, no. 1, pp. 123–143, 2007.
44. K. Ozawa, "Classic: A hierarchical clustering algorithm based on asymmetric similarities," *Pattern Recognition*, vol. 16, pp. 201–211, 1983.
45. S. Song and C. Li, "Tcuap: A novel approach of text clustering using asymmetric proximity," in *IICAI*, pp. 676–685, 2005.
46. S. Guha, R. Rastogi, and K. Shim, "Rock: a robust clustering algorithm for categorical attributes," *Inf. Syst.*, vol. 25, no. 5, pp. 345–366, 2000.
47. R. Gil-García, J. M. Badía-Contelles, and A. Pons-Porrata, "Dynamic hierarchical compact clustering algorithm," in *Progress in Pattern Recognition, Image Analysis and Applications*, pp. 302–310, 2005.
48. "Text REtrieval Conference (TREC)." http://trec.nist.gov.
49. "TDT2 collection, version 4.0, 1998." http://www.nist.gov/speech/tests/tdt.html.
50. S. M. Beitzel, *On understanding and classifying web queries*. PhD thesis, Chicago, IL, USA, 2006. Adviser-Frieder, Ophir.
51. "Zoo data set." http://archive.ics.uci.edu/ml/index.html.

RT_028, junio 2010
Aprobado por el Consejo Científico CENATAV

Editor: Lic. Lucía González Bayona
Diseño de Portada: DCG Matilde Galindo Sánchez
RNPS No. 2142
ISSN 2072-6287
Indicaciones para los Autores:
Seguir la plantilla que aparece en www.cenatav.co.cu
C E N A T A V
7ma. No. 21812 e/218 y 222, Rpto. Siboney, Playa;
Ciudad de La Habana. Cuba. C.P. 12200
Impreso en Cuba