Algoritmos de Agrupamiento para la Generación de Cubrimientos en Colecciones de Documentos

por

AIREL PÉREZ SUÁREZ

Tesis sometida como requisito parcial para obtener el grado **de Maestro en Ciencias** en la especialidad de **Ciencias Computacionales** en el Instituto Nacional de Astrofísica, Óptica y Electrónica.

Supervisada por:

Dr. JOSÉ FRANCISCO MARTÍNEZ TRINIDAD
Dr. JOSÉ ELADIO MEDINA PAGOLA

Tonantzintla, Puebla
2008

©INAOE 2008
Derechos reservados
El autor otorga al INAOE el permiso de reproducir y distribuir copias en su totalidad o en partes de esta tesis

A mis abuelos Tata y Cheo,
a mi madre, a mi padre,
y a mi hermano

Agradecimientos

Agradezco a mis asesores Dr. José Francisco Martínez Trinidad y Dr. José Eladio Medina Pagola, por todo el apoyo, guía y consejos brindados para el desarrollo de esta tesis.

Al Dr. Jesús Ariel Carrasco Ochoa y a su esposa Lucía de la Vega por toda la ayuda brindada.

A mis sinodales Dr. Aurelio López López, Dr. Luis Villaseñor Pineda y Dr. Manuel Montes y Gómez por sus observaciones y sugerencias que ayudaron a mejorar la calidad de esta tesis.

Un agradecimiento muy especial a mis padres Rafael y Mercedes por todo el cariño que siempre me han brindado, por su apoyo incondicional y todo lo que me han enseñado estos años, gracias a ustedes he llegado hasta donde estoy.

A mi hermano Arian, por toda la confianza depositada en mí, por todo el cariño que me tiene y por siempre impulsarme y apoyarme en toda mi vida.

A mi esposa Yaraidys por todo el amor que me brinda, y por haber cambiado mi vida desde el primer momento en que la conocí.

A mis abuelas Mima y Cristina por su cariño; al resto de mi familia.

Agradezco a Raudel por todos los momentos que hemos pasado, por siempre estar presente para ayudarme incondicionalmente y por ser mi mejor amigo; ahora sí terminamos!!.

A mi cuñada Suley por ser pinareña y protestona, pero sobretodo por pinareña.

A todos mis amigos de mi barrio en Cuba: Abraham, Ronald, Reynol, Osniel y Osmany.

A a todos mis compañeros de trabajo del CENATAV.

A mis compañeros y amigos: Caro, Roberto, Zapotecalt, Gaby, Julio P., Julio H., Mario, Adelina, Ángel, Cris, Irving y Xalteno. De todo corazón, gracias a todos por su amistad.

Agradezco de manera especial a los padres de Adelina por toda la ayuda que me brindaron en el tiempo que estuve en México.

Finalmente, agradezco a todos los que sinceramente confiaron en mí y también a los que no.

Airel Pérez Suárez.
Tonantzintla, 15 de julio de 2008.

Resumen

El agrupamiento es el proceso de organizar o estructurar una colección de objetos en clases o *clusters* de forma tal que exista una semejanza relativamente alta (o mínima distancia) entre los objetos del mismo grupo en contraposición con la existente entre objetos de grupos diferentes.

Un aspecto a tener en cuenta, en el caso del agrupamiento de documentos, es cómo será considerada la pertenencia de un documento a los grupos. Es muy común que un documento no trate de un único tema o tópico en particular sino que en el mismo estén presentes varios tópicos; este documento debería pertenecer a cada uno de los grupos que representan a estos tópicos. No obstante, la mayoría de los algoritmos de agrupamiento de documentos producen particiones, es decir grupos disjuntos.

En esta tesis se presenta un nuevo algoritmo de agrupamiento basado en grafos denominado CStar, el cual permite generar cubrimientos en bases documentales considerando que un documento pueda pertenecer a más de un grupo. El algoritmo CStar define un nuevo tipo de sub-grafo llamado *sub-grafo en forma de estrella condensada* y aplica una heurística, a partir de la cual obtiene un cubrimiento del grafo que representa la colección utilizando sub-grafos de este tipo. La heurística que sigue CStar para obtener el agrupamiento evita el efecto de encadenamiento entre documentos, el cual es común en muchos algoritmos como el Single-Link. CStar también reduce limitaciones presentes en algoritmos basados en grafos como Star, Extended Star y Generalized Star.

Los experimentos realizados en colecciones estándar de documentos, considerando varias medidas de calidad externas e internas, mostraron que CStar obtiene mejores resultados que los algoritmos basados en grafos y que los algoritmos jerárquicos más usados para el agrupamiento de documentos. Además, CStar reduce la cantidad de grupos generados por la mayoría de los algoritmos y obtiene grupos más densos igualmente en la mayoría de los casos.

Adicionalmente, se propone una variante del algoritmo CStar denominada CStar+ que permite, a través de la detección de algunos tipos de componentes conexas, aumentar la cohesión del agrupamiento y reducir el procesamiento del algoritmo.

Abstract

Clustering is the process of grouping a set of prototypes objects into a set of meaningful subclasses, called clusters, such that prototypes belonging to the same cluster have a higher similarity in comparison to prototypes that belonging to different clusters.

One important aspect to consider, in document clustering, is how will be the resulting clusters, i.e. they will be disjoint or overlapped. Usually, there are many topics included in a single document; therefore, this document should belong to more than one cluster. However, most of document clustering algorithms produce disjoint clusters.

In this thesis, a new graph-based clustering algorithm called CStar is presented. The CStar algorithm builds a set of clusters, from one document collection, where a document may belong to more than one cluster. In CStar algorithm, a new type of star-shaped subgraph and a new heuristic that allows obtaining a cover of the thresholded similarity graph representing the document collection, are introduced. The heuristic followed by CStar avoids the chaining effect presented in some algorithms like Single-link and it reduces some drawbacks presented in some graph-based algorithms like Star, Extended Star and Generalized Star.

The experimentation using two standard document collections, considering internal and external evaluation measures, showed that CStar outperforms other graph-based algorithms and some widely used hierarchical algorithms. Besides, CStar obtains fewer clusters than most of the aforementioned algorithms and, in almost all cases, their clusters have more elements.

Additionally, in this thesis, an improved version of CStar algorithm called CStar+ is presented. This version allows, through the detection of some types of connected components in the thresholded similarity graph, to increase the internal cohesion of the resulting clustering and to reduce algorithm's processing.

Índice general

1. Introducción **1**

1.1. Descripción del problema 3

1.2. Objetivos 4

1.3. Estructura de la tesis 5

2. Algoritmos de agrupamiento **7**

2.1. Modelos de representación de documentos 9

2.1.1. Pre-procesamiento de los documentos 9

2.1.2. Modelo de Espacio Vectorial 11

2.1.3. Modelo de Distancia Asociativa Global 13

2.1.4. Modelo Basado en Conjuntos de Secuencias Frecuentes Maximales . 14

2.2. Medidas de semejanza 15

2.3. Trabajo relacionado 16

2.3.1. Algoritmos de pasada simple 16

2.3.2. Algoritmos basados en optimización 17

2.3.3. Algoritmos jerárquicos aglomerativos 19

2.3.4. Algoritmos jerárquicos divisivos 20

2.3.5. Algoritmos basados en árboles 21

2.3.6. Algoritmos basados en densidad 23

2.3.7. Algoritmos basados en conjuntos frecuentes de ítems 24

2.3.8. Algoritmos basados en técnicas genéticas 27

2.3.9. Algoritmos basados en factorización de matrices 28

2.3.10. Algoritmos basados en grafos 30
2.3.11. Algoritmos híbridos 43
2.4. Medidas de evaluación de la calidad 45
2.4.1. Medidas internas de calidad 45
2.4.2. Medidas externas de calidad 46
2.5. Recapitulación 47

3. Agrupamiento basado en sub-grafos condensados 51
3.1. Algoritmo CStar 52
3.1.1. Conceptos básicos 52
3.1.2. Algoritmo CStar 55
3.1.3. Análisis de la complejidad computacional 59
3.2. Algoritmo CStar+ 62
3.3. Evaluación experimental 64
3.3.1. Descripción de las colecciones 66
3.3.2. Representación de los documentos 67
3.3.3. Experimentos 67
3.4. Recapitulación 73

4. Conclusiones y Trabajo Futuro 75
4.1. Conclusiones 75
4.2. Trabajo Futuro 76

Bibliografía 78

Capítulo 1

Introducción

En la actualidad, con el desarrollo de Internet, las comunicaciones y la creación de grandes repositorios de documentos y almacenes de datos con contenidos textuales, se ha observado un significativo incremento del interés en el desarrollo de métodos asociados con la minería de datos para el procesamiento de textos, como son la clasificación, el análisis de tendencias, el descubrimiento de patrones frecuentes, el agrupamiento, etc.

El término de agrupamiento (clustering) suele asociarse con todo método que estructure conjuntos de objetos en grupos (clusters) de forma tal que exista una semejanza relativamente alta (o mínima distancia) entre los objetos del mismo grupo en contraposición con la existente entre objetos de grupos diferentes. Esa semejanza (o distancia) puede ser medida de muy diversas formas, atendiendo a los diferentes modelos desarrollados y a la diversidad de tipos de datos y aplicaciones. A su vez, los grupos descubiertos pueden: (i) formar particiones (grupos disjuntos), (ii) formar cubrimientos con mayores o menores grados de solapamientos en dependencia de los requerimientos de los usuarios, ó (iii) ser difusos.

El agrupamiento es una herramienta que, en el contexto del procesamiento de documentos ha sido utilizada eficientemente en la solución de diversas tareas, como son indexación, recuperación de información, sistemas de búsquedas de respuestas, generación automática de resúmenes, detección y seguimiento de tópicos, etc.

Estas múltiples aplicaciones han favorecido el desarrollo de varios algoritmos de agrupamiento de documentos. Estos algoritmos son variados en cuanto a: (i) el modelo uti-

lizado para representar a los documentos, (*ii*) la función empleada para el cálculo de la semejanza entre un par de documentos y (*iii*) el método o heurística utilizada en la construcción del conjunto de clusters.

Un aspecto a tener en cuenta en el caso del agrupamiento de documentos es cómo será considerada la pertenencia de un documento a los grupos. Es muy común que un documento no trate de un único tema o tópico en particular, sino que en el mismo estén presentes varios tópicos cada uno de ellos posiblemente con distintos grados de relevancia. Teniendo esto en cuenta, este documento debería pertenecer a cada uno de los grupos que representan a estos tópicos. No obstante a lo mencionado anteriormente, la mayoría de los algoritmos de agrupamiento producen particiones, es decir grupos disjuntos.

Es importante agregar que existen algunas aplicaciones en las cuales se requiere la inclusión de un documento en varios tópicos de discusión. Este tipo de estructuración sólo es posible si se consideran algoritmos cuyos grupos puedan formar cubrimientos y no particiones. Un ejemplo de estas aplicaciones se puede encontrar en bibliotecas digitales o sistemas de información científica en general, donde es común que un documento se clasifique de acuerdo a varios tópicos. En este tipo de sistemas, tener los documentos agrupados de esta forma, puede resultar de vital importancia para el trabajo de especialistas y analistas de información.

Este trabajo centrará su atención en algoritmos de agrupamiento que permitan la generación de cubrimientos en bases documentales, de forma tal que se considere la posible pertenencia de un documento a más de un grupo. Los algoritmos propuestos permitirán el descubrimiento de grupos no necesariamente disjuntos, donde cada documento de la colección pertenecerá a uno o más grupos en dependencia de los tópicos subyacentes que el mismo contenga.

Este capítulo está organizado como sigue: en la sección 1.1 se realiza una descripción del problema a tratar en este documento de tesis. Los objetivos principales y específicos del trabajo se presentan en la sección 1.2 y por último en la sección 1.3 se describe cómo está organizado el documento de tesis.

1.1. Descripción del problema

La obtención de un algoritmo de agrupamiento que pueda organizar una colección de documentos de forma tal que se logre una cohesión interna elevada, continua siendo un problema de investigación abierto. Aunque se han desarrollado varios algoritmos para tal fin, la mayoría de ellos presentan diferentes problemas. Por ejemplo, en algunos algoritmos el agrupamiento resultante depende del orden de análisis de los documentos. Esta dependencia no sólo trae como consecuencia que los grupos sean diferentes cada vez que se ejecute el algoritmo sobre la misma colección, sino que además en muchos casos se afecta la calidad del agrupamiento.

Otros algoritmos por su parte, incorporan heurísticas que eliminan esta dependencia, sin embargo:

1. Presentan el efecto de encadenamiento, es decir varios documentos diferentes entre sí, pero asociados transitivamente pueden ser agrupados en un mismo grupo, reduciéndose de esta forma la cohesión interna de dicho grupo.

2. Obtienen muchos grupos y éstos generalmente tienen pocos elementos, dificultándose el análisis de los resultados, ó

3. Requieren varios parámetros externos para los que no siempre es conocido cuál o cuáles valores asignarles.

Uno de estos parámetros que muchos de los algoritmos necesitan es el número de grupos en los que se desea organizar la colección, lo cual puede afectar directamente la calidad del agrupamiento puesto que en la mayoría de los casos no se conoce a priori la cantidad de clases en las que se distribuyen los documentos que se desean agrupar. Una alternativa a este problema sería descubrir automáticamente el número de grupos idóneo en los que debiera agruparse la colección, a partir de las características de los documentos que la componen.

Por último, como ya se mencionó, en el contexto del procesamiento de documentos puede resultar muy útil permitir la asignación de un documento a más de un tópico o

tema. Sin embargo, la mayoría de los algoritmos de agrupamiento desarrollados consideran un documento como perteneciente a un único grupo.

El análisis de las limitantes planteadas conduce a la siguiente pregunta: ¿Será posible desarrollar un algoritmo de agrupamiento que considere la generación de cubrimientos y que a la vez ataque las deficiencias planteadas?

1.2. Objetivos

En el conjunto de los algoritmos de agrupamiento de documentos, una clase muy importante es la de los algoritmos basados en grafos. Un algoritmo de agrupamiento basado en grafo se caracteriza por representar la colección a agrupar por su correspondiente grafo de semejanza. Un grafo de semejanza $G = \langle V, E, w \rangle$, es un grafo en el cual los vértices representan a los documentos de la colección, cada arista (d_i, d_j) está etiquetada con el valor de semejanza existente entre los documentos que la componen y w es la función que asigna los pesos a las aristas. Debido a que el grafo de semejanza formado es completo, existen algoritmos que trabajan sobre variantes de este grafo. Por ejemplo: (i) grafos de máxima semejanza; los cuales sólo consideran que un vértice tenga arista con otro si la semejanza entre ellos es máxima, (ii) grafos de β-semejanza; los cuales consideran que dos vértices forman una arista si la semejanza entre ellos es superior a un umbral β.

Una característica importante de estos algoritmos de agrupamiento es que no imponen restricciones al espacio de representación de los objetos de la colección, así como tampoco restringen la medida de semejanza utilizada en el cálculo del peso de las aristas. Estas características aumentan sin lugar a dudas el campo de aplicación de dichos algoritmos.

En este trabajo está centrado en algoritmos de agrupamiento basados en grafos.

Objetivo general

Teniendo en cuenta las bondades de los algoritmos basados en grafos, el objetivo de este trabajo es desarrollar un algoritmo de agrupamiento de este tipo, que permita la generación de cubrimientos en bases documentales, de forma tal que se considere la pertenencia de un documento a más de un grupo y que además:

1. Reduzca la cantidad de grupos obtenidos.

2. Reduzca la influencia del orden de análisis de los documentos en el agrupamiento obtenido.

3. No presente el efecto encadenamiento.

4. No necesite para su funcionamiento conocer a priori el número de grupos a obtener.

5. Reduzca la dependencia de parámetros externos.

El desarrollo de los puntos 1, 2 y 3 tiene como objetivo mantener o mejorar la calidad del agrupamiento resultante respecto al agrupamiento obtenido por otros algoritmos relacionados. El desarrollo de los puntos 4 y 5 tiene como objetivo facilitar la aplicación del algoritmo.

Objetivos específicos

Como objetivos específicos de este trabajo se encuentran los siguientes:

1. Proponer un nuevo criterio de cubrimiento de grafo de semejanza utilizando subgrafos que reduzca su dependencia a parámetros adicionales.

2. Desarrollar un algoritmo de agrupamiento que utilice el criterio de cubrimiento propuesto en (1) y que permita la generación de cubrimientos en bases documentales.

3. Evaluar la eficacia del algoritmo desarrollado en bases documentales estándares, comparándolo con otros métodos de agrupamiento.

1.3. Estructura de la tesis

Este documento de tesis consta de 4 capítulos. En el capítulo 2 se exponen los pasos involucrados en el proceso de agrupamiento de documentos así como una descripción de los principales trabajos reportados en la literatura referentes a este tipo de agrupamiento. En el capítulo 3 se describe en detalle el algoritmo presentado en esta tesis. Por último el

capítulo 4 contiene las conclusiones de la tesis y una descripción breve de futuras líneas de trabajo.

Capítulo 2

Algoritmos de agrupamiento

En la década de los noventa, disciplinas como la Minería de Datos, el Aprendizaje Automático, la Minería de Textos y el Reconocimiento de Patrones resultaron de gran ayuda para el hombre, en su constante afán de extraer información relevante de un conjunto de datos. Hoy en día, el volumen de información almacenada crece a una velocidad verdaderamente increíble por lo que, el proceso de análisis de esta información resulta cada vez más complejo e importante. Es en este punto donde la aplicación de técnicas de las disciplinas mencionadas adquieren un papel importante.

Una de las técnicas que ha demostrado ser de utilidad en varios contextos es el agrupamiento o estructuración de datos. El agrupamiento es el proceso de organizar o estructurar una colección de objetos en clases o *clusters* de forma que los objetos contenidos en un mismo grupo sean más semejantes en relación con objetos que pertenezcan a grupos diferentes [28]. Los algoritmos de agrupamiento pueden ser *Restringidos* o *Libres* de acuerdo a si se conoce a priori o no el número de grupos que se desean obtener [74].

Algunas de las aplicaciones de los algoritmos de agrupamiento están asociadas con áreas como la biología, donde se han utilizado algoritmos de agrupamiento para el estudio de enfermedades como el cáncer de mamas [47] o para el estudio de secuencias de genes [26, 86], en contextos como la detección y seguimiento de sucesos en flujos de noticias o correos [10, 66, 15], el procesamiento de imágenes [40, 9, 42], el procesamiento de datos espaciales [32, 92, 52], donde se busca descubrir estructuras, así como analizar flujos en datos que son generados continuamente. En otros contextos como el procesamiento de documentos

web se ha utilizado el agrupamiento como herramienta para el descubrimiento de patrones en el comportamiento de usuarios que navegan en distintos sitios web [61, 62, 57], para determinar las relaciones presentes entre los tópicos de diferentes sitios visitados [29], así como para personalizar, en dependencia de los gustos de ciertos clientes, los servicios brindados por un portal web [90].

Los algoritmos de agrupamiento pueden clasificarse atendiendo a varios criterios [94, 36, 28]. Algunos de estos criterios son: (*i*) la forma en que procesan los objetos, (*ii*) atendiendo a cómo se considera la pertenencia de los objetos a los grupos formados, (*iii*) la forma en que se organizan o se relacionan los grupos obtenidos y (*iv*) el mecanismo en que se basan para agrupar los objetos.

En la literatura existen diversos algoritmos de agrupamiento reportados, observándose en todos un conjunto de pasos involucrados: (*i*) Representación de los objetos (opcionalmente incluye extracción o selección de variables), (*ii*) Definición de una medida de proximidad o semejanza entre los objetos, (*iii*) Agrupar los objetos utilizando algún criterio de agrupamiento o heurística. Además de estos pasos, y aunque no forma parte como tal del proceso de agrupamiento, es común incluir un proceso de evaluación de los resultados obtenidos.

Como ya se mencionó anteriormente, este trabajo centra su atención en los algoritmos de agrupamiento de documentos. En este capítulo se presentará en profundidad cada uno de los pasos presentes en el proceso de agrupamiento de documentos, exponiéndose una panorámica de los trabajos más importantes reportados en la literatura referente a este tipo de algoritmo.

El capítulo está organizado como sigue: las secciones 2.1 y 2.2 abordan en detalle los primeros dos pasos involucrados en el proceso de agrupamiento. La sección 2.3 por su parte, describe las características de los principales trabajos relacionados. Por último, las secciones 2.4 y 2.5 están dedicadas a detallar las medidas de evaluación que utilizaremos para estimar la calidad de los resultados obtenidos y a realizar un resumen de lo abordado en el capítulo.

2.1. Modelos de representación de documentos

Si a una persona se le pidiera que agrupara una colección de documentos según su contenido, ésta sin lugar a duda procedería a leerlos uno a uno (o al menos el resumen del documento) y con base en la información extraída lograría el agrupamiento deseado. En el caso de un programa automático la tarea de "leer" el documento, como lo haría una persona, es mucho más difícil ya que incluye extracción de información tanto sintáctica como semántica y posteriormente el uso de esta información para determinar cuándo dos documentos abordan el mismo tema. En el ser humano sin embargo, estas acciones ocurren casi inconsientemente.

De lo anteriormente planteado se induce la necesidad de buscar una alternativa de representación para los documentos, diferente al lenguaje natural en que normalmente éstos están escritos, de forma tal que dicha representación sea fácil de utilizar por un algoritmo de agrupamiento de documentos.

En esta sección se describen tres modelos de representación de documentos que han sido utilizados con cierta frecuencia en los algoritmos de agrupamiento y en otras tareas de procesamiento de documentos: el Modelo de Espacio Vectorial, el Modelo de Distancia Asociativa Global y el Modelo Basado en Conjuntos de Secuencias Frecuentes Maximales. Con el objetivo de describir los modelos anteriormente planteados se hace necesario primeramente describir el pre-procesamiento que se realiza sobre los documentos de una colección.

2.1.1. Pre-procesamiento de los documentos

Como es conocido, un documento está compuesto por varios elementos: palabras, signos de puntuación, caracteres de espacio, así como posiblemente de algunos caracteres especiales. Como es de suponer, una persona que quiere analizar dos documentos para determinar si los mismos abordan un mismo tema no utiliza todos estos elementos, sino que solamente utiliza las palabras y de hecho un sub-conjunto de ellas, el que mejor representa a cada documento. Estas palabras generalmente reciben el nombre de *términos*

y representan al documento al que pertenecen.

El pre-procesamiento de un documento es un proceso que tiene el objetivo de modificar el documento de forma tal que el mismo sólo esté compuesto por este conjunto de términos. Entre las tareas más comunes de este proceso están:

1. Eliminación de mayúsculas, signos ortográficos, espacios, etiquetas (en el caso de las páginas HTML), así como el reconocimiento de palabras (*tokenización*). Al finalizar esta tarea el documento queda representado solamente por las palabras que fueron identificadas.

2. Eliminación de las *palabras vacías*. Este conjunto o lista está formado por palabras que carecen de utilidad para resaltar el contenido del documento al que pertenecen. Algunos de los elementos que conforman esta lista son preposiciones, artículos, pronombres, conjunciones así como otras palabras que no son discriminantes debido a que aparecen muy frecuentemente en todos los documentos. Por consiguiente cada palabra que se identificó como resultado de la tarea anterior es verificada y sólo aquellas que no estén en la lista son utilizadas para representar el documento. Este paso reduce notablemente la cantidad de palabras que representa al documento.

3. Lematización o stemming. Este paso tiene como objetivo reducir aún más la dimensionalidad del documento al tratar de reducir las palabras a una forma canónica y de esta forma poder identificar palabras que representan variaciones morfológicas de una misma palabra. Un ejemplo de estas variaciones morfológicas se observa en palabras como *jóvenes* y *jovencitos* las cuales se reducen a joven. Este proceso por consiguiente tiene implícito eliminación de plurales, prefijos, sufijos, conjugaciones verbales así como redución al infinitivo de los verbos.

Es común que no se realicen siempre todas las tareas de pre-procesamiento. Hay muchos algoritmos que sólo realizan las dos primeras y otros que además incluyen un proceso de identificación de entidades nombradas como nombres propios, nombres de lugares, organizaciones, etc.

2.1.2. Modelo de Espacio Vectorial

Uno de los modelos de representación más utilizados por los algoritmos que procesan documentos es el Modelo de Espacio Vectorial (VSM, por sus siglas en inglés) [71]. En este modelo, dada una colección de documentos D, cada documento $d \in D$ es representado como un vector de la forma $d = (w_1, w_2, \ldots, w_N)$, donde N es la cantidad total de términos de la colección (la cantidad de términos que aparecen al menos una vez en al menos un documento de la colección) y cada w_i representa el *peso* del termino i-ésimo dentro del documento, es decir un valor que identifica la utilidad de ese término para describir al documento d.

Los términos que componen la colección de documentos y que representan a cada uno de sus documentos en este modelo pueden ser de distintos tipos: (i) palabras, (ii) n-gramas ó (iii) secuencias frecuentes maximales, etc. El caso de las palabras es sencillo puesto que éstas serían las mismas que se obtuvieron en la etapa de pre-procesamiento de los documentos. Los n-gramas por su parte son subsecuencias de n elementos consecutivos presente en un documento. Normalmente el valor de n que más se utiliza es 2, por lo que estos 2-gramas son generalmente llamados *bigramas*.

Una secuencia de palabras se considera frecuente si la misma aparece en la colección de documentos una cantidad de veces no menor que un valor mínimo α predefinido. Una secuencia frecuente S es maximal si se cumple que $\nexists S'$, secuencia frecuente tal que $S \subset S'$. Existen en la literatura reportados varios algoritmos que calculan estas SFM [2, 20, 21].

Como se mencionó anteriormente, en el Modelo de Espacio Vectorial cada término presente en el vector de un documento es representado por un peso que determina la importancia de este término para representar al documento. Existen diferentes formas de calcular este peso [25], entre las más utilizadas se encuentran:

- **Pesado Booleano**. Este tipo de pesado es el más sencillo que se utiliza, el mismo asigna un peso igual a 1 si el término está presente en el documento y 0 en caso contrario.

- **Frecuencia del término**. Este tipo de pesado también es conocido como TF por

sus siglas en inglés. A cada término en un documento se asigna un peso proporcional a la cantidad de veces que el mismo aparece en dicho documento. Luego, si se denota $TF(d, t_i)$ como la frecuencia de aparición del término t_i dentro del documento d, entonces el peso de t_i en d es $w(d, t_i) = TF(d, t_i)$. Generalmente estas frecuencias de aparición son normalizadas con el objetivo de que: (i) este peso asociado no dependa de la frecuencia relativa de este término con otros términos y (ii) evitar favorecer a documentos más largos que otros. Existen diferentes tipos de normalizaciones reportadas en la literatura [25], algunas de estas normalizaciones son:

- *Normalización por la frecuencia.* La frecuencia de cada término se divide por la frecuencia máxima de todos los términos del documento, logrando que este peso esté entre 0 y 1. Existen algunas variantes de esta normalización, entre ellas podemos mencionar:
  - *Augmented Normalized Term Frequency.* Esta normalización produce valores entre $0,5$ y 1 mediante la aplicación de la fórmula:

$$w(d, t_i) = 0{,}5 + \frac{0{,}5 \cdot TF(d, t_i)}{\text{máx}_{\bar{d} \in D}\, TF(\bar{d}, t_i)}$$

  - *Logarithmic Term Frequency* . Esta normalización reduce la importancia de la frecuencia absoluta de un término en aquellas colecciones con gran variabilidad en la longitud de los documentos.

$$w(d, t_i) = 1 + \log\left(TF(d, t_i)\right)$$

  - *Average Term Frequency.*

$$w(d, t_i) = \frac{1 + \log\left(TF(d, t_i)\right)}{1 + \log\left(avg_{\bar{d} \in D}(TF(\bar{d}, t_i))\right)}$$

- *Normalización por la longitud del documento.* Esta normalización consiste en dividir la frecuencia de cada término por la longitud del documento.

- **TF-IDF**. El factor $IDF(d, t_i)$ de un término t_i, a diferencia de $TF(t_i)$, representa la frecuencia del término t_i en la colección de documentos. Este factor $IDF(d, t_i)$ da un

estimado de la importancia del término t_i para describir al documento d en función de cuántos documentos en la colección contienen a t_i; mientras menos documentos contengan a t_i mayor será la utilidad de éste para describir al documento d. Luego el peso del término t_i en el documento $d \in D$ será:

$$w(d, t_i) = TF(d, t_i) \cdot IDF(d, t_i) \qquad , donde \quad IDF(d, t_i) = \log \frac{N}{df(t_i)},$$

N es la cantidad de documentos de la colección y $df(t_i)$ el número de estos que contienen al término t_i. Este tipo de pesado permite asignar un peso mayor a aquellos términos que ocurren con una alta frecuencia en el documento pero que son poco frecuentes en la colección.

Estas formas de pesado son utilizadas para cada uno de los posibles tipos de términos que se mencionaron anteriormente. En el caso de las SFM, se ha definido también otra forma de asignar los pesos que utiliza la longitud de dichas SFM [31].

2.1.3. Modelo de Distancia Asociativa Global

Este modelo se propuso con el objetivo de solucionar las desventajas del Modelo de Grafo de Asociación [54]. Este modelo se apoya en las ideas preliminares del Modelo de Espacio Vectorial y a diferencia de este último estima la relevancia de un término a través de su fuerza global asociativa [56].

El modelo GADM, por sus siglas en inglés, asume las siguientes hipotesis:

H_1**:** *La relación semántica entre dos términos de un documento disminuye con el aumento de la distancia entre ellos.*

H_2**:** *La relevancia de un término en un documento está dada por la fuerza de sus relaciones con el resto de los términos del documento.*

Como primer paso se modela cada documento $d \in D$ a través de un grafo G_{d_i}, denominado *Grafo de Asociación*, en el que los vértices representan a los términos presentes en el mismo y una arista entre un par de términos t_i y t_j representa la distancia entre

ellos. La fórmula para calcular la distancia entre los términos tiene en cuenta la distancia por párrafos existente entre los términos, así como la co-ocurrencia natural cuando estos términos aparecen en la misma oración.

Sean (p_i, n_i), (p_j, n_j), el número del párrafo y el número de la oración en que se encuentran el término t_i y el término t_j respectivamente, la distancia entre estos términos en el documento d se define como:

$$D_{ij} = \begin{cases} 1 & (i = j) \vee [(p_i = p_j) \wedge (n_i = n_j)] \\ |p_i - p_j| + 2 & \text{en otro caso} \end{cases} \tag{2.1}$$

Formalmente, en el modelo de Distancia Asociativa Global cada documento es representado como un vector de la forma $d = (g_{t_1}, g_{t_2}, \ldots, g_{t_N})$, donde el peso de cada término t_i se calcula a través de su fuerza global asociativa g_{t_i} definida por:

$$g_{t_i} = \sum_{t_j \in d} \frac{1}{\sqrt{D_{ij}}}, \tag{2.2}$$

donde D_{ij} representa la distancia existente entre los términos i y j, o sea el peso de la arista que une a los vértices v_i y v_j en el grafo G_{d_i} calculada por la expresión 2.1. Existen otros trabajos que abordan nuevas formas de definir la expresión 2.2, ver [55] para mayor información.

2.1.4. Modelo Basado en Conjuntos de Secuencias Frecuentes Maximales

Este modelo fue propuesto en [31] como una alternativa al modelo de espacio vectorial y el mismo pretende reducir el espacio de representación del documento y aprovechar las características de las SFM.

Formalmente, dado un documento $d \in D$, el Modelo de Secuencias Frecuentes Maximales representa dicho documento a través del conjunto de las SFM que el mismo contiene, o sea $d = \{SFM_1, SFM_2, \ldots, SFM_k\}$. Adicionalmente, como una estrategia para tener en cuenta la presencia de cada SFM en la colección D, cada SFM puede tener asociado un peso. Para la asignación de los pesos a estas SFM se utilizó el pesado TF-IDF [31].

2.2. Medidas de semejanza

Una vez que se tienen los documentos representados en algunos de los modelos anteriormente mencionados, resulta mucho más fácil para un algoritmo de agrupamiento procesarlos y encontrar una estructuración de la colección. Una de las operaciones básicas para el agrupamiento es la determinación del grado de similitud entre un par de documentos d_i y d_j denotada por $sim(d_i, d_j)$.

En el caso del Modelo de Espacio Vectorial, la medida más utilizada para determinar la semejanza entre un par de documentos es la medida del coseno [8]. Esta medida se define como:

$$sim(d_i, d_j) = cos(d_i, d_j) = \frac{(d_i \times d_j)}{||d_i|| \cdot ||d_j||} = \frac{\sum w_{ir} \cdot w_{jr}}{\sqrt{\sum w_{ir}^2 \cdot \sum w_{jr}^2}},$$

donde d_i, d_j son los vectores asociados a los documentos i, j, $||d_i||$, $||d_j||$ las normas de los vectores, y w_{ir}, w_{jr} son los pesos de los términos en los vectores d_i, d_j, respectivamente.

Otras medidas utilizadas también en este modelo son la distancia Euclidiana [49], el coficiente Dice y el coeficiente Jaccard [89].

En el caso del Modelo Basado en Conjuntos de SFM la semejanza entre un par de documentos d_i y d_j se calcula teniendo en cuenta el conjunto de secuencias comunes en ambos documentos [31]. Estas medidas de semejanzas se definen en 2.3 y 2.4.

$$sim(d_i, d_j) = 2 \cdot \frac{|SFM(d_i) \cap SFM(d_j)|}{|SFM(d_i)| + |SFM(d_j)|}, \tag{2.3}$$

donde $SFM(d_i)$ y $SFM(d_j)$ identifican a los conjuntos de secuencias frecuentes maximales que representan a los documentos d_i y d_j. Es importante mencionar que la fórmula (2.3) coincide con el coeficiente Dice [89] aplicado a los vectores de secuencias de cada documento.

Sea $C = SFM(d_i) \cap SFM(d_j)$, el conjunto de SFM comunes en ambos documentos. La semejanza entre d_i y d_j también se puede calcular como:

$$sim(d_i, d_j) = 2 \cdot \frac{\sum_{S_q \in C} W_{d_i}(S_q) \cdot W_{d_j}(S_q)}{\sum_{q=1}^{k} W_{d_i}(S_q) + \sum_{q=1}^{p} W_{d_j}(S_q)}, \tag{2.4}$$

donde $W_{d_i}(S_q)$ y $W_{d_j}(S_q)$ identifican a los pesos de la SFM S_q en los documentos d_i y d_j.

En el caso del modelo de Distancia Asociativa Global puede ser utilizada cualquiera de las medidas de semejanzas utilizadas en el modelo de espacio vectorial [56].

2.3. Trabajo relacionado

En esta sección se describen los principales trabajos reportados en la literatura referentes al agrupamiento de documentos, mostrándose para cada uno sus características así como limitaciones. Los algoritmos basados en grafos por estar más relacionados con el trabajo de esta tesis se expondrán detalladamente.

2.3.1. Algoritmos de pasada simple

Los algoritmos de *pasada simple* son aquellos en los que cada vez que se tiene que agrupar un documento, éste se compara con cada uno de los grupos existentes y se adiciona al grupo que logró el máximo valor de semejanza. En caso de no existir semejanza con algún grupo el documento conforma un grupo independiente.

De forma general estos algoritmos definen una función de semejanza entre un grupo y un objeto. Utilizando esta función y un umbral especificado previamente se decide a que grupo asignar dicho objeto. Las funciones que se han usado con frecuencia son:

- *Promedio*. Dado el grupo G y el elemento O, la semejanza entre ellos se calcula como el promedio de las semejanzas existentes entre O y todos los elementos de G.

- *Máxima semejanza*. Dado el grupo G y el elemento O, la semejanza entre ellos se calcula como el máximo entre las semejanzas existentes entre O y todos los elementos de G.

- *Representante*. En esta función se asume que cada grupo está representado por un objeto que, pertenece al grupo o se calcula a través de los elementos del grupo. Teniendo esto en cuenta, la semejanza entre el grupo G y el elemento O, se calcula como la semejanza entre O y R_G, siendo este último el representante del grupo G.

Un ejemplo de este tipo de algoritmos es el *Single-Pass* que ha sido utilizado en sistemas para la detección de tópicos [12, 78]. Este algoritmo representa cada grupo a través de su representante, el cual se calcula como la media de los objetos pertenecientes al grupo. El algoritmo funciona de la siguiente forma: cada elemento se compara con los representantes de cada grupo existente para determinar su semejanza, seleccionándose el grupo más semejante al objeto mientras esta semejanza sea superior a un cierto umbral β predefinido. A continuación el objeto es asignado al grupo determinado en el paso anterior, ajustando consecuentemente su representante. En caso de que no existan grupos con semejanzas superiores a β, se crea un nuevo grupo que contenga al objeto. La complejidad computacional de este algoritmo es lineal.

El Single-Pass es un algoritmo incremental y los grupos que se obtienen a través del mismo son disjuntos. La principal limitación de este algoritmo es que los grupos formados pueden variar de acuerdo al orden en que se procesan los documentos. Esta variabilidad en el resultado del agrupamiento también está provocada por el criterio de asignación de elementos a grupos, ya que cuando existe más de un grupo con semejanza máxima al documento que se está procesando la selección del grupo al cual se asigne el documento determinará el agrupamiento final.

2.3.2. Algoritmos basados en optimización

Estos algoritmos producen una estructuración de la colección de documentos a través de la optimización de una función que estima la calidad del agrupamiento realizado hasta el momento. Teniendo en cuenta esta función, comúnmente llamada *función objetivo*, el algoritmo iterativamente procede a "re-asignar" documentos de un grupo a otro siempre y cuando se optimice el valor de dicha función.

La definición de esta función objetivo, puede estar relacionada con medidas de semejanza o distancia entre los documentos que componen a cada uno de los grupos. Existen varias funciones objetivos reportadas en la literatura, sin embargo la más utilizada en este tipo de algoritmos es la que estima el error cuadrático [94].

Esta función puede ser definida utilizando una medida de distancia (2.5) ó de semejanza (2.6).

$$E = \sum_{i=1}^{k} \sum_{p \in C_i}^{|C_i|} |p - m_i|^2 \tag{2.5}$$

$$E = \sum_{i=1}^{k} \sum_{p \in C_i}^{|C_i|} F(p, m_i) \tag{2.6}$$

donde p representa a los documentos que pertenecen a cada grupo $C_i, i = 1...k$ que tiene por representante a m_i y $F(p, m_i)$ representa la función que estima la semejanza entre p y m_i. En el caso de documentos se utiliza con frecuencia como función de semejanza la medida del coseno.

Un algoritmo de optimización que utiliza esta función y que además resulta el más simple y frecuentemente utilizado es el *k-means* [30, 44, 60]. Este algoritmo representa los grupos a través de representantes o centroides que se calculan por la media (o promedio pesado) de los elementos pertenecientes al grupo. El algoritmo k-means comienza seleccionando aleatoriamente k elementos como centros iniciales de los grupos, y a cada objeto restante se le asigna el grupo con el cual tenga la mínima distancia o mayor semejanza. A partir de este punto se recalculan los centroides y se vuelve a iterar por los objetos re-asignando cada uno a su grupo más cercano.

Este proceso se repite hasta que se alcance algún criterio de convergencia. Usualmente el criterio puede ser: (a) no se afecta el agrupamiento, es decir ningún objeto cambia de grupo, o (b) la función objetivo deja de crecer ó disminuir, en dependencia si se utilizó la función 2.6 ó la 2.5. Los grupos que se obtienen producto de este algoritmo son grupos disjuntos. La complejidad computacional de este algoritmo es $O(nkT)$, donde T representa la cantidad de iteraciones hasta la convergencia del algoritmo.

Este algoritmo tiene varias deficiencias importantes. En primer lugar sólo puede ser

utilizado si los objetos están descritos en un espacio métrico, no agrupa correctamente los objetos aislados y puede caer en mínimos locales de la función objetivo. Otra deficiencia importante es que la calidad del agrupamiento resultante está determinada por la selección inicial de los centroides. Por último es importante mencionar que en este algoritmo es necesario determinar a priori el número de grupos en los que se desea particionar la colección.

2.3.3. Algoritmos jerárquicos aglomerativos

Este tipo de algoritmos comienza considerando como un grupo a cada documento de la colección y a partir de este punto comienza un proceso en el que iterativamente va mezclando los dos grupos más semejantes hasta obtener un solo grupo que contiene a todos los elementos. Adicionalmente a este criterio de convergencia y por motivos prácticos puede ser de utilidad no trabajar con toda la jerarquía sino sólo con un nivel de ésta. Este nivel o partición de la jerarquía puede obtenerse aplicando otros criterios como por ejemplo: (a) la inclusión de un parámetro que estime la semejanza mínima que deben tener dos grupos para poder ser mezclados y (b) la definición de una cantidad de grupos a obtener.

El punto clave en estos algoritmos es la función o método que determine cuándo dos grupos son cercanos o semejantes. Varios trabajos se han desarrollado con este propósito [35, 39]. En dependencia de la función que se considere pueden obtenerse diferentes métodos; los más utilizados para agrupar documentos son el *Single-link*, *Complete-Link* y *Average-Link*.

El Single-link [15, 75] se caracteriza por calcular la semejanza entre dos grupos C_i y C_j a través de la mayor semejanza existente entre un elemento $p \in C_i$ y uno $q \in C_j$. Teniendo en cuenta esta definición de semejanza entre grupos, el algoritmo iterativamente une los grupos que tengan mayor semejanza.

El Complete-Link [87] calcula la semejanza entre dos grupos C_i y C_j a través de la menor semejanza existente entre un elemento $p \in C_i$ y uno $q \in C_j$. Teniendo en cuenta esta definición de semejanza entre grupos, el algoritmo iterativamente une los grupos que

tengan mayor semejanza.

De forma general estos algoritmos obtienen malos resultados debido a que utilizan muy poca información (Single-link) o a que asumen que todos los documentos en un cluster están fuertemente relacionados (Complete-link) [94]. Un algoritmo que soluciona estos problemas, al utilizar una estrategia que representa un punto medio entre las anteriores, es el Average-link o UPGMA como también es conocido [87, 35].

El Average-link calcula la semejanza entre un par de grupos como el promedio de semejanza existente entre los objetos que componen a dichos grupos. Utilizando esta función, este algoritmo logra obtener grupos más cohesionados y que a la vez son menos sensibles al ruido.

La complejidad computacional de estos algoritmos es de $O(n^3)$; sin embargo se pueden lograr implementaciones con complejidad computacional de $O(n^2)$ [22]. La principal deficiencia de estos algoritmos es que los grupos formados dependen del orden de análisis de los elementos, además en el caso del Single-link los grupos formados presentan encadenamiento y como consecuencia son poco cohesionados. Por otra parte, los grupos formados por Complete-link y Average-link tienden a tener forma esférica.

2.3.4. Algoritmos jerárquicos divisivos

Los algoritmos de este tipo asumen inicialmente que todos los documentos pertenecen a un mismo grupo. A partir de este punto y aplicando técnicas particionales se realiza un proceso en el que iterativamente se selecciona un grupo para ser dividido en dos. Este proceso termina cuando se satisface algún criterio de convergencia. Generalmente los criterios utilizados pueden son: (i) se alcanza una cantidad de grupos determinada o (ii) los grupos formados satisfacen algún criterio de cohesión.

Teniendo en cuenta lo anteriormente mencionado podría utilizarse cualquier algoritmo particional con el cual ir realizando divisiones sucesivas y de esta forma lograr la jerarquía deseada. Algunos de los algoritmos particionales utilizados con estos propósitos pueden consultarse en [94].

El algoritmo divisivo más utilizado es el *Bisecting K-means* [81], que utiliza el al-

goritmo particional K-means para iterativamente ir dividiendo un grupo seleccionado. Generalmente el criterio para seleccionar el grupo puede ser: (i) el grupo más grande, (ii) el de menor semejanza intra-grupo o (iii) una combinación de ambos.

La complejidad computacional de este algoritmo es de $O(n^2)$ y al utilizar K-means como algoritmo para dividir los grupos presenta las mismas deficiencias de éste (ver 2.3.2). Ademas de lo anterior, existen trabajos que plantean que, respecto a la calidad del agrupamiento, estos algoritmos obtienen peores resultados que sus contrapartes aglomerativas [44].

2.3.5. Algoritmos basados en árboles

Este tipo de algoritmos como su nombre lo indica crean un árbol con los documentos de acuerdo a determinados criterios; de esta forma los nodos del árbol resumen las características de los documentos que contienen. Ejemplos de este tipo de algoritmo son el STC [93] y el DC-tree [88].

El algoritmo STC (*Suffix Tree Clustering*) fue creado para agrupar los *snippets*[1] devueltos en una consulta web. Este algoritmo, a diferencia de muchos otros algoritmos de agrupamiento, representa a los documentos como una secuencia ordenada de palabras logrando de esta forma utilizar la información sintáctica que se puede extraer de esta secuencia para realizar el agrupamiento.

Este algoritmo está compuesto de tres pasos. El primer paso se encarga de realizar un proceso de lematización de los documentos (snippets) a agrupar y eliminar todo aquello que no sea una palabra. Al final de este paso cada documento está representado por una secuencia de palabras que mantienen el orden de acuerdo al documento original. El segundo paso tiene como objetivo construir los *grupos base* que serán utilizados para obtener el agrupamiento final.

Estos grupos base se obtienen del árbol de sufijos [84, 27] que se puede construir a partir de todos los posibles sufijos que se pueden descubrir utilizando todos los documentos

[1] Los snippets son los pequeños textos utilizados por los sistemas de búsqueda como Google para describir los resultados de las búsquedas

de la colección. Cada nodo de este árbol contiene el conjunto de documentos que tienen en común el sufijo formado por el camino desde la raíz del árbol hasta el nodo en cuestión. Cada conjunto de documentos perteneciente a un nodo formará un grupo base.

El último paso aplica una heurística, similar a la del algoritmo Single-link, en la que utilizando cierta medida de semejanza entre grupos, iterativamente se unen los grupos base que sean semejantes. El agrupamiento resultante estará formado por los grupos construidos en este tercer paso. Los grupos obtenidos pueden ser solapados.

El algoritmo DC-Tree (*Document Clustering Tree*) fue creado para agrupar páginas web. Este algoritmo representa los documentos (páginas web) a partir de un vector de pesos $W_i = (w_{i1}, w_{i2}, \ldots, w_{in})$, donde cada valor w_{ik} es el peso asociado al término t_k. El peso de un término es 1 si el mismo pertenece al documento y 0 en otro caso. El número de términos utilizados para representar a los documentos se calcula utilizando un método de selección de variables aplicado a una muestra de la colección. Teniendo los documentos representados en este modelo este algoritmo construye iterativamente un árbol denominado DC-tree.

El DC-tree es un árbol que se contruye de forma iterativa insertando cada uno de los documentos. En este proceso de inserción, de forma similar al B^+-tree, se realizan operaciones de ajuste para mantener la estructura del árbol. Ver [88] para ver detalles de esta estructura.

Luego de construido el DC-tree se realiza un proceso de búsqueda en amplitud sobre dicho árbol para identificar aquellos nodos que representen grupos "interesantes". Este criterio de interés se define teniendo en cuenta la cantidad de documentos almacenados en dicho nodo y el vector de pesos perteneciente al nodo. El agrupamiento resultante está formado por estos grupos interesantes. Los grupos que se obtienen de este algoritmo son disjuntos.

Estos dos algoritmos son incrementales por lo que pueden ser utilizados en colecciones en las cuales se adicionen documentos a través del tiempo. Sin embargo, estos algoritmos presentan algunas deficiencias. En el caso del STC, la construcción del árbol de sufijos resulta extremadamente costosa respecto a la cantidad de términos de los documentos

por lo que sólo ha sido probado con documentos de pequeña dimensionalidad como los snippets que no sobrepasan las 35 palabras. Otros aspectos negativos en este algoritmo es que los grupos resultantes del mismo pueden presentar encadenamiento y que requiere prefijar valor para 1 parámetro.

En el caso del DC-tree, el algoritmo puede dejar documentos sin agrupar, debido a los criterios definidos para considerar un grupo interesante. Otro aspecto negativo de este algoritmo es que requiere prefijar valores para 9 parámetros. Es importante mencionar además que este algoritmo sólo ha sido probado con páginas web las cuales tienen características diferentes a los documentos como por ejemplo la frecuencia de los términos. Por lo antes planteado, el algoritmo DC-tree puede resultar ineficaz para agrupar documentos ya que en éstos el pesado binario puede ser inadecuado.

2.3.6. Algoritmos basados en densidad

Los algoritmos de este tipo, dado un conjunto de datos D, definen un criterio de densidad para un grupo y tratan de encontrar grupos o divisiones $D_1, D_2, \ldots, D_k$ de este conjunto de datos de forma tal que las densidades de estas divisiones sean las más cercanas posibles. En el mejor de los casos, estos algoritmos son capaces de detectar automáticamente el número de grupos k en los que se deben dividir los datos, así como son capaces de descubrir grupos de diferentes formas y tamaños. Un ejemplo de estos algoritmos es el MajorClust [79].

En el algoritmo MajorClust el criterio de densidad de un grupo está asociado a la *atracción* que ejerce dicho grupo sobre los elementos que lo componen. Este criterio de atracción está relacionado con la cantidad de elementos del grupo así como también a la semejanza existente entre los elementos del mismo [45]. Formalmente, dado un grupo C y un documento d, la atracción que ejerce C sobre d se calcula como:

$$\sum_{p \in C} \varphi(p, d),$$

donde $\varphi(p, d)$ representa la semejanza existente entre el documento $p \in C$ y el documento

d.

Este algoritmo, dada una colección de documentos D, contruye grupos disjuntos asignando cada documento d en el grupo que ejerza sobre él la mayor atracción. MajorClust comienza formando un grupo unitario con cada documento, a partir de este punto comienza un proceso en el que iterativamente para cada documento d se calcula la atracción que ejerse cada grupo sobre él teniendo en cuenta solamente aquellos elementos del grupo cuya semejanza con d sea mayor que cierto umbral t. Una vez que se calculan dichos valores se asigna a d en el grupo con mayor valor de atracción. Si existen varios grupos se selecciona uno aleatoriamente. Este proceso se repite hasta que no hayan cambios en los grupos.

Este algoritmo tiene una complejidad computacional de $O(n^2)$ y aunque necesita del parámetro t los autores comentan que puede asignársele siempre el valor 0,3. Sin embargo este algoritmo es no determinista puesto que cuando existen varios grupos con el mismo valor de atracción se selecciona uno aleatoriamente. Esta deficiencia puede llevar a que ciertos grupos no sean correctamente identificados [79]. Por último, cabe decir que este algoritmo ha sido utilizado para agrupar colecciones de resúmenes [3] así como documentos de mayor tamaño [63].

2.3.7. Algoritmos basados en conjuntos frecuentes de ítems

La alta dimensionalidad de los documentos es una característica que hace muy costoso utilizar muchos de los algoritmos clásicos en el contexto del agrupamiento de documentos. Una alternativa a este problema son los agrupamientos basados en conjuntos frecuentes de ítems (FI, por sus siglas en inglés) [1]. Ejemplos de estos algoritmos son el *FTC* [7, 73] y el *FIHC* [18].

El algoritmo FTC construye un conjunto de grupos disjuntos utilizando los conceptos de *cubrimiento de un FI*, *conjunto descriptivo de un agrupamiento* y la definición de una medida para estimar el solapamiento de un grupo en relación con el resto de los grupos determinados.

Sea $F = \{F_1, F_2, \ldots, F_N\}$, el conjunto de todos los FI presentes en una colección de documentos D y que fueron calculados por algún algoritmo, como por ejemplo *Apriori*

[1]. El cubrimiento de F_k se denota como $cov(F_k)$ y define al conjunto de documentos $d_j \in D$ que contienen a los ítems contenidos en F_k. Por otra parte el conjunto descriptivo de un agrupamiento es el sub-conjunto $CD \subseteq F$, tal que se cumple la condición siguiente:

$$\bigcup_{F_k \in CD} cov(F_k) = D \tag{2.7}$$

Sea $f_j \in F$ un conjunto frecuente de ítems, y G el grupo formado a partir de los documentos en $cov(f_j)$, una medida de solapamiento aplicada a G permite estimar la cantidad de conjuntos frecuentes que tiene en común éste con los grupos que se forman a partir del cubrimiento de cada conjunto $f_k \in F$, $k = 1...N, k \neq j$. Mientras menos conjuntos frecuentes contienen los documentos $d \in G$ menor será el valor de esta medida y por consiguiente el solapamiento de este grupo. Las medidas que se han definido para calcular dicho solapamiento son el *Standard overlap* (SO), *Entropy overlap* (EO), *Improve Standard overlap* (ISO) e *Improve Entropy overlap* (IEO) [7, 73].

Utilizando lo anteriormente mencionado, FTC construye iterativamente un conjunto de grupos $CG = \{g_1, g_2, \ldots, g_M\}$ de forma tal que cada grupo adicionado tenga el menor solapamiento respecto a los grupos anteriormente calculados. Para construir los grupos se parte del conjunto $F = \{F_1, F_2, \ldots, F_N\}$ e iterativamente se selecciona el F_k cuyo grupo formado a partir de los documentos $d_j \in cov(F_k)$ tenga el menor valor de solapamiento. Luego de esta selección, se elimina de F al elemento F_k y se adiciona a CG el grupo formado por los documentos $d_j \in cov(F_k)$ que no pertenezcan a algún grupo ya calculado.

El algoritmo FIHC permite obtener un conjunto de grupos disjuntos y a partir de este conjunto puede obtenerse además una jerarquía de grupos. Este algoritmo se apoya en los conceptos de *ítem frecuente global* e *ítem grupo-frecuente*. El concepto de ítem frecuente global es el mismo que se utiliza para definir a un ítem frecuente en [1]. Un ítem f es grupo-frecuente respecto al grupo G_i si, dado un umbral α, el ítem está contenido en al menos α documentos que pertenecen a G_i.

Para obtener el conjunto de grupos disjuntos este algoritmo parte del conjunto $F = \{F_1, F_2, \ldots, F_M\}$ de todos los FI de la colección. Como primer paso se contruye un conjunto de grupos $CG = \{G_1, G_2, \ldots, G_k\}$ donde cada $G_i = cov(F_k)$, $i = 1...k$. Como

se puede notar los grupos formados pueden ser solapados. Debido a lo anterior, en el segundo paso se procede a asignar cada documento $d_j \in D$ en el grupo más adecuado.

Un grupo G_i es adecuado para contener a un documento d_j si existen muchos ítems globalmente frecuentes en d_j que son grupo-frecuentes en G_i. La siguiente fórmula es utilizada para calcular el índice de pertenencia de un documento a un grupo:

$$Score(G_i, d_j) = \left(\sum_x n(x) \cdot s - grupo(x) \right) - \left(\sum_{x'} n(x') \cdot s - global(x') \right) \tag{2.8}$$

donde x representa a aquellos ítems en d_j que son frecuentes globalmente y que a la vez son grupo-frecuentes respecto a G_i; x' representa a aquellos ítems en d_j que son frecuentes globalmente y que no son grupo-frecuentes respecto a G_i; $n(x)$ y $n(x')$ representan la frecuencia de los ítems x y x' en d_j.

Utilizando la expresión 2.8 se determina para cada documento d_j el grupo con mayor índice de pertenencia, se asigna d_j a dicho grupo y se elimina del resto de los grupos. En el caso de que existan varios grupos con el mismo valor de índice, se selecciona aquel que fue formado por el FI de mayor cardinalidad. El grupo que resulte vacío producto de la operación anterior es eliminado.

Luego de obtener este conjunto de grupos disjuntos, se puede construir una jerarquía de grupos utilizando una estratégia aglomerativa que parte de los grupos formados por los FI de mayor cardinalidad o longitud y que utiliza una variante de la función definida en 2.8 [18].

La principal deficiencia de estos métodos es que los grupos que se obtienen dependen tanto del valor de umbral de soporte mínimo para el cálculo de los FI, así como (en el caso del FIHC) del umbral definido para determinar los ítems grupo-frecuentes. Además, los grupos que se obtengan para un cierto valor de los umbrales dependen del orden de análisis de los FI en el caso del FTC, así como del orden de los documentos en el caso del FIHC. Otro aspecto negativo del FIHC es que puede dejar documentos sin agrupar.

2.3.8. Algoritmos basados en técnicas genéticas

Los algoritmos genéticos son heurísticas que ayudan a solucionar problemas que por vías analíticas serían difíciles de resolver debido al costo computacional involucrado. Este tipo de heurísticas tratan de emular el comportamiento evolutivo de los seres vivos a través de procesos como cruzamiento, mutación o selección natural [33]. Ejemplos de algoritmos de agrupamiento de documentos que utilizan estas técnicas ó heurísticas se pueden apreciar en [13] y [77].

El algoritmo desarrollado en [13] determina dinámicamente el número de grupos para particionar la colección de documentos. Para realizar este paso se selecciona un subconjunto o muestra de los documentos y si la cantidad de documentos de ésta es menor a 15 se aplica el algoritmo de Calinski y Harabasz [11], en otro caso se aplica un algoritmo genético. El algoritmo genético determina el número de grupos óptimo partiendo del árbol de cubrimiento mínimo que se puede obtener a través del grafo de semejanza que se forma con los documentos de la muestra.

Una vez que se determina el número de grupos a obtener, éstos se construyen utilizando algún método de agrupamiento de la librería de métodos CLUTO (Single-link, Complete-link ó UPGMA)[38].

En [77] se desarrolla un algoritmo llamado MVGA, el cual calcula dinámicamente el número óptimo de grupos en los que se debe agrupar la colección y durante este proceso obtiene también dicho conjunto de grupos. Para obtener dicho conjunto de grupos, este algoritmo se basa en una estrategia similar a la del algoritmo k-means y en una implementación de un algoritmo genético. En este algoritmo los individuos son vectores donde, cada elemento del vector representa a un centroide de un grupo del agrupamiento a obtener. Estos centroides son seleccionados aleatoriamente del conjunto de documentos de la colección, los cuales son previamente representados utilizando el modelo de Espacio Vectorial.

Partiendo de una población de 100 individuos y utilizando como función de aptitud el índice DB [17], este algoritmo realiza un proceso evolutivo mediante los operadores

de selección, cruzamiento y mutación con los cuales se exploran al mismo tiempo diferentes formas de particionar al conjunto de documentos. Al ejecutar este proceso un cierto número de iteraciones o al alcanzar la condición de convergencia, el individuo con mejor valor de la función aptitud y que pertene a la última generación, representa al agrupamiento óptimo entre todos los explorados.

Estos algoritmos presentan varias deficiencias, una de ellas es que resultan costosos cuando trabajan con datos de alta dimensionalidad, ya que el proceso evolutivo se vuelve muy complejo. Debido a esto en el caso del algoritmo MVGD se realizó un proceso de selección de términos para reducir la dimensionalidad, es importante notar que este proceso puede afectar la calidad del agrupamiento resultante. Otro aspecto negativo de este algoritmo es que necesita para su ejecución asignar valores a 5 parámetros.

Una deficiencia notable en el algoritmo desarrollado en [13] es que el proceso de determinar dinámicamente el número de grupos depende en gran medida de la calidad de la muestra seleccionada, si ésta no es representativa de la colección entonces el número de grupos calculados puede no ser el más idóneo. Otra deficiciencia de este algoritmo es que sólo utiliza 10 individuos para conformar la población inicial por lo que la diversidad de ésta puede resultar baja y por consiguiente aumenta la posibilidad de que el algoritmo converja a un óptimo local o que aumente el tiempo de convergencia al óptimo global.

Por último es importante comentar que los experimentos realizados con cada uno de los algoritmos anteriores utilizaron colecciones pequeñas de documentos que contenían 200 documentos como máximo, por lo que es muy posible que resulten ineficientes o incluso imposibles de aplicar con respositorios de documentos de mayor tamaño.

2.3.9. Algoritmos basados en factorización de matrices

Teniendo en cuenta el modelo de Espacio Vectorial (ver sección 2.1.2), una colección de documentos puede representarse como una matriz de V de dimensión $m \times n$, donde n representa a la cantidad de documentos de la colección y m a la cantidad de términos presentes en ésta. Cada elemento V_{ji} representa el peso del término t_j en el documento d_i, o sea un valor mayor o igual a cero, por lo que se puede decir que V es una matriz no

negativa.

Los algoritmos basados en factorización de matrices, y en específico de matrices no negativas (como el caso de V), se basan en conceptos del álgebra vectorial. Asumiendo que en la colección hay presentes k tópicos o grupos, cada uno expresado a través de un vector de términos, un documento d_i de la colección pudiera verse como una combinación lineal de estos vectores de tópicos si consideramos el espacio semántico de dimensión k del cual este conjunto de vectores es base.

Dado un valor k que representa el número de grupos disjuntos en los que se desea agrupar una colección D, un algoritmo de este tipo aproxima la matriz no-negativa V que representa a D mediante el producto de dos matrices no-negativas W y H de dimensiones $m \times k$ y $k \times n$ respectivamente. Cada columna de W contiene al vector que define a un grupo presente en la colección mientras que cada columna de H contiene el vector de coeficientes utilizado para aproximar el vector de la correspondiente columna de V. Una vez que se tienen estas matrices, se recorre cada documento d_j asignándolo al grupo que más aportó en la combinación lineal que forma a dicho documento.

Un ejemplo de algoritmos de este tipo se puede apreciar en los trabajos de Xu *et al* [91] y los de Shahnaz [72]. Ambos métodos parten de inicializar las matrices W y H con valores no negativos y a partir de este punto realizan un proceso en el cual iterativamente se va ajustando los valores de W_{ji} y H_{ji} mediante transformaciones sucesivas. Los métodos son diferentes en cuanto a formas de actualizar los valores de cada matriz. La complejidad computacional de estos algoritmos es $O(tkn)$ y $O(tkmn)$, donde t es el número de iteraciones realizadas.

La principal deficiencia de estos algoritmos es que el resultado del agrupamiento depende de qué tan buena sea la aproximación realizada de V. Si durante el proceso de aproximación no se llegó a la condición de convergencia sino que dicho proceso se detuvo al realizar un número de iteraciones prefijado, es muy posible que la calidad del agrupamiento sea baja. Otro aspecto a señalar es que este algoritmo necesita fijar a priori el número de grupos a obtener por lo que el resultado depende de este valor.

2.3.10. Algoritmos basados en grafos

Como se mencionó anteriormente, los algoritmos de agrupamiento de documentos basados en grafos representan la colección de documentos a través de un grafo $G = \langle V, E, w \rangle$ en el cual los vértices representan a los documentos y las aristas están etiquetadas con el valor de semejanza existente entre los vértices que la componen. Una vez representada la colección, estos algoritmos construyen un cubrimiento de este grafo utilizando propiedades de la Teoría de Grafos. Los grupos resultantes de este algoritmo estarán formados por los elementos de este cubrimiento.

Algunos de los algoritmos de agrupamiento de documentos que utilizan esta técnica son: GLC [70, 76], Compacto Incremental [65, 64], Fuertemente Compacto Incremental [67, 64], Star [4, 5], Exteded Star [23, 24, 22] y Generalized Star [69]. Todos estos algoritmos varían respecto al tipo de grafo que utilizan, a la heurística utilizada para obtener el cubrimiento de dicho grafo e incluso respecto al tipo de cubrimiento que obtienen. Con el objetivo de describir estos algoritmos se hace imprescindible introducir algunos conceptos básicos.

Conceptos básicos

Sea $D = \{d_1, d_2, \ldots, d_N\}$ una colección de documentos y S una función de semejanza simétrica entre documentos, es decir dado el par de documentos d_i y d_j se cumple que $S(d_i, d_j) = S(d_j, d_i)$. Un *grafo de semejanza* $G = \langle V, E, w \rangle$, es un grafo no dirigido etiquetado en el cual los vértices representan a los documentos de la colección, cada arista (d_i, d_j) representa la semejanza existente entre los documentos que la componen y w es la función que asigna los pesos a las aristas.

Sea $\beta \in \Re$, tal que $\beta \in [0, 1]$, un umbral de semejanza y G un grafo de semejanza. Un *grafo de β-semejanza* se denota por $G_\beta = \langle V, E_\beta \rangle$ y es el grafo no dirigido que se obtiene a partir de G si se eliminan todas las aristas $e \in E$ tal que $w(e) < \beta$.

Dado el grafo $G_\beta = \langle V, E_\beta \rangle$, dos documentos d_i, d_j representados por los vértices $v_i, v_j \in V$ serán denominados *β-semejantes* si existe la arista $(v_i, v_j) \in E_\beta$. Los vértices v_i para los que no exista $v_j \in V, v_j \neq v_i$ tal que v_i y v_j sean β-semejantes, serán denominados

β-aislados.

Un *grafo de máxima semejanza* se denota por $G_{max} = \langle V, E \rangle$, y es el grafo dirigido donde los vértices representan a los documentos $d_i \in D$, y donde existe una arista $(v_i, v_j) \in E$, si se cumple que $Sim(d_i, d_j) = max_{d_p \in D} Sim(d_i, d_p)$.

Dado un grafo dirigido $G = \langle V, E \rangle$, llamaremos *Clausura-O de G* al grafo no dirigido $\bar{G} = \langle V, \bar{E} \rangle$, tal que para cada par de vértices $v_i, v_j \in V$ existe una arista no dirigida $(v_i, v_j) \in \bar{E}$, si y sólo si $(v_i, v_j) \in E$ o $(v_j, v_i) \in E$.

Algoritmo GLC

Uno de los casos más simples de algoritmos basados en grafos es el algoritmo GLC, el cual obtiene de forma incremental las componentes conexas del grafo de β-semejanza las cuales representan a los grupos que se desean obtener.

El algoritmo GLC realiza un proceso en el cual cada documento d_i se analiza una sola vez, determinándose para éste cuáles son los documentos pertenecientes a los grupos existentes que son β-semejantes con él. Si el documento d_i no tiene elementos β-semejantes entonces se crea un nuevo grupo unitario que contiene a d_i. En caso contrario todos aquellos grupos que contienen algún documento β-semejante con d_i, se unen y forman un nuevo grupo en el que también se adiciona a d_i.

La complejidad computacional de este algoritmo es $O(n^2)$ ya que para cada documento es necesario calcular la semejanza de éste con los restantes. Sin embargo, es importante mencionar que en la práctica este algoritmo en muchos casos no requiere calcular siempre todas estas semejanzas. Lo anterior se debe a que cuando se encuentra un documento d_j que es β-semejante con d_i, ya no es necesario calcular la semejanza de d_i con el resto de los documentos que pertenecen al mismo grupo que d_j.

El resultado del algoritmo GLC es independiente del orden de análisis de los documentos y no impone restricciones a la forma de los grupos obtenidos. Su mayor limitación es que las componentes conexas sobre el grafo de β-semejanza presentan un elevado efecto de encadenamiento por lo que pueden obtenerse grupos poco cohesionados, es decir grupos que incluyan documentos muy poco semejantes.

Algoritmo Compacto Incremental

Este algoritmo, dado el grafo de máxima semejanza $G_{max} = \langle V, E \rangle$ que representa la colección de documentos D, obtiene de forma incremental un conjunto de grupos disjuntos $G = \{G_1, G_2, \ldots, G_K\}$ en el que cada G_i es un *conjunto compacto*. Los conjuntos compactos coinciden con las componentes conexas del grafo $\bar{G}_{max}$ [64].

Cada vez que se inserta un objeto O en el grafo, se calcula la semejanza de éste con los objetos existentes y se actualiza G_{max}. Esta actualización involucra eliminar aristas existentes en G_{max} y adicionar otras nuevas, por lo tanto puede producir cambios en el agrupamiento, ya que pueden surgir nuevos conjuntos compactos y otros conjuntos compactos existentes pueden dejar de serlo. Debido a lo anterior, luego de que se produce la actualización del grafo, se realiza un proceso de reconstrucción de los conjuntos compactos a partir del objeto O y de aquellos grupos que pueden dejar de ser compactos.

Los grupos a tener en cuenta en este proceso son aquellos que están conectados con el objeto O. Un grupo G está conectado con O si existe un objeto $O' \in G$ que cumple que O es el objeto más β-semejante de O' o viceversa. De lo anterior podemos deducir que aquellos grupos que no estén conectados con O no pierden la propiedad de ser compactos así como que, si no existe ningún grupo conectado con O, entonces se crea un nuevo grupo formado únicamente por el propio O.

Una vez identificados los grupos conectados con O el algoritmo construye los siguientes conjuntos:

- ***Grupos a Procesar***. Estos grupos representan a aquellos conjuntos compactos que pueden perder la propiedad de ser compactos y por consecuencia deben de ser reconstruidos.

- ***Objetos a Unir***. A este conjunto pertenecen todos aquellos objetos O' que tienen como único objeto más β-semejante a O y que antes de la inserción de éste eran β-aislados o tenián solamente un objeto más β-semejante.

- ***Grupos a Unir***. Un grupo G_i pertenece a este conjunto si no pertenece a *Grupos*

a Procesar y cumple además que: (i) está conectado con O y (ii) la componentes conexa que lo representa en $\bar{G}_{max}$ no perdió ninguna arista producto de la inserción de O. Los elementos de estos conjuntos pertenecen a la misma componente conexa de O en $\bar{G}_{max}$ y por lo tanto a su mismo conjunto compacto.

Una vez construidos estos conjuntos, se forma un nuevo grupo compacto que contiene al objeto O y a todos los objetos en *Objetos a Unir*. Posteriormente se procesan cada uno de los grupos insertados en *Grupos a Unir* para reconstruir los conjuntos compactos. Para cada grupo $g_i \in$ *Grupos a Unir*, se sigue un proceso iterativo en el cual se selecciona un objeto $O' \in g_i$ y se construye su componente conexa en $\bar{G}_{max}$; esta componente conexa es un nuevo conjunto compacto y por lo tanto se adiciona al conjunto de grupos determinados. Este proceso termina cuando todos los objetos de g_i pertenecen a algún conjunto compacto.

Al finalizar el proceso de reconstrucción de los conjuntos compactos se procede a eliminar todos los grupos pertenecientes a *Objetos a Unir* y *Grupos a Procesar* de la lista de grupos existentes.

La complejidad computacional de este algoritmo es de $O(n \cdot (e+nm)))$, donde $e = |E|$, $n = |V|$ y m es la cantidad de rasgos que describen a los objetos. No obstante a lo anterior y considerando que en la práctica la cantidad de objetos más semejantes a un objeto dado es pequeña (1 en muchos casos) se toma $e = cn$, donde c es la cantidad máxima de objetos más semejantes a uno dado. Teniendo en cuenta la consideración anterior la complejidad computacional de este algoritmo es $O(n^2)$.

Por último cabe mencionar que los agrupamientos construidos por el algoritmo compacto incremental son independientes del orden de análisis los documentos. Sin embargo, este algoritmo presenta algunas deficiencias: (i) por lo general la cantidad de grupos que se obtienen es grande, (ii) los grupos pueden tener encadenamiento y ser poco cohesionados, aunque en menor magnitud que los grupos generados por el GLC.

Algoritmo Fuertemente Compacto Incremental

Este algoritmo también representa la colección de documentos a través de su grafo de máxima semejanza pero, a diferencia del algoritmo Compacto Incremental, permite obtener un conjunto de grupos que pueden ser solapados, los cuales son relativamente pequeños y densos [67, 64].

El algoritmo Fuertemente Compacto Incremental basa su funcionamiento en la relación existente entre los conjuntos compactos y los fuertemente compactos la cual establece que *"Todo conjunto compacto es la unión finita de conjuntos fuertemente compactos"* [50]. Utilizando la relación anterior, los conceptos de *grafo reducido según la relación de fuerte--conexidad* y *base de un grafo dirigido*, este algoritmo construye un conjunto de grupos solapados en los que cada grupo es un conjunto fuertemente compacto en G_{max}.

Se llamará *base del grafo dirigido* $G = \langle V, E \rangle$ al conjunto de vértices $B \subseteq V$ tal que el mismo satisface las condiciones siguientes [37]:

1. Para todo $v_i \in V$ existe un camino desde un vértice en B a v_i.

2. El conjunto B es el menor que cumple esta propiedad.

Sea $G = \langle V, E \rangle$ un grafo dirigido, se llamará *grafo reducido según la fuerte-conexidad de* G al grafo denotado por $G_r/f\text{-}C$ cuyos vértices son las componentes fuertemente conexas de G y en el cual existe una arista del vértice v a otro v' si existe al menos una arista de un vértice que pertenece a la componente fuertemente conexa representada por v a otro vértice en aquella componente representada por v'.

De forma similar al algoritmo Compacto Incremental, cada vez que se inserta un nuevo elemento O en $\bar{G}_{max}$ se calcula la semejanza de éste con todos los objetos existentes y como consecuencia se actualiza el grafo. Posteriormente se reconstruyen los conjuntos compactos y a partir de éstos los conjuntos fuertemente compactos. Para lograr esto, para cada sub-grafo determinado por los conjuntos compactos, se construye el grafo reducido asociado y su correspondiente base. Una vez que se tiene la base del grafo reducido, los conjuntos fuertemente compactos se construyen utilizando los vértices que componen dicha base [64].

Este algoritmo es incremental, su complejidad computacional es $O(n^2)$, no depende del orden de análisis de los documentos, no necesita conocer el número de grupos a determinar y estos grupos obtenidos pueden tener formas arbitrarias. La principal limitante de este algoritmo es que obtiene muchos grupos y generalmente cada grupo contiene pocos elementos.

Algoritmo Star

El algoritmo Star fue propuesto por Aslam [4, 5] y ha sido utilizado en varias aplicaciones relacionadas con el filtrado y organización de información. Este algoritmo construye un conjunto de grupos G, que pueden ser solapados, a partir del cubrimiento del grafo G_β utilizando *sub-grafos en forma de estrella*, cada sub-grafo determina un grupo de G. El cubrimiento realizado por el algoritmo Star es similar al cubrimiento que se obtiene si se utilizaran cliques (el cubrimiento a partir de cliques es un problema NP-completo [95]); de esta forma Star construye un conjunto de grupos menos cohesionados que los que se formarían utilizando los cliques, pero con una complejidad computacional mucho menor.

Un *sub-grafo en forma de estrella*, es un sub-grafo de $m+1$ vértices, en el cual existe un vértice llamado "centro", m vértices denominados "satélites" y se cumple que: (i) el centro tiene un grado mayor o igual que el resto de los vértices del sub-grafo y (ii) existe una arista del centro a cada uno de los satélites. El problema de encontrar los sub-grafos en forma de estrella se reduce al problema de determinar el conjunto X de vértices centro.

El algoritmo Star sigue una estrategia *greedy*. Como primer paso, se construye el grafo $G_\beta = \langle V, E_\beta \rangle$ y crea una lista L con todos los vértices del grafo ordenados de mayor a menor respecto a su grado. Este ordenamiento permite seleccionar primero los vértices centro que forman sub-grafos más densos. Seguidamente se realiza un proceso que selecciona iterativamente el vértice $v \in L$ de mayor grado que no pertenezca a ningún sub-grafo determinado en iteraciones anteriores. Luego, el vértice v se adiciona al conjunto X y se elimina de L junto con sus vértices adyacentes. Este proceso se repite hasta que lista L queda vacía, garantizando que cada vértice pertenezca al menos a un sub-grafo.

Este algoritmo no necesita conocer el número de grupos a obtener sino que lo descubre

a través de las propias características de los documentos de la colección. El mismo tiene una complejidad computacional de $O(n^2)$ y garantiza una semejanza de al menos β entre los centros y sus respetivos satelites de cada sub-grafo (grupo) determinado. Aunque entre dos satélites de un mismo sub-grafo no se garantiza este valor de semejanza en [5] se exponen resultados que argumentan que el valor de esta semejanza es relativamente alta y que los sub-grafos de este tipo conforman grupos con una semejanza promedio, entre los elementos que componen a los grupos, relativamente elevada. No obstante a lo anterior este algoritmo tiene algunas deficiencias.

La primera deficiencia radica en que el agrupamiento resultante depende del orden de análisis de los vértices. Esta deficiencia se pone de manifiesto en el momento en que se tiene que seleccionar el vértice de mayor grado para ser incluido en el conjunto X; si en este momento existe más de un vértice con el mismo grado y éstos son adyacentes, el agrupamiento resultante puede variar. Esta situación se expone en los grafos (A) y (B) de la Figura 2.1, en la cual los vértices negros representan a aquellos incluidos en X.

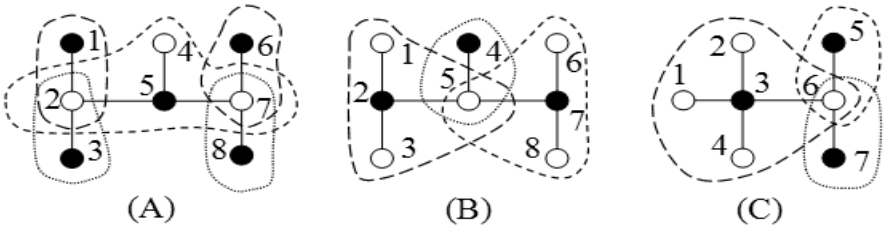

Figura 2.1: Deficiencias del algoritmo Star

Como se puede observar en la Figura 2.1, El grafo (A) muestra el agrupamiento resultante si el algoritmo analiza primero el vértice 5. Sin embargo, si se analizan primero cualquiera de los vértices 2 ó 7, el conjunto de grupos que se obtiene difiere notablemente como se puede observar en el grafo (B). Es importante mencionar que este ejemplo no sólo muestra cuan diferentes pueden ser los agrupamientos de un algoritmo que sea dependiente del orden, sino cómo esta dependencia puede afectar la calidad del agrupamiento al no permitir que se descubran grupos densos.

La segunda deficiencia de este algoritmo es que, independientememte del orden en que

se procesen los elementos, se pueden obtener grupos "ilógicos". En esta tesis se entenderá que un grupo g_1 es ilógico si cumple las siguientes condiciones:

- Existe un elemento $e \in g_i$ que es más denso[2] que el vértice centro c que define a g_1.
- El elemento e puede agrupar, si se considera como centro, a los vértices que son agrupados sólo por el centro c.

Esta limitación se debe a que el algoritmo no permite que dos vértices centro sean adyacentes. Como se puede observar en el grafo (C) de la Figura 2.1, el vértice 6 debería haber sido seleccionado como centro (ser incluido en X) y no sus correspondientes adyacentes menos densos.

Algoritmo Extended Star

El algoritmo Extended Star fue desarrollado por Gil *et al.* [23] con el objetivo de solucionar las deficiencias detectadas en Star. Este algoritmo sigue una heurística semejante a la que aplica Star para la construcción del agrupamiento de G_β, pero a diferencia de este último, define un nuevo concepto de *centro* que determina un conjunto de vértices X diferente al que obtiene Star y por consecuencia, un conjunto de grupos (sub-grafos) diferentes.

Un vértice $v \in V$ se considera como *centro* si tiene un adyacente v' tal que en éste se satisface alguna de las siguientes condiciones:

- v' no tiene vértices centro adyacentes.
- Si se considera a v'' como el vértice centro de mayor grado que es adyacente a v', entonces se cumple que v'' tiene un grado no mayor al de v.

La heurística que sigue este algoritmo para determinar el conjunto de vértices X, que definen el agrupamiento, utiliza además del grado de los vértices su *grado complemento*. Dado un vértice v, su *grado complemento* se denota como $v.GC$ y es la cantidad de

[2]El criterio de densidad que utiliza el algoritmo Star tiene en cuenta solamente el grado del vértice centro del sub-grafo que forma al grupo

adjacentes que tiene v que no pertenecen a ninguno de los grupos formados a partir de los vértices insertados en el conjunto X hasta el momento.

El funcionamiento del algoritmo Extended Star tiene como primer paso la construcción de G_β para lo cual se calcula la semejanza entre todo par de vértices. Posteriormente se insertan en X todos los vértices aislados y se construye una lista L con el resto de los vértices del grafo. Una vez formada L se comienza un proceso en el cual, iterativamente se selecciona el sub-conjunto $M \subseteq L$ de vértices con mayor grado complemento y de éste último, el sub-conjunto $M' \subseteq M$ de mayor grado. Cada uno de los elementos incluidos en M' son procesados y aquellos que, cumplan las condiciones para ser centro y formen un grupo diferente a los determinados hasta el momento, son adicionados a X. A continuación se eliminan los elementos de M' de la lista L, se actualiza el grado complemento de los vértices que quedaron en L y se repite el proceso hasta que todos los elementos estén agrupados. Este criterio de parada puede hacer que el algoritmo caiga en un ciclo infinito, dejando de agrupar elementos; incluso si se modifica el criterio de paro para evitar el ciclo infinito, el algoritmo puede dejar elementos sin agrupar. Esta situación se ilustra en el grafo (A) de la Figura 2.2.

La complejidad computacional de este algoritmo es $O(n^2)$ y el conjunto de grupos que se obtiene es independiente del orden de análisis de los vertices en G_β. Sin embargo, el algoritmo Extended Star tiene algunas deficiencias considerables. La primera deficiencia es que puede dejar elementos sin agrupar.

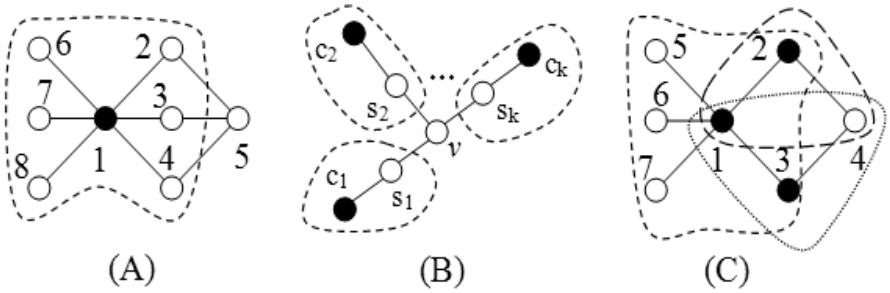

Figura 2.2: Deficiencias del algoritmo Extended Star

El proceso que realiza el algoritmo para el grafo (A) es el siguiente: en la primera

iteración se selecciona al vértice 1 y este se adiciona a X. Posteriormente, en la segunda iteración, se forma el conjunto $M' = \{2, 3, 4\}$ de los vértices con mayor grado complemento (1) y grado (2). Como ninguno de los elementos en M' cumple las condiciones para ser centro éstos se eliminan de L y dejan como candidatos para la proxima iteración a los vértices 5,6,7 y 8. En la siguiente iteración el conjunto M' sólo está formado por el vértice 5 el cual tampoco satisface las condiciones de centro y es eliminado de L. Por último en la cuarta iteración, los vértices restantes en L son eliminados al no cumplir las condiciones exigidas. En este punto la lista L esta vacía y todavía queda el vértice 5 sin agrupar, luego el proceso queda atrapado en un ciclo infinito. Como se mencionó anteriormente, aunque se modifique la condición de parada para evitar el ciclo infinito, el algoritmo todavía dejaría al vértice 5 sin agrupar.

La configuración de vértices del grafo (A) que da lugar a esta situación no es única. En general se cumple que siempre que exista un vértice v, como el del grafo (B) de la Figura 2.2, y éste cumple las condiciones 2.9 y 2.10, el algoritmo dejará a v sin agrupar.

$$\forall s_i, 1 \leq i \leq k, |v.Adj| > |s_i.Adj| \tag{2.9}$$

$$\forall c_i, 1 \leq i \leq k, |c_i.Adj| > |v.Adj| \tag{2.10}$$

En este grafo, cada s_i es un vértice adyacente de v, y cada c_i es el centro de mayor grado adyacente a s_i. En las expresiones 2.9 y 2.10, así como en el resto del documento, $v.Adj$ denotará al conjunto de adyacentes de v.

La segunda deficiencia de este algoritmo es que obtiene grupos redundantes. En esta tesis, se entenderá por grupo redundante a aquel donde todos sus documentos pertenecen al menos a otro grupo. Esta deficiencia tiene lugar debido a que el algoritmo para poder ser independiente del orden permite que más de un vértice que cumpla las condiciones requeridas sea insertado en X. Como se puede apreciar en el grafo (C) de la Figura 2.2 el vértice 2 y el 3 no deberían ser centros a la vez, puesto que sólo se necesita uno para cubrir al vértice 4. Esta deficiencia provoca que aumente la cantidad de grupos producidos.

Una versión diferente de este algoritmo fue presentada por Gil *et al.* en [24, 22] con el objetivo de ser utilizada en un algoritmo paralelo. Esta nueva versión eliminó en el

proceso de agrupamiento la verificación de las condiciones de centro y sólo verifica que el elemento a insertar en X no forme un grupo ya existente. Como resultado se mantiene la independencia del orden, la complejidad computacional y se resuelve la primera deficiencia del Extended Star. No obstante a lo anterior, dicha versión produce grupos redundantes y grupos ilógicos.

Algoritmo Generalized Star

El algoritmo Generalized Star fue propuesto por Pérez y Medina [69], y se apoya en el plantamiento de Aslam [5] de que, el cubrimiento de G_β a través de sub-grafos en forma de estrella permite obtener grupos con una semejanza relativamente alta entre los documentos que lo componen. Utilizando este planteamiento y definiendo un nuevo tipo de sub-grafo en forma de estrella, este algoritmo construye un conjunto de grupos que pueden ser solapados a partir del cubrimiento de G_β con sub-grafos de este tipo.

A diferencia de Star, donde sólo se utiliza el grado de los vértices para definir el sub--grafo, Generalized Star define un grupo de conjuntos para cada uno de los vértices del grafo, y apoyados en estos conjuntos define lo que es un *sub-grafo en forma de estrella generalizada* y posteriormente, una heurística que define cómo será el agrupamiento que se obtenga.

Los conjuntos de *Satélites Débiles* (WeakSats) y *Satélites Potenciales* (PotSats) de un vértice v, se definen por las siguientes expresiones:

$$v.WeakSats = \{s \in v.Adj \mid |v.Adj| \geq |s.Adj|\}$$

$$v.PotSats = \{s \in v.Adj \mid |v.WeakSats| \geq |s.WeakSats|\}$$

El *grado WeakSats y PotSats* de un vértice v, se define como la cardinalidad de los conjuntos de satélites débiles y potenciales de v respectivamente.

Teniendo en cuenta las definiciones anteriores, un *sub-grafo en forma de estrella generalizada* (sub-grafo EG) se define como un sub-grafo de $m + 1$ vértices, en el cual existe un vértice c denominado "centro" y m vértices llamados "satélites" cumpliéndose que: (i)

existe una arista entre cada satélite y el centro, (ii) el centro satisface la expresión 2.11.

$$\forall s \in c.PotSats, |c.PotSats| \geq |s.PotSats| \tag{2.11}$$

El funcionamiento de este algoritmo, se basa en una heurística que asume como criterio para estimar la densidad de un sub-grafo EG, el grado PotSats de su vértice centro. Como primer paso del algoritmo se construye el grafo G_β que representa a la colección. Posteriormente se construye una lista L con todos los vértices del grafo y utilizando ésta, se realiza un proceso iterativo que selecciona en L el vértice v de mayor grado PotSats, se adiciona éste a X, se actualizan sus vértices adyacentes consecuentemente y se elimina de L. El proceso termina cuando la lista L quede vacía. El conjunto de vértices centro X determina un cubrimiento de G_β a través de los sub-grafos en forma de estrella generalizada formados a partir de dichos centros. Al terminar este proceso se ordenan los vértices en X de acuerdo a su grado PotSat y se verifica cada centro con el objetivo de eliminar aquellos *centros redundantes* que pudieron ser seleccionados en el proceso anterior. Para determinar si un centro es redundante se utiliza el concepto de conjunto de *Centros Potenciales* de un vértice [69].

El conjunto de *Centros Potenciales* (PotCenters) de un vértice v, es el conjunto de todos los vértices adyacentes a v que potencialmente pueden formar un sub-grafo EG en el cual esté incluido v. Este conjunto está definido por la siguiente expresión:

$$v.PotCenters = \{c \in v.Adj \mid c.PotSats \neq\}$$

Un centro $c \in X$ se considera *redundante* si cumple las condiciones siguientes:

1. $c.PotCenters \cap X \neq \emptyset$; es decir, c tiene algun vértice centro adyacente.

2. $\forall s \in c.PotSats$, $(s \in X) \vee (s.PotCenters \cap (X \setminus \{c\}) \neq \emptyset)$; es decir, cada satélite potencial de c, es centro o tiene algún centro c' adyacente en X tal que $c' \neq c$.

Este algoritmo tiene una complejidad computacional de $O(n^3)$, no necesita del número de grupos a obtener, no produce grupos ilógicos ni redundantes y no deja elementos sin cubrir, solucionando deficiencias de los algoritmos Star y Extended Star.

Este algoritmo puede construir diferentes agrupamientos para una misma colección puesto que sólo selecciona en cada iteración un único elemento para incluir en X, por lo que es dependiente de los datos. Sin embargo, al permitir que los centros de los sub-grafos sean adyacentes, el algoritmo GStar evita descubrir grupos ilógicos, reduce el número de configuraciones en las que el algoritmo pueda dar soluciones diferentes, además de que garantiza obtener una cantidad de grupos menor que los algoritmos anteriores, reduce la influencia del orden de análisis en la calidad de los grupos y obteniendo agrupamientos con valores de calidad muy cercanos.

No obstante a lo anterior, este algoritmo tiene algunas deficiencias. La primera deficiencia es que el algoritmo elimina grupos densos previamente descubiertos. En esta tesis se entenderá que un grupo g_i es denso, si el vértice centro c_i que lo forma es más denso[3] que todos los otros vértices centro contenidos en el grupo. Esta deficiencia tiene lugar debido al criterio adoptado para decidir cuándo un centro es redundante, el cual no tiene en cuenta la densidad del centro evaluado para tomar dicha decisión. Esta deficiencia puede afectar la calidad del agrupamiento pues, basándose en el criterio de agrupamiento utilizado, el grupo eliminado tiene una semejanza interna relativamente alta. Esta deficiencia se muestra en el grafo (A) de la Figura 2.3.

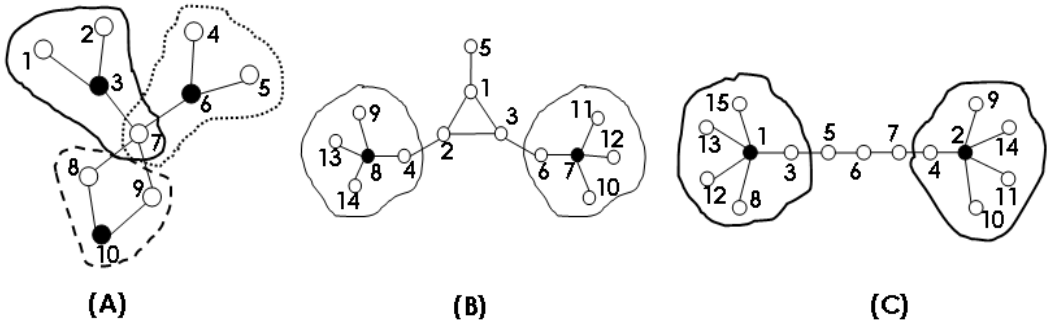

Figura 2.3: Deficiencias del algoritmo Generalized Star.

La segunda deficiencia de este algoritmo está relacionada con cómo se seleccionan el conjunto de centros. Aunque GStar reduce la influencia del orden de los datos en la

[3]El criterio de densidad que utiliza el algoritmo GStar tiene en cuenta solamente el grado PotSats del vértice centro del sub-grafo que forma al grupo

calidad del agrupamiento resultante al permitir que los centros puedan ser adyacentes, el criterio de agrupamiento utilizado podría mejorarse para reducir aun más el número de configuraciones en las que existe más de un conjunto de grupos como solución.

Considérese la ejecución del algoritmo GStar sobre los grafos (B) y (C) de la Figura 2.3. En la ejecución sobre el grafo (B), una vez que se han insertado en X los vértices 7 y 8, el criterio de agrupamiento considera tres candidatos en la iteración actual, los vértices 1,2 y 3. De esta forma al terminar el proceso de construcción de X se pueden obtener varios conjuntos posibles: (1) $X = \{7, 8, 3, 1\}$, (2) $X = \{7, 8, 2, 1\}$ o (3) $X = \{7, 8, 1\}$; afortunadamente el proceso de eliminación de centros redundantes, sin importar cual fue el conjunto X formado, siempre produciría el mismo agrupamiento final ($X = \{7, 8, 1\}$). Por otra parte, en la ejecución sobre el grafo (C), luego de seleccionar los vértices 1 y 2, los posibles candidatos en la iteración actual son los vértices 5,6 y 7; luego, en dependencia de la selección, X podría tener diferentes posibles valores; lamentablemente en este caso el proceso de eliminación de centros redundantes no soluciona este problema.

La última deficiencia de este algoritmo está relacionada con el consumo de memoria. El generalized Star necesita para su funcionamiento el cálculo de varios conjuntos para cada vértice, como por ejemplo los adyacentes, WeakSats, PotSats, PotCenters e incluso NecSats los cuales debe mantener en memoria hasta el final del algoritmo. Debido a lo anterior, este algoritmo pudiera resultar ineficiente con grandes colecciones de documentos.

2.3.11. Algoritmos híbridos

Este tipo de algoritmos como su nombre lo indica utilizan una combinación de algoritmos de distintos tipos para encontrar la estructuración de la colección. Ejemplos de estos algoritmos son el Scatter/Gather [16], el ICT [48] y el UMASS [14].

El Scatter/Gather es un algoritmo que combina las ventajas de un algoritmo jerárquico aglomerativo y uno basado en optimización. Scatter/Gather representa los documentos mediante el modelo de Espacio Vectorial y utiliza la medida del coseno para calcular la semejanza entre los documentos. Como primer paso Scatter/Gather aplica un algoritmo

aglomerativo para encontrar los k centroides y una vez encontrados aplica un algoritmo de optimización para refinar los grupos.

El algoritmo aglomerativo que se aplica es el Average-link (ver sección 2.3.3) y puede aplicarse siguiendo una estrategia *Buckshot* o *Fractionation*. Luego de que se tienen determinados los k centroides se aplica un proceso de refinamiento de la partición en el cual puede utilizarce: (i) una variante del algoritmo k-means, (ii) el algoritmo Split-Merge el cual iterativamente divide los grupos con menor semejanza intracluster y une aquellos grupos muy semejantes, o (iii) una combinación de ambos. Ver [16] para más información.

Los algoritmos ICT y UMASS han sido utilizados para la detección jerárquica de tópicos en colecciones de noticias [82]. Como paso previo para el agrupamiento estos algoritmos utilizan la información temporal de las noticias para ordenarlas cronológicamente. En el algoritmo ICT, se aplica un algoritmo aglomerativo para construir la jerarquía y un algoritmo de optimización o de una sola pasada para refinar cada nivel de ésta. El algoritmo aglomerativo utilizado fue el Single-link utilizando representantes para los grupos y definiendo un valor de umbral de semejanza para cada nivel. Para refinar las particiones de cada nivel se utilizó el Single-pass o el K-means.

El algoritmo UMASS una vez que se ha ordenado las noticias cronológicamente, utiliza el algoritmo 1-NN[74] para agruparlas. Para realizar esto, al procesar cada noticia q se calcula la semejanza de ésta con las k noticias anteriores, seleccionándose aquellas cuya semejanza excede cierto umbral β. Posteriormente la noticia se le asigna a q el grupo al que pertenece su noticia más semejante; si no existe tal grupo, se forma un nuevo grupo unitario que contiene a q. Luego del paso anterior, los grupos obtenidos se dividen en sub--conjuntos y sobre cada uno de ellos se ejecuta un algoritmo aglomerativo para reducir el número de grupos determinados. Como último paso, los grupos resultantes de cada sub-conjunto son agrupados con el algoritmo aglomerativo para formar la jerarquía.

Estos algoritmos son dependientes del orden. En Scatter/Gather además la calidad del agrupamiento puede depender de la muestra seleccionada si se utiliza una estrategia *Buckshot*. Los grupos formados por el ICT pueden presentar encadenamientos al utilizarse el Single-link. Por último cabe mencionar que los tres algoritmos necesitan prefijar valores

de parámetros para ejecutarse.

2.4. Medidas de evaluación de la calidad

Existen variadas medidas para estimar la calidad de los agrupamientos de documentos obtenidos por un algoritmo. Estas medidas se clasifican en medidas *internas* y *externas*, atendiendo a si utilizan o no alguna información externa al agrupamiento. A continuación se describen las medidas que se utilizan en este documento para evaluar la calidad de los agrupamientos obtenidos.

2.4.1. Medidas internas de calidad

Las medidas internas permiten evaluar la calidad del agrupamiento cuando no se tiene información de cual es el grupo al que realmente pertenece cada documento. Estas medidas calculan un valor de calidad en función de la cohesión interna de los grupos, la cual tiene en cuenta que tanto se parecen los documentos en un mismo grupo y que tan diferentes son éstos de los documentos de otro grupo. En este documento se utiliza como medida interna la *similitud global* [81].

Para obtener esta medida es necesario calcular primero la similitud interna de cada grupo. Sea $C = \{C_1, C_2, \ldots, C_k\}$ un conjunto de grupos de documentos, la similitud interna de cada grupo C_i se calcula utilizando la expresión siguiente:

$$Similitud_{int}(C_i) = \frac{1}{|C_i|^2} \cdot \sum_{d,d' \in C_i} F(d, d')$$

donde $|C_i|$ es la cantidad de documentos presentes en el grupo C_i y $F(d, d')$ es una función que determina la semejanza entre los documentos d y d'. Una de las funciones que se puede utilizar para el cálculo de la semejanza entre documentos es la medida del coseno (ver sección 2.2).

Teniendo en cuenta la definición anterior, la similitud global del agrupamiento se calcula utilizando la siguiente expresión:

$$similitud_{global}(C) = \frac{\sum_{i=1}^{k} semejanza_{int}(C_i)}{k} \tag{2.12}$$

el valor de esta medida está en el intervalo [0;1] y entre mayor sea el valor de la misma mejor será la calidad del agrupamiento obtenido.

Es importante resaltar que en el cálculo de la similitud global, según la expresión 2.12, no se tendrá en cuenta a aquellos grupos formados por un único elemento ya que la cohesión interna de estos grupos es 1 y por consecuente un agrupamiento en el cual se formara un grupo unitario por cada documento de la colección tuviera el valor máximo de esta medida. Esta forma de proceder evita favorecer a algoritmos de agrupamiento que producen muchos grupos unitarios.

Otros ejemplos de medidas internas utilizadas son el *índice de Davies-Boulding* y el *índice de Dunn* [80] y la medida Λ [34].

2.4.2. Medidas externas de calidad

Este tipo de medida, determina la calidad del agrupamiento comparando los grupos obtenidos con un conjunto de clases previamente definidas y que generalemnte son determinadas manualmente por expertos. Ejemplos de estas medidas son *Fmeasure* [6, 58] y *Jaccard-index* [41].

La medida *Fmeasure* es una medida que combina las clásicas medidas de *precision* y *recall* para dar un indicador global de calidad del agrupamiento. Las expresiones para calcular esta medida se muestran a continuación:

$$Fmeasure = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{2.13}$$

donde precision y recall se definen como:

$$precision = \frac{n_{11}}{n_{11} + n_{10}} \qquad recall = \frac{n_{11}}{n_{11} + n_{01}}$$

donde n_{10} representa la cantidad de pares de documentos que están en el mismo grupo y en diferentes clases, mientras que n_{01} es la cantidad de pares que están en diferentes grupos y en la misma clase. Esta medida permite, para cada par de documentos que comparten al menos un grupo en el agrupamiento, estimar cuándo esta predicción de asignarlos a un mismo grupo es correcta respecto a las clases predefinidas.

Es importante resaltar que esta definición de la medida *Fmeasure* fue propuesta en [6] específicamente para evaluar algoritmos que producen cubrimientos y la misma es una variante de la clásica medida $F1$ propuesta por van Rijsbergen en [85] que ha sido utilizada con frecuencia para evaluar la calidad de algoritmos que producen particiones.

El Jaccard index se denota por J y es una medida que establece una forma simple de comparar el agrupamiento resultante con el conjunto de clases predefinidas a través de la co-ocurrencia de los documentos en estos conjuntos. Esta medida se calcula utilizando la siguiente expresión:

$$J(A, B) = \frac{n_{11}}{\frac{N \cdot (N-1)}{2} - n_{00}} \tag{2.14}$$

donde N es la cantidad de documentos de la colección, A representa al conjunto de grupos obtenidos, B al conjunto de clases definidas, n_{11} denota al número de pares de documentos que coinciden en un mismo grupo en A y en una misma clase en B, mientras n_{00} denota a la cantidad de pares que se encuentran en diferentes grupos en A y diferentes clases en B. Los valores de esta medida están en el intervalo $[0, 1]$ y mientras mayor sea el valor de la misma, mejor es la calidad del agrupamiento.

En el caso de la medida Jaccard index, aunque ha sido utilizada para evaluar particiones, la forma en que se define sigue el mismo principio de la medida *Fmeasure* al utilizar para estimar la calidad del agrupamiento la cantidad de pares que comparten al menos un grupo en el conjunto de clases predefinidas y que en el agrupamiento resultante tambien comparten al menos un grupo. Esta forma de calcular la eficacia hace al Jaccard index independiente de la forma de los grupos y por lo tanto aplicable al problema del cubrimiento.

2.5. Recapitulación

Los algoritmos de agrupamiento constituyen una herramienta útil en diferentes contextos. En el caso específico del procesamiento de documentos se han desarrollado diversos algoritmos bajo diferentes criterios. De forma general, estos algoritmos presentan algunos problemas como se pudo apreciar en la sección 2.3.

Los problemas o limitaciones de estos algoritmos se pueden resumir en los siguientes aspectos:

- Los grupos formados presentan encadenamientos. Esta limitación afecta la cohesión interna de los grupos, ya que en un mismo grupo pueden ser asignados documentos que tengan muy baja semejanza, pero que se encuentren asociados transitivamente a través de otros documentos muy semejantes. Ejemplos de algoritmos que presentan este problema son: Single-link, STC, GLC, Compacto Incremental y ICT.

- Necesitan determinar a priori el número de grupos a obtener. En la mayoría de los casos, se desconoce la cantidad de clases en las que se distribuyen los documentos de la colección; luego, fijar un valor para la cantidad de grupos puede disminuir la cohesión interna de los grupos al asignar documentos poco semejantes en un mismo grupo. Ejemplos de algoritmos que presentan esta limitación son: K-means, NMF y Scatter/Gather.

- Requieren determinar a priori valores para varios parámetros[4] (diferentes del número de grupos). De forma similar al punto anterior, estos parámetros son desconocidos a priori para cada colección de documentos que se desee agrupar y la adecuada selección de los mismos determina la calidad del agrupamiento obtenido. Ejemplos de algoritmos con esta limitación son: STC, DC-tree, FTC, FIHC, Scatter/Gather, ICT, UMASS y MVGD.

- Los grupos determinados dependen de la selección inicial de objetos. Ejemplos de algoritmos con esta limitación son: K-means, Bisecting K-means, Scatter/Gather y los algoritmos genéticos.

- Obtienen muchos grupos, en algunos casos con pocos elementos, respecto a otros algoritmos para los mismos valores de los parámetros de entrada. Esta limitación dificulta el análisis de los resultados. Ejemplos de algoritmos con este problema son: Compacto Incremental, Fuertemente Compacto Incremental, Star y Extended Star.

[4]En este punto se asumió que el mínimo de parámetros que puede necesitar un algoritmo de agrupamiento para trabajar es 1

- Pueden dejar documentos sin agrupar, es decir elementos que no pertenecen a ningún grupo de la solución. Ejemplos de algoritmos con este problema son: DC-tree, FIHC, FTC y Extended Star.

Además de las limitaciones anteriores, existen algoritmos cuyo proceso de agrupamiento resulta muy costoso y que por lo tanto no pueden aplicarse o resultarían ineficientes con colecciones muy grandes o de gran dimensionalidad, por lo que sólo se han aplicado a resúmenes, snippets o simplemente a documentos en los cuales se realiza un proceso de selección de términos. Ejemplos de algoritmos con este problema son: MVGD, DC-tree, STC.

Otro problema, presente en la mayoría de los algoritmos descritos, es que pueden obtener diferentes agrupamientos cuando se ejecutan sobre una misma colección, producto de la dependencia del orden de análisis de los documentos, el proceso de asignación de elementos a los grupos o simplemente del criterio de agrupamiento utilizado. Existen algoritmos como GLC, Compacto Incremental, Fuertemente Compacto Incremental y Extended que solucionan esta dependencia; sin embargo, presentan otros problemas como encadenamiento, dejan elementos sin agrupar, etc.

No obstante a lo anterior, existen trabajos que plantean que la dependencia del orden de análisis de los documentos es una característica de interés más que una deficiencia, al permitir explorar las diferentes formas en que se pueden agrupar los elementos sin aumentar la cardinalidad del agrupamiento [5]. Otros autores, plantean que el problema real acerca de la dependencia es si ésta afecta o no la calidad del agrupamiento, es decir, el problema no es que se obtengan diferentes agrupamientos sino cuánto varía la calidad de éstos [69].

Como se mencionó en el capítulo 1, es común que un documento aborde varios temas o tópicos, lo cual debería permitir que se asigne dicho documento a varios grupos; sin embargo, la mayoría de los algoritmos sólo permiten la asignación de un documento a un único grupo. Algunos de los algoritmos que permiten obtener grupos solapados o con traslape son: Star, Extended Star, Fuertemente Compacto Incremental, Generalized Star y STC los cuales presentan, como se explicó en este capítulo, varias limitaciones de acuerdo

a diferentes aspectos. Es por ello, que es necesario continuar desarrollando algoritmos que generen cubrimientos y reduzcan las limitaciones señaladas.

Capítulo 3

Agrupamiento basado en sub-grafos condensados

En el capítulo anterior se realizó una descripción de los principales trabajos relacionados con el agrupamiento de documentos, señalando para cada uno sus bondades y sus limitaciones.

Considerando las ventajas de los algoritmos basados en grafos (sección 1.2), específicamente aquellos que realizan cubrimientos de G_β a través de sub-grafos densos (Star, Extended Star y Generalized Star), se presenta en este capítulo un algoritmo denominado CStar (Estrellas Condensadas).

El algoritmo CStar define un nuevo concepto de sub-grafo llamado *sub-grafo en forma de estrella condensada* y aplica una heurística, a partir de la cual obtiene un cubrimiento de G_β utilizando sub-grafos de este tipo. Este algoritmo obtiene un conjunto de grupos que pueden tener traslape y además, mantiene las bondades de sus predecesores a la vez que reduce algunas de las limitaciones presentes en éstos.

Además del algoritmo CStar, en este capítulo se propone una variante de éste llamada CStar+. Esta variante utiliza la heurística de CStar para realizar un cubrimiento en las componentes conexas de G_β e introduce la detección de algunos tipos de componentes conexas con el objetivo de aumentar la calidad del agrupamiento y reducir el tiempo de ejecución del algoritmo.

Este capítulo está organizado como sigue: el algoritmo CStar y su variante CStar+ se presentan en las secciones 3.1 y 3.2. Los resultados de la evaluación del algoritmo CStar y

CStar+ en colecciones estándar de documentos se muestran en la sección 3.3. Por último, en la sección 3.4 se hace una recapitulación.

3.1. Algoritmo CStar

Antes de describir el algoritmo CStar, se necesita introducir algunos conceptos básicos que se utilizan para definir el nuevo concepto de sub-grafo en forma de estrella y el nuevo criterio de agrupamiento.

3.1.1. Conceptos básicos

Sea $G_\beta = \langle V, E_\beta \rangle$ un grafo de β-semejanza, llamaremos *0-Transición* de G_β al grafo dirigido $G_\beta^{(0)} = \langle V, E_\beta^{(0)} \rangle$ que se obtiene adicionando por cada arista no-dirigida $(u,v) \in E_\beta$ las aristas dirigidas (u,v) y (v,u) en $E_\beta^{(0)}$. En la Figura 3.1 se muestra un ejemplo de la *0-Transición* de un grafo de β-semejanza donde cada vértice está etiquetado con el valor de su grado saliente.

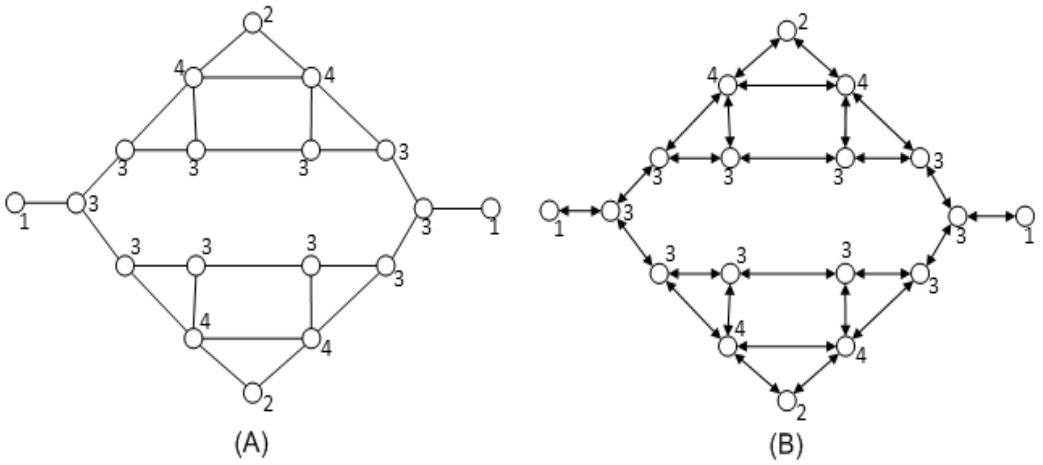

Figura 3.1: (A) $G_\beta = \langle V, E_\beta \rangle$, $|E_\beta| = 26$, (B) $G_\beta^{(0)} = \langle V, E_\beta^{(0)} \rangle$, $\left|E_\beta^{(0)}\right| = 52$

Sea Γ una función definida como:

$$\Gamma : G = \langle V, E \rangle \longrightarrow \tilde{G} = \langle V, \tilde{E} \rangle$$

donde G y $\tilde{G}$ son grafos dirigidos. El grafo $\tilde{G}$ se construye a partir de G eliminando de E

las aristas dirigidas (u,v) que no satisfacen:

$$u.out \geq v.out \tag{3.1}$$

para formar $\tilde{E}$, donde $v.out$ y $u.out$ son los grados salientes de v y u respectivamente. Al grafo $\tilde{G}$ se le denomina Γ-*Transición* de G.

Teniendo en cuenta los conceptos previos, se construye una secuencia de grafos $\{G_\beta^{(0)}, G_\beta^{(1)}, G_\beta^{(2)}, \ldots, G_\beta^{(n)}, \ldots\}$, de forma tal que para todo $i > 0$, $G_\beta^{(i)} = \langle V, E_\beta^{(i)} \rangle$ es la Γ-*Transición* de $G_\beta^{(i-1)}$. Esta secuencia se llama *secuencia de* Γ-*Transiciones de* G_β y cumple que la sucesión de números enteros no-negativos $\{e_n\}_{n=0}^{\infty}$, donde $e_n = \left|E_\beta^{(n)}\right|$, es monótona decreciente acotada inferiormente por 0.

En la figura 3.2, se muestran los grafos $G_\beta^{(1)}, G_\beta^{(2)}, G_\beta^{(3)}, G_\beta^{(4)}$ de la secuencia de Γ-Transiciones del grafo G_β de la figura 3.1 (A).

Como todos los grafos $G_\beta^{(i)}$ de la secuencia de Γ-Transiciones de G_β comparten el mismo conjunto de vértices, se referenciará al grado saliente del vértice v en el grafo $G_\beta^{(i)}$ como $v.out[i]$.

A continuación se presenta un conjunto de teoremas y corolarios que permiten asegurar que el proceso de construcción de la secuencia de Γ-Transiciones de G_β es finito y no genera vértices aislados.

Teorema 3.1 *Dada la secuencia de* Γ-*Transiciones* $\{G_\beta^{(0)}, G_\beta^{(1)}, G_\beta^{(2)}, \ldots, G_\beta^{(n)}, \ldots\}$ *de* G_β*, se cumple que* $\bar{G_\beta^{(n)}} = G_\beta$*,* $\forall n \geq 0$

Demostración.

Si $n = 0$ queda demostrado por definición de $\bar{G_\beta^{(0)}}$

Si $n > 0$ se cumple $G_\beta^{(n)} = \Gamma(G_\beta^{(n-1)})$ por definición de *secuencia de* Γ-*Transiciones de* G_β. En la construcción de $G_\beta^{(n)}$, se verifica cada arista $(u,v) \in E_\beta^{(n-1)}$ y se eliminan de $E_\beta^{(n)}$ aquellas que no satisfacen la desigualdad 3.1, la cual siempre se cumple en uno de los dos sentidos. Es válido resaltar que si existe al menos una arista entre los vértices u y v en $G_\beta^{(n)}$ se garantiza que esta arista esté en $\bar{G_\beta^{(n)}}$. $\square$

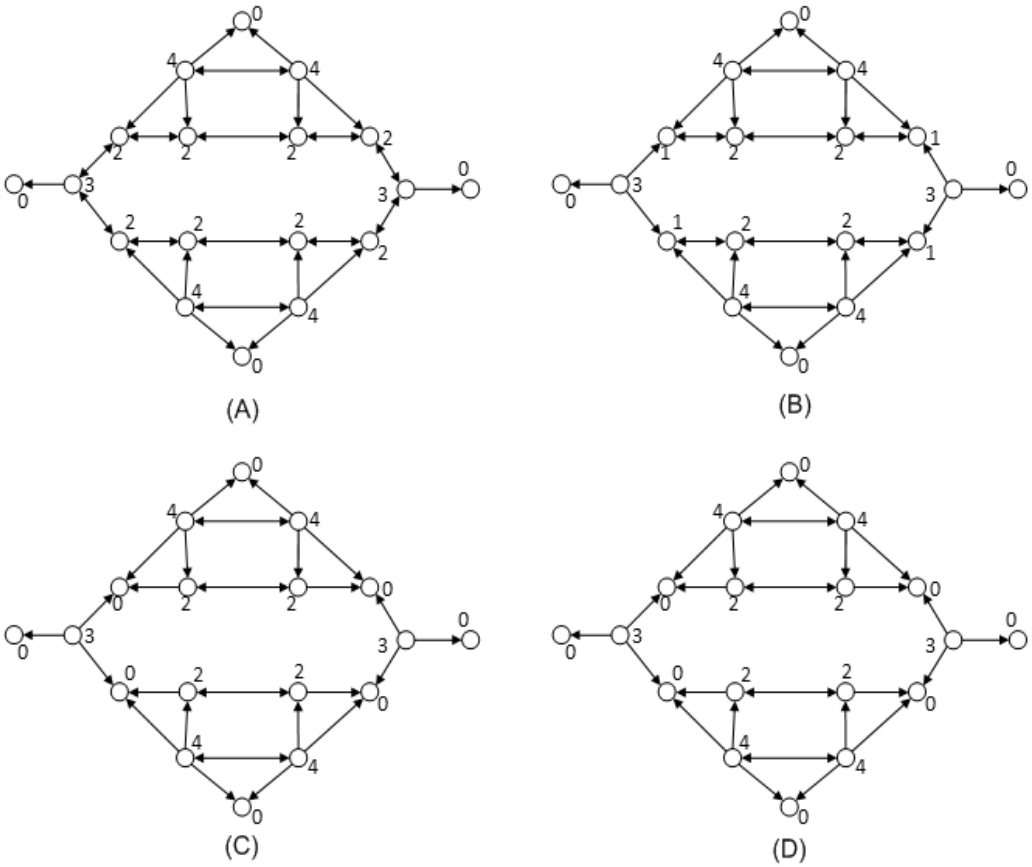

Figura 3.2: (A) $G_\beta^{(1)}$, $\left|E_\beta^{(1)}\right| = 38$, (B) $G_\beta^{(2)}$, $\left|E_\beta^{(2)}\right| = 34$, (C) $G_\beta^{(3)}$, $\left|E_\beta^{(3)}\right| = 30$, (D) $G_\beta^{(4)}$, $\left|E_\beta^{(4)}\right| = 30$

Corolario 3.1 $\forall n \geq 0, \left|E_\beta^{(n)}\right| \geq |E_\beta|$.

Teorema 3.2 *Existe $h \geq 0$ tal que se cumple $G_\beta^{(h+1)} = G_\beta^{(h)}$*

Demostración.

A partir del corolario 3.1 se tiene que la sucesión $\{e_n\}_{n=0}^{\infty}$, donde $e_n = \left|E_\beta^{(n)}\right|$ está acotada inferiormente por $|E_\beta|$ y se cumple además que $\left|E_\beta^{(n)}\right| \geq \left|E_\beta^{(n+1)}\right|$.

Supóngase que no existe $h \geq 0$, tal que $G_\beta^{(h+1)} = G_\beta^{(h)}$. Luego la sucesión $\{e_n\}_{n=0}^{\infty}$ sería estrictamente decreciente y no acotada, contradiciendo el corolario 3.1. $\square$

Corolario 3.2 *Existe $h \geq 0$ tal que si $G_\beta^{(h+1)} = G_\beta^{(h)}$ entonces se cumple $G_\beta^{(h+k)} = G_\beta^{(h)}$ para todo $k \geq 0$*

Ahora se introducen algunos conceptos que permiten definir el concepto de sub-grafo en forma de Estrella Condensada.

Definición 3.1 *Sea h el primer entero tal que $G_\beta^{(h+1)} = G_\beta^{(h)}$, a éste entero h se llamará* cota *del grafo G_β y a $G_\beta^{(h)}$* última transición *del grafo G_β*

Sea $G_\beta = \langle V, E_\beta \rangle$ un grafo de β-semejanza, h la cota de G_β y $u, v \in V$; u es un *r-satélite* de v, con $0 \leq r \leq h$, si se cumple que la arista dirigida $(v,u) \in E_\beta^{(r)}$. El conjunto de todos los r-satélites de v se denota por $v.Sats[r]$ y se define por la siguiente expresión:

$$v.Sats[r] = \{u \in V \mid u \text{ es un } r\text{-satélite de } v\}$$

Sea $G_\beta = \langle V, E_\beta \rangle$ un grafo de β-semejanza, h la cota de G_β, un *sub-grafo en forma de Estrella Condensada* (sub-grafo EC), es un sub-grafo de $m+1$ vértices, en el cual existe un vértice c denominado "centro" y m vértices llamados "satélites", cumpliéndose que: (i) $c.out[h] > 0$ y (ii) existe una arista entre cada satélite y el centro c en G_β. Cada vértice aislado $u \in V$ se considera un sub-grafo EC degenerado de un solo vértice.

3.1.2. Algoritmo CStar

Teniendo en cuenta los conceptos definidos en la sección anterior, la idea principal del algoritmo CStar es definir un criterio que establezca cuando un sub-grafo EC es más denso que otro y a partir de éste, realizar un cubrimiento de G_β utilizando los sub-grafos EC más densos y posteriormente aplicar un proceso de filtrado que permita reducir la cantidad de grupos construidos.

El problema de encontrar un cubrimiento de G_β puede transformarse en el problema de encontrar el conjunto X de vértices, tal que cada cada vértice $c_i \in X$ determine un sub-grafo EC. Un aspecto importante para solucionar este problema es reducir el conjunto de vértices candidatos que puedan ser insertados en X.

Sea $G_\beta^{(h)}$ la última transición de $G_\beta = \langle V, E_\beta \rangle$ y $v \in V$, el conjunto de vértices adyacentes a v que mayor posibilidad tienen de agrupar a éste se denota por $v.Electees$ y está definido por la siguiente expresión:

$$v.Electees = \arg max_x \{x.out[h] \mid x \in v.Adj \cup \{v\}\}$$

Sea $G_\beta^{(h)}$ la última transición de $G_\beta = \langle V, E_\beta \rangle$ y $v \in V$ un vértice no aislado. El *grado de votación* de v se denota por $v.vd$ y se define por la siguiente expresión:

$$v.vd = |\{u \in v.Adj \mid v \in u.Electees\}|$$

El grado de votación de un vértice v permite estimar cuántos de sus h-satélites necesitan de él para pertenecer a algún sub-grafo EC; es decir, la mayor cantidad de vértices no agrupados, por otros vértices con grado de votación superior al de v, que puede agrupar v si se adiciona al conjunto X. El concepto de grado de votación de un vértice guarda relación con el PageRank de las páginas de Google [43], lo que a diferencia de éste, la relevancia de cada nodo en el grafo se estima a través de sus aristas salientes.

La lista de candidatos L que considera el algoritmo CStar está compuesta por todos los vértices $v \in V$ tal que $v.vd > 0$; de esta forma, CStar evita seleccionar como centros a vertices que forman grupos que a priori son redundantes al no agrupar a ningún vértice no cubierto.

A partir de esta lista, CStar selecciona iterativamente un vértice v y lo inserta en X si v forma el sub-grafo EC más denso. Existen varios criterios para determinar la densidad de un sub-grafo, como por ejemplo los utilizados por los algoritmos Star y Generalized Star. El algoritmo CStar considera, para determinar entre dos sub-grafos el más denso, el grado de votación de sus vértices centro. Por lo tanto, seleccionar en cada iteración el vértice centro de mayor grado de votación, permite a CStar formar el grupo con más vértices no cubiertos y reducir el número de vértices necesarios para cubrir a G_β.

El pseudocódigo de CStar se muestra en el Algoritmo 1.

Las funciones $BuildTransition0$ y $FindLastTransition$ utilizan los conceptos introducidos en 3.1.1 para construir $G_\beta^{(0)}$ y encontrar $G_\beta^{(h)}$ respectivamente. La función $CalculateVotingDegree$ calcula el grado de votación para cada vértice de $G_\beta^{(h)}$.

El algoritmo CStar está compuesto por 4 fases o etapas: (1) construcción de la lista de vértices candidatos L; (2) construcción del conjunto inicial de centros X y la lista de vértices *dudosos* U; (3) selección de nuevos vértices centro en U y adición de éstos a X; (4) eliminación del conjunto X de todo vértice que forme un grupo *redundante*. A

Input: $D = \{d_1, d_2, \ldots, d_N\}$- colección de documentos
β-umbral de semejanza
Output: SC-conjunto de grupos

```
// Etapa 1 - Construcción de lista de candidatos
```

1 "Construir $G_\beta = \langle V, E_\beta \rangle$";
2 $G_\beta^{(0)}$:= `BuildTransition0`(G_β);
3 $G_\beta^{(h)}$:= `FindLastTransition`($G_\beta^{(0)}$);
4 `CalculateVotingDegree`(V);
5 $L := \{v \in V \mid v.vd > 0\}$;

```
// Etapa 2 - Construcción de conjunto inicial de centros
```

6 $X := \{v \in V \mid v.Adj = \emptyset\}$;
7 $U := \emptyset$;
8 **while** $L \neq \emptyset$ **do**
9 $\quad v := \arg max_x\{x.vd \mid x \in L\}$; `// Sólo se selecciona un vértice`
10 $\quad$ **if** $v.Adj \cap X = \emptyset$ **then** $X := X \cup \{v\}$
11 $\quad$ **else**
12 $\quad\quad F = \{u \in v.Sats[h] \mid u.Adj \cap X = \emptyset\}$;
13 $\quad\quad$ **if** $F \neq \emptyset$ **then**
14 $\quad\quad\quad$ **if** $\exists f \in F,\ v.vd > f.vd$ **then** $X := X \cup \{v\}$
15 $\quad\quad\quad$ **else** $U := U \cup \{v\}$;
16 $\quad\quad$ **end**
17 $\quad$ **end**
18 $\quad L := L \setminus \{v\}$;
19 **end**

```
// Etapa 3 - Procesamiento de vértices dudosos
```

20 **forall** *vertex* $v \in U$ **do**
21 $\quad$ **if** $\exists u \in v.Sats[h],\ u.Adj \cap X = \emptyset$ **then** $X := X \cup \{v\}$;
22 **end**

```
// Etapa 4 - Eliminación de grupos redundantes
```

23 "Ordenar los vértices $v \in X$ respecto a su grado de votación";
24 $SC := \emptyset$;
25 **forall** *center* $c \in X$ **do**
26 $\quad$ **if** c *is redundant* **then** $X := X \setminus \{c\}$
27 $\quad$ **else** $SC := SC \cup \{\{c\} \cup c.Adj\}$;
28 **end**

Algoritmo 1. Algoritmo CStar

continuación se describe en detalle el funcionamiento del algoritmo.

La primera etapa obtiene un conjunto reducido de vértices con posibilidades reales de formar sub-grafos EC y por tanto ser insertados en X. Como la lista de candidatos L está formada por vértices con grado de votación no nulo, los vértices que no pertenezcan a L son aislados (forman un sub-grafo EC degenerado y por tanto pertenecen a X) o tienen algún vértice adyacente en L. En caso de que se inserten en X todos los vértices de L, se garantiza que no queden vértices sin agrupar.

Al comienzo de la etapa 2, se inicializa el conjunto X con todos aquellos vértices aislados de G_β. Posteriormente, la lista L es procesada iterativamente en orden decreciente respecto al grado de votación; este orden garantiza que el vértice $v \in L$ seleccionado en cada iteración sea el que forma el sub-grafo más denso de los candidatos en esa iteración y por lo tanto se evita descubrir grupos ilógicos. El vértice v es procesado considerando las siguientes situaciones:

1. Si v no está agrupado, entonces se inserta en X. Al priorizar, en el proceso de selección a los vértices no agrupados, se permite realizar un cubrimiento más rápido de G_β y se evita forzar el cubrimiento entre los grupos.

2. Si v está agrupado, entonces la decisión de insertar a v en X depende de los vértices no agrupados de su conjunto de h-satélites. Sea F el conjunto de vértices no agrupados del conjunto v.Sats[h]. Si $F = \emptyset$, v no puede ser seleccionado ya que forma un grupo redundante. En otro caso, como v es necesario todavía para agrupar a algún vértice, se procede considerando las condiciones siguientes:

   a) Si v tiene al menos un vértice $f \in F$ tal que $v.vd > f.vd$, entonces v se inserta en X. Esta condición permite agrupar a f en un sub-grafo más denso que el que f podría formar si se seleccionara en alguna iteración posterior. De esta forma se evita construir grupos ilógicos.

   b) En otro caso, v se clasifica como dudoso, se inserta en la lista U y se postpone la decisión de insertar v en X. Esta condición permite explorar otras posibles selecciones de vértices adyacentes a v, que tengan su mismo valor de densidad

y que puedan, agrupar a éste y a algún adyacente menos denso. Nótese que, aunque este paso puede hacer que el agrupamiento resultante sea diferente en cada ejecución, se garantiza que los sub-grafos formados tengan la misma densidad, pues en otro caso se hubiese seleccionado v por la condición a.

Después de este proceso, el vértice v es eliminado de la lista L garantizando de esta forma que se alcance la condición de parada de la etapa 2. Durante la etapa 3, todos los vértices de U son verificados en el mismo orden en que fueron insertados, garantizando de esta forma siempre procesar al vértice que forma el sub-grafo EC más denso del resto de la lista U. Para cada uno de estos vértices v, se verifica si tienen algún h-satélite no agrupado y en ese caso, se inserta v en X.

Es importante resaltar que una vez concluidas las etapas 2 y 3 los vértices que no fueron seleccionados como centro en dichas etapas cumplen que: (i) ya están agrupados y (ii) todos sus h-satélites ya están agrupados. Luego, queda garantizado que cada vértice de G_β pertenece al menos a un grupo.

Finalmente la etapa 4 ordena los vértices en X ascendentemente de acuerdo a su grado de votación y elimina de X aquellos vértices centro que sean redundantes. Al finalizar, se forma el agrupamiento resultante con los restantes vértices en X.

Un vértice centro $c \in X$ se considera *redundante* si satisface las siguientes condiciones:

1. $\exists d \in c.Adj \cap X, d.vd > c.vd$; es decir, el vértice c tiene adyacente a él al menos un vértice centro con mayor grado de votación.

2. $\forall s \in c.Sats[h], s \in X \vee (s.Adj \cap (X \setminus \{c\})) \neq \emptyset$; es decir, cada h-satélite s de c es centro o tiene algún vértice centro diferente de c adyacente a él.

3.1.3. Análisis de la complejidad computacional

Para facilitar la comprensión del análisis de la complejidad computacional del algoritmo CStar, éste se dividirá por etapas del algoritmo.

La primera etapa del algoritmo se encarga de la construcción de la lista de vértices candidatos a centro por lo cual, en sus primeros dos pasos se construye el grafo G_β y a

partir de éste el grafo $G_\beta^{(0)}$. Los dos pasos anteriores tienen una complejidad de $O(n^2)$, ya que el primero calcula la semejanza entre todo par de vértices y el segundo itera por cada arista de E_β para construir las aristas de $E_\beta^{(0)}$; la cantidad de aristas en E_β es $\frac{n\cdot(n-1)}{2}$ en el peor de los casos. El paso 3, teniendo en cuenta el proceso de construcción de la secuencia de Γ-transiciones de G_β, tiene una complejidad de $O(h \cdot |E|)$, donde h es la cota de G_β.

A continuación se presentan un conjunto de teoremas y corolarios que permiten acotar el valor de h.

Teorema 3.3 *Sea r un entero positivo tal que $0 \leq r \leq h$ y v un vértice de G_β, se cumple que si v no pierde aristas en la transición r entonces no pierde arista en ninguna transición $r' > r$.*

Demostración.

Como se demostró en 3.1.1, el proceso de construcción de la secuencia de Γ-transiciones de G_β cumple que para toda $n > 0$, $\left|E_\beta^{(n)}\right| \geq \left|E_\beta^{(n+1)}\right|$, por lo que para cada vértice $v \in V$ se cumple que éste reduce o mantiene su grado saliente. Luego, si v no perdió aristas en la transición r, se cumple que:

$$\forall u \in v.Adj, v.out[r] \geq u.out[r] \tag{3.2}$$

Supóngase que existe una transición $r' > r$ tal que v pierde al menos una arista. Luego, se cumple que existe un vértice $u \in v.Adj$ tal que $u.out[r'] > v.out[r']$, lo cual contradice que $\left|E_\beta^{(r'-1)}\right| \geq \left|E_\beta^{(r')}\right|$ ya que según (3.2), u tenía en la transición r un grado saliente menor o igual que el de v. □

Corolario 3.3 *Existe al menos un vértice que perdió aristas en todas las transiciones anteriores a h.*

Corolario 3.4 *Sea v el vértice de mayor grado saliente en $G_\beta^{(0)}$, se cumple que el valor de h es menor o igual que $v.out[0]$.*

Corolario 3.5 $h \leq (|V| - 1)$.

Teorema 3.4 *Sea v el vértice que tiene, en $G_\beta^{(0)}$, mayor cantidad de adyacentes con mayor o igual valor de grado saliente que él. Si Z representa a dicha cantidad de adyacentes entonces se cumple $h \leq Z$.*

Demostración.

Supóngase que la cantidad de transiciones es $Z' > Z$.

Por el corolario 3.3 se sabe que entonces existe al menos un vértice u que perdió aristas en todas las transiciones. Luego, se cumple que u es el vértice que tiene mayor cantidad de adyacentes con mayor o igual valor de grado saliente que él, contradiciendo la condición inicial. □

Es válido resaltar que, aunque teóricamente el valor Z que se presenta como cota de h en el teorema 3.4 puede ser $n = |V|$, en ese caso el vértice v no perdería aristas al ser el vértice de mayor grado saliente. Además, en los teoremas presentados se está asumiendo el peor de los casos, en el cual el vértice que perdió aristas en todas las transiciones, sólo pierde una sola arista en cada transición. Por lo general un vértice puede perder más de una arista en una transición, lo que llevaría a CStar realizar menos iteraciones. En experimentos realizados el número de iteraciones no pasó de 22.

La complejidad computacional de la primera etapa está dominada según la regla del máximo por el cálculo de $G_\beta^{(h)}$, ya que el cálculo del grado de votación de todos los vértices tiene una complejidad de $O(n^2)$. Luego, la complejidad computacional de la primera etapa es $O(Z \cdot n^2)$.

En la segunda etapa, cada vértice $v \in L$ es procesado y en dicho proceso se debe recorrer $v.Adj$ según la condición de la línea 10 y en caso de que ésta no se cumpla, se debe recorrer su lista de h-satélites. Luego, como cada una de estas listas en el peor caso tiene longitud n y la longitud de la lista L está acotada también por n, esta etapa tiene una complejidad computacional de $O(n^2)$. La etapa 3, procesa cada vértice dudoso $u \in U$ determinando si éste debe ser insertado en X; como en este proceso se recorre la lista de h-satélites de u y conociendo que en el peor caso $|U| = (|V| - 1)$, la complejidad computacional de la tercera etapa es $O(n^2)$.

Por último, en la etapa 4 para eliminar los centros redundantes se comienza ordenando los vértices de X respecto a su grado de votación. Esta ordenación, teniendo en cuenta que el grado de votación de un vértice es a lo sumo $|V| - 1$, puede realizarce en $O(n)$ utilizando el principio *pigeonhole*. Luego, como la cantidad máxima de centros está acotada por n y para determinar si un centro c es redundante se verifican los vértices en $c.Adj$, la complejidad computacional de esta cuarta etapa es $O(n^2)$.

Teniendo en cuenta la complejidad computacional de cada una de las etapas y según la regla del máximo, la complejidad computacional del algoritmo CStar es $O(Z \cdot n^2)$, donde Z es el valor máximo que puede alcanzar la cota de G_β.

3.2. Algoritmo CStar+

Como se comentó en la sección 2.3.10, el algoritmo GLC construye un agrupamiento a partir de G_β construyendo componentes conexas del grafo. La mayor limitación de este algoritmo es la baja cohesión de los grupos formados producto del encadenamiento entre los documentos en las componentes conexas. Este encadenamiento puede romperse si se realiza un agrupamiento a través de sub-grafos EC (Estrella Condensada) sobre el conjunto de grupos resultantes del algoritmo GLC. En esta sección se presenta el algoritmo CStar+, el cual es una variante del CStar y realiza un cubrimiento sobre las componentes conexas de G_β utilizando sub-grafos EC.

Si se considera al conjunto $CP = \{CP_1, CP_2, \ldots, CP_k\}$ de componentes conexas de G_β, el problema de determinar un agrupamiento de G_β a través de sub-grafos EC puede transformarse en el problema de realizar un cubrimiento utilizando sub-grafos EC de cada componente conexa CP_i, $i = 1...k$ y este último, en el problema de encontrar el conjunto X_i de vértices centro que forman dichos sub-grafos.

Es importante tener en cuenta que aunque obtener un cubrimiento de estas componentes a través de sub-grafos EC reduce el encadenamiento, también podría afectar la calidad del agrupamiento si dicha componente tiene un alto grado de conexión entre sus vértices pues se estaría dividiendo en sub-grupos un grupo que ya es altamente cohesio-

nado. Un ejemplo de grafo conexo altamente conectado es el *cuasi-clique*.

Sea $G = \langle V, E \rangle$ un grafo conexo, G se considera un *cuasi-clique* si se cumple que:

$$\forall v \in V, |v.Adj| \geq \alpha \cdot (n-1)$$

donde $n = |V|$ y α es un número real $0 < \alpha \leq 1$. Este tipo de grafo puede considerarse como un clique en el cual cada vértice ha perdido un número de aristas. Existen trabajos que han utilizado este tipo de grafo para el análisis de relaciones entre entidades [53] y en el análisis de genes y proteínas [51].

Es importante mencionar que α sólo representa, de forma similar al umbral β, un valor de confianza definido por el usuario para la aceptación o no de una componente conexa como grupo cohesionado. El valor de α está muy relacionado con el valor de β ya que al ser menor el valor de éste último existen más enlaces entre los vértices de G_β y aumente la probabilidad de que las componentes conexas de G_β sean cuasi-cliques; por lo tanto el valor de α debe ser mayor para detectar grupos que realmente son altamente conectados.

No obstante a lo anterior según Pei [59], para lograr detectar sub-grafos realmente cohesionados (de menor diámetro) los valores de α deben encontrarse en los intervalos $(\frac{n-2}{n-1};1]$ y $[0.50;\frac{n-2}{n-1}]$, pues es en éstos para los que se obtienen sub-grafos con diámetros menor que 2 y por lo tanto altamente cohesionados. En los intervalos n representa la cantidad de vértices del sub-grafo.

Como se explicó en la sección anterior, uno de los pasos del algoritmo CStar es la construcción de una lista de candidatos L a partir de $G_\beta^{(h)}$. Considerando que el grafo G_β se divide en componentes conexas $CP_1, CP_2, \ldots, CP_k$ sobre las cuales se realiza un cubrimiento a través de sub-grafos EC, puede ocurrir que existan componentes en las cuales no sea necesario construir la secuencia de transiciones; estas componentes conexas son los *k-plex*.

Sea $G = \langle V, E \rangle$ un grafo conexo, G se considera un *k-plex* si se cumple que:

$$\forall v \in V, v.Adj = (n-k)$$

donde $n = |V|$. Detectar este tipo de componente permite evitar la construcción del grafo $\tilde{G}_\beta^{(0)}$ y la búsqueda de $\tilde{G}_\beta^{(h)}$, lo cual ayuda a reducir el procesamiento del algoritmo.

La idea del algoritmo CStar+ consiste en dada una colección de documentos, construir su grafo de semejanza G_β y el conjunto de componentes conexas CP que forman a dicho grafo. Posteriormente se procesa cada componente conexa CP_i y se procede según las condiciones siguientes:

1. Si CP_i es un cuasi-clique entonces es un grupo del agrupamiento resultante.

2. Si CP_i es un k-plex entonces no es necesario realizar las transiciones sobre CP_i y se realiza un cubrimiento de ésta utilizando el algoritmo CStar considerando la lista L compuesta por todos los vértices de la componente.

3. En otro caso utilizar al algoritmo CStar para obtener el cubrimiento de CP_i.

Al terminar el proceso anterior, el agrupamiento resultante se forma por la unión de los grupos detectados en cada componente conexa. El pseudocódigo del algoritmo CStar+ se muestra en el Algoritmo 2.

La complejidad computacional de este algoritmo es $O(h \cdot n^2)$, puesto que el paso de calcular las componentes conexas es $O(n^2)$ y los otros pasos se reducen a aplicar el algoritmo CStar en el grafo que determina cada componente conexa. Es importante resaltar que, al construir G_β y sus componentes conexas, para cada componente conexa $\tilde{G}_\beta = \left\langle V', \tilde{E}_\beta \right\rangle$ se almacena el menor y mayor grado de un vértice $v \in V'$. Luego, en el proceso de agrupamiento se utilizan estos valores para determinar en $O(1)$, el tipo de componente de $\tilde{G}_\beta$.

Es importante resaltar que, aunque el valor del parámetro (α) sólo representa un umbral para lo que el usuario considera o no un grupo cohesionado, el algoritmo CStar+ es sensible al valor asignado a α, pues valores pequeños podrían formar grupos con poca cohesión interna ya que se asumirían como grupos sub-grafos poco conectados.

3.3. Evaluación experimental

En esta sección se presenta una serie de experimentos en los que se evalúa el comportamiento de los algoritmos CStar y CStar+ respecto a otros algoritmos de agrupamiento.

Input: $D = \{d_1, d_2, \ldots, d_N\}$- colección de documentos
β-umbral de semejanza
α-umbral de conexión
Output: SC-conjunto de grupos

1 "Construir G_β y el conjunto $CP = \{CP_1, CP_2 \ldots, CP_N\}$";
2 $SC := \emptyset$;
3 **foreach** *componente* $\tilde{G}_\beta = \left\langle V', \tilde{E}_\beta \right\rangle \in CP$ **do**
4 **if** $\tilde{G}_\beta$ *is a cuasi-clique* **then** $SC := SC \cup V'$;
5 **else**
6 **if** $\tilde{G}_\beta$ *is a k-plex* **then** $L := V'$;
7 **else**
8 $\tilde{G}_\beta^{(0)} :=$ `BuildTransition0(`$\tilde{G}_\beta$`)`;
9 $\tilde{G}_\beta^{(h)} :=$ `FindLastTransition(`$\tilde{G}_\beta^{(0)}$`)`;
10 `CalculateVotingDegree(`V'`)`;
11 $L := \{v \in V' \mid v.vd > 0\}$;
12 **end**
13 $X := \emptyset$;
14 $U := \emptyset$;
15 **while** $L \neq \emptyset$ **do**
16 $v := \arg max_x\{x.vd \mid x \in L\}$; `// Sólo se selecciona un vértice`
17 **if** $v.Adj \cap X = \emptyset$ **then** $X := X \cup \{v\}$
18 **else**
19 $F = \{u \in v.Sats[h] \mid u.Adj \cap X = \emptyset\}$;
20 **if** $F \neq \emptyset$ **then**
21 **if** $\exists f \in F$, $v.vd > f.vd$ **then** $X := X \cup \{v\}$
22 **else** $U := U \cup \{v\}$;
23 **end**
24 **end**
25 $L := L \setminus \{v\}$;
26 **end**
27 **forall** *vertex* $v \in U$ **do**
28 **if** $\exists u \in v.Sats[h]$, $u.Adj \cap X = \emptyset$ **then** $X := X \cup \{v\}$;
29 **end**
30 "Ordenar los vértices $v \in X$ respecto a su grado de votación";
31 $SC := \emptyset$;
32 **forall** *center* $c \in X$ **do**
33 **if** c *is redundant* **then** $X := X \setminus \{c\}$
34 **else** $SC := SC \cup \{\{c\} \cup c.Adj\}$;
35 **end**
36 **end**
37 **end**

Algoritmo 2. Algoritmo CStar+

Los algoritmos seleccionados para realizar la evaluación son: Star, Extended Star, Generalized Star, Fuertemente Compacto Incremental, una versión del algoritmo FTC que produce solapamientos y los algoritmos clásicos Single-link y Average-link. Todos los algoritmos fueron implementados en C++ ya que los códigos originales no se encontraron disponibles.

3.3.1. Descripción de las colecciones

Para evaluar la calidad del agrupamiento obtenido por los algoritmos, se seleccionaron dos colecciones de documentos ampliamente utilizadas en evaluaciones de algoritmos de agrupamiento de documentos: TREC-5 [83] y *Reuters*-21578 [46]. Estas colecciones son variadas en cuanto al tamaño de los documentos, número de clases, tamaño de las clases, dimensionalidad, etc.

La colección TREC-5 contiene 695 artículos periodísticos en castellano publicados por la agencia AFP en el año 1994 y que han sido clasificados en 25 tópicos o clases. La colección *Reuters*-21578 contiene 21578 artículos periodísticos en inglés de la agencia Reuters publicados en el año 1987 y que están organizados en 120 clases. De esta última colección se seleccionaron 10377 documentos que están clasificados en uno o más topicos y que tienen al menos un término.

En lo siguiente se referirá a las colecciones TREC-5 y *Reuters*-21578 como AFP y *Reuters* respectivamente. Algunas de las características de estas colecciones se presentan en la Tabla 3.1.

Colección	**N. Doc.**	**Solapamiento**	**N. Clases**	**N. térm.**	**Prom. térm**	**Idioma**
AFP	695	16	25	11839	126	Español
Reuters	10377	1722	119	21218	43	Inglés

Tabla 3.1: Características de las colecciones de documentos AFP y *Reuters*

El término *Solapamiento* en esta tabla hace referencia a la cantidad de documentos que están asignados manualmente a más de una clase; “N. térm.” y “Prom. térm.” son el número de términos presentes en la colección luego de lematizar sus documentos y el promedio de términos por documentos respectivamente.

3.3.2. Representación de los documentos

Los documentos de ambas colecciones fueron representados utilizando el Modelo de Espacio Vectorial. El peso de cada término en el vector documento se determinó a partir del valor de la frecuencia del término normalizada por la frecuencia máxima de los términos en el documento (sección 2.1.2).

Los términos en los documentos representan la forma reducida (lema) de las palabras que aparecen en el documento, por ejemplo, todas las conjugaciones de un verbo se representaron por su infinitivo, el plural de la palabra por su forma singular, etc. En este conjunto de términos no se consideraron las palabras vacías presentes en el documento. Algunas de las palabras vacías no consideradas como términos son: artículos, preposiciones, adverbios, conjuciones, etc.

Para obtener los lemas de los términos de ambas colecciones se utilizó el etiquetador léxico-morfológico *TreeTagger*[1], desarrollado en el Instituto de Lingüística Computacional de la Universidad de Stuttgart. La función fundamental del TreeTagger es analizar y desambiguar las categorías léxicas de las palabras. Para ello se basa en los modelos de Markov de orden k y en los árboles de decisión con objeto de obtener estimaciones más fiables.

Para determinar la semejanza entre los documentos se utilizó la medida del coseno (sección 2.2).

3.3.3. Experimentos

Para comparar la calidad de los grupos obtenidos por los algoritmos se utilizó la medida interna y las dos medidas externas que fueron definidas en la sección 2.4. La primera permitió estimar la cohesión interna de los grupos determinados mientras que ambas medidas externas permitieron estimar la calidad de los grupos de acuerdo a las clases determinadas por expertos en cada colección y que son suministradas con éstas. Además de las medidas anteriores se evaluaron otros indicadores como el número de grupos

[1]Una descripción detallada de este etiquetador se puede encontrar en http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/.

generados y el promedio de elementos por grupo.

En los experimentos se referirá a los algoritmos Extended Star, Generalized Star, Fuertemente Compacto Incremental, Single-link y Average-link por los nombres EStar, GStar, FCI, SLINK y ALINK respectivamente.

Para la evaluación del algoritmo CStar+ se selecionó, según la explicación al final de la sección 3.2, un intervalo en el que se pudiera detectar sub-grafos cohesionados; este intervalo es [0.50;0.85]. Luego, para cada experimento se ejecutó CStar+ con valores de α desde 0.50 a 0.85 con incremento de 0.05. Los valores mostrados para el algoritmo CStar+, en cada uno de los experimentos, corresponden al valor de umbral de conexión α=0.75.

Experimento 1

En este experimento se comparó la calidad de los grupos obtenidos por cada algoritmo teniendo en cuenta las medidas externas. En el caso de los algoritmos Star, EStar, GStar, CStar y CStar+, se muestra cómo se comportó la calidad del agrupamiento teniendo en cuenta el umbral de semejanza β; este umbral tomó valores en el intervalo en que los algoritmos obtuvieron los mejores valores de calidad ([0.25;0.45]). En la comparación con los algoritmos FTC, SLINK, ALINK, CStar y CStar+, el resultado mostrado corresponde al valor promedio de calidad obtenido.

Los resultados de las comparaciones utilizando como medida el *Jaccard-index* se observan en la Tabla 3.2 y en la Figura 3.3.

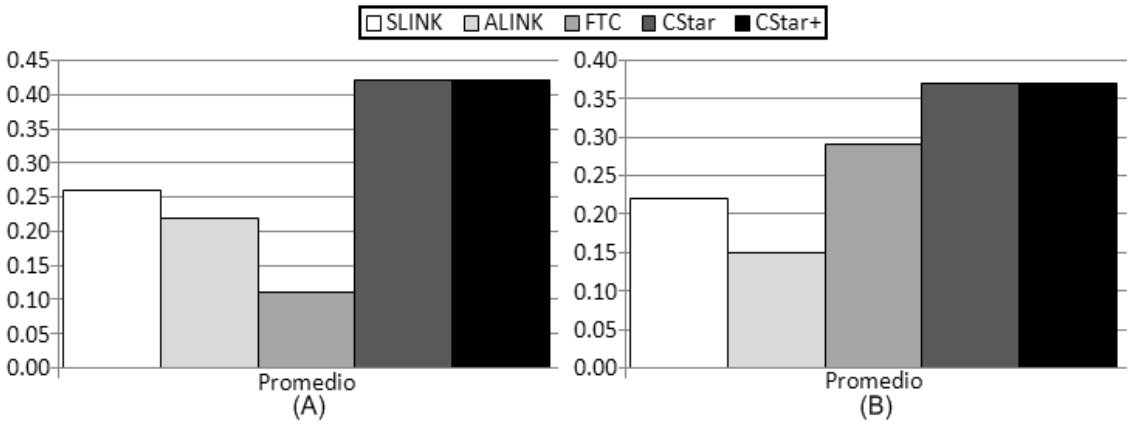

Figura 3.3: Evaluación considerando el Jaccard-index: (A) AFP, (B) *Reuters*

		Valores de β					
Colecc.	**Algoritmo**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**Prom.**
AFP	Star	**0.44**	0.51	0.44	0.33	0.20	0.38
	EStar	0.34	0.52	0.50	0.40	**0.26**	0.40
	GStar	0.35	0.53	0.49	0.40	**0.26**	0.41
	CStar	0.36	**0.54**	**0.51**	**0.41**	**0.26**	**0.42**
	CStar+	0.36	**0.54**	**0.51**	**0.41**	**0.26**	**0.42**
Reuters	Star	0.42	0.33	0.31	0.26	0.21	0.31
	EStar	0.45	0.35	0.36	0.30	0.24	0.34
	GStar	0.45	0.36	**0.38**	0.31	**0.26**	0.35
	CStar	**0.46**	**0.41**	**0.38**	**0.32**	**0.26**	**0.37**
	CStar+	**0.46**	**0.41**	**0.38**	**0.32**	**0.26**	**0.37**

Tabla 3.2: Evaluación considerando el Jaccard-index

Los resultados de las comparaciones utilizando la medida *Fmeasure* se observan en la Tabla 3.3 y en la Figura 3.4.

		Valores de β					
Colecc.	**Algoritmo**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**	**Prom.**
AFP	Star	**0.61**	0.66	0.61	0.48	0.33	0.54
	EStar	0.51	0.67	0.64	**0.58**	**0.41**	0.56
	GStar	0.54	0.69	0.65	**0.58**	0.40	0.57
	CStar	0.54	**0.70**	**0.67**	**0.58**	**0.41**	**0.58**
	CStar+	0.54	**0.70**	**0.67**	**0.58**	**0.41**	**0.58**
Reuters	Star	0.60	0.50	0.48	0.40	0.35	0.47
	EStar	**0.63**	0.52	0.53	0.46	0.41	0.51
	GStar	**0.63**	0.51	0.54	0.47	**0.42**	0.51
	CStar	**0.63**	**0.59**	**0.55**	**0.49**	**0.42**	**0.54**
	CStar+	**0.63**	**0.59**	**0.55**	**0.49**	**0.42**	**0.54**

Tabla 3.3: Evaluación considerando el *Fmeasure*

Como se puede apreciar en las Tablas 3.2 y 3.3, a excepción del caso en que el umbral de semejanza es 0.25 para la colección AFP, los algoritmos CStar y CStar+ obtienen en la mayoría de los casos mejores valores de calidad que los algoritmos Star, EStar y GStar. Si se considera la calidad promedio obtenida por cada algoritmo respecto al *Jaccard-index*, CStar y CStar+ representan un 10.53 % de mejora respecto al Star, un 5.0 % de mejora respecto al EStar y un 2.44 % de mejora respecto al GStar en la colección AFP; en la colección *Reuters*, los algoritmos CStar y CStar+ logran una mejora del 19.35 % respecto

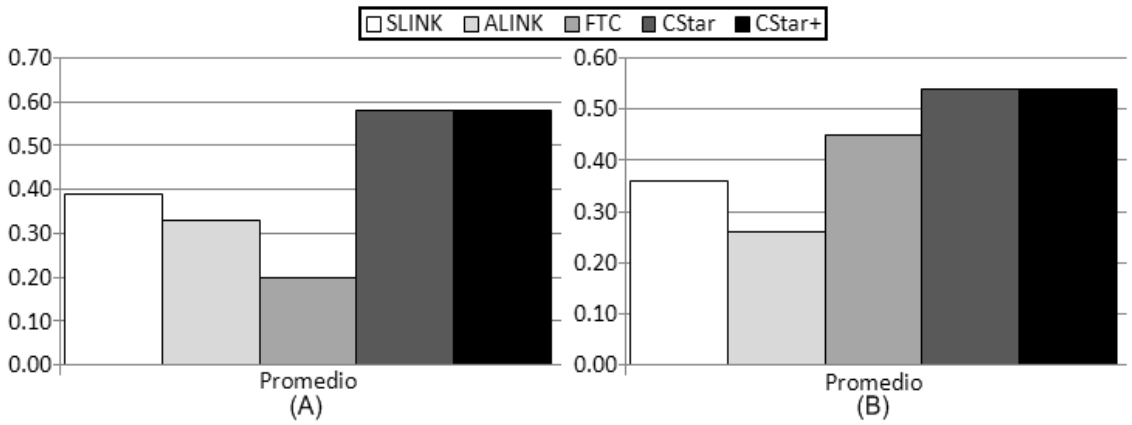

Figura 3.4: Evaluación considerando el *Fmeasure*: (A) AFP, (B) *Reuters*

al Star, 8.82 % respecto al EStar y un 5.71 % respecto al GStar.

Si se tiene en cuenta los resultados para la medida *Fmeasure* en AFP, CStar y CStar+ obtienen un incremento en la calidad del 7.41 %, 3.57 % y 1.75 % respecto a los algoritmos Star, EStar y GStar respectivamente; en la colección *Reuters*, los incrementos son del 14.89 % respecto al algoritmo Star y 5.88 % respecto a los algoritmos EStar y GStar.

Aunque las mejoras obtenidas por CStar y CStar+ respecto a los algoritmos EStar y GStar en el caso promedio no son elevadas, sobre todo en la colección AFP, en el experimento 2 se muestra que los grupos obtenidos por CStar y CStar+ son más cohesionados que los obtenidos por EStar y GStar.

Por último, como se aprecia en las Figuras 3.3 y 3.4, los valores de calidad obtenidos por CStar y CStar+ respecto a los algoritmos clásicos SLINK y ALINK son mucho mejores, sobre todo en la colección AFP. En el caso del algoritmo FTC, aunque alcanza un valor de calidad elevado respecto a ALINK y SLINK en la colección *Reuters*, los algoritmos CStar y CStar+ en esa misma colección lo superan en un 27.59 % respecto al *Jaccard-index* y en un 20.0 % considerando la medida *Fmeasure*.

Experimento 2

En este experimento se utiliza la medida interna definida en la sección 2.4.1 para comparar la cohesión interna de los grupos formados por los algoritmos CStar y CStar+ respecto a aquellos formados por los algoritmos EStar y GStar. Los resultados de esta

comparación en las colecciones AFP y *Reuters* se muestran en la Tabla 3.4.

		Valores de β				
Colecc.	**Algoritmo**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**
AFP	EStar	0.273	0.347	0.401	0.458	0.500
	GStar	0.275	0.340	0.406	0.461	0.504
	CStar	**0.353**	**0.437**	**0.505**	**0.570**	**0.601**
	CStar+	**0.353**	**0.437**	**0.505**	**0.570**	**0.601**
Reuters	EStar	0.256	0.319	0.391	0.455	0.518
	GStar	0.260	0.318	0.391	0.459	0.521
	CStar	**0.284**	**0.373**	**0.460**	**0.543**	**0.608**
	CStar+	**0.284**	**0.373**	**0.460**	**0.543**	**0.608**

Tabla 3.4: Evaluación considerando la cohesión interna de los grupos

Como se puede apreciar, CStar y CStar+ obtienen en todos los casos grupos con una cohesión interna superior a los que obtienen los algoritmos GStar y EStar. Es importante mencionar que los resultados obtenidos por cada algoritmo superan los valores de cohesión interna que presentan las clases calculadas manualmente por los expertos (AFP tiene 0.24 y *Reuters* tiene 0.23).

El resultado de los algoritmos CStar y CStar+ de acuerdo a esta medida refuerza los resultados obtenidos en el experimento 1.

Experimento 3

En este experimento se comparan la cantidad de grupos obtenidos y el promedio de elementos de éstos teniendo en cuenta los grupos obtenidos por los algoritmos Star, EStar, GStar, FCI, CStar y CStar+ en ambas colecciones. Los resultados de esta comparación se muestran en las Tablas 3.5 y 3.6.

Como se aprecia en la Tabla 3.5, sólo el algoritmo GStar obtiene menos grupos que los algoritmos CStar y CStar+. Sin embargo, es importante mencionar que la diferencia en cuanto al número de grupos es, en el peor caso, sólo de 22 grupos para el valor de β=0.45 en la colección *Reuters*.

Si se considera el promedio de elementos por grupos, como se puede apreciar en la Tabla 3.6, los algoritmos CStar y CStar+ obtienen grupos con mayor cantidad de elementos,

		Valores de β				
Colecc.	**Algoritmo**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**
AFP	Star	143	196	265	326	411
	EStar	114	179	258	317	397
	FCI	351	373	404	439	489
	GStar	**108**	**173**	**245**	**308**	**389**
	CStar	109	175	250	310	391
	CStar+	109	175	250	310	391
Reuters	Star	1060	1646	2351	2997	3650
	EStar	662	1203	1923	2622	3378
	FCI	4528	4695	4950	5252	5600
	GStar	**648**	**1153**	**1845**	**2522**	**3279**
	CStar	651	1155	1856	2544	3295
	CStar+	651	1155	1856	2544	3295

Tabla 3.5: Evaluación considerando la cantidad de grupos generados

		Valores de β				
Colecc.	**Algoritmo**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**
AFP	Star	9.32	5.43	3.57	2.83	2.13
	EStar	18.61	11.43	6.85	4.92	3.42
	FCI	3.29	2.96	2.56	2.27	1.92
	GStar	18.73	11.54	7.10	5.01	3.43
	CStar	**18.92**	**12.10**	**7.23**	**5.03**	**3.55**
	CStar+	**18.92**	**12.10**	**7.23**	**5.03**	**3.55**
Reuters	Star	47.21	24.97	15.77	10.93	7.74
	EStar	176.27	72.74	40.40	23.34	16.61
	FCI	4.62	4.39	4.04	3.68	3.34
	GStar	**225.36**	**87.29**	**49.28**	24.07	19.32
	CStar	218.03	83.73	46.98	**27.17**	**20.64**
	CStar+	218.03	83.73	46.98	**27.17**	**20.64**

Tabla 3.6: Evaluación considerando el promedio de elementos por grupos

solamente superados por el algoritmo GStar en la colección *Reuters* para los valores de β iguales a 0.25, 0.30 y 0.35.

Es importante mencionar que, respecto al promedio de elementos por grupo que poseen las clases etiquetadas manualmente de las colecciones AFP (28,44) y *Reuters* (110,261), los mejores resultados los obtienen los algoritmos CStar y CStar+ para el valor de β=0.25 en el caso de AFP y el algoritmo GStar en el caso de *Reuters*.

Experimento 4

Para mostrar que CStar, a pesar de ser dependiente del orden de análisis de los elementos, reduce la influencia de esta dependencia en la calidad del agrupamiento resultante, se realizaron 10 ejecuciones de CStar para cada valor de β variando el orden de los documentos en las colecciones. Posteriormente se calculó la calidad de cada agrupamiento resultante utilizando la medida interna y las medidas externas utilizadas en los experimentos 1 y 2. Utilizando estos valores de calidad se determinó la desviación estándard presente en los resultados para cada valor de β. El resultado de este experimento se puede observar en la Tabla 3.7.

		Valores de β				
Colecc.	**Med.**	**0.25**	**0.30**	**0.35**	**0.40**	**0.45**
AFP	Jaccard-index	1.05E-05	1.04E-03	1.14E-03	7.25E-05	9.01E-04
	Fmeasure	1.58E-05	1.08E-03	1.17E-16	7.91E-05	4.31E-04
	Sim. Global	4.83E-04	6.67E-04	1.40E-03	6.32E-04	5.16E-04
Reuters	Jaccard-index	1.58E-03	2.00E-03	8.41E-04	6.05E-04	1.24E-03
	Fmeasure	1.34E-03	1.33E-03	5.89E-04	9.30E-04	6.54E-04
	Sim. Global	5.16E-04	7.38E-04	4.83E-04	5.68E-04	4.22E-04

Tabla 3.7: Desviación estándar presente en los resultados

Como se puede apreciar en la Tabla 3.7, la desviación estándar de los resultados es muy pequeña para cada valor de β, evidenciándose que aunque los agrupamientos sean diferentes, la calidad de éstos es muy similar.

3.4. Recapitulación

En este capítulo se presentó el algoritmo CStar, el cual introduce un nuevo tipo de sub-grafo en forma de estrella y una heurística que permite realizar el cubrimiento de G_β utilizando estos sub-grafos. El cubrimiento que obtiene CStar de G_β determina un conjunto de grupos en los cuales se considera la pertenencia de un documento a más de un grupo.

El algoritmo CStar reduce las limitaciones presentes en los algoritmos Star, EStar y GStar como son: (i) obtención de grupos ilógicos, (ii) obtención grupos redundantes, (iii)

dejan elementos sin agrupar, (iv) eliminación de grupos densos previamente descubiertos y (v) alto consumo de memoria. Además de lo anterior, como se describió en la sección 3.1.2, la heurística que sigue CStar para obtener el cubrimiento de G_β garantiza que aunque el conjunto de centros que determina los grupos puedan variar, la densidad de los centros se mantiene.

Los experimentos realizados en colecciones estándar de documentos, frecuentemente utilizadas para evaluar el desempeño de los algoritmos de agrupamiento de documentos, mostraron que CStar obtiene mejores resultados que los algoritmos Star, EStar, GStar, FTC, FCI, Single-link y Average-link, de acuerdo a las medidas de evaluación utilizadas y respecto a otros parámetros como: el número de grupos generados y densidad de los mismos. La experimentación también mostró que la heurística que sigue CStar permite reducir la influencia del orden de análisis de los datos en la calidad del agrupamiento resultante.

Además del algoritmo CStar, se propuso una variante de éste denominado CStar+ que permite, a través de la detección de algunos tipos de componentes conexas, aumentar la calidad del agrupamiento y reducir el procesamiento del algoritmo. Aunque esta variante fue evaluada en las colecciones estándar, la calidad de los grupos obtenidos fue la misma que la que obtuvo el algoritmo CStar en dichas colecciones. La explicación de estos resultados se debe a que en las colecciones evaluadas y para los valores del parámetro α en el intervalo [0.50;0.85] con los cuales se experimentó no se formaron cuasi-cliques.

Capítulo 4

Conclusiones y Trabajo Futuro

4.1. Conclusiones

En la presente tesis se presentó un nuevo algoritmo de agrupamiento basado en grafos denominado CStar, el cual permite generar cubrimientos en bases documentales considerando que un documento pueda pertenecer a más de un grupo.

El algoritmo CStar sólo necesita del parámetro β para su funcionamiento y no necesita fijar el número de grupos a obtener, sino que descubre dinámicamente dicho número a través de un cubrimiento del grafo de semejanza que representa la colección a agrupar. La heurística que sigue CStar para obtener el agrupamiento evita el encadenamiento entre documentos, el cual es común en muchos algoritmos como el Single-Link. CStar también reduce las limitaciones presentes en algoritmos basados en grafos como Star, Extended Star y Generalized Star; estas limitaciones son: (a) obtención de grupos ilógicos, (b) obtención grupos redundantes, (c) alto consumo de memoria, (d) eliminación de grupos densos previamente descubiertos y (e) no agrupamiento de todos los elementos de la colección.

Las evaluaciones realizadas en colecciones estándar, considerando varias medidas de calidad externas e internas, mostraron que CStar obtiene mejores resultados que los algoritmos basados en grafos y que los algoritmos jerárquicos más usados para el agrupamiento de documentos. El algoritmo CStar obtiene menor cantidad de grupos que la mayoría de los algoritmos basados en grafos con excepción del algoritmo GStar. Los grupos obtenidos

por CStar son más densos que los obtenidos por la mayoría de los algoritmos con los que se evaluó.

Además del algoritmo CStar, se propuso una variante de éste denominada CStar+ la cual transforma el problema de cubrimiento del grafo G_β en el problema de cubrimiento del conjunto de componentes conexas de G_β aplicando la heurística de CStar. Este algoritmo introduce además la detección de algunos tipos de componentes conexas como los cuasi-cliques y los k-plex con el objetivo de aumentar la calidad del agrupamiento y reducir el procesamiento del algoritmo.

Aunque las evaluaciones en las colecciones utilizadas mostraron que CStar+ obtuvo el mismo resultado de calidad que CStar, los beneficios que puede aportar CStar+ respecto a la reducción de procesamiento y calidad del agrupamiento lo convierten en una alternativa a tener en cuenta en el procesamiento de otras colecciones de documentos.

Con el desarrollo del algoritmo CStar se considera que se han alcanzado los objetivos planteados al inicio de la tesis.

Los primeros resultados de esta tesis fueron reportados en [19] y del resto se tiene un artículo listo para enviarse a un evento internacional arbitrado [68].

4.2. Trabajo Futuro

Con base en los resultados obtenidos en esta tesis, existen algunas ideas que se pueden seguir como posibles trabajos futuros.

Primeramente, existen aplicaciones en las cuales las colecciones de documentos cambian con frecuencia al eliminar documentos existentes o adicionar nuevos, por ejemplo: colecciones de noticias, análisis del comportamiento de las facturas en una empresa, etc. En este tipo de colecciones ejecutar un algoritmo de agrupamiento estático como CStar resulta ineficiente ya que se debería ejecutar cada vez que exista un cambio en la colección o luego de acumular un número considerable de cambios para evitar procesamiento excesivo. Una posible línea a seguir es el desarrollo de una versión de CStar, en principio incremental, que permita actualizar el agrupamiento a medida que se adicionen nuevos

documentos en la colección sin necesidad de volver a procesar el conjunto completo de documentos.

Otra posible línea es incorporar como parte de la heurística que sigue CStar, para realizar el cubrimiento de G_β, el valor de semejanza entre los documentos. Este cambio, permitiría mejorar la calidad de los grupos y posibilita el desarrollo de una variante difusa que considere grados de pertenencia de los documentos a los grupos y no la pertenencia Booleana que se ha utilizado hasta el momento.

Como tarea inmediata, se pretende:

1. Desarrollar un estudio para definir un criterio de "grupo interesante" el cual pueda incluirse como condición en el criterio utilizado para detectar grupos redundantes. Este criterio de interés podría influir positivamente en la calidad de los grupos obtenidos por CStar.

2. Evaluar el desempeño de CStar+ en otras colecciones de documentos.

Bibliografía

[1] R. Agrawal and R. Srikant. Fast algorithms for mining association rules. *In Proceedings of the 20*th *VLDB Conference*, pages 487–499, 1994.

[2] H. Ahonen-Myka. Finding all maximal frequent sequences in text. *Proc. of the ICML99 Workshop on Machine Learning in Text Data Analysis*, pages 11–17, 1999.

[3] M. Alexandrov, A. Gelbukh, and P. Rosso. An approach to clustering abstracts. *In Proceeding of NLDB 2005 Conference, LNCS 3513, Springer, Verlag*, page 275–285, 2005.

[4] J. Aslam, K. Pelekhov K., and D. Rus. Static and dynamic information organization with star clusters. *In Proceedings of the seventh international conference on Information and knowledge management*, pages 208–217, 1998.

[5] J. Aslam, E. Pelekhov, and D. Rus. The star clustering algorithm for static and dynamic information organization. *Journal of Graph Algorithms and Applications, Vol. 8, No. 1*, pages 95–129, 2004.

[6] A. Banerjee, C. Krumpelman, S. Basu, R. Mooney, and J. Ghosh. Model based overlapping clustering. *In Proceedings of International Conference on Knowledge Discovery and Data Mining(KDD)*, pages 532–537, 2005.

[7] F. Beil, M. Ester, and X. Xu. Frequent term-based text clustering. In *Proceedings of the 8th Int. Conf. on Knowledge Discovery and Data Mining (KDD)'2002.*, pages 436–442, 2002.

[8] M. Berry. *Survey of Text Mining, Clustering, Classification and Retrieval*. Springer-Verlag, 2004.

[9] G. J. Bloy. Blind camera fingerprinting and image clustering. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 30(3):532–534, March 2008.

[10] T. Brants, F. Chen, and A. Farahat. A system for new event detection. *Proceedings of the 26th annual international ACM SIGIR conference on Research and Development in Information Retrieval, SIGIR2003*, pages 330–337, 2003.

[11] T. Calinski and J. Harabasz. A dendrite method for cluster analysis. *Communications in Statistics, 3*, pages 1–27, 1974.

[12] J. Carbonell, Y. Yang, J. Lafferty, R.D. Brown, T. Pierce, and X. Liu. Cmu report on tdt-2: Segmentation, detection and tracking. In *Proceedings of DARPA Broadcast News Workshop*, pages 117–120, 1999.

[13] A. Casillas, M. T. González de Lena, and R. Martínez. Document clustering into an unknown number of clusters using a genetic algorithm. *In Proceedings of the 6th International Conference of Text, Speech and Dialogue, TSD 2003*, pages 43–49, 2003.

[14] M. Connell, A. Feng, G. Kumaran, H. Raghavan, C. Shah, and J. Allan. Umass at tdt 2004. *In TDT2004 Workshop*, 2004.

[15] G. Cselle, K. Albrecht, and R. Wattenhofer. Buzztrack: Topic detection and tracking in email. *IUI2007*, pages 446–453, 2007.

[16] D. Cutting, D. Karger, J. Pedersen, and J. Tukey. Scatter/gather: A cluster-based approach to browsing large document collections. *In 15th ACM SIGIR*, pages 318–329, 1992.

[17] D. L. Davies and D. W. Bouldin. A cluster separation measure. *IEEE Trans. Patt. Anal. Mach. Intell. 1*, pages 224–227, 1979.

[18] B. C. M. Fung and et al. Hierarchical document clustering using frequent itemsets. In *Proceedings of the SIAM International Conference on Data Mining*, pages 59–70, 2003.

[19] A. Gago-Alonso, A. Pérez-Suárez, and J. Medina-Pagola. Acons[1] : a new algorithm for clustering documents. *In Proc. of XII CIARP, LNCS 4756, Springer-Verlag Berlin Heidelberg, CIARP2007*, pages 664–673, 2007.

[20] R.A. García-Hernández, J.Fco. Martínez-Trinidad, and J. A. Carrasco-Ochoa. A fast algorithm to find all the maximal frequent sequences in a text. *9th Iberoamerican Congress on Pattern Recognition (CIARP'2004), Lecture Notes in Computer Science*, 3287:478–486, 2004.

[21] R.A. García-Hernández, J.Fco. Martínez-Trinidad, and J. A. Carrasco-Ochoa. A new algorithm for fast discovery of maximal sequential patterns in a document collection. *7th Intelligent Text Processing and Computational Linguistics (CICLing'2006), Lecture Notes in Computer Science*, 3878:514–523, 2006.

[22] R. J. Gil-García. *Algoritmos de agrupamiento sobre grafos y su paralelización*. PhD thesis, Escuela Superior de Tecnología y Ciencias Experimentales, Departamento de Ingeniería y Ciencia de los Computadores, Universidad Jaume I, 2005.

[23] R. J. Gil-García, J. M. Badía-Contelles, and A. Pons-Porrata. Extended star clustering algorithm. *In Proceedings of CIARP'03, LNCS 2905*, pages 480–487, 2003.

[24] R. J. Gil-García, J. M. Badía-Contelles, and A. Pons-Porrata. Parallel algorithm for extended star clustering. *In Proceedings of CIARP'04, LNCS 3287*, pages 402–409, 2004.

[25] E. Greengrass. Information retrieval: A survey. Ed Greengrass. DOD Technical Report TR-R52-008-001, 2001.

[1] Este algoritmo fue presentado en el capítulo 3 con el nombre de CStar

[26] G. Gupta, A. Liu, and J. Ghosh. Automated hierarchical density shaving: A robust, automated clustering and visualization framework for large biological datasets. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, March 2008.

[27] D. Gusfield. *Algorithms on strings, trees and sequences: computer science and computational biology, chapter 6*. Cambridge University Press, 1997.

[28] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. The Morgan Kaufmann Series in Data Management Systems. Morgan Kaufmann Publishers, 2000.

[29] J. Han, T. Kim, and J. Choi. Web document clustering by using automatic keyphrase extraction. *IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology - Workshops*, pages 56–59, 2007.

[30] J. Hartigan. *Clustering Algorithms*. John Wiley & Sons, New York, NY, 1975.

[31] E. Hernández-Reyes. Agrupamiento de documentos basado en secuencias frecuentes maximales. Master's thesis, Instituto Nacional de Astrofísica, Óptica y Electrónica, 2007.

[32] J. Hershberger, N. Shrivastava, and S. Suri. Cluster hull: A technique for summarizing spatial data streams. *22nd International Conference on Data Engineering (ICDE'06)*, pages 138–140, 2006.

[33] J. H. Holland. Adaptation in natural and artificial systems. Technical report, Ann Arbor, MI: The University of Michigan Press, 1975.

[34] D. Ingaramo, D. Pinto, P. Rosso, and M. Errecalde. Evaluation of internal validity measures in short-text corpora. *In Proc. of the 9th Int. Conf. on Comput. Linguistics and Intelligent Text Processing, CICLing-2008, Springer-Verlag, LNCS(4919)*, pages 555–567, 2008.

[35] A. Jain and R. C. Dubes. *Algorithms for Clustering Data*. Prentice Hall, 1988.

[36] A. K. Jain, M.Ñ. Murty, and P. J. Flynn. Data clustering: a review. *ACM Computing Surveys*, 31(3):264–323, 1999.

[37] A. Kakes and O. Casas. *Teoría de grafos*. Editorial Pueblo y Educación, 1987.

[38] G. Karypis. Cluto: A clustering toolkit. Technical report, University of Minnesota, Department of Computer Science, Minneapolis, MN 55455, 2002.

[39] G. Karypis, E. Han, and V. Kumar. Chameleon: A hierarchical clustering algorithm using dynamic modeling. In *IEEE Computer, 32(8)*, page 68–75, 1999.

[40] W. H. Kim and T. Sikora. Image denoising method using diffusion equation and edge map estimated with k-means clustering algorithm. *Eight International Workshop on Image Analysis for Multimedia Interactive Services (WIAMIS '07)*, pages 21–24, 2007.

[41] L. Kuncheva and S. Hadjitodorov. Using diversity in cluster ensembles. *In Proceedings of IEEE SMC 2004, The Netherlands*, pages 1214– 1219, 2004.

[42] I. O. Kyrgyzov, H. Maitre, and M. Campedel. A method of clustering combination applied to satellite image analysis. *14th International Conference on Image Analysis and Processing (ICIAP 2007)*, pages 81–86, 2007.

[43] A.N. Langville and C. D. Meyer. Google's pagerank and beyond: The science of search engine rankings. *Princeton University Press. ISBN 0-691-12202-4*, 2006.

[44] B. Larsen and C. Aone. Fast and effective text mining using linear-time document clustering. In *KDD'99*, pages 16–22, 1999.

[45] E. Levner, D. Pinto, P. Rosso, D. Alcaide, and R. Sharma. Fuzzifying clustering algorithms: The case study of majorclust. *In Proceeding of the 6th Mexican International Conference on Artificial Intelligence, MICAI-2007, Springer-Verlag, LNAI (4827)*, pages 821–830, 2007.

[46] D. Lewis. Reuters-21578 text categorization text collection 1.0. http://kdd.ics.uci.edu.

[47] P. Mahata. Exploratory consensus of hierarchical clusterings for melanoma and breast cancer. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, March 2008.

[48] Y. Man-Quan, Luo Wei-Hua, Z. Zhao-Tao, and B. Shuo. Ict's approaches to htd and tracking at tdt2004. *In TDT2004 Workshop*, 2004.

[49] C. Manning and H. Schutze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 1999.

[50] J. F. Martínez-Trinidad, J. Ruiz-Shulcloper, and M. Lazo-Cortés. Structuralization of universes. *Fuzzy Sets and Systems, vol. 112 (3)*, pages 485–500, 2000.

[51] H. Matsuda, T. Ishihara, and A. Hashimoto. Classifying molecular sequences using a linkage graph with their pairwise similarities. *Theoretical Computer Science*, 210(2):305–325, 1999.

[52] R.T. Ng and J. Han. Clarans: A method for clustering objects for spatial data mining. *IEEE Transactions on Knowledge and Data Engineering*, pages 1003–1016, 2002.

[53] B. On, E. Elmacioglu, D. Lee, J. Kang, and J. Pei. Improving grouped-entity resolution using quasi-cliques. *Sixth International Conference on Data Mining, ICDM2006*, pages 1008–1015, 2006.

[54] J. E. Medina Pagola, E. Guevara Martinez, J. Hernández Palancar, A. Hechavarría Díaz, and R. Hernández León. Similarity measures in documents using association graphs. In *Proceedings of X CIARP, LNCS 3773*, pages 741–751, 2005.

[55] J. E. Medina Pagola, A. Y. Rodríguez, and A. Hechavarría. Formal distance vs. association strength in text procesing. In *Proceedings of CIARP2007, LNCS 4756*, pages 930–939, 2007.

[56] J. E. Medina Pagola, A. Y. Rodríguez, A. Hechavarría, and J. Hernández Palancar. Document representation using global association distance model. In *Proceedings of ECIR 2007, LNCS 4425*, pages 565–572, 2007.

[57] G. Pallis, L. Angelis, and A. Vakali. Model-based cluster analysis for web users sessions. *Proc. 15th Int'l Symp. Methodologies for Intelligent Systems (ISMIS'05)*, pages 219–227, 2005.

[58] A. Peñas, A. Rodrigo, V. Sama, and F. Verdejo. Overview of the answer validation exercise 2006. *In Working Notes for the CLEF 2006 Workshop. Alicante, Spain*, 2006.

[59] J. Pei, D. Jiang, and A. Zhang. On mining cross-graph quasi-cliques. *In Proceedings of the eleventh ACM SIGKDD international conference on Knowledge discovery in data mining*, pages 228 – 238, 2005.

[60] J. Peng and J. Zhu. Refining spherical k-means for clustering documents. In *International Joint Conference on Neural Networks (IJCNN '06)*, pages 4146–4150, 2006.

[61] S. Petridou, V. Koutsonikola, A. Vakali, and G. Papadimitriou. A divergence-oriented approach for web users clustering. *Proc. Int'l Conf. Computational Science and Its Applications (ICCSA '06)*, pages 1229–1238, 2006.

[62] S. Petridou, V. A. Koutsonikola, A. I. Vakali, and G. I. Papadimitriou. Time-aware web users'clustering. *IEEE Transactions on Knowledge and Data Engineering*, pages 653–667, 2008.

[63] D. Pinto and P. Rosso. On the relative hardness of clustering corpora. *In Proceedings of the 10th Int. Conf. on Text, Speech and Dialogue, TSD-2007, Springer-Verlag, LNAI (4629)*, pages 155–161, 2007.

[64] A. Pons-Porrata. *Algoritmos de agrupamiento para la detección de sucesos en noticias*. PhD thesis, Centro de Investigación en Computación, 2004.

[65] A. Pons-Porrata, R. Berlanga-Llavori, and J. Ruiz-Shulcloper. On-line event and topic detection by using the compact sets clustering algorithm. *Journal of Intelligent and Fuzzy Systems, 3-4*, pages 185–194, 2002.

[66] A. Pons-Porrata, R. Berlanga-Llavori, J. Ruiz-Shulcloper, and J. M. Pérez-Martínez. Jerartop: A new topic detection system. *In Proceedings of the 9th Iberoamerican Congress on Pattern Recognition,LNCS 3287, Springer Verlag*, pages 446–453, 2004.

[67] A. Pons-Porrata, J. Ruiz-Shulcloper, R. Berlanga-Llavori, and Y. Santiesteban-Alganza. Un algoritmo incremental para la obtención de cubrimientos con datos mezclados. *Reconocimiento de Patrones. Avances y Perspectivas. Research on Computing Science, CIARP'2002*, pages 405–416, 2002.

[68] A. Pérez-Suárez, J. F. Martínez-Trinidad, J. A. Carrasco-Ochoa, and J. Medina-Pagola. A new graph-based algorithm for clustering documents. *enviado al ICDM2008*.

[69] A. Pérez-Suárez and J.E Medina-Pagola. A clustering algorithm based on generalized stars. *In Proceedings of the 5th International Conference on Machine Learning and Data Mining in Pattern Recognition, MLDM 2007, LNAI 4571, Leipzig, Germany, July 18-20*, pages 248–262, 2007.

[70] J. Ruiz-Shulcloper, G. Sanchez, and M.A Abidi. Clustering in mixed incomplete data. *Heuristics and Optimization for Knowledge Discovery*, pages 88–106, 2002.

[71] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, 1975.

[72] F. Shahnaz. Clustering method based on nonnegative matrix factorization for text mining. Master's thesis, Department of Computer Science, University of Tennessee, Knoxville, TN, 2004.

[73] Z. Shi, , and M. Ester. Performance improvement for frequent term-based text clustering algorithm. In *Research Project of CMPT 843*, 2003.

[74] J. R. Shulcloper, E. Alba, and M. Lazo. *Introducción al Reconocimiento de Patrones (Enfoque Lógico-Combinatorio)*, volume 51 of *Serie Verde*. Grupo de Reconocimiento de Patrones Cuba-México, Centro de Investigación y de estudios avanzados del IPN, Dpto. de Ingeniería Eléctrica, 1995.

[75] R. Sibson. An optimally effcient algorithm for the single link cluster method. In *Computer Journal, 16*, pages 30–34, 1973.

[76] G. Sánchez-Díaz. *Desarrollo de algoritmos para el agrupamiento de grandes volúmenes de datos mezclados*. PhD thesis, Centro de Investigación en Computación, 2001.

[77] W. Song and S. C. Park. Genetic algorithm-based text clustering technique: Automatic evolution of clusters with high efficiency. *In Proceedings of the Seventh International Conference on Web-Age Information Management Workshops (WAIMW'06)*, page 17, 2006.

[78] M. Spitters and W. Kraaij. Tno at tdt2001: Language model-based topic detection. In *Topic Detection and Tracking (TDT) Workshop*, November 2001.

[79] B. Stein and M. Busch. Density-based cluster algorithms in low-dimensional and high-dimensional applications. *In Proceedings of Second International Workshop on Text-Based Information Retrieval, TIR05*, pages 45–56, 2005.

[80] B. Stein, S. Meyer, and F. Witbrock. On cluster validity and the information need of users. *In Proceedings of the 3rd IASTED*, pages 216–221, 2003.

[81] M. Steinbach, G. Karypis, and V. Kumar. A comparison of document clustering techniques. In *KDD Workshop on Text Mining*, 2000.

[82] TDT. The 2004 topic detection and tracking (tdt2004). *Task Definition and Evaluation Plan, version 1.0*, 2004.

[83] TREC. Text retrieval conference. http://trec.nist.gov.

[84] Esko Ukkonen. On-line construction of suffix trees. *Algorithmica*, 14(3):249–260, 1995.

[85] C. J. van Rijsbergen. *Information Retrieval*. Butterworths, London, second edition, 1979.

[86] M. Vignes and F. Forbes. Gene clustering via integrated markov models combining individual and pairwise features. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, August 2007.

[87] E.M. Voorhees. Implementing agglomerative hierarchical clustering algorithms for use in document retrieval. In *Information Processing and Management, 22*, pages 465–476, 1986.

[88] W. Wai-chiu and A. W. Fu. Incremental document clustering for web page classification. *In IEEE 2000 International Conference on Information Society in the 21st Century: Emerging technologies and new challenges*, 2000.

[89] X. Wan and Y. Peng. A measure based on optimal matching in graph theory for document similarity. In *LNCS 3411*, pages 227–238, 2005.

[90] K. L. Wu, C. Aggarwal, and P. Yu. Personalization with dynamic profiler. *Proc. Third Int'l Workshop Advanced Issues of E-commerce and Web-Based Information Systems (WECWIS '01)*, pages 12–20, 2001.

[91] W. Xu, X. Liu, and Y. Gong. Document-clustering based on non-negative matrix factorization. *In Proceedings of SIGIR'03*, pages 267–273, 2003.

[92] O. R. ZaIane and C. H. Lee. Clustering spatial data when facing physical constraints. *Second IEEE International Conference on Data Mining (ICDM'02)*, pages 737–740, 2002.

[93] O. Zamir and O. Etziony. Web document clustering: A feasibility demonstration. *In Proceedings of the 21st Annual International ACM SIGIR Conference*, pages 46–54, 1998.

[94] Y. Zhao and G. Karypis. Evaluation of hierarchical clustering algorithms for document datasets. In *11th Conference of Information and Knowledge Management (CIKM)*, pages 515–524, 2002.

[95] D. Zuckerman. Np-complete problems have a version that's hard to approximate. *In Proceedings of the Eight Annual Structure in Complexity Theory Conference, IEEE Computer Society*, page 305–312, 1993.