Escola Tècnica Superior d'Enginyeria Informàtica
Universitat Politècnica de València

CLUSTERING DE DOCUMENTOS CON RESTRICCIONES DE TAMAÑO

Trabajo Fin de Máster

Master Universitario en Gestión de la Información

Autor: Diego Fernando Vallejo Huanga

Tutor: Cèsar Ferri Ramírez

2015-2016

A Pauly

A María, Hugo y Oscar

A mis amigos

Agradecimiento

Primeramente deseo expresar mi más sincera muestra de agradecimiento, a DIOS sobre todas las cosas, por mantenerme en su camino y permitirme seguirle.

Agradezco a las personas e instituciones que sin su valioso aporte hubiera sido imposible que este trabajo se hubiese podido concretar.

Al Gobierno de la República del Ecuador, que mediante la Secretaría de Educación Superior, Ciencia, Tecnología e Innovación SENESCYT, y su programa de becas ha financiado mis estudios y me ha permitido alcanzar uno de mis objetivos inmediatos.

Agradezco a mi tutor, Dr. Cèsar Ferri Ramírez, por sus dirección, apoyo y confianza en mi trabajo. La oportunidad de trabajar en este proyecto, el tiempo brindado, la experiencia y rigurosidad en sus metodologías ha sido un valioso aporte, para mi vida académica. Realmente ha sido una experiencia enriquecedora el trabajar en el Departamento de Sistemas Informáticos y Computación de la UPV.

Deseo extender este agradecimiento también hacia mis profesores del Máster Universitario en Gestión de la Información, por su invalorable aporte a mi formación durante este período, así como también a mis compañeros del MUGI.

Finalmente, me queda agradecer a mi esposa, familia y amigos, seres extraordinarios y condición sine qua non para que este trabajo se encuentre plasmado ahora como una memoria.

Diego Fernando Vallejo Huanga

Valencia, Julio 2016.

Resumen

El análisis de *clusters* tiene por objetivo dividir objetos de datos en grupos, de tal manera que los objetos dentro de un mismo grupo sean muy similares entre sí y diferentes de los objetos de otros grupos. Tradicionalmente, el *clustering* es visto como un método de aprendizaje no supervisado, que agrupa los objetos de datos basándose únicamente en la información presentada en el conjunto de datos, sin información externa. El K-Medoides es uno de los más famosos y sencillos algoritmos de agrupamiento, donde el usuario define el número de *clusters* deseados.

En muchas aplicaciones del mundo real, tales como: codificación de imágenes, agrupamientos espaciales en geo-informática, segmentación de clientes o agrupamiento de documentos, por lo general hay restricciones o prioridades en la definición del problema que limitan, el espacio de posibles soluciones, al problema o rango de interés de las soluciones. Este tipo de problemas se tratan mediante métodos de agrupamiento semi-supervisados.

El presente trabajo pretende diseñar, implementar y probar modificaciones en los algoritmos de *clustering* tradicionales, para incorporar restricciones de tamaño en cada *cluster*. Específicamente, se proponen dos nuevos algoritmos de agrupamiento semi-supervisado, basados en: programación lineal entera binaria con restricciones del tipo *cannot-link* y en una variación del algoritmo K-Medoides, respectivamente.

Para mostrar la aplicabilidad de los métodos de agrupación semi-supervisados propuestos, se aborda el problema de configuración automática del programa de una conferencia, con agrupación de artículos por similitud. Se incluyen experimentos, aplicando las nuevas técnicas, sobre conjuntos de datos de conferencias reales: ICMLA-2014, AAAI-2013 y AAAI-2014. Los resultados de estos experimentos muestran que los nuevos métodos son capaces de resolver problemas prácticos y reales.

Palabras clave: *Minería de datos, Agrupamiento con restricciones, Restricción de tamaño, K-Medoides, Programación lineal*.

Abstract

Cluster analysis aims to divide data objects in groups, so that objects within a group are very similar and different of those objects from other groups. Traditionally, clustering is known as a method of unsupervised learning, which groups data objects only based on the information presented in the dataset, without external information. The K-Medoids is one of the most famous and simple clustering algorithms, where the user defines the desired number of *clusters*.

In many real-world applications, such as image coding, spatial clustering in geo-informatics, customer segmentation or grouping of documents, there are usually constraints or priorities in the problem definition that limit the space of possible solutions to the problem or rank the interest of the solutions. This kind of problems are addressed by semi-supervised clustering methods.

This paper aims to design, implement and test modifications in traditional clustering algorithms to incorporate size constraints in each cluster. Specifically, two new algorithms are proposed to semi-supervised clustering, based on: binary integer linear programming with cannot-link constraints and a variation of the K-Medoids algorithm, respectively.

The applicability of the proposed semi-supervised clustering methods is illustrated by addressing the problem of automatic configuration of conference schedules by clustering articles by similarity. We include experiments, applying the new techniques, over real conferences datasets: ICMLA-2014, AAAI-2013 and AAAI-2014. The results of these experiments show that the new methods are able to solve practical and real problems.

Keywords: *Data Mining, Clustering with constraints, Size constraint, K-Medoids, Linear programming.*

Índice de Contenidos

Índice de Figuras

Índice de Tablas

1. Introducción

"In God we trust, all others must bring data"

William Edwards Deming

A través de una amplia variedad de campos, los datos son recogidos y acumulados en un contexto cada vez más complejo y a un ritmo dramáticamente acelerado. La aparición de nuevas tecnologías y la incesante necesidad de tener un mundo cada vez más conectado y globalizado, ha contribuido a que esta cantidad de datos se multiplique exponencialmente.

Conforme creció el volumen de datos digitales fue necesario el desarrollo de estructuras tecnológicas adecuadas para gestionar, analizar y representar dichos datos. Con la aparición de computadores cada vez más potentes y con mayores prestaciones, se pudieron empezar a generar índices para los grandes volúmenes de datos de forma automática, para su posterior análisis.

Esto último provocó una necesidad urgente de una nueva generación de teorías computacionales y herramientas para ayudar a los seres humanos a la extracción de información útil (conocimiento) [64] y desembocó en la aparición de nuevos campos del conocimiento como la minería de datos, la inteligencia artificial y el *machine learning*, además de un fenómeno disruptivo denominado "*Big Data*". Surge así también un nuevo tipo de análisis predictivo, como evolución lógica del simple análisis descriptivo.

El manejo y procesamiento de estos grandes volúmenes de información no es una tarea trivial y requiere de conocimientos transversales y técnicas que sean capaces de explotar estos datos para generar y descubrir conocimiento novedoso y apropiado.

El *machine learning* (aprendizaje automático) se define como [72] una rama de la inteligencia artificial que aborda el estudio y construcción de modelos capaces de aprender de los datos. Una técnica de aprendizaje que no necesita supervisión es el denominado *clustering*.

El desarrollo de la metodología de *clustering* ha sido interdisciplinaria. Investigadores de muchas áreas han contribuido: taxonomistas, psicólogos, biólogos, estadísticos, cuentistas sociales, ingenieros e informáticos. Por ello el *clustering* se puede encontrar bajo diferentes nombres en diferentes contextos, como: aprendizaje no supervisado (en *Machine Learning* y *Pattern Recognition*), taxonomía numérica (en Biología, Ecología), tipología (en Ciencias Sociales) y partición (en Teoría de Grafos). Además de diferentes terminologías, estas comunidades tienen diferentes supuestos acerca de los componentes del proceso de *clustering* y el contexto en el que se usa [60] [61]. Este trabajo usa la terminología propuesta en los campos de reconocimiento de patrones y *machine learning*.

Aunque las técnicas de *clustering* y *data mining* suelen estar orientadas a dar solución a problemas computacionales, también han sido usadas para la segmentación de clientes, clasificación de documentos, segmentación de imágenes, redes de sensores inalámbricas, sistemas recomendadores de productos y como una solución efectiva para atender las necesidades de relación con el cliente por parte de las empresas [11]. En muchas ocasiones los datos o los problemas, per se, poseen ciertas restricciones implícitas, que los algoritmos tradicionales de agrupamiento no aprovechan.

Últimamente se han incorporado al proceso de *clustering* ciertas restricciones de tamaño y relaciones de pertenencia de objetos a los *clusters*, que han demostrado que el rendimiento de los algoritmos planteados para la solución de este tipo de problemas, aumenta significativamente [3]. En el *clustering* con restricciones de tamaño, el tamaño del *cluster* hace referencia al número total de objetos en cada *cluster*, y la distribución de clases es la configuración de los tamaños de los *clusters* en la solución final de *clustering* [17].

Una de las tareas típicas en los estudios de marketing es la segmentación de clientes, dónde los clientes son divididos dentro de diferentes grupos con la finalidad de que un equipo de ventas en particular le sea asignado o porque es necesario estipular una cantidad específica de dinero para las campañas de marketing en cada grupo. Si cada equipo de ventas es de diferente tamaño o la cantidad de recursos necesarios para las campañas tienen un diferente monto, entonces el problema de segmentación de clientes se convierte en un problema de *clustering* de datos con restricciones de tamaño [3]. La importancia de segmentar a los clientes estriba en los beneficios económicos que recibe la empresa por la fidelización de sus clientes, debido a la sensación de bienestar que provoca en el consumidor cuando este recibe productos personalizados.

Por otra parte, el agrupamiento de documentos, debido a su misión y campo de acción presenta ciertas particularidades a las que es menester atender. En forma genérica se debe concebir al *clustering* de documentos como [46] el particionamiento de una colección de documentos en varios grupos, de tal manera que los documentos en el mismo *cluster* sean lo más similares posibles, y a su vez dos grupos cualquiera, deben ser tan diferentes como sea posible. El agrupamiento de documentos se ha aplicado a muchos campos de estudio, como: la recuperación de información, la detección de temas y el seguimiento de contenidos; todos ellos en intrínseca relación con el lenguaje.

Se debe entender que el lenguaje es más que una colección de palabras, expresa relaciones y conceptos semánticos en los que se puede observar dependencias estructurales, correferencias, roles semánticos, intenciones, etc. [56].

En principio, se pensaría que campos como la informática y la lingüística se encuentran en las antípodas, pero la explotación y tratamiento sistemático del lenguaje ha podido ser trasladado a campos más técnicos gracias al desarrollo del procesamiento del lenguaje natural (NLP). Esto ha supuesto un mejor entendimiento de las solicitudes de información, y ha permitido mejorar la efectividad de tareas ejecutadas sobre documentos o colecciones de documentos. El NLP busca utilizar información semántica, además de información estadística, para analizar textos, extraer y almacenar información, y posteriormente realizar tareas de tratamiento de información y de *clustering* de documentos, de manera más adecuada [56].

Un artículo científico, llamado coloquialmente *paper* como anglicismo, es un trabajo de investigación publicado en revistas especializadas y en conferencias. Uno de los principales inconvenientes que se presentan en el momento de organizar las sesiones de una conferencia es la gran cantidad de temáticas abordadas por los *papers* presentados, que se encuentran diseminadas en diferentes áreas de conocimiento y con estructuras que, a priori, parecen no tener relación alguna. Además el problema se vuelve mucho más complejo si los tiempos que son asignados para cada sesión en una conferencia son limitados, por lo que la asignación del número de *papers* a ser expuestos en una sesión viene restringido por una cantidad específica. Este escenario puede ser catalogado como un problema de *clustering* de documentos con restricciones de tamaño.

Ergo, este estudio tiene como objetivo principal, usar técnicas de minería de datos y *clustering* orientadas al agrupamiento de documentos con restricciones de tamaño con un enfoque hacia la configuración de sesiones en conferencias. Para lo cual se pretende diseñar, implementar y probar modificaciones en algoritmos de *clustering* tradicionales

incorporando las restricciones de tamaño a los *clusters*. Se proponen dos nuevos algoritmos para agrupar documentos, con sustento teórico en: programación lineal entera binaria con restricciones del tipo *cannot-link* y en una variación del algoritmo K-Medoids, respectivamente.

Bajo toda la premisa antes planteada, el segundo capítulo de este trabajo hablará del marco teórico y conceptual bajo el cual se despliega este estudio.

En el tercer capítulo, se estudiarán los algoritmos de *clustering* tradicionales, sus características principales y los mecanismos de validación de sus resultados. Además se revisan algunas de las particularidades asociadas al *clustering* de documentos.

El cuarto capítulo abordará el problema de *clustering* semi-supervisado y los problemas de agrupamiento con restricciones. Se hará una revisión del estado de arte y trabajo relacionado con la temática propuesta, esto último servirá como referencia para el desarrollo de los dos nuevos algoritmos de solución propuestos en el capítulo 5.

En el sexto capítulo se presentará la metodología planteada para abarcar de manera holística el problema de *clustering* de documentos y finalmente se realizarán experimentos con los nuevos algoritmos. En primera instancia, sobre *datasets* de *benchmarking* multivariantes y con atributos de tipo numérico con la finalidad de validar el funcionamiento de dichos algoritmos. Y a posteriori, se emplearán *datasets* multivariantes con atributos de tipo texto, que representarán a *papers* de conferencias en el área del *machine learning*. Una forma sencilla con la que se representó a los datos agrupados fue mediante visualizaciones.

Finalmente, en el capítulo 7, se pueden encontrar una serie de conclusiones y recomendaciones derivadas del trabajo y algunas prospectivas de mejora a los planteamientos desarrollados, para un trabajo futuro.

2. Minería de Datos

Resumen: En este capítulo se introducirán definiciones fundamentales en el campo de la minería de datos y aspectos relacionados con el aprendizaje automático. Se abordarán tópicos complementarios como: el tratamiento automatizado de la información, representación gráfica de instancias, modelos de recuperación de información y técnicas de optimización que son primordiales para definir los posteriores esquemas de agrupamiento de documentos. Estas definiciones servirán como base para los algoritmos propuestos en el Capítulo 5.

2.1. Introducción a la Minería de Datos

El objetivo de este trabajo es buscar el agrupamiento de documentos utilizando restricciones de tamaño para cada grupo, por tanto, es necesario entender el funcionamiento de los métodos de descubrimiento de información y específicamente cómo funciona el proceso de *clustering*.

En la literatura existen muchas definiciones para caracterizar expresiones como el KDD *(Knowledge Discovery in Databases*) y *Data Mining*, pero de manera resumida se debe entender que la minería de datos es parte de un proceso más global que es el KDD, y que este último es un proceso que consta de una secuencia interactiva e iterativa de pasos para la extracción de conocimiento en bases de datos: comprensión del problema, selección del *dataset*, limpieza de datos y pre-procesamiento, reducción y proyección de datos, análisis exploratorio y selección del modelo, minería de datos, interpretación de los resultados, utilización del conocimiento [64].

El *data mining* entrega diversas técnicas para encontrar patrones y descubrir relaciones insospechadas en grandes conjuntos de datos, bajo un enfoque multidisciplinario que combina a ramas científicas como la estadística, el *machine learning*, tecnologías difusas, etc. Las relaciones obtenidas mediante técnicas de minería de datos son conocidas como modelos o patrones y se puede distinguir dos tipos de paradigmas [7]: el paradigma orientado a la verificación, donde el sistema comprueba la hipótesis del usuario y el paradigma orientado hacia el descubrimiento, donde el sistema encuentra nuevas normas y patrones de manera autónoma.

La Figura 1 ilustra una de las posibles taxonomías, que componen a la minería de datos:

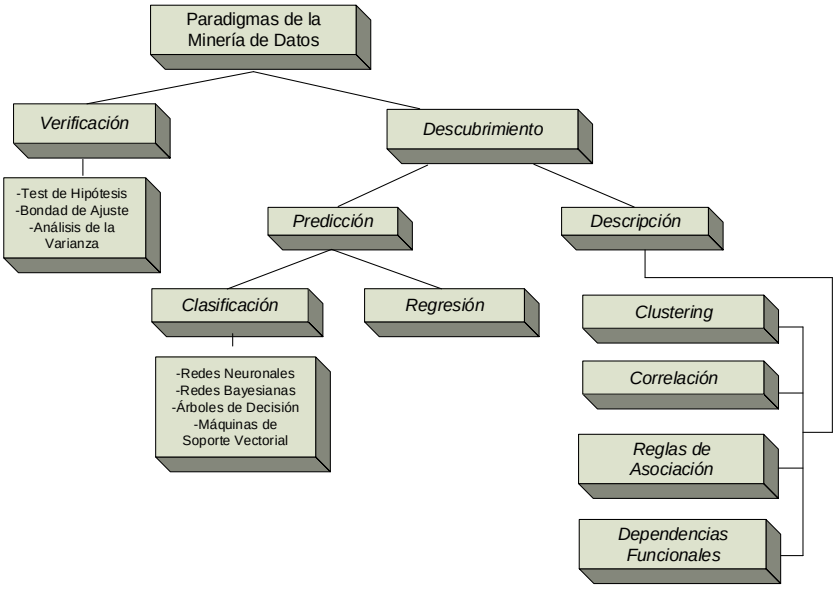

Figura 1. *Paradigmas de la Minería de Datos [7]*

En el campo del *Machine Learning*, los métodos de predicción se conocen como Aprendizaje Supervisado y los métodos de descripción como Aprendizaje no Supervisado.

El aprendizaje supervisado es una técnica para deducir una función a partir de datos de entrenamiento, estos datos de entrenamiento proveen las etiquetas de las clases (*true labels*) a las que pertenecen los patrones. En esta misma línea temática, el término *ground truth* es utilizado para hacer referencia a la información proporcionada por la observación directa en contraposición a la información proporcionada por la inferencia.

En el aprendizaje no supervisado, por otra parte, el modelo se ajusta a las observaciones y al contrario que en el aprendizaje supervisado no existe un conocimiento, a priori, de las clases y no se provee al modelo de datos de entrenamiento [1].

En estos últimos es donde se centrará la atención de este estudio, ya que el *clustering* es catalogado como un tipo de aprendizaje no supervisado y su estudio se ampliará en el Capítulo 3.

2.2. Distancias, Matriz de Distancias y Matriz de Similaridad

Para tareas de minería de datos y la aplicación de distintos tipos de algoritmos en el campo de la inteligencia artificial, se suele aplicar distintas métricas de distancia o de similitud, dependiendo del contexto de uso. La utilización de un patrón de medida adecuado supone un impacto importante en el rendimiento de una tarea o proceso llevado a cabo por dicho algoritmo. Las distancias suelen ser utilizadas en tareas de clasificación, agrupamiento, regresión, entre otras.

Para la formulación matemática de los algoritmos, que se plantearán en capítulos posteriores, es necesario identificar *ex-ante* ciertos conceptos y definiciones relacionados con las distancias, similitudes y sus propiedades.

Se debe entender a una distancia como una medida de proximidad entre los objetos o instancias, que puede ser usado tanto para variables cuantitativas, como, cuando las variables observadas son de tipo más general, o incluso cuando no se dispone de variables propiamente dichas. El entendimiento del concepto de distancia estadística

junto a sus propiedades constituye una importante herramienta, tanto en la estadística matemática como en el análisis de datos, ya que permite interpretar geométricamente muchas técnicas clásicas del análisis multivariante, equivalentes a representar estos objetos como puntos de un espacio métrico adecuado [53].

Para que una distancia d_{ij}, se pueda considerar como tal, debe cumplir con las propiedades generales enumeradas en la Tabla 1 [53]:

Id.	Propiedades	Nombre
P1	$d_{ij} \geq 0$	No negatividad
P2	$d_{ii} = 0$	Reflexividad
P3	$d_{ij} = d_{ji}$	Simetría
P4	$d_{ij} \leq d_{ik} + d_{jk}$	Desigualdad Triangular
P5	$d_{ij} = 0 \Leftrightarrow i = j$	Identidad de los indiscernibles

Tabla 1. *Propiedades Generales de las Distancias [53]*

En la Tabla 2, se presentan algunos calificativos que adquieren las distancias según sus propiedades [53]:

Calificación	Propiedades
Disimilaridad	P1, P2, P3
Distancia Métrica	P1, P2, P3, P4, P5

Tabla 2. *Calificación de una Distancia según sus Propiedades [53]*

Existen múltiples distancias que cumplen con estas propiedades como la Euclídea (o Euclideana), Euclídea Normalizada, *Mahalanobis*, *Manhattan*, *Chebyshev*, *Minkowski*, etc. más detalles acerca de sus características se detallan en [54].

Bajo esta premisa, se dice que $DM \in M_{nxn}$, donde n es el número de instancias u objetos, es una matriz de distancias. De esta manera, cada elemento d_{ij} de la matriz DM representaría la distancia o proximidad existente entre la instancia i y la instancia j.

$$DM_{ij} = \begin{bmatrix} d_{11} & d_{12} & \dots & d_{1n} \\ d_{21} & d_{22} & \dots & d_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1} & d_{n2} & \cdots & d_{nn} \end{bmatrix}$$

Por otra parte se debe entender que una similaridad S_{ij}, es una medida del grado de semejanza entre dos elementos i y j, de tal manera que si ambos son muy parecidos entonces s_{ij} se aproxima a 1. Si se trabaja sobre un conjunto con n instancias, y esta métrica es aplicada en un espacio $\mathbb{R}$ vectorial, se cumplen las siguientes propiedades [53]:

$$S_{ij} = \begin{bmatrix} s_{11} & s_{12} & \dots & s_{1n} \\ s_{21} & s_{22} & \dots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix};\ \{\forall\, s_{ij}\} \in [0,1];\ s_{ii} = 1;\ s_{ij} = s_{ji}$$

Dependiendo de la literatura tratada a esta matriz se la conoce también como matriz de proximidades [52] y en el campo de la minería de datos, que es objeto de este estudio, se suele manejar la matriz de disimilaridad [57], que únicamente es el valor complementario de $s_{ij} \rightarrow (1 - s_{ij})$.

La similaridad entre dos instancias i y j, cuyos atributos o características son de tipo: nominal, categórico o binario, puede ser descrita por medio del siguiente grupo de ecuaciones:

$$a = \sum_{k=1}^{n} x_{ik}x_{jk}\,; \quad b = \sum_{k=1}^{p}(1 - x_{ik})x_{jk}\,; \quad c = \sum_{k=1}^{p} x_{ik}(1 - x_{jk})\,; \quad d = \sum_{k=1}^{p}(1 - x_{ik})(1 - x_{jk})$$

Donde:

x_{ik}: característica k de la instancia i

x_{jk}: característica k de la instancia j

Varios autores han propuesto diversos tipos de coeficientes de similaridad y algunos de ellos junto a sus propiedades se exponen en [53] y se resumen en la Tabla 3.

Autor/Nombre	Similaridad	Rango	$S \geq 0$	Métrica	Euclídea
Kulczynsky	$\frac{a}{b+c}$	$[0, \infty[$	Si		
Jaccard	$\frac{a}{a+b+c}$	$[0, 1]$	Si	Si (Si)	Si
Sokal y Michener	$\frac{a+d}{a+b+c+d}$	$[0, 1]$	Si	Si (Si)	Si
Anderberg	$\frac{a}{a+2(b+c)}$	$[0, 1]$	Si	Si (Si)	Si
Rogers y Tanimoto	$\frac{a+d}{a+2(b+c)+d}$	$[0, 1]$	Si	Si (Si)	Si
Sneath y Sokal	$\frac{a+d}{a+\frac{1}{2}(b+c)+d}$	$[0, 1]$	No	Si (No)	No
Pearson	$\frac{ad-bc}{\sqrt{(a+b)(a+c)(b+d)(c+d)}}$	$[-1, 1]$	Si	Si (No)	Si
Yule	$\frac{ad-bc}{ad+bc}$	$[-1, 1]$	No	No (No)	No
Coseno de Salton	$\frac{\overrightarrow{X_1} \cdot \overrightarrow{X_2}}{\lVert\overrightarrow{X_1}\rVert\lVert\overrightarrow{X_2}\rVert}$	$[0, 1]$	Si	Si (Si)	Si
1. $S \geq 0$ significa que la matriz de similaridades es (semi) definida positiva.					
2. $d_{ij} = 1 - s_{ij}$ (entre paréntesis).					
3. Ninguna de las distancias d_{ij} es euclídea.					
4. Para el coseno se consideran dos vectores en un espacio que posee un producto interior					

Tabla 3. *Coeficientes de Similaridad [53]*

2.3. Escalamiento Multidimensional

Es muy común en el campo del *data mining* encontrarnos con representaciones gráficas de los objetos que conforman un *dataset*, sobre el cual se desea encontrar patrones.

Los gráficos en dos dimensiones permiten mostrar las relaciones que existen entre los diferentes elementos, pero en la mayoría de los casos dichos elementos no se pueden representar en un espacio vectorial $\mathbb{R}^2$, ya que contienen más de dos atributos.

El escalamiento multidimensional MDS (*Multidimensional Scaling*) [52] permite transformar los datos aglutinando la información en un número menor de atributos (dimensiones) permitiendo de esta manera su representación gráfica si esta se limita a dos o tres, aunque en este caso se pierde el significado de las dimensiones. Por lo tanto, el MDS es una técnica que intenta modelar los datos de similitud o disimilitud como distancias en un espacio geométrico. Los datos de entrada pueden ser, por ejemplo: tablas de contingencia, puntuaciones de similitud entre objetos, las frecuencias de interacción de moléculas, o los índices de comercio entre diferentes países.

Si se cumple que las variables se han medido objetivamente, el escalamiento multidimensional podría ser utilizado como técnica de reducción de datos ya que permitirá calcular las distancias a partir de los datos multivariados. Existen otras técnicas multivariantes, como son el análisis factorial y el análisis *cluster*, que persiguen objetivos muy similares al MDS pero que tienen sus particularidades respectivas. Sin embargo, la utilización de alguna de estas técnicas no supone que no se pueda utilizar el escalamiento multidimensional, sino que esta última técnica puede servir como alternativa o bien como complemento a las otras técnicas multivariantes [52].

De forma genérica el escalamiento multidimensional toma como entrada una matriz de distancias (o disimilaridades), DM ó $(1 - S)$, y entrega como salida una matriz $X \in M_{nxm}$, donde n, representa al número de instancias, mientras que m es el número que determina la cantidad de dimensiones. En este caso, cada valor X_{ij} es la coordenada de la instancia i en la dimensión j.

$$X_{ij} = \begin{bmatrix} X_{11} & X_{12} & \dots & X_{1m} \\ X_{21} & X_{22} & \dots & X_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ X_{n1} & X_{n2} & \cdots & X_{nm} \end{bmatrix}$$

Dada la matriz X, se puede calcular la distancia existente entre dos instancias i y j aplicando la fórmula general de alguna distancia como, por ejemplo, la Euclídea o *Minkowski*. Con estas distancias se puede obtener una nueva matriz de distancias, denotada por $D' \in M_{nxn}$:

$$D'_{ij} = \begin{bmatrix} d_{11}' & d_{12}' & \dots & d_{1n}' \\ d_{21}' & d_{22}' & \dots & d_{2n}' \\ \vdots & \vdots & \ddots & \vdots \\ d_{n1}' & d_{n2}' & \cdots & d_{nn}' \end{bmatrix}$$

La solución proporcionada por el escalamiento multidimensional debe ofrecer la máxima correspondencia entre la matriz de distancias inicial DM y la nueva matriz de distancias D'. Para que esto ocurra el MDS debe cumplir con alguna medida que permita determinar la bondad del modelo.

Una de las medidas de bondad más utilizadas considerando que $d_{ij}' \approx f(d_{ij})$ es el *Stress*, que *Kruskal* lo definió como:

$$Stress = \sqrt{\frac{\sum_{i,j}\left(f(d_{ij}) - d_{ij}'\right)^2}{\sum_{i,j} d_{ij}^2}}$$

El Stress no es propiamente una medida de la bondad del ajuste, sino su complemento. También se suele usar una modificación de esta última medida llamada *S-Stress* y el coeficiente de correlación al cuadrado (RSQ), como medidas de bondad.

Para cualquier modelo de escalamiento se debe cumplir que las distancias sean una función de las proximidades, es decir, $d_{ij}' = f(d_{ij})$. Existen dos modelos básicos de MDS:

2.3.1. Modelo de escalamiento métrico

En este primer modelo se considera que los datos están medidos en escala de razón o en escala de intervalo y se parte de la premisa que la relación entre las proximidades y las distancias es de tipo lineal:

$$d_{ij}' = a + b(d_{ij}).$$

La Figura 2, muestra un ejemplo de escalamiento multidimensional métrico.

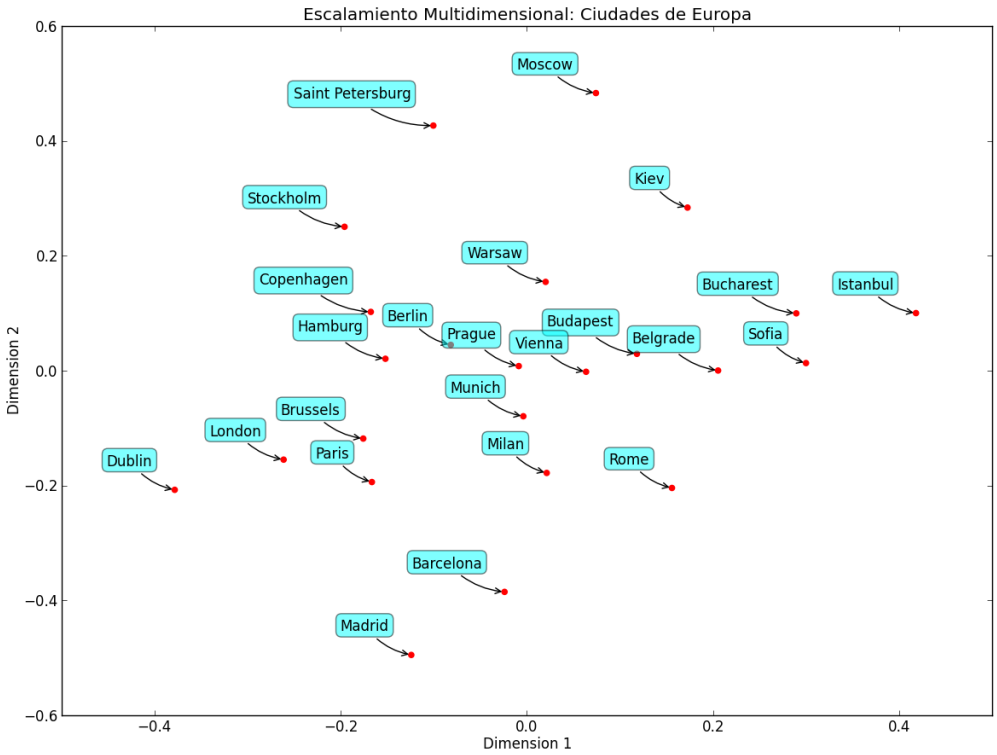

***Figura 2**. Ejemplo de Escalamiento Multidimensional Métrico*

2.3.2. Modelo de escalamiento no métrico

En el segundo consideramos que los datos están medidos en escala ordinal [52]. Aquí no se presupone una relación lineal entre las proximidades y las distancias, sino que establece una relación de monotonía creciente entre ambas, es decir que:

$$d_{ij} < d_{kl} \Rightarrow d_{ij}' \leq d_{kl}'$$

2.4. Procesamiento Natural del Lenguaje

Antes de ejecutar un proceso de *clustering* de documentos sobre un *dataset* en concreto, ciertamente es necesario el procesamiento de los corpus de dichos documentos por medio de técnicas de computación automatizadas, como el NLP.

El Procesamiento Natural del Lenguaje o NLP (*Natural Language Processing*) [58] por sus siglas en inglés, es el estudio del modelamiento matemático y computacional de varios aspectos del lenguaje y su desarrollo ha supuesto el uso de un amplio rango de técnicas y sistemas que permiten el análisis y manipulación del lenguaje natural. Dada la complejidad intrínseca que acompaña cualquier proceso de NLP, muchas de las tareas están relacionadas de manera interdisciplinaria con otras áreas como la inteligencia artificial, informática, lingüística, psicología y en general todo lo referente a las ciencias cognitivas.

El procesamiento natural del lenguaje tiene asociada una serie de tareas que permitirán posteriormente construir modelos para la recuperación de la información, así como para

el *clustering* de documentos. Por lo tanto es necesario, de manera previa, la determinación del vocabulario de términos que luego será procesado en un determinado modelo. Algunos de los procesos y terminologías inherentemente asociadas a estas tareas se describen a continuación [55]:

2.4.1. Tokenización

Dada una secuencia de caracteres y una unidad de documento definida, se entiende por tokenización al proceso de segmentación de una sentencia en unidades más simples denominadas *tokens*, al mismo tiempo se busca remover ciertos caracteres especiales y signos de puntuación. Aunque puede ser un proceso simple para textos escritos en algunas lenguas, especialmente en el caso de las románicas donde el separador de *tokens* es un espacio, en otras lenguas como las altaicas, extraer los *tokens* de una sentencia es un proceso mucho más complejo debido a su sintaxis y semántica. Un ejemplo de tokenización es:

Entrada:	En un lugar de la Mancha, de cuyo nombre no quiero acordarme											
Salida:	En	un	lugar	de	la	Mancha	de	cuyo	nombre	no	quiero	acordarme

Se debe entender al *token* como la unidad más simple de procesamiento y representa una secuencia de caracteres que generalmente se expresa como una palabra en el texto, mientras que la sentencia es la secuencia ordenada de los *tokens*. El corpus hace referencia al cuerpo del mensaje que se encuentra compuesto por un conjunto de sentencias.

2.4.2. Part-of-speech (POS)

Permite, dependiendo de la semántica del lenguaje, que cada *token* que compone una sentencia pueda ser clasificado como un verbo, un adjetivo, un pronombre, un artículo, etc. De esta manera se consigue identificar el significado de cada *token* y las partes clave de cada sentencia.

2.4.3. Stopwords

Llamadas también palabras vacías, son un conjunto de palabras con poco significado, debido a que son términos muy comunes y que aportan poco valor en la tarea de selección de coincidencias en documentos. Generalmente suelen ser eliminadas del vocabulario en un proceso de limpieza, pero esto depende mucho del contexto y del sistema de recuperación de información manejado. Dependiendo del idioma se suelen utilizar diferentes métodos como: la creación de listas fijas de palabras para cada idioma o la creación de listas de categorías léxicas como: preposiciones, artículos, conjunciones, pronombres, adverbios.

2.4.4. Lematización

Tiene por objetivo eliminar las variantes y/o inflexiones de los *tokens*, de manera que se almacene sólo el lema o forma base.

Tokens	**Reducción**
am	be
is	
are	

Para comprender lo anterior, es necesario saber que cada *token* tiene una morfología que determina la estructura interna de la palabra. La morfología de una palabra puede dar lugar a nuevas palabras a partir de su morfema base, por ello es importante diferenciar la morfología de la sintaxis de una palabra.

2.4.5. Stemming

Es un proceso que permite la reducción de los *tokens* a sus "raíces", eliminando de esta manera los sufijos. Ejemplo:

Tokens	Reducción
automate(s)	automat
automatic	
automation	

La complejidad de un proceso de *stemming* es alta, debido a las características propias de cada lenguaje. Ergo, en muchas de las ocasiones solamente se encuentran procesos automatizados y algoritmos desarrollados para un conjunto limitado de idiomas.

El algoritmo más común para realizar el proceso de *stemming* en el idioma inglés, y uno de los que ha demostrado de manera empírica ser muy efectivo, es el Algoritmo de *Porter*. En términos generales este algoritmo se puede resumir en varias fases secuenciales que permiten la reducción de las palabras a su raíz. Existen algunos otros *stemmers* como el de *Lovins* o el de *Paice/Husk*. [55].

A manera de resumen, en este análisis morfológico, tanto el *stemming* como la lematización buscan catalogar cada *token* de una sentencia para extraer sus "morfemas" y "raíces" para su posterior análisis.

2.5. Modelos de Recuperación de la Información

Debido a que este trabajo busca la agrupación de documentos con restricciones de tamaño, es menester conocer algunos de los modelos más utilizados para la recuperación de información (RI) que permitan la extracción de dicha información de ficheros o bases de datos de textos, de manera sistemática, para su posterior comparación. Algunas aproximaciones planteadas para la recuperación de la información se resumen en la Tabla 4.

Modelo	Representación	Función de Relevancia
Booleano	Conjuntos	Función Booleana
Conjuntos Difusos	Conjuntos Difusos	Función con Valores Asociados
Vectorial	Vectores	Coseno
Indexación Semántica Latente (LSI)	Matriz	Producto Punto
Redes Neuronales	Nodos	Nivel de Activación
Probabilístico	Conjunto de Probabilidades	Suma de Probabilidades
Modelo de Lenguaje	Conjunto de Probabilidades	Producto de Probabilidades
Redes de Inferencia	Nodos	Función de Probabilidad
Gravitational - Based Model (GBM)	Objetos	Fórmula de Gravedad
Latent Relation Discovery (LRD)	Vectores Extendidos	Coseno
Topic-Based Vector Space Model (TVMS)	Vectores	Producto Punto

Tabla 4. *Modelos de Recuperación de la Información [56]*

La RI puede definirse como un problema de selección, donde se busca extraer de una colección de documentos un subconjunto de objetos cuyo contenido es relevante para las necesidades de información, expresadas por un usuario en una consulta [56]. Las técnicas clásicas descansan en el siguiente supuesto: si un documento y una consulta (*query*) tienen una palabra en común, entonces el documento se refiere a la consulta, si el número de palabras en común aumenta, entonces la relación es mayor.

En el modelo booleano, el problema de RI se fundamenta en la teoría de conjuntos y el álgebra booleana. Debido a su simplicidad este modelo fue adoptado por los primeros sistemas bibliográficos, pero presenta algunos inconvenientes ya que su estrategia de recuperación está basada en un criterio de decisión binario, donde el documento es relevante o no. Las expresiones booleanas tienen una semántica precisa y limitada, y no siempre el usuario puede traducir sus necesidades de información en operadores booleanos [56].

El problema de RI se ve desde la perspectiva del algebra lineal cuando se usa el modelo vectorial, que considera similitud parcial entre consultas y documentos asignando pesos (no binarios) a los términos. Se suele considerar a esta representación vectorial como el modelo estándar en RI [55] para:

- Puntuación de documentos sobre una consulta.
- Clasificación de documentos.
- *Clustering* de documentos.

Ya que el modelo vectorial no sólo sirve para la consulta de documentos sino que también extiende la posibilidad de comparación de similitudes entre documentos, que es uno de los objetivos de este estudio, es importante conocer su funcionamiento.

2.5.1. Modelo vectorial

También se lo conoce como modelo de espacio vectorial, y es uno de los modelos más interesantes y ampliamente usados. Actualmente existen múltiples estudios como [25][26][59], en los que se detallan mejoras significativas con variaciones de este modelo. Este acercamiento representa los documentos como vectores de términos, donde cada término es una dimensión, y el valor de cada dimensión tiene asociado un peso que representa la relevancia del término T_q en el documento D_p . Esta representación vectorial no considera el orden de las palabras en un documento y se la conoce con el nombre de Bolsa de Palabras (*Bag-of-Words*) [59].

De manera general la bolsa de palabras suele ser escrita en un modelo matricial W_{pxq}, donde cada fila D_p representa un documento y cada T_q columna es un término, de tal manera que cada elemento w_{pq}, cumple $\{\forall\, w_{pq}\} \in \mathbb{N}^+$ [44].

$$W_{pxq} = \begin{array}{c} \\ D_1 \\ D_2 \\ \vdots \\ D_p \end{array} \begin{array}{c} \begin{array}{cccccc} T_1 & T_2 & T_3 & T_4 & \dots & T_q \end{array} \\ \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & \dots & w_{1q} \\ w_{21} & w_{22} & w_{23} & w_{24} & \dots & w_{2q} \\ \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ w_{p1} & w_{p2} & w_{p3} & w_{p4} & \dots & w_{pq} \end{bmatrix} \end{array}$$

La matriz de bolsa de palabras, también conocida como DTM–*Document Term Matrix* o TFM–*Term Frequency Matrix* suele ser bastante dispersa cuando existen grandes colecciones de documentos o cuando la cantidad de palabras que se encuentren en el corpus de los documentos es muy amplia. La mayoría de los documentos suele utilizar un subconjunto muy pequeño de las palabras utilizadas en todo el corpus, por lo tanto, la matriz resultante tendrá muchos valores que son ceros (por lo general más del 99% de ellos) [55]. Para solucionar, esto en ocasiones se usa el índice invertido, que almacena

solamente los valores diferentes a cero, es decir, el conteo de cada palabra en que documentos está.

Para modelar el aporte de información de un término en un documento, se define la medida TF (*Term Frequency*) [55] que hace referencia a la frecuencia del término T_q en el documento D_p. Esta medida debe ser normalizada por la frecuencia máxima de T_q en el documento, para que la longitud de los documentos no afecte a la relevancia del término:

$$TF_{T,D} = \frac{f_{T,D}}{max(f_{T,D})}$$

Por otra parte, es necesario cuantificar si un término que aparece en muy pocos documentos de la colección aporta más o menos información que uno que aparece varias veces. Para esto se hace uso de la frecuencia inversa de documento, que se define como:

$$IDF_T = log\left(\frac{n}{DF_T}\right)$$

Donde n es el número total de documentos en la colección y DF es la frecuencia de documento de T_q, es decir, el número de documentos de la colección que contienen T_q.

El $TF - IDF$, es un esquema de pesado que asigna al término T_q un peso en el documento D_p, y se puede expresar matemáticamente como [55]:

$$TF - IDF = TF_{T,D}\ x\ IDF_T$$

En otras palabras, $TF - IDF$ asigna al término T un peso en el documento D que es:

- Más alto, cuando T aparece muchas veces dentro de un pequeño número de documentos (dando así la máxima capacidad de discriminación para esos documentos).
- Más bajo, cuando el término T aparece pocas veces en un documento o se presenta en muchos documentos (ofreciendo así una señal de relevancia menos pronunciada).
- El más bajo cuando el término aparece prácticamente en todos los documentos.
- Cero cuando el término no aparece en el documento.

Una vez representados, los documentos y consultas como vectores, podemos medir su similitud. Una alternativa sería usar la distancia euclidiana, pero la variabilidad del largo entre documentos afectaría a la métrica. Lo más frecuente es usar el coseno del ángulo entre los vectores como medida de similitud (coseno de *Salton*). Si los documentos son iguales, el ángulo vale 0 y por lo tanto el coseno 1. Mientras que si los dos vectores (documentos) son ortogonales el coseno vale 0.

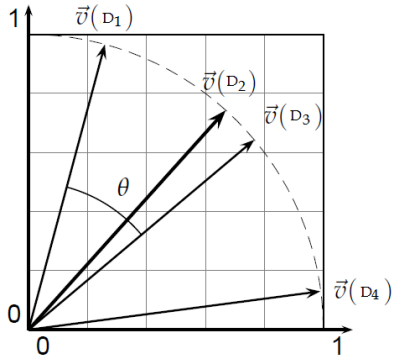

Figura 3. *Similaridad Coseno, cos(θ) = sim(D_1, D_3) [55]*

Los vectores, deben ser normalizados por su norma euclidiana $\sqrt{\sum_{i=1}^{n} \vec{v}_i^2(D)}$. De esta manera, y particularizando la fórmula de la Tabla 3, la similitud coseno para dos documentos estaría expresada por:

$$sim(D_1, D_2) = \cos(\theta) = \frac{\vec{v}(D_1) \cdot \vec{v}(D_2)}{\|\vec{v}(D_1)\| \|\vec{v}(D_2)\|}$$

2.6. Técnicas de Optimización

Se hace necesario una revisión de algunas de las técnicas de optimización existentes ya que un problema de *clustering* puede ser abordado como un problema de optimización [65], donde se busca maximizar o minimizar algún criterio, para mejorar la calidad de la agrupación. En particular, técnicas como la programación lineal, actualmente se han convertido en un recurso útil para resolver problemas de aprendizaje semi-supervisado. Como parte de la solución propuesta en este estudio "*Clustering de documentos con restricciones de tamaño*", en el Capítulo 5, se ha incorporado este tipo de técnicas como una medida efectiva para restringir los tamaños de los grupos.

El objetivo de la optimización, de manera global, es encontrar la mejor solución de modelos de decisiones difíciles frente a un conjunto de múltiples soluciones locales [29]. Desde el punto de vista más particular, un problema de optimización consiste en maximizar o minimizar alguna característica de sistema, eligiendo de manera sistemáticamente valores de entrada, dado un dominio definido, y calculando el valor de la función.

Existen diversas clasificaciones de las técnicas de optimización entre las cuales se puede mencionar las sumariadas por Singiresu Rao [27], en la Tabla 5.

<table>
<tr><th>Técnicas de Optimización Matemática</th><th>Técnicas de Procesos Estocásticos</th><th>Métodos Estadísticos</th></tr>
<tr><td>Métodos de cálculo</td><td rowspan="3">Teoría de decisión estadística</td><td rowspan="3">Análisis de regresión</td></tr>
<tr><td>Cálculos de variaciones</td></tr>
<tr><td>Programación lineal</td></tr>
<tr><td>Programación no lineal</td><td rowspan="2">Procesos de Markov</td><td rowspan="5">Análisis de clusters, pattern recognition</td></tr>
<tr><td>Programación geométrica</td></tr>
<tr><td>Programación cuadrática</td><td rowspan="2">Queueing theory</td></tr>
<tr><td>Programación dinámica</td></tr>
<tr><td>Programación de variables enteras</td><td rowspan="2">Métodos de simulación</td></tr>
<tr><td>Programación estocástica</td><td rowspan="5">Experimentos de diseño</td></tr>
<tr><td>Programación separable</td><td rowspan="4">Teoría de fiabilidad</td></tr>
<tr><td>Programación de funciones multiobjetivo</td></tr>
<tr><td>Métodos de Redes: PERT y CPM</td></tr>
<tr><td>Teoría de juegos</td><td rowspan="4">Análisis discriminante</td></tr>
<tr><td>Simulated annealing</td><td rowspan="3">Teoría renovable</td></tr>
<tr><td>Algoritmos genéticos</td></tr>
<tr><td>Redes neuronales</td></tr>
</table>

Tabla 5. *Clasificación general de las Técnicas de Optimización [27]*

De acuerdo al tipo de problema de optimización que se aborda, una de las muchas taxonomías existentes se puede observar en la Figura 4, considerando a los problemas de optimización como exactos y aproximados. Los algoritmos exactos dan una solución

exacta al problema, como su nombre lo sugiere, mientras que los algoritmos aproximados pueden dar, o no, una solución exacta [28].

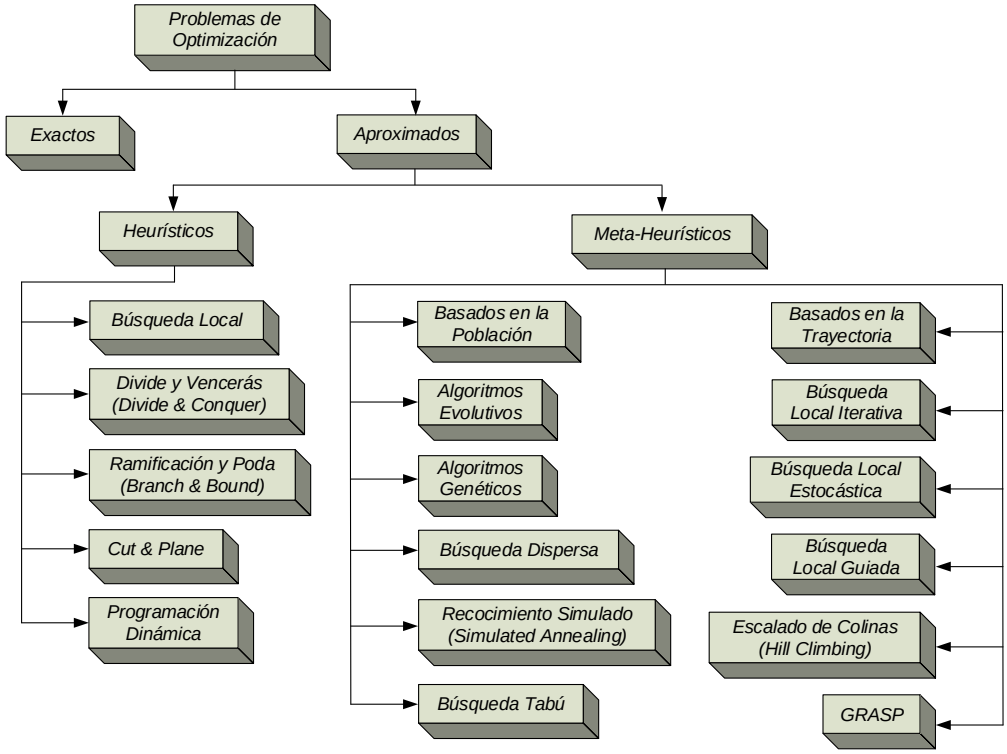

Figura 4. *Taxonomía de los Problemas de Optimización [28]*

La programación lineal por su parte, se utiliza en campos como la ingeniería, la economía, la gestión, y muchas otras áreas de la ciencia, la técnica y la industria [67] y es considerada un desarrollo revolucionario que permite tomar decisiones óptimas en situaciones complejas [27].

Muchos de los problemas de optimización combinatoria se pueden formular como problemas de programación lineal entera mixta. Por ejemplo, los métodos *Branch & Cut*, son algoritmos exactos que consisten en una combinación del método *Cut & Plane* con el algoritmo *Branch & Bound* [66].

2.6.1. Programación lineal (PL):

Estudios recientes en el campo del *machine learning* han empezado a incorporar técnicas de programación lineal, como una solución viable, para la solución de problemas de agrupamiento especialmente cuando estos problemas incorporan restricciones de alguna índole.

La programación lineal [27] fue reconocida por primera vez como un problema de optimización en el año 1930 por un grupo de economistas que desarrollaban métodos para la asignación óptima de recursos y fue usada durante la Segunda Guerra Mundial por la fuerza aérea de los EE.UU con la finalidad de buscar procedimientos más eficaces para la distribución de los recursos armamentísticos.

La programación lineal tiene como hipótesis la perfecta divisibilidad y tiene por objetivo [67] optimizar (minimizar o maximizar) una función lineal de n variables $x = \{x_1, x_2, \dots, x_n\}$, denominada función objetivo o función de coste, sujeta a restricciones lineales expresadas por un sistema de ecuaciones o inecuaciones. Más formalmente, se dice que un problema de programación lineal consiste en encontrar el óptimo (máximo o mínimo) de una función lineal en un conjunto que puede expresarse como la intersección de un número finito de hiperplanos y semiespacios en $\mathbb{R}^n$.

En cualquier problema de programación lineal es necesario la identificación de los siguientes elementos [29]:

1. El conjunto de datos.
2. El conjunto de variables involucradas en el problema, junto con sus dominios respectivos de definición. Atendiendo al tipo de variables, los problemas de programación lineal pueden ser clasificados como:

   Enteros puros (PLE): son aquellos en que todas las variables únicamente pueden tomar valores enteros. Se distinguen dentro de estos los problemas totalmente enteros como aquellos en que tanto las variables como todos los coeficientes que intervienen en el problema han de ser enteros.

   Mixtos (PLE): son aquellos en los que hay al mismo tiempo variables continuas y variables que sólo pueden tomar valores enteros.

   Binarios (PLEB): las variables sólo pueden tomar los valores cero o uno (0/1).

   Aunque en un principio pueda parecer que los problemas lineales enteros son más fáciles de resolver que los continuos, dado que el número de soluciones factibles a analizar cuando el conjunto de oportunidades está acotado, es finito, éste número suele ser lo suficientemente grande (en un problema binario con n variables el número de soluciones factibles a estudiar es 2^n) como para que resulte imposible su comparación.

   Mientras que bajo el criterio del tipo de problema, los problemas de programación lineal pueden ser catalogados como:

   Directo: Si el problema de decisión involucra variables enteras.

   Codificado: Cuando se trata de un problema que contiene además de aspectos cuantitativos, alguna consideración de tipo cualitativos, y por ello para tratar este tipo de aspectos se requiere el uso de variables enteras o binarias.

   Transformado: Cuando el problema no incluye variables enteras, pero para ser tratado analíticamente requiere el uso de variable enteras "artificiales".

3. La función lineal que debe ser optimizada (minimizada o maximizada) y puede ser expresada bajo la siguiente expresión general:

$$Z = f(x) = \sum_{j=1}^{n} c_j x_j$$

   Donde:

   c_j: son los coeficientes de las variables independientes,

   x_j: son las variables independientes.

4. El conjunto de restricciones lineales del problema que definen el conjunto de soluciones admisibles. Donde $\{\forall\, p, q, m\} \in \mathbb{Z}^+ \;/\; 1 \leq p \leq q \leq m$, se cumple que:

$$\sum_{j=1}^{n} a_{ij} x_j = b_i \quad ; \quad i = 1,2, \dots, p-1$$

$$\sum_{j=1}^{n} a_{ij}x_j \geq b_i \quad ; \quad i = p, \dots, q-1$$

$$\sum_{j=1}^{n} a_{ij}x_j \leq b_i \quad ; \quad i = q, \dots, m$$

Siendo:

a_{ij}, b_i: coeficientes conocidos de las restricciones asociadas a la variable x_j.

De todo lo expuesto anteriormente, se desprenden dos definiciones:

- **Solución factible:** Es un punto $x = \{x_1, x_2, \dots, x_n\}$ que satisfice todas las restricciones de igualdad y desigualdad (apartado 4). El conjunto de todas esas soluciones es la región de factibilidad.
- **Solución óptima:** Un punto factible x" tal que $f(x) \geq f(x")$, para cualquier otro punto factible x, se denomina una solución óptima del problema.

2.6.1.1. Problemas de programación lineal en forma estándar

Debido a que un modelo lineal puede adoptar varias formas: la función objetivo puede ser de máximo o mínimo, y las restricciones pueden ser de mayor o igual, menor o igual o de signo igual, incluso tener restricciones de varios tipos [29], la solución y el análisis de un modelo de programación ideal deberán formularse según una forma determinada de programa lineal y expresarse en forma canónica. Para describir un PPL en forma estándar son necesarios tres elementos [67]:

1. Un vector $c \in \mathbb{R}^n$
2. Un vector no negativo $b \in \mathbb{R}^m$
3. Una matriz mxn, A.

Con estos elementos, el problema lineal asociado y en forma estándar tiene la siguiente forma. Minimizar:

$$Z = c^T x$$

Sujeto a:

$$Ax = b$$

$$x \geq 0$$

Donde $c^T x$ indica el producto escalar de los vectores c y x, Ax es el producto de la matriz A, y el vector x, y $x \geq 0$ hace que todas las componentes de los vectores factibles sean no negativas. Típicamente, n es mucho mayor que m.

Cualquier problema de programación lineal puede expresarse siempre en forma estándar mediante manipulaciones algebraicas, así:

- Las variables no restringidas en signo se pueden expresar como diferencias de variables que si están restringidas en signo, es decir variables no negativas. Si algunas (o todas) de las variables no están restringidas en signo, éstas se pueden expresar mediante sus partes positiva y negativa.
- Las restricciones de desigualdad pueden convertirse en restricciones equivalentes de igualdad introduciendo nuevas variables que se denominan variables de holgura.

- Un problema de maximización es equivalente a uno de minimización sin más que cambiar el signo de la función objetivo.
- Una restricción con término independiente b no positivo puede reemplazarse por otra equivalente cuyo término independiente es no negativo.

2.6.1.2. Métodos de solución de programación lineal

Una vez que se ha conseguido representar una situación mediante un modelo lineal, es preciso encontrar y explotar la solución de ese modelo. Existen varias formas de encontrar esta solución [29]:

- En el caso (poco frecuente) que el modelo se haya representado con dos variables de decisión, éste se puede resolver gráficamente. La solución gráfica de un modelo lineal tiene fundamentalmente un interés pedagógico, dado que permite introducir diversos conceptos asociados a los modelos lineales de forma gráfica e intuitiva.
- Para modelos pequeños o medianos (hasta decenas de miles de variables y restricciones) resulta adecuado el algoritmo simplex, que busca el conjunto de soluciones posibles, de manera que se alcance el óptimo explorando únicamente un subconjunto pequeño de éstas.
- Para modelos de gran tamaño, el procedimiento del punto interior, técnica exportada de la programación no lineal, permite en ocasiones obtener una excelente aproximación a la solución de forma más rápida que el algoritmo simplex.

Aunque varios métodos adicionales se han desarrollado durante los años para solucionar problemas PL, el método simplex sigue siendo el método más eficiente y popular para la solución de problemas generales. Por ejemplo, el método de *Karmarkar*, desarrollado en 1984, ha demostrado que es hasta 50 veces más rápido que el algoritmo simplex de *Dantzig* [27].

3. Clustering

Resumen: A lo largo de este capítulo se describirá al proceso de *clustering*, sus características principales, los algoritmos asociados a este tipo de aprendizaje no supervisado y los mecanismos de validación para sus resultados. Además se revisan algunas de las particularidades asociadas al *clustering* de documentos.

3.1. Introducción al Proceso de Clustering

Como se ha mencionado el *clustering* es un tipo de aprendizaje no supervisado. Específicamente, existe un algoritmo de *clustering* que tiene vital importancia para este trabajo ya que será el que permitirá caracterizar a los documentos y agruparlos dentro de sesiones temáticas, y es el *K-Means* (K-Medias) con una de sus variaciones llamada *K-Medoids*.

Dado un conjunto de puntos (elementos u objetos) $x = \{x_1, x_2, \dots, x_n\}$ de tamaño n, un *cluster* c_j es un conjunto de puntos, que basados en una medida de proximidad, son similares entre sí. El *clustering* es un proceso que permite dividir un conjunto en k grupos $c = \{c_1, c_2, \dots, c_k\}$ de datos distintos, por medio de algún criterio de agrupamiento, como una función de coste o algún otro tipo de regla de asociación [16] [18] [61]. Al igual que en el aprendizaje supervisado en el *clustering* también se maneja el concepto de etiquetas (*labels* o *class labels*), pero las etiquetas se basan exclusivamente en la información, es decir, que son únicamente obtenidas de los datos. Así, cada grupo (clase), tiene una etiqueta que la identifica de los demás [1]. Para el análisis planteado los puntos estarán representados por documentos y lo que se busca es agruparlos por ciertas características que el algoritmo lo determinará.

De esta manera, se dice que el principio fundamental del *clustering* es garantizar que los grupos sean lo más heterogéneos entre sí, pero que los elementos del grupo sean lo más homogéneos posibles, basados en un criterio de optimización. En otras palabras, lo que se busca es minimizar la distancia/similaridad intra-*cluster* (cohesión) $mín\ d(x_1, x_2)$, y a la vez maximizar la distancia/similaridad inter-*cluster* (separación) $max\ d(c_1, c_3)$ [2]. La Figura 5 expresa de manera gráfica lo expuesto anteriormente.

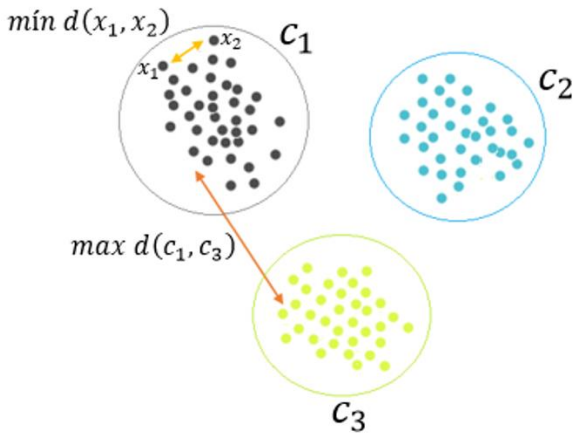

Figura 5. *Representación Gráfica de la Idea Fundamental del Clustering*

La elección de un algoritmo de agrupamiento no es algo trivial. Los factores a considerar en la elección de un algoritmo incluyen: la naturaleza de la aplicación, las características de los objetos a analizar, el número esperado y la forma de los grupos, y la complejidad del problema frente a la potencia de cálculo disponible. En algunos casos un algoritmo muy simple puede ser apropiado para hacer frente a un problema, pero en muchas situaciones diferentes se requiere de un algoritmo más complejo que tome en cuenta la mayor cantidad de parámetros que afectan su desempeño. De esta manera se busca tener una sinergia entre rendimiento y viabilidad, con la finalidad de obtener un alcance más adecuado para el trabajo de dicho algoritmo [4].

Actualmente resulta difícil imaginar la variedad de tareas computacionales existentes y el número de algoritmos desarrollados para resolverlos. A los algoritmos que dan una respuesta aproximada a la respuesta correcta o no ofrecen una solución para todas las instancias del problema se los denomina algoritmos heurísticos. Este grupo de algoritmos incluye un gran espectro de métodos basados en técnicas tradicionales y específicas [10].

Por lo general, los algoritmos heurísticos se utilizan para solucionar problemas que no se pueden resolver fácilmente. Las clases, en términos de complejidad computacional, se definen para distinguir los problemas de acuerdo a su "dureza".

- **Clase P (Polinomial):** se compone de todos aquellos problemas de decisión que pueden ser resueltos en una máquina de Turing determinista en un período de tiempo polinómico proporcional al tamaño de los datos de entrada. La máquina de Turing es una abstracción que se utiliza para formalizar la noción de algoritmo y complejidad computacional. Y se debe entender por tiempo polinómico a la posibilidad de encontrar soluciones “rápidamente”.
- **Clase NP (Polinomial No Determinística):** consiste en todos aquellos problemas cuya solución se la puede encontrar en un período de tiempo polinómico en una máquina de *Turing* no determinista. Ya que una máquina de este tipo no existe, en la práctica esto significa que un algoritmo exponencial puede ser descrito por un problema NP, aunque no se pueda asegurar que exista el algoritmo polinómico.

Existe una multitud de algoritmos de *clustering* propuestos y su clasificación depende de la literatura que se revise. Pero de manera genérica los algoritmos de *clustering* se pueden clasificar de acuerdo a [61]:

- El tipo de datos de entrada para el algoritmo.
- El criterio de agrupamiento que define la similitud entre los objetos.
- La teoría y los conceptos fundamentales en los que se basan sus técnicas de análisis (ej.: teoría difusa, estadísticas, etc.).

Así, de acuerdo con el método adoptado para definir las agrupaciones, los algoritmos se pueden clasificar en [61]: jerárquicos, particionales, basados en la densidad, basados en el *grid, etc*. En este trabajo se describirán con mayor profundidad los dos primeros ya que se utilizarán a posteriori.

3.2. Algoritmos de Clustering Jerárquicos

Los algoritmos de *clustering* jerárquicos tienen por objetivo agrupar *clusters* para formar un nuevo o bien separar alguno ya existente para dar origen a otros dos, de modo que si el proceso se efectúa de manera sucesiva, este proceso minimice o maximice alguna medida de similitud. Una de las características principales en la agrupación jerárquica es que no requiere que se especifique el número de *k clusters* [43].

El resultado del algoritmo es un árbol de *clusters*, llamado dendograma, que muestra cómo se relacionan los *clusters*. Cortando el dendograma a un nivel deseado (umbral), se obtiene una agrupación de los elementos en grupos disjuntos [61].

Los algoritmos de *clustering* jerárquicos, de acuerdo al método que usan para formar los *clusters*, pueden ser divididos en: divisivos o aglomerativos. Cada una de estas categorías presenta una gran diversidad de variantes [16].

3.2.1. Clustering jerárquico divisivo - DHC (*Divisive Hierarchical Clustering*)

Son también conocidos como descendentes (*top-down*) y construyen su jerarquía comenzando con un conglomerado que contiene todos los *n* objetos, y por medio de divisiones sucesivas, se van formando grupos cada vez más pequeños. En el estado final hay un total de *n clusters*, que contienen un solo objeto [43].

3.2.2. Clustering jerárquico aglomerativo - AHC (*Aglomerative Hierarchical Clustering*)

A este tipo de algoritmos se los conoce alternativamente como ascendentes (*bottom-up*), constituyen el proceso inverso a los métodos divisivos.

Comienzan el análisis con *n* grupos, formados por un solo objeto. A partir de estas unidades iniciales se van formando grupos, de forma ascendente, hasta que al final del proceso todos los objetos están agrupados en un mismo *cluster*.

De las dos aproximaciones, AHC es la más comúnmente utilizada en las aplicaciones ya que el proceso DHC es más complejo y más costoso en términos computacionales. Existen diferentes métodos para poder calcular las similaridades y distancias entre los pares de *clusters*, pero las más comunes son [16]:

3.2.2.1. Agrupación de enlace simple (single-link)

La distancia entre dos grupos se calcula como la distancia entre los dos elementos más cercanos en los dos grupos. Este también es conocido como *Nearest Point Algorthm (NPA)*.

$$d(c_i, c_j) = min\left(d(x_i, x_j)\right) \;;\; \{\forall x_i \in c_i, \forall x_j \in c_j\}$$

La Figura 6, muestra un dendograma trazado para un conjunto de 40 puntos, generados de manera aleatoria, bajo el método *Nearest Point Algorthm* y cortado a un umbral de 0.325 con la finalidad de obtener 7 grupos.

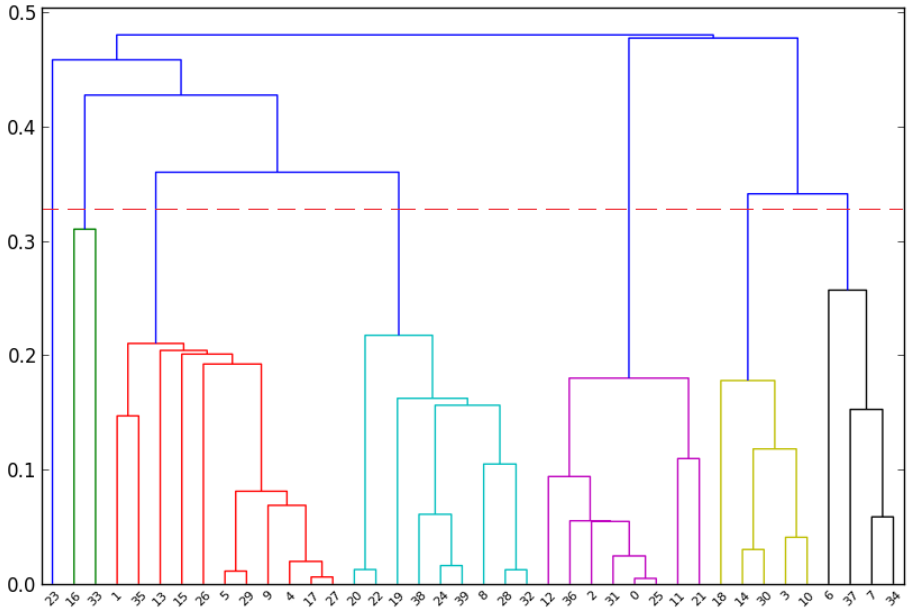

Figura 6. *Dendograma - Nearest Point Algorithm, k=7.*

3.2.2.2. Agrupación de enlace completo (complete-link)

La distancia entre dos grupos es calculada como la distancia entre los elementos más lejanos en los dos grupos. Otro nombre que suele adoptar es *Farthest Point Algorthm (FPA)* o Algoritmo de *Voor Hees*.

$$d(c_i, c_j) = max\left(d(x_i, x_j)\right) \;;\; \{\forall x_i \in c_i, \forall x_j \in c_j\}$$

3.2.2.3. Agrupación de enlace promedio (average-link)

La distancia entre dos grupos se calcula como la distancia promedio entre los elementos del primer *cluster* y los elementos del segundo *cluster*. También se le conoce como Algoritmo UPGMA.

$$d(c_i, c_j) = \frac{1}{|c_i||c_j|}\sum_{\substack{x_i \in c_i \\ x_j \in c_j}} d(x_i, x_j)$$

Además de los antes mencionados, existen otros métodos para poder calcular las similaridades basados en el: centroide, centroide ponderado, mediana, método de *Ward*, etc. [43].

AHC incluye muchos otros enfoques más allá de los descritos anteriormente. Algunas de estas alternativas incluyen modelos híbridos, como el BIRCH (*Balanced Iterative Reducing and Clustering using Hierarchies*). O algoritmos como el CURE (*Clustering Using REpresentatives*), ROCK (*RObust Clustering using linKs*), CHAMALEON y el SOTM [16].

3.3. Algoritmos de Clustering Particionales

Un algoritmo de agrupamiento particional, también conocido como "*flat clustering*" [16], obtiene una partición única de los datos en vez de una estructura de agrupación, como lo hacía el dendograma producido por una técnica jerárquica. Una de las ventajas de los métodos particionales sobre los métodos jerárquicos se da en aplicaciones que implican el manejo de grandes conjuntos de datos donde la construcción de un dendograma es computacionalmente prohibitivo [1].

Los métodos particionales dividen a los datos en k conjuntos disjuntos de manera simultánea y como resultado de eso producen una clasificación en k clases sin relación entre ellas. Las técnicas particionales suelen producir *clusters* optimizando alguna función objetivo (criterio) definida en forma local (usando parte de los patrones) o global (usando la totalidad de los datos). El criterio más comúnmente usado es el error cuadrático [60] y uno de los problemas asociados al uso de un algoritmo particional es la elección del k número de grupos y puntos iniciales [1].

La búsqueda combinatoria del conjunto de posibles clases para el valor óptimo de la función objetivo es computacionalmente prohibitiva. Por esta razón, en la práctica, el algoritmo se ejecuta típicamente múltiples veces con diferentes estados iniciales, y la mejor configuración obtenida de todas las ejecuciones es la que se utiliza como *clustering* de salida [1], es decir que la optimización de la función objetivo se realiza mediante un proceso iterativo [61]. Algunos de los algoritmos particionales más comunes se describen a continuación.

3.3.1. Algoritmo K – Means

Este algoritmo ha sido redescubierto varias veces en la literatura asociada al *clustering* y estadística bajo diversos nombres. El término *K-Means* fue utilizado por primera vez por *MacQueen* en 1967, aunque la idea se remonta a la propuesta de *Steinhaus* en 1956. El algoritmo fue redescubierto de forma independiente por *Lloyd* en 1957 como una técnica de cuantificación para la modulación por impulsos codificados, pero no se publicó hasta mucho más tarde, en 1982. En consecuencia, en el campo de la compresión de datos, este algoritmo a menudo se lo denomina como algoritmo de *Lloyd* o *Lloyd-Max*. En 1965, E. W. Forgy publicó esencialmente el mismo método, por lo que en ocasiones se lo denomina como *Lloyd-Forgy* [12].

El algoritmo de *K-Means* propuesto por *MacQueen* es la versión del algoritmo más conocida y ampliamente usada, ya que guarda un equilibrio entre la facilidad de implementación, velocidad y eficacia [43]. Se lo denomina *K-Medias* porque representa a cada uno de los grupos por la media (o media ponderada) de sus puntos, denominado centroide [3]. De acuerdo a la literatura que se revise a estos puntos también se les suele llamar centros de gravedad, centro geométrico o puntos medios del *cluster* [14].

En el algoritmo *K-Means* es necesario que el usuario especifique, a priori, el número k de grupos que desea formar para que el proceso se inicie. Una vez iniciado el proceso se escogen k puntos iniciales en todo el conjunto de datos $x = \{x_1, x_2, \dots, x_n\}$ y se calcula la distancia que hay entre los centroides (puntos iniciales) $u = \{u_1, u_2, \dots, u_k\}$ y el resto de puntos, luego se resignan a los puntos que estén más próximos. Una vez que se concluye esta reasignación, se vuelven a calcular los centroides de cada uno de los *clusters* $c = \{c_1, c_2, \dots, c_k\}$ y se repite el proceso de reasignación hasta que no se presente un nuevo cambio de los k centroides [3].

En términos generales, se puede decir que el objetivo del algoritmo *K-Means* es reducir al mínimo el cálculo de la distancia entre los puntos de cada grupo y su centroide. Para ello se utiliza la siguiente función objetivo, que minimiza la suma de los cuadrados *SSE* entre cada patrón y el centroide de su *cluster* [60]:

$$J_c = SSE = \sum_{i=1}^{k} \sum_{x \in c_i} \|x - u_i\|^2$$

Donde el centroide u_i del *cluster* c_i es:

$$u_i = \frac{1}{n_i} \sum_{x \in c_i}^{n} d(x_i, u_i)$$

Si bien el algoritmo realiza una minimización de la función objetivo, esto no garantiza que se alcance el mínimo global [16]. Se dice más bien que el *K-Means* converge a un mínimo local del vector de cuantificación del error, y por lo tanto debe ser reiniciado muchas veces, lo que convierte a esta tarea muy cara computacionalmente, especialmente, cuando se trata de grandes conjuntos de datos [6].

La calidad de la solución obtenida depende de la partición inicial de los datos, por lo que una heurística para la obtención de una mejor solución (aunque no una solución óptima) es repetir el algoritmo, con diferentes puntos iniciales y escoger la solución con el valor mínimo de la función objetivo. El pseudocódigo del algoritmo *K-Means* se describe a continuación [16]:

Algoritmo: K-Means

ENTRADA:

Conjunto de Datos x
Número de *Clusters* k

SALIDA:

Resultado del Agrupamiento Etiquetado

MÉTODO:

1. Inicializa los centroides de los *clusters* $u_1, \dots, u_k$, de manera aleatoria o con cualquier otro método
2. **Mientras** los *clusters* estén cambiando **Hacer**
3. Reasignación de los puntos
4. **Desde** $i = 1$ **Hasta** n **Hacer**
5. Asignación punto x_i al *cluster* cuyo centroide u_j este más cerca
6. **Fin**
7. **Fin**
8. Actualizar los centroides
9. **Desde** $j = 1$ **Hasta** k **Hacer**
10. n_j = número de puntos en c_j
11. $u_j = \frac{1}{n_j} \sum_{x_i \in c_j} x_i$
12. **Fin**

La Figura 7 muestra un ejemplo del proceso de agrupación con el uso del algoritmo *K-Means*, para formar dos agrupaciones de distinto tamaño, donde los puntos en rojo representan a los centroides.

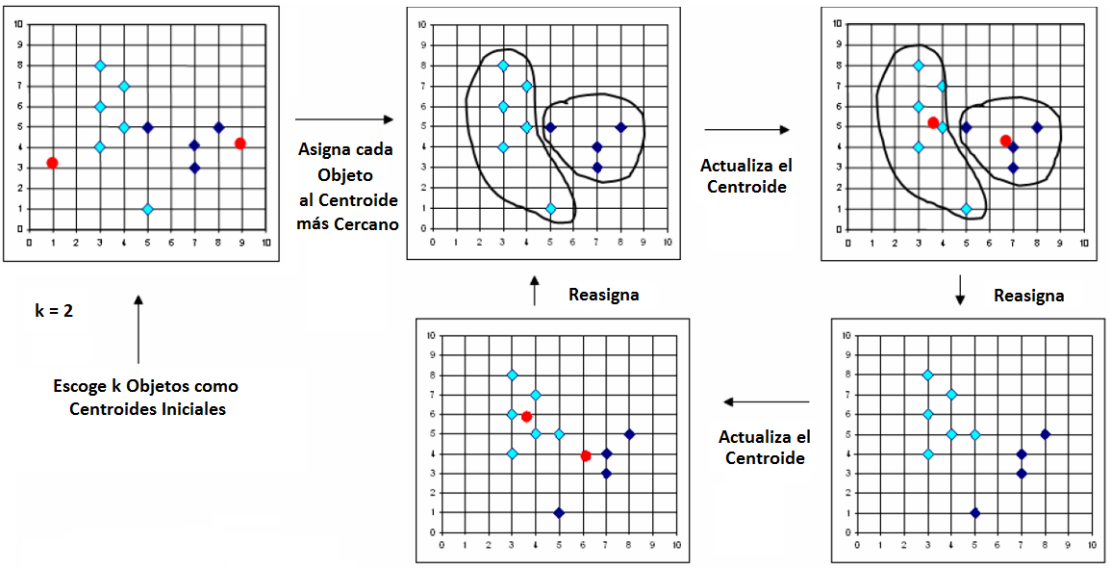

Figura 7. *Representación Visual del Clustering con el Algoritmo K-Means [8]*

El *dataset* Iris de *Fisher* [50] es un conjunto de datos bastante conocido en el campo de la minería de datos y es frecuentemente usado para tareas de *clustering* y clasificación. Este *dataset* fue construido a partir de los largos y anchos de pétalos y sépalos de 150 flores. Hay tres variedades de iris: setosa, versicolor y virgínica, en las que existe 50 iris de cada variedad. Este conjunto de datos sólo contiene dos racimos, con una separación obvia y clara. Uno de los racimos contiene iris setosa, mientras el otro contiene ambos iris virgínica e iris versicolor y no es separable.

La Figura 8, muestra la distribución de los objetos, en función del ancho del sépalo y el largo del pétalo, luego de un proceso de *clustering* con el algoritmo *K-Means* tradicional con tres grupos ($k = 3$) sobre el *dataset* Iris.

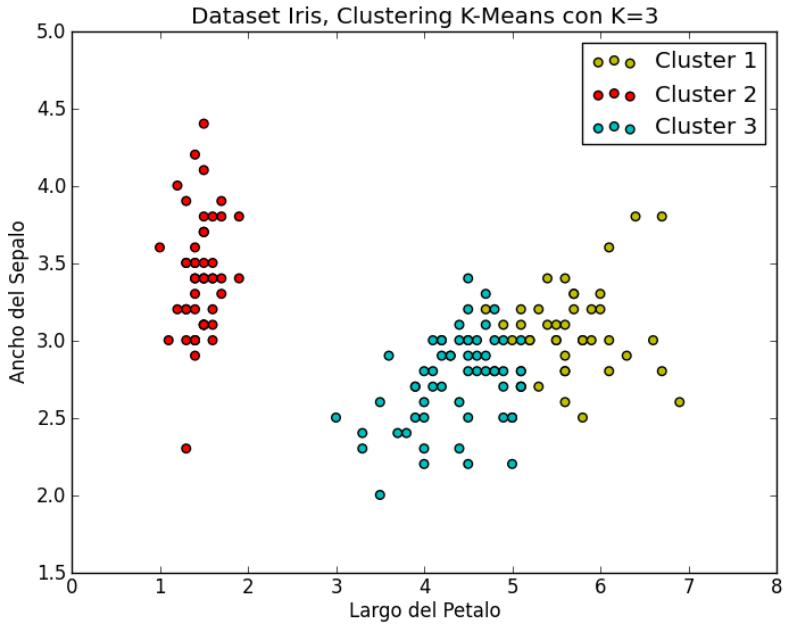

Figura 8. *Dataset Iris con Algoritmo K-Means, k=3*

Muchas variantes del algoritmo de *K-Means* se han reportado en la literatura con diferentes propósitos de estudio, algunas de ellas intentan seleccionar una buena agrupación inicial de modo que sea más probable que el algoritmo encuentre el mínimo global [1]. Una de estas variantes es el algoritmo *K-Medoids*.

3.3.2. Algoritmo K – Medoids (PAM - Partitioning Around Medoids)

Este algoritmo particional, fue propuesto en el año 1987 por *Kaufman* y *Rousseeuw* [62]. Se considera como una variación del *K-Means* ya que usa la misma función objetivo con la restricción de que los centros de los *clusters* deben pertenecer al *dataset*, su objetivo es determinar el mejor representante del centro de cada *cluster* (medoide) [60]. La función objetivo particularizada para el caso de la distancia entre un objeto x y su medoide u_i en un *cluster* c_i, es:

$$min \sum_{i=1}^{k} \sum_{x \in c_i} d(x, u_i)$$

Un medoide, de manera más formal, puede definirse como aquel objeto de un *cluster* cuyo promedio de disimilitud a todos los objetos en el *cluster* es mínimo [62].

En primera instancia el algoritmo PAM calcula k objetos representativos, llamados medoides, y luego cada punto que no es un centro se agrupa a su medoide más cercano. PAM intercambia los medoides con otros puntos candidatos, hasta obtener una configuración que minimice la función objetivo. El proceso continúa hasta que no se produzcan más intercambios de medoides [60].

Una de las mayores ventajas de este algoritmo es que puede tomar como entrada la matriz de disimilaridades. Estas disimilaridades, pueden ser por ejemplo, la evaluación subjetiva de relación entre objetos, en donde las medidas no son precisamente distancias. Esta importante condición permite que el algoritmo trabaje con diferentes métricas, que es algo muy común en el caso de *clustering* de documentos. Además, el algoritmo *K-Medoids* es menos sensible a la presencia de valores atípicos (*outliers*) [62].

Se considera que PAM es un algoritmo costoso en cuanto a la búsqueda de los medoides, ya que se compara un objeto con todo el conjunto de datos [61]. Por lo tanto, es computacionalmente ineficiente para valores grandes de n y k [2].

Uno de las principales desventajas de este algoritmo es que no escala bien para grandes conjuntos de datos, por esta razón, años más tarde se propusieron soluciones factibles para grandes *datasets* basadas en la misma idea de PAM en los algoritmos CLARA y CLARANS [61].

El pseudocódigo del algoritmo *K-Medoids* se describe a continuación [63]:

Algoritmo: K-Medoids

ENTRADA:
Conjunto de Datos x
Número de *Clusters* k

SALIDA:
Resultado del Agrupamiento Etiquetado

MÉTODO:

1. Inicializa los medoides de los *clusters* $u_1, \dots, u_k$, de manera aleatoria o con cualquier otro método
2. **Para** cada objeto no medoide x y el medoide u_i, calcular el costo de intercambiar $d(x, u_i)$
3. **Si** $d < 0$
4. u_1 es reemplazado por x y se asigna cada objeto al nuevo medoide
5. **Fin**
6. **Fin**
7. **Repetir** pasos 2, 3, 4, 5 y 6 hasta que los medoides no cambien

En [61], se resume de manera clara las principales características de algunos algoritmos de *clustering*. La Tabla 6, compendia algunas de las características más importantes de los algoritmos antes citados y se amplían con algunos otros más.

Nombre	Tipo de Dato	Complejidad*	Geometría	Resistencia al Ruido	Parámetros de Entrada	Resultados
Algoritmos Particionales						
K-Means	Numérico	$O(n)$	Formas No Convexas	No	Número de *Clusters*	Centroides de los *clusters*
K-Modas	Categórico	$O(n)$	Formas No Convexas	No	Número de *Clusters*	Modas de los *clusters*
PAM	Numérico	$O(k(n-k)^2)$	Formas No Convexas	No	Número de *Clusters*	Medoides de los *clusters*
CLARA	Numérico	$O(k(40+k)^2 + k(n-k))$	Formas No Convexas	No	Número de *Clusters*	Medoides de los *clusters*
CLARANS	Numérico	$O(kn^2)$	Formas No Convexas	No	Número de *Clusters*, Número Máximo de vecinos examinados	Medoides de los *clusters*
FCM	Numérico	$O(n)$	Formas No Convexas	No	Número de *Clusters*	Centroides de los *clusters*, Beliefs
Fuzzy C-Means	N/A	N/A	N/A	N/A	N/A	N/A
Algoritmos Jerárquicos						
BIRCH	Numérico	$O(n)$	Formas No Convexas	Si	Radios de los *Clusters*, Factor de Derivación	CF = (número de puntos en el *cluster* N, suma lineal de los puntos en el *cluster* LS, la suma cuadrada de los N datos SS) puntos
CURE	Numérico	$O(n^2 \log n)$	Formas Arbitrarias	Si	Número de *Clusters*, Número de *Clusters* Representativos	Asignación de los datos a los *clusters*
ROCK	Categórico	$O(n^2+nm_m m_a+ n^2 \log n)$, $O(n^2, nm_m m_a)$ Donde: m_m es el número máximo de vecinos para un punto y m_a es el número promedio de vecinos para un punto	Formas Arbitrarias	Si	Número de *Clusters*	Asignación de los datos a los *clusters*
Algoritmos Basados en la Densidad						
DBSCAN	Numérico	$O(n \log n)$	Formas Arbitrarias	Si	Radio del *Cluster*, Número Mínimo de Objetos	Asignación de los datos a los *Clusters*

DECLUE	Numérico	*O(n*log*n)*	Formas Arbitrarias	Si	Radio del *Cluster* σ, Número Mínimo de Objetos ε	Asignación de los datos a los *clusters*
Algoritmos Basados en el Grid						
Wave-Cluster	Datos Especiales	*O(n)*	Formas Arbitrarias	Si	*Wavelets*, el Número de *grid cells* para cada dimensión, el Número de Aplicación de la Transformada *Wavelet*	Objetos Agrupados
STING	Datos Especiales	*O(K)* *K* es el número de *grid cells* en el más bajo nivel	Formas Arbitrarias	Si	Número de Objetos en una *cell*	Objetos Agrupados
* ***n:*** número de puntos en el *dataset*. ***k:*** número de *clusters*.						

Tabla 6. *Características de los Tipos de Algoritmos de Clustering [61]*

3.4. Selección del Número de Grupos

Un problema fundamental en el análisis de *clusters* es determinar el número de grupos, que por lo general es una de las entradas de la mayoría de los algoritmos de agrupación particionales. Las soluciones del proceso de *clustering* varían significativamente dependiendo del número de grupos que se especifique. Por lo tanto, una técnica de agrupamiento adecuada debería recuperar la estructura del *cluster* subyacente, dada una buena estimación del verdadero número de *clusters* [43].

Una variedad de métodos se han propuesto para estimar el número de grupos. El libro "*Classification*" de A.D. Gordon [45], ha dividido estos métodos en dos categorías: métodos globales y métodos locales.

En los métodos globales [43], la calidad del *clustering* dado un número k específico de *clusters* se mide por un criterio y la estimación óptima de k denominada $k"$, se obtiene mediante la comparación de los valores del criterio calculado en un rango de valores de k. Pertenecen a esta categoría los métodos de: *Calinski* y *Harabasz's*, *Hartigan's*, *Krzanowski* y *Lai's*, *Gap*, *Jump*, *Elbow* o el método del coeficiente de silueta que se explicará en la Sección 3.6.1.2 de este trabajo.

El método de *Elbow* conocido también como método del "codo" [60], examina la curva generada por el índice de la suma de los cuadrados intra-*cluster* (*SSE*) en función del número de *clusters*. El valor de *SSE* siempre decrece a medida que k aumenta pero si en realidad hay $k"$ *clusters* en los datos entonces el decrecimiento será más lento a partir de $k" + 1$. Ese "codo" que se forma en el cambio de la pendiente de la curva indica la cantidad de *clusters* adecuada.

El método puede ser atribuido a los estudios de Robert L. Thorndike en 1953 [43]. Un ejemplo de este método aplicado al *dataset* Iris con el algoritmo *K-Means*, se muestra en la Figura 9.

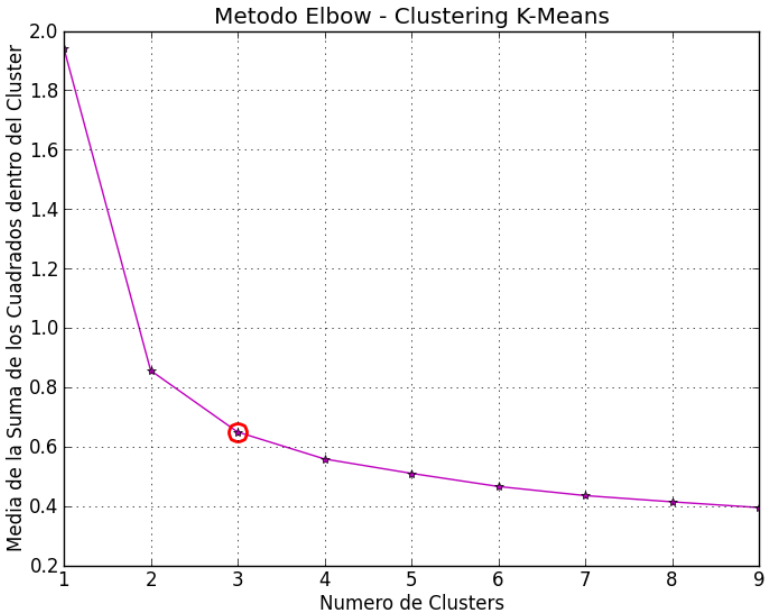

Figura 9. *Método Elbow del Dataset Iris con Algoritmo K-Means*

Los métodos locales están enfocados a probar la hipótesis de que un par de grupos deben ser fusionados, por lo que son adecuados para evaluar particiones que están anidadas jerárquicamente. El nivel significativo no debe interpretarse estrictamente ya que hay múltiples pruebas que están involucradas en el procedimiento. Ejemplos de métodos locales pueden ser los propuestos por *Duda* y *Hart* o el método de *Beale* [43].

Además de los antes mencionados existen muchos otros métodos adicionales, como: *Akaike Information Criterion* (AIC), *Bayesian Information Criterion* (BIC), *Deviance Information criterion* (DIC), métodos de validación cruzada o el análisis de la matriz *Kernel*.

3.5. Selección de los Puntos Iniciales

Uno de los inconvenientes de los métodos de agrupación que usan como criterio de entrada el número *k* de grupos, es la elección de los puntos iniciales, también llamados puntos semilla o atractores iniciales [49]. Los algoritmos de refinamiento iterativo como el *K-Means* (y algunas de sus variaciones) son muy sensibles a los *outliers* y a este parámetro de inicialización *k*, por lo que la elección incorrecta de estos puntos semilla será determinante en el resultado final de los conglomerados. Una elección bajo un criterio no apropiado determinará, a priori, un bajo rendimiento del algoritmo.

Existen muchas variantes, técnicas y algoritmos para este propósito. Algunas de los procedimientos que son más comúnmente utilizados para generar los puntos iniciales se describen a continuación.

3.5.1. Generación aleatoria

Suele aparecer como el método de facto en los algoritmos de *clustering* particionales. Esta técnica es la más rápida y simple, pero también la más arriesgada, ya que la elección de los *k* puntos representantes iniciales es al azar, lo que puede ser una mala representación de la colección de objetos. Lo más común en este tipo de procedimiento es ejecutar varias veces el algoritmo de *clustering* para distintas selecciones aleatorias y escoger el mejor resultado [49].

3.5.2. Técnica del vecino más lejano (Farthest Neighbor Technique)

Otra forma de colocar los atractores iniciales, es mediante el algoritmo del vecino más lejano, en donde se propugna que si se seleccionan los k patrones más alejados entre sí, entonces el *K-Means* (o *K-Medoids*) siempre converge hacia el óptimo global [68]. A esta técnica también se le conoce como el algoritmo de *González*, a quien se le atribuye formalmente su autoría, pero probablemente puede ser mucho más antiguo. El pseudocódigo del algoritmo de *González*, particularizado para el trabajo con una matriz de distancias/disimilaridades, se describe a continuación:

Algoritmo: González

ENTRADA:

Matriz de Disimilaridades S_{ij}

Número de *Clusters* k

SALIDA:

Puntos Iniciales $u = \{u_1, u_2, \dots, u_k\}$

MÉTODO:

1. $u_1 = argmax \sum_{i=1}^{n} s_{ij}$; $\forall j: 1, \dots, n$

2. $u_2 = argmax\ s_{iu_1}$; $i: 1, \dots, n$

3. **Desde** $i = 3$ **Hasta** k **Hacer**

4. $u_i = argmax \sum_{1}^{m-1} s_{iu_m}$; $i: 1, \dots, n$

5. **Fin**

En la Figura 10, se muestran los $k = 5$ puntos iniciales para un conjunto de 40 puntos, generados de manera aleatoria, bajo el método *Farthest Neighbor Technique* y en los que se ha trabajado con las distancias entre elementos procedentes de la matriz de disimilaridades.

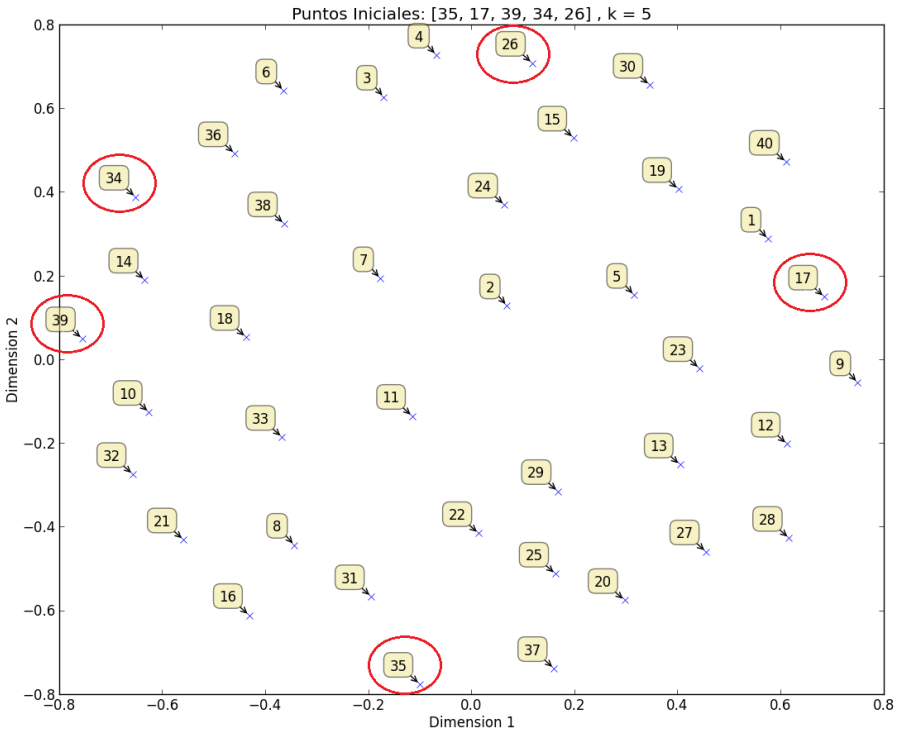

Figura 10. *Puntos Iniciales – Farthest Neighbor Technique, k=5.*

3.5.3. Algoritmo Buckshot

Buckshot es una técnica híbrida que combina un método de *clustering* jerárquico (AHC) con una técnica de *clustering* particional (*K-Means*/*K-Medoids*), en un intento de mezclar la alta calidad de los algoritmos jerárquicos con la eficiencia de los algoritmos particionales [48].

La idea del algoritmo de *Buckshot* es bastante simple [47]. Primero se traza un dendograma con el algoritmo AHC y se corta el dendograma en un umbral determinado con la finalidad de obtener los k *clusters*, que contengan a los k puntos iniciales, que servirán como entrada para el algoritmo de *clustering* particional. El algoritmo AHC trabaja únicamente con una muestra aleatoria de la totalidad de puntos de tamaño $\sqrt{kn}$, donde los k puntos iniciales serán los centros de los *clusters* encontrados. Este algoritmo claramente tiene una complejidad $O(kn)$.

Dado que se emplea un muestreo aleatorio, el algoritmo *Buckshot* es no determinístico. Esto quiere decir que si se ejecuta el algoritmo de manera repetida sobre el mismo corpus, este puede producir diferentes particiones. Sin embargo la experiencia ha demostrado, mediante varios ensayos heurísticos, que generalmente se producen particiones cualitativamente similares.

3.5.4. Técnica de fraccionamiento

La técnica de fraccionamiento utiliza una aplicación sucesiva de la subrutina de algún algoritmo de *clustering* (*K-Means*/*K-Medoids*), a través de grupos de tamaño fijo para encontrar los puntos iniciales.

El algoritmo primero encuentra k puntos iniciales mediante el fraccionamiento de n en n/m "*buckets*" de un tamaño fijo, donde se cumple que $m > k$. Se aplica alguna subrutina de *clustering* para cada uno de estos *buckets* por separado, con la finalidad de agrupar las instancias en *clusters*. Los centroides de los *clusters* generados, son recursivamente divididos dentro de *buckets* una vez más. La iteración termina sólo cuando se haya alcanzado k centroides iniciales. El fraccionamiento se puede ver como la construcción de la parte inferior de un árbol de $1/\rho$ ramificaciones hacia arriba, donde las hojas son instancias (o documentos) individuales y el procedimiento termina cuando solo existan k raíces [47].

El algoritmo de fraccionamiento ha demostrado ser un procedimiento más preciso que el algoritmo *Buckshot* para encontrar los puntos iniciales, pero este último es significativamente más rápido [47].

3.6. Validación del Agrupamiento

La evaluación del rendimiento de un algoritmo de *clustering* no es algo tan trivial como contar el número de errores o calcular el *precision* y el *recall* como se suele hacer en los algoritmos de clasificación supervisada. Históricamente, se han propuesto una serie de medidas para la evaluación de los resultados de la agrupación basadas en diferentes criterios [4].

En particular, cualquier métrica de evaluación no debe tomar en cuenta los valores absolutos de los *cluster labels* sino más bien si en la agrupación están bien definidas las separaciones de los datos similares con respecto a algún conjunto de clases tomado como referencia (*ground truth*). O satisfacer algún supuesto de tal manera que los miembros que pertenecen a la misma clase sean más similares que los miembros de clases diferentes, de acuerdo con alguna métrica de similitud.

3.6.1. Índices de validación internos

Los índices de validación internos utilizan sólo la distribución espacial de los puntos y las etiquetas de los *clusters* generados por el algoritmo, para el cálculo de propiedades en las agrupaciones resultantes, tales como: compactación, separación de las clases y redondez. Este enfoque no requiere información adicional acerca de los datos, por lo que se considera que es la forma más sencilla de evaluar un algoritmo de agrupación aplicado a un conjunto de datos [4].

3.6.1.1. Índice de Dunn

Este índice de validación es conceptualmente el más simple dentro de los índices de validación interna y compara el tamaño de los grupos con las distancia entre los grupos [4]. Fue propuesto por J.C. Dunn en el año 1974 con el objetivo de identificar un conjunto de *clusters* que sean compactos, con una varianza pequeña entre los miembros del *cluster*, y que éstos estén bien separados de los miembros de otros *clusters* [61].

Este índice está definido por la siguiente ecuación, para un número específico de *clusters*:

$$D_{dunn} = \min_{1\le i\le n}\left\{\min_{1\le j\le n, i\neq j}\left(\frac{d(c_i,c_j)}{\max_{1\le k\le n} diam(c_k)}\right)\right\}$$

Donde $d(c_i, c_j)$ es la función de disimilaridad entre dos *clusters* c_i y c_j definido como:

$$d(c_i,c_j)=\min_{x\in c_i, y\in c_j} d(x,y)$$

Y $diam(c_k)$ es el diámetro de un *cluster*, que puede ser considerada como una medida de dispersión de los grupos. El diámetro de un grupo c se puede definir de la siguiente manera:

$$diam(c_k) = \max_{x,y\in c} d(x,y)$$

Existen muchos métodos para calcular tanto $d(c_i, c_j)$ como $diam(c_k)$, y cada combinación de éstos define un valor del índice de Dunn diferente, pero sí es obligatorio usar la misma en ambos conceptos [4].

El índice de *Dunn* está definido para $D_{dunn} \in [0, \infty[$, y lo deseable es tener un valor lo más alto posible ya que esto indicará un mejor rendimiento del algoritmo de *clustering*. También suele ser usado como métrica para encontrar el número óptimo de *cluster*s en un conjunto de datos [61].

3.6.1.2. Índice de silueta

El índice de silueta es una métrica interna que permite evaluar el buen funcionamiento de los algoritmos de aprendizaje no supervisado. El objetivo de este índice es identificar el número óptimo de agrupamientos. Para obtener el valor de $S(i)$ sólo es necesario tener dos cosas: los grupos obtenidos por la aplicación de un algoritmo de *clustering* y la colección de todas las proximidades entre los objetos. Posteriormente, se tomará como referencia cualquier objeto i en el *dataset* y se calcularán los valores [5]:

- $a(i)$ que es la distancia media entre el objeto y todos los otros objetos de la misma clase y,
- $b(i)$ que es la distancia media entre el objeto y todos los otros objetos del *cluster* más cercano.

El valor de $S(i)$ puede ser obtenido mediante la combinación de los valores de $a(i)$ y $b(i)$:

$$S(i) = \begin{cases} 1 - \dfrac{a(i)}{b(i)} & ; \quad si\ a(i) < b(i) \\ 0 & ; \quad si\ a(i) = b(i) \\ \dfrac{b(i)}{a(i)} - 1 & ; \quad si\ a(i) > b(i) \end{cases}$$

De esta manera es posible expresar de forma genérica el valor del coeficiente de silueta bajo la siguiente ecuación:

$$S(i) = \frac{b(i) - a(i)}{max\{a(i), b(i)\}}$$

Cuando uno de los *clusters* tiene un único objeto $S(i) = 0$. El dominio de este índice fluctúa entre:

$$-1 \leq S(i) \leq 1$$

Un valor más alto de este índice indica un mejor rendimiento del agrupamiento, ya que se está garantizando que la distancia inter-*cluster* es pequeña y la distancia intra-*cluster* es grande. Este coeficiente suele generalmente ser más alto para grupos convexos, bien separados y con una densidad alta.

Este índice puede ser representado de manera gráfica, calculando el coeficiente de silueta para cada uno de los objetos del *dataset*. La silueta de cada *cluster* es graficada en orden decreciente para todos los objetos que conforman dicho agrupamiento. Con el fin de obtener una visión general, las siluetas de los diferentes grupos se imprimen una debajo de otra. De esta manera toda la agrupación se puede mostrar por medio de una sola gráfica, que permite distinguir las agrupaciones 'fuertes' de las 'débiles'.

Un ancho de la silueta amplio, indica valores altos de $S(i)$ y por lo tanto un *cluster* más compacto, mientras que la otra dimensión de la silueta, la altura, indica simplemente el número de objetos en un determinado *cluster*. Así, el diagrama de silueta muestra cuáles objetos se encuentran bien agrupados dentro de su *cluster*, y cuáles están colocados de manera forzada o artificial.

En la parte izquierda de la Figura 11, se muestran los diagramas de silueta luego de un proceso de *clustering* con el algoritmo *K-Medoids* tradicional, con dos y tres grupos ($k = 2$ y $k = 3$) sobre el *dataset* Iris. Se puede observar claramente la disminución del valor del índice de silueta promedio, desde 0.881 hasta 0.714, cuándo se aumenta el valor de k, como era de esperarse, ya que como se mencionó este *dataset* aunque tiene tres clases solo posee dos racimos. En la parte derecha de la Figura 11 se muestra el MDS del *dataset* y sus respectivos grupos.

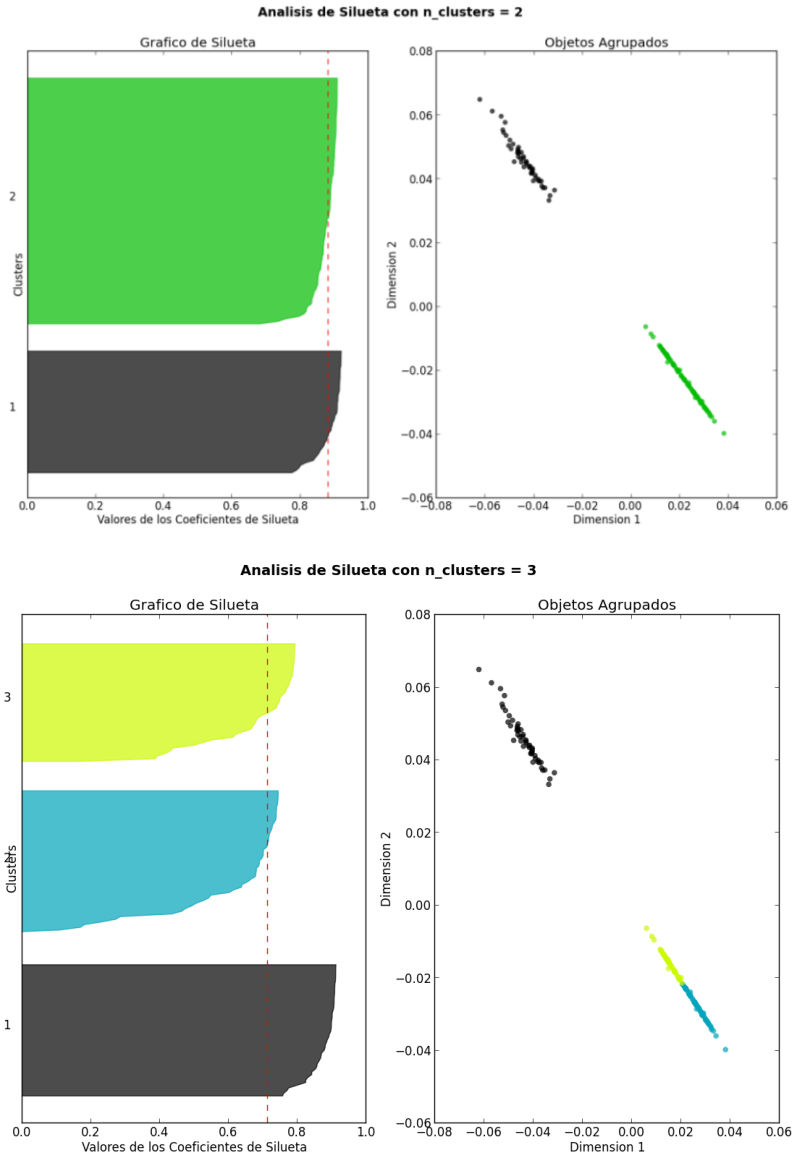

Figura 11. *Diagrama de Silueta del Dataset Iris, con Algoritmo K-Medoids, k=2 y k=3*

3.6.2. Índices de validación externos

El enfoque de validación externa, compara el agrupamiento generado por el algoritmo de *clustering* con otro que es considerado como el "mejor agrupamiento", por lo que de cierta manera, corresponde a una especie de medida de error, ya sea directa o indirectamente [4].

Para el caso de los índices externos en donde se comparan los resultados de los agrupamientos con una referencia de solución de *clustering* "ideal", resulta extremadamente importante evaluar la calidad de la agrupación. Sin embargo, no existe una medida estandarizada para poder comparar los *clusters* [41]. A continuación se presentan algunas de las medidas más utilizadas.

3.6.2.1. Adjusted Rand Index (ARI)

ARI [3] es un índice que mide las coincidencias entre las etiquetas de las clases calculadas por el algoritmo de agrupación y las etiquetas de clase del agrupamiento "ideal". *ARI* es un valor que oscila entre $-1 \leq ARI \leq 1$, y cuanto mayor sea, mayor será la semejanza entre los resultados de la agrupación y el *ground truth*. La ventaja de este índice es [23]

que no toma en cuenta a las permutaciones y existe la posibilidad de normalización de sus valores.

Sea A el *ground truth* de la asignación de las clases y B las clases predichas por el algoritmo de *clustering*, entonces [23]:

- a es el número de pares de elementos que están en el mismo conjunto en A y en el mismo conjunto en B.
- b es el número de pares de elementos que están diferentes conjuntos en A y en diferentes conjuntos en B.

El Índice Aleatorio RI, sin ajuste, entonces podrá ser expresado como:

$$RI = \frac{a + b}{A_2^n}$$

Donde A_2^n es el número total de pares posibles en el conjunto de datos, sin ordenar.

Sin embargo, el índice RI no garantiza que las asignaciones de etiquetas al azar alcanzarán un valor cercano a cero, especialmente si el número de grupos es del mismo orden de magnitud que el número de muestras.

Para contrarrestar este efecto, se puede descontar el RI esperado $E[RI]$ de las asignaciones etiquetadas al azar, para poder obtener un Índice Aleatorio Ajustado ARI:

$$ARI = \frac{RI - E[RI]}{max(RI) - E[RI]}$$

3.6.2.2. Normalized Mutual Information (NMI)

En teoría de la información, la Información Mutua MI (*Mutual Information)* o transinformación cuantifica el valor de la información compartida entre dos variables aleatorias, es decir, mide la reducción de la incertidumbre (entropía) de una variable aleatoria, A, debido al conocimiento del valor de otra variable aleatoria B [41].

Si se conoce *ground truth* de la asignación de las clases y las clases predichas por el algoritmo de *clustering*, se puede calcular la MI como una función que mide las coincidencias entre estos parámetros, sin tomar en cuenta las permutaciones.

$$MI(A, B) = H(A) + H(B) - H(A, B) = \sum_{i=1}^{c} \sum_{j=1}^{k} \frac{n_{ij}}{n} log \frac{n_{ij} n}{n_i n_j}$$

Donde $H(A)$ y $H(B)$ son las entropías de cada variable independiente, mientras que $H(A, B)$ es la entropía conjunta entre agrupamientos. La entropía para una agrupación se define como el valor esperado de la información contenida si esta es vista como una variable aleatoria.

Dos versiones normalizadas y muy conocidas de esta medida MI son: la NMI y la AMI. La Información Mutua Normalizada, NMI se utiliza de manera más frecuente en la literatura, mientras que la AMI fue propuesta más recientemente y se normaliza contra el azar.

El NMI mide [3] la cantidad de información estadística compartida por las dos variables aleatorias que representan a la asignación de *clusters* y a la etiqueta de clase subyacente. Suponiendo que la entrada n_{ij}denota la cantidad de elementos que pertenecen al grupo i y la clase j. Entonces, NMI se calcula como:

$$NMI(A,B) = \frac{MI(A,B)}{\sqrt{H(A)H(B)}} = \frac{\sum_{i=1}^{c}\sum_{j=1}^{k}\frac{n_{ij}}{n}log\frac{n_{ij}n}{n_i n_j}}{\sqrt{\left(\sum_{i=1}^{c}\frac{-n_i}{n}log\frac{n_i}{n}\right)\left(\sum_{j=1}^{k}\frac{-n_j}{n}log\frac{n_j}{n}\right)}}$$

Donde $n_i = \sum_{j=1}^{k} n_{ij}$, $n_j = \sum_{i=1}^{c} n_{ij}$, n, c, k denota el número total de objetos, el número de grupos y el número de clases, respectivamente. Basados en el conocimiento previo del número de clases, por lo general, se fija el número de grupos igual al número real de clases, es decir, $c = k$.

3.6.2.3. Adjusted Mutual Information (AMI)

La Información Mutua Ajustada, es el valor esperado de la información mutua y se puede calcular utilizando la siguiente ecuación, propuesta por Vinh et al. (2009) [41]:

$$AMI(A,B) = \frac{MI - E[MI]}{max\big(H(A),H(B)\big) - E[MI]} = \frac{MI - E[MI]}{\sqrt{H(A)H(B)} - E[MI]}$$

El valor de AMI se calcula usando una forma similar a la del índice ARI. El dominio de esta función es el mismo para todos los valores normalizados de MI y está definido entre [42]:

$$-1 \leq AMI \leq 1$$

3.6.3. Índices de validación relativos

Debido a que los índices de validación internos evalúan únicamente el resultado del agrupamiento efectuado por el algoritmo de *clustering*, estos no pueden percibir la estabilidad del algoritmo contra las variaciones de los datos o la consistencia de los resultados en caso de redundancia. Los índices de validación relativa son una familia de índices más complejos que los internos e intentan medir la coherencia de un algoritmo mediante la comparación de los grupos obtenidos, por el mismo algoritmo, en diferentes condiciones. Estos índices a menudo intentan explotar la redundancia en los datos y al igual que en los índices de validación interna tampoco requieren de información adicional al proceso de *clustering* [4].

Dentro de este grupo de índices de validación encontramos [4]: al índice FOM (*Figure of Merit*) donde para medir la consistencia se asume que la redundancia esta embebida en los datos de la muestra. Y al índice de estabilidad que mide la capacidad de un *dataset,* que ha sido agrupado previamente, de predecir la agrupación de otro conjunto de datos muestreados de la misma fuente. Esto último se lo hace dividiendo el *dataset* que se desea agrupar en dos partes. El algoritmo de agrupamiento a ser utilizado se aplica sobre la primera parte, y las etiquetas obtenidas sobre estos puntos se utilizan para entrenar un clasificador que divide a todo el espacio.

El índice de estabilidad depende del número de grupos, y por lo tanto necesita ser normalizado cuando se utiliza para el modelo de selección. Otro problema que afecta el índice de estabilidad es la selección de la regla de clasificación, que puede influir fuertemente en los resultados [4].

3.6.4. Análisis gráfico de clusters mediante la matriz de distancias

La matriz de distancias/disimilaridades puede ser representada en un espacio vectorial $\mathbb{R}^2$, donde la diagonal de la misma expresará la comparación entre los objetos $x_{ij} = x_{ji}$. Aunque per se, el análisis gráfico de una matriz de distancias no es un método para la validación del agrupamiento, si permite de manera eficaz visualizar los resultados del proceso de *clustering*.

En la Figura 12 se muestra la representación gráfica de una matriz de disimilaridades de 40 puntos en un mapa de calor, sobre la cual se ha ejecutado un agrupamiento jerárquico AHC bajo el criterio del *Nearest Point Algorithm*. En esta gráfica se pueden observar claramente cómo los *clusters* se distribuyen sobre la diagonal de dicha matriz. Las zonas con colores más claros representan una mayor similaridad entre los objetos.

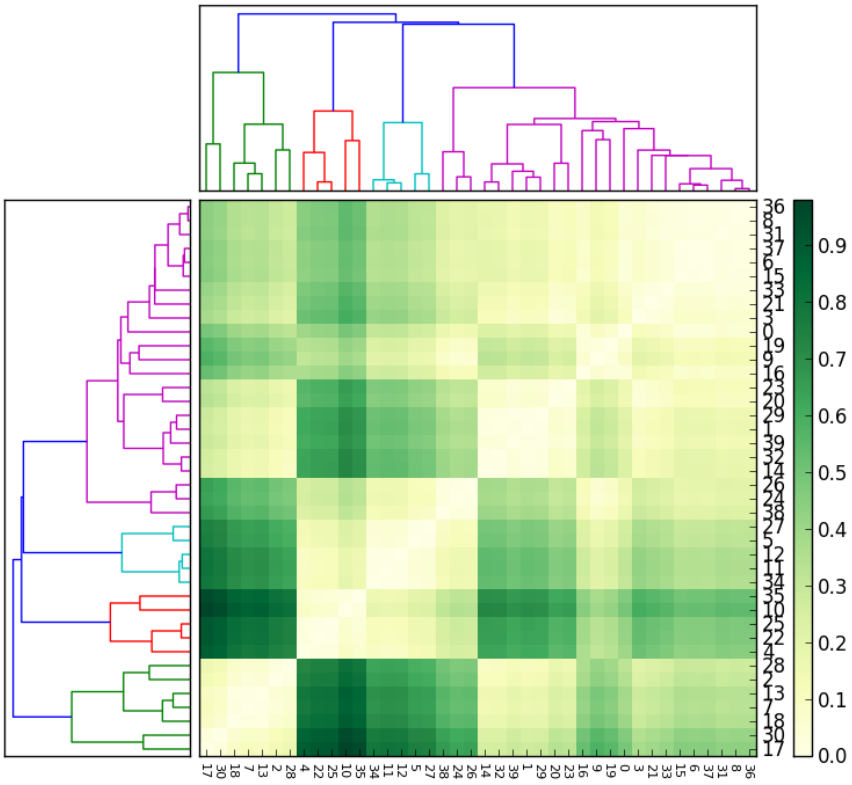

Figura 12. *Matriz de Disimilaridades, con Algoritmo AHC, k=4*

3.7. Clustering de Documentos

El *clustering* de documentos se basa en una hipótesis que reza de la siguiente manera [65]:

"*Documentos en el mismo cluster se comportan de manera similar con respecto a la relevancia de las necesidades de información*"

Esto quiere decir que si existe un documento en un *cluster* que es relevante para una búsqueda, entonces, es probable que los otros documentos del mismo *cluster* también sean relevantes para la misma búsqueda.

El *clustering* de documentos es un tópico que se ha estudiado y ha sido ampliamente investigado como una metodología para mejorar la búsqueda y recuperación de documentos [47], se han planteado diferentes esquemas y técnicas que permitan procesar los corpus de los documentos y posteriormente ejecutar un proceso de agrupamiento. En el trabajo de Peter Willett [69] se tiene una excelente revisión de los conceptos y trabajos relacionados con el *clustering* de documentos.

El agrupamiento de documentos, tiene como finalidad analizar colecciones, dividiéndolas en grupos de documentos similares, tradicionalmente los documentos son representados como bolsas de palabras mediante un modelo vectorial, aunque en ocasiones este esquema desaprovecha las relaciones existentes entre las mismas. La

aproximación de bolsa de palabras se utiliza ampliamente dado que genera de manera rápida resultados aceptables; sin embargo, una de sus desventajas es que no considera variaciones lingüísticas como la morfológica, que origina palabras con diferente número, género, tiempo, modo; la léxica, en la que diferentes palabras tienen el mismo significado; la sintáctica, en la que el orden de las palabras cambia el significado y la semántica, en la que una palabra puede tener diferentes significados [56].

Además del modelo vectorial, existen otras medidas de similaridad como los coeficientes de *Dice* y *Jaccard*, que normalizan los recuentos de solapamiento entre las palabras [47]. Por lo que se podría plantear esquemas para la agrupación de documentos que combinen varias medidas de similitud en un parámetro único de entrada hacia el algoritmo de *clustering*, una posible solución es el trabajo con la matriz de distancias/disimilaridades.

Muy pocos estudios se han propuesto donde se trabaje sin un modelo vectorial y más bien se utilice directamente la matriz de disimilaridades para el proceso de agrupamiento. Esto sobre todo, por las dificultades que genera la adaptación de un algoritmo existente a este tipo de esquemas. Como se mencionó en la Sección 3.3.2, el algoritmo *K-Medoids* opera con la matriz de similaridad de un conjunto de datos en lugar del espacio original de características. El artículo [44] propone, una alternativa de agrupamiento de documentos con el uso del algoritmo *K-Medoids*.

Willett en [69] ha sugerido que la elección de la medida de similitud, tiene menos impacto cualitativo en los resultados de la agrupación, que la elección del algoritmo de agrupamiento [47].

En el trabajo de Moran et al. [57], se presenta una propuesta para encontrar las mejores palabras clave en una conferencia de *Machine Learning* utilizando *clustering* espectral, este último es un esquema de agrupamiento que también opera con la matriz de disimilaridades. Su propuesta se basa en las opiniones de expertos para la validación de los resultados. El *dataset* con el que se trabajó fue liberado para su reutilización, por lo que parte del trabajo experimental de este estudio, en la Sección 6.3, se ha valido de este conjunto de datos para el análisis de *clustering* con restricciones de tamaño.

Además de los algoritmos particionales, en el *clustering* de documentos también se ha trabajado con esquemas basado en *clustering* jerárquico [47][69] y métodos basados en grafos. Los métodos basados en grafos intentan modelar a los documentos como vértices de un grafo ponderado. El peso del borde se determina por la similitud de los dos documentos correspondientes. Entonces, el problema de la agrupación de documentos se transforma en un esquema de representación gráfica de partición basada en un cierto criterio [46].

El estudio de Steinbach et al. [71], presenta una comparativa de las diferentes técnicas de *clustering* para documentos, haciendo énfasis en los rendimientos y centrado en los algoritmos de agrupamiento particionales y jerárquicos.

Dentro de la literatura revisada, existe un grupo de artículos e investigaciones cuya orientación, talvez es la más importante, ya que combina los conceptos de *clustering* con restricciones (que se abordarán en el Capítulo 4) y el agrupamiento de documentos.

En el año 2008, Hu et al., publicaron el artículo "*Towards effective document clustering: A constrained K-Means based approach*" [46] inspirados en el trabajo de Ji et al. (2006) [70], donde se hace una aproximación mediante la integración de restricciones del tipo *instance-level* en un esquema de *clustering* de documentos. El artículo presenta una serie de comparaciones con otras metodologías propuestas y los resultados son bastante aceptables a nivel de rendimiento.

4. Clustering Semi-Supervisado

Resumen: Este capítulo abordará el problema de *clustering* semi-supervisado, los problemas de agrupamiento con restricciones y las nuevas definiciones asociadas a esta variación de aprendizaje. Se hará una revisión del estado de arte y trabajo relacionado, con un especial énfasis en los algoritmos de aprendizaje semi-supervisados.

4.1. Algoritmos de Clustering Semi-Supervisados

En el campo del *machine learning*, los métodos no supervisados se han utilizado en una variedad de tareas, principalmente enfocadas al análisis de datos exploratorio. Algunos ejemplos de estas tareas incluyen el descubrimiento de grupos, la compresión de datos, la reducción de la dimensionalidad y la detección de *outliers* [16].

En los problemas del mundo real, los usuarios a menudo tienen alguna información adicional acerca de los *clusters* como: etiquetas, relación entre los datos, o conocimiento global de parte de sus dominios, que proceden del conocimiento externo del *dataset*, de la problemática abordada o del conocimiento de un experto [21]. Estudios recientes han demostrado que la incorporación de esta información en los algoritmos de *clustering* no supervisados tradicionales puede aumentar el rendimiento de la agrupación [3] [17] [18] [21].

A este tipo de *clustering*, que utiliza esta pequeña cantidad de información adicional para guiar al algoritmo en el proceso de agrupación, se le conoce con el nombre de *clustering* semi-supervisado [16] [21] o *clustering* restringido (*constrained clustering*) [17][18].

Las posibles restricciones, que pueden ser especificadas por el usuario, son de dos tipos [65]:

1. ***Cluster-level constraints:*** en las restricciones a nivel de *cluster* se definen algunos requerimientos específicos de los *clusters* tales como: el número mínimo o máximo de elementos, es decir el tamaño de los posibles grupos.
2. ***Instance-level constraints:*** en las restricciones del tipo nivel de instancia se definen propiedades entre pares de objetos tales como la pertenencia o no de elementos al mismo *cluster* [65]. El uso de restricciones por pares (*pairwise constraints*) también puede contribuir al proceso de inicialización del *cluster*. Por ejemplo, durante la inicialización del *cluster*, los puntos *must-links* deben comenzar en el mismo grupo, mientras que los puntos *cannot-links* deberían iniciar en diferentes *clusters* [16]. Esto es fundamental cuando se usa una metodología estratificada para la inicialización de los puntos semilla en algoritmos de refinamiento iterativo como el *K-Means* o el *K-Medoids*.

A su vez, los métodos existentes para el *clustering* semi-supervisado pueden ser vistos desde dos enfoques generales: los basados en las restricciones (*constraint-based*) y los basados en métricas (*metric-based*). En la aproximación basada en restricciones, el propio algoritmo de *clustering* es modificado de manera que las etiquetas proporcionadas por el usuario o las restricciones por pares se utilizan para guiar el algoritmo hacia una partición de datos más apropiada. Esto se lleva a cabo mediante la modificación de la función objetivo del *clustering* para que incluya el cumplimiento de las restricciones durante el proceso de agrupación. Por otra parte, en la aproximación basada en métricas, se emplea un algoritmo de *clustering* que utilice una distancia métrica. Esta métrica es previamente entrenada para que cumpla con las etiquetas o

restricciones en datos supervisados. Se han utilizado varias medidas de distancia para este tipo de *clustering*, incluyendo: la distancia euclidiana entrenada por un algoritmo de la ruta más corta, la cadena de distancia de edición entrenada con *Expectation Maximization* (EM), la divergencia *Kullback-Leibler* (KL) usando el gradiente descendiente y las distancias de *Mahalanobis* entrenadas por medio de optimización convexa [21].

De los dos tipos de restricciones descritas anteriormente, se han derivado varias definiciones de restricciones específicas como [65]:

- **Restricción *Must-Link*:** requiere que dos instancias (objetos) x_i y x_j deben ser colocadas en el mismo *cluster* c_j, del conjunto de *clusters* c. Esta relación puede ser expresada como: $\exists c_j \in c : x_i \in c_j \wedge x_j \in c_j$.
- **Restricción *Cannot-Link*:** requiere que dos instancias (objetos) x_i y x_j deben ser colocadas en un diferente *cluster* c_j, del conjunto de *clusters* c. Esta relación puede ser expresada como: $\forall c_j \in c : \neg(x_i \in c_j \wedge x_j \in c_j)$
- **Restricciones de Tamaño del *Cluster*:** requiere que se encuentren *clusters* que tengan un mínimo o máximo número de elementos. Estas restricciones son expresadas respectivamente por: $\forall c_j \in c : |c_j| \geq \alpha$ y $\forall c_j \in c : |c_j| \leq \alpha$.
- **Restricciones de Diámetro del *Cluster*:** requiere que cada *cluster* tenga un diámetro de máximo ϵ. A esta restricción también se la conoce como *ϵ-constraint* y puede ser expresada como:
  $\forall c_j \in c, \forall x_i, x_j \in c_j : d(x_i, x_j) \leq \epsilon$.
- **Restricción de Margen:** requiere que el margen entre dos *clusters* diferentes deben estar por encima de un cierto umbral δ. A esta restricción también se la conoce como *δ-constraint* y puede ser expresada como:
  $\forall c_j, c_i \in c, \forall x_i \in c_j, x_j \in c_i : d(x_i, x_j) \geq \delta$
- **Restricción de Equilibrio:** Las *balancing constraints* se pueden ver como un caso especial de las restricciones de tamaño del *cluster*, donde todos los grupos tienen el mismo tamaño. Varias aplicaciones de *clustering* en la vida real requieren que los tamaños de los grupos sean fijados, pero no necesariamente con el mismo tamaño para todos los grupos [3]. Los *datasets* no balanceados se observan ampliamente en aplicaciones prácticas como por ejemplo, cuando se realiza la clasificación en diagnóstico médico, el número de puntos en la clase enfermedad será mucho menor que en la clase normal (sin enfermedad) [17].

La Figura 13 ilustra los conceptos de restricciones *must-link* y *cannot-link* [16].

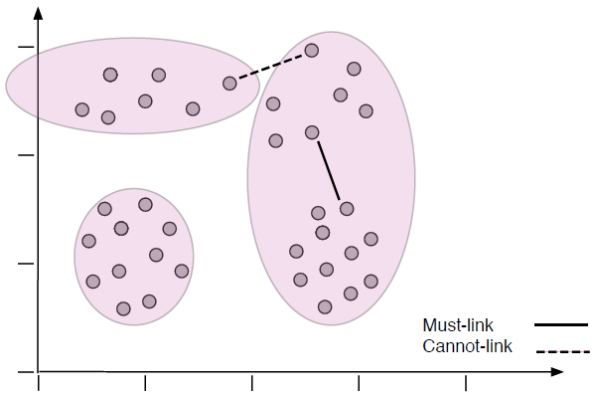

Figura 13. *Representación Visual de las Restricciones Must-Link y Cannot-Link [16]*

Los resultados de complejidad computacional para algunos de los problemas de restricciones, se resumen en la Tabla 7.

Restricción	Complejidad
Must-Link	P
Cannot-Link	NP-Completo
δ-constraint	P
ϵ-constraint	P
Must-Link y δ	P
Must-Link y ϵ	NP-Completo
δ y ϵ	P

Tabla 7. *Complejidad Computacional de las Restricciones [6]*

Algunos de los algoritmos y métodos semi-supervisados que hacen uso de las restricciones antes mencionadas, se citan a continuación.

4.1.1. COP-Kmeans

Constraint-Partitioning K-Means, es un algoritmo que surge como una variación del algoritmo COP-COBWEB que se formuló en [15] y se caracteriza porque permite utilizar restricciones del tipo *must-link* y *cannot-link*, siendo ambas restricciones consideradas como *hard constraints*. Este algoritmo, cuyo fundamento es el algoritmo *K-Means,* actualiza las asignaciones del *cluster*, con la certeza de que ninguna de las restricciones especificadas se van a violar durante el proceso [14].

4.1.2. MPCK-Means

El algoritmo *MPCK-Means*, incorpora tanto el aprendizaje semi-supervisado basado en métricas, como el uso de *pairwise constraints (must-link* y *cannot-link). MPCK-Means* entrena la distancia-métrica en cada iteración del proceso de *clustering*, utilizando tanto los datos no etiquetados como las restricciones por pares. El algoritmo es capaz de aprender métricas individuales para cada grupo, lo que le permite obtener *clusters* de diferentes formas. *MPCK-Means* también permite la violación de las restricciones si esto conduce a una agrupación más cohesiva [21].

Existen algunas variaciones del algoritmo *MPCK-Means* como el *PCKmeans* (*Partial Constrained K-Means*) o el *PCCKmeans* (*Partial Closurebased Constrained K-Means*) donde se optimizan sus correspondientes funciones de coste, y se toman en cuenta tanto a las distancias como también las restricciones [17].

4.1.3. Clustering temporal semi-supervisado

El *clustering* temporal hace referencia a la partición de una serie de tiempo en múltiples segmentos no superpuestos que pertenecen a *k clusters* temporales, de tal manera que los segmentos en el mismo grupo sean más similares entre sí. El *clustering* temporal utiliza un método completamente no supervisado, que en algunos casos, puede producir resultados poco satisfactorios. Al igual que en los métodos de *clustering* tradicionales, el *clustering* temporal también se puede beneficiar del conocimiento disponible en el *dataset*. De esta manera surge una alternativa llamada *clustering* temporal semi-supervisado que es una estrategia en la que el conocimiento adicional, en forma de restricciones por pares, se incorpora en los datos temporales para ayudar a particionar el *dataset* [16].

4.1.4. Clustering jerárquico semi-supervisado

[15] Las restricciones, por su misma naturaleza, se enfocan inicialmente en los algoritmos particionales. Los algoritmos de agrupamiento jerárquico no admiten una interpretación directa de las restricciones *must-link* y *cannot-link*. Una restricción *cannot-link* entre dos instancias, por ejemplo, podría significar que no pueden estar en el mismo nodo, o que no pueden tener el mismo nodo padre, o que no pueden estar a una "distancia" pre-especificada de la otra jerarquía, etc. Una ambigüedad similar ocurre con

las restricciones del tipo *must-link*. La dificultad de incorporar restricciones duras de tipo *instance-level* dentro de los algoritmos de agrupamiento jerárquico es una posible desventaja del enfoque de los algoritmos con restricciones, aunque en los trabajos recientes de [19] y [20] se ofrecen algunas posibles soluciones para solventar dichas limitaciones.

4.1.5. Trabajos relacionados

En la literatura revisada para la construcción de la propuesta "*Clustering de documentos con restricciones de tamaño*", se ha podido observar que hay una orientación de los trabajos previos con dos enfoques:

El primer enfoque temático de la literatura tratada se proyecta en la definición de los conceptos básicos del *clustering* de documentos y en algunos de los trabajos que han propuesto esquemas de *clustering* para documentos con la incorporación de restricciones, que no necesariamente implican la restricción del tamaño del *cluster* y que ya fueron mencionados en la Sección 3.7.

Por otra parte, el segundo grupo de investigaciones está orientado a la explicación de los modelos matemáticos y las propuestas de nuevos algoritmos que resuelven de manera parcial o total la problemática del *clustering* de objetos con restricciones de tamaño, sin centrarse en el caso particular de los documentos.

De este conjunto de artículos hay dos que son los que mayor impacto [3] y [17]. El primero se basa en un modelo de agrupamiento particional combinado con programación lineal entera, mientras que el segundo artículo usa una formulación basada en penalizaciones por incumplimiento de los tamaños de los *clusters*, que previamente fueron indicados por el usuario. Los procedimientos descritos en estos dos artículos han servido de base e inspiración para las soluciones propuestas en la Sección 5 de este estudio.

La única aproximación propuesta para el algoritmo *K-Medoids* con restricciones de tamaño se presenta en [65] donde se utiliza restricciones del tipo *intance-level* para el proceso de optimización de la función objetivo.

En la Tabla 8, se resumen las principales características del trabajo relacionado con el *clustering* que incorpora restricciones de tamaño. Todos los algoritmos y métodos planteados han sido enfocados a algoritmos de tipo no jerárquico.

Autores	Título	Algoritmo	Restricciones	Ventajas	Simulaciones y Consideraciones
[22] Höppner, Frank; Klawonn, Frank	Clustering with size constraint, 2008	Modifica el FCM (Fuzzy C-Means, que es una variación del K-Means). Transforma el problema combinacional discreto en un problema continuo, de tal manera que se puedan utilizar métodos numéricos para la resolución del problema, para esto se modifica la función objetivo. La idea de esta modificación es añadir una restricción adicional (con multiplicadores de Lagrange) en la función objetivo que obligue a cumplir la restricción del tamaño.	Multiplicadores de Lagrange	Se puede usar con *clusters* del mismo tamaño o de diferentes tamaños. En el caso que el *dataset* no pueda cumplir con un tamaño determinado (por complicación del *dataset*) se pide prestado un elemento a otro *cluster* cercano. Dicho elemento es tomado del centro del *cluster* cercano.	Pruebas en *datasets* extremadamente grandes o pequeños. *Clusters* del mismo tamaño solo se pueden conseguir si la densidad de los datos es uniforme.
[3] Shunzi Zhu, Dingding Wang, Tao Li	Data Clustering with Size Constraint, 2010	Algoritmo heurístico toma como punto de partida el K-Means o el MPCK-Means. Transforma el problema de restricción de tamaño en un problema de programación lineal entera (PLE). Uso del *instance-level* en forma de inecuación para que se pueda incorporar en el problema de programación lineal.	Uso del instance-level: *Cannot Link* (CL) (opcional) Prioriza las restricciones de tamaño sobre las restricciones *instance-level*	Se puede usar con *clusters* del mismo tamaño o de diferentes tamaños. Si no se conoce el tamaño exacto del *cluster* se puede incorporar un treshold.	Comparaciones con K-Means y MPCK-Means. Los pares de restricciones *cannot-link* son generadas de manera aleatoria.
[12] David Rebollo-Monedero, Marc Solé, Jordi Nin, Jordi Forné	A modification of the k-means method for quasi-unsupervised learning, 2013	Método heurístico basado en la modificación del K-Means (denominado PCL) con un procedimiento de optimización. Está dotado de ponderaciones para cada *cluster* que controla el tamaño de los *clusters* formados, y que es ajustado por medio del algoritmo de Levenberg-Marquardt.	Se propone también algunas variaciones adicionales sobre esta modificación, en el que diferentes tipos de información adicional estén presentes.	Se puede usar con *clusters* del mismo tamaño o de diferentes tamaños.	La complejidad en tiempo de funcionamiento se evalúa experimentalmente por medio de un análisis de regresión.
[11] Nuwan Ganganath, Chi-Tsun Cheng, Chi K. Tse	Data Clustering with Size Constraints Using a Modified k-Means Algorithm, 2014	Modificación del K-Means con dos variaciones: 1) Inicialización aleatoria del centroide, o 2) Inicialización selectiva del centroide. La similitud entre los objetos del *cluster* tiene que ser maximizada.	Uso del *instance-level*: *Must Link* (ML) y *Cannot Link* (CL). Centroides selectivos son los ML o CL Necesario conocer por los menos un elemento en cada *cluster*.	Se puede usar con *clusters* del mismo tamaño o de diferentes tamaños. Permite tener restricciones de tamaño para cada *cluster* de manera separada.	Comparaciones con K-Means *Datasets* Multidimensionales (m): 2 y 3 dimensiones Los resultados con K-Means pueden ser variables (mejores o peores) para un mismo *dataset* dependiendo de donde se escoja los centroides iniciales.

[17] Zhang, Shaohong; Wong, Hau-San; Xie, Dongqing.	Semi-supervised clustering with pairwise and size constraint, 2014	Combinación de KmeansS (Kmeans with Size constraint) el cual usa solo la restricción de tamaño, y PCKmeans el cual usa solo la restricción del *pairwise* (se entiende que el *pairwise* es lo mismo que el *instance level*: *must-link, cannot-link*). El nuevo algoritmo se denomina PCS.	Uso del *instance-level*: *Must Link* (ML) y *Cannot Link* (CL). Se deben ordenar las restricciones.	No solo se toma en cuenta el costo de la distancia sino que también se usa el costo de la violación de una restricción del *pairwise* y se penaliza la inconsistencia entre la distribución de tamaño del *clustering* y el tamaño del *cluster*.	Comparaciones con K-Means y PCKmeans.
[65] Grossi, Valerio; Monreal, Anna; Nanni, Mirco; Pedreschi, Dino; Turini, Franco	Clustering Formulation Using Constraint Optimization 2015	Este documento propone una formulación de programación con restricciones para tres métodos de agrupamiento: K-Medoids, DBSCAN y Label Propagation	Se utiliza *Cluster-level constraints e Instance-level constraints*. Las primeras para los algoritmos DBSCAN y Label Propagation, y las otras para el algoritmo K-Medoids.	Se puede usar con *clusters* del mismo tamaño o de diferentes tamaños.	No se realizan simulaciones sobre *datasets*, sino únicamente los planteamientos de los algoritmos.

Tabla 8. *Trabajos Relacionados al Clustering con Restricciones de Tamaño*

5. Algoritmos de Clustering con Restricciones de Tamaño: CSCLP y K-MedoidsSC

Resumen: En este capítulo se describen dos nuevos algoritmos que se plantean para solucionar el problema de *clustering* con restricciones de tamaño. La primera propuesta se fundamenta en restricciones del tipo *cannot-link* y programación lineal entera binaria, mientras que el segundo enfoque se basa en el funcionamiento del algoritmo *K-Medoids*. Antes de describir las dos nuevas propuestas se realiza la formulación matemática que sustenta el proceso genérico de agrupamiento con restricciones de tamaño.

5.1. Formulación Matemática del Problema

Ya que los algoritmos tradicionales de *clustering* no proporcionan mecanismos eficaces para la incorporación de las restricciones de tamaño en cada *cluster*, se ha pensado en incorporar esta información adicional. De tal manera que el problema de *clustering* de documentos con restricciones de tamaño podría ser expresado de la siguiente manera [11]:

Dado el conjunto de datos $x = \{x_1, x_2, \dots, x_n\}$ de tamaño n, donde $x_i \in \mathbb{R}^m$, dicho conjunto de datos puede ser expresado en términos de documentos como: $D = \{D_1, D_2, \dots, D_n\}$ donde n es el número de documentos en la colección [47]. En un problema de *clustering* sin restricciones de tamaño, el objetivo del algoritmo de *clustering* es encontrar un valor de k *clusters*, tal que se cumpla $1 \leq k \leq n$, y donde los tamaños de los *clusters* $c = \{c_1, c_2, \dots, c_k\}$ son determinados de manera automática por el algoritmo, cumpliéndose que la similitud entre objetos de cada *cluster* sea la máxima. El criterio de similitud puede ser diferente dependiendo del algoritmo de *clustering* utilizado. Aquí, el tamaño total del conjunto de datos es $|c| = \sum_{\forall j} |c_j|$, donde $|c_j|$ denota el tamaño de un *cluster* c_j y $1 \leq j \leq k$. En el *clustering* de documentos con restricciones de tamaño, el tamaño máximo del *cluster* E_j es un valor conocido para cada *cluster* c_j. Por lo tanto, un algoritmo de *clustering* con restricciones de tamaño debería satisfacer una condición adicional, además del valor k, y es que $|c_j| \leq E_j$ de tal manera que $\sum_{j=1}^{k} E_j \geq |x|$.

5.2. Algoritmo de Clustering con Restricción de Tamaño y Programación Lineal (CSCLP – Clustering with Size Constraints and Linear Programming)

Como ya se ha mencionado, en el problema de *clustering* con restricciones de tamaño se conoce de antemano el número de objetos (documentos) que serán asignados en cada k *cluster*. En el trabajo desarrollado por Shunzi Zhu et al. "*Data Clustering with Size Constraints*" [3], se propone un algoritmo que toma como punto de partida los algoritmos *K-Means* o *MPCK-Means*, para transformar el problema de restricción de tamaño en un problema de programación lineal entera (PLE). Se hace uso de restricciones *instance-level* en forma de inecuaciones para que esta información se pueda incorporar en el problema de programación lineal. Para este planteamiento:

1. Se ejecuta el algoritmo *K-Means* o *MPCK-Means*
2. Se optimiza los grupos obtenidos en el paso anterior utilizando PLE y restringiendo los tamaños de cada *cluster*.

Uno de los principales inconvenientes de este planteamiento es la alta dependencia de los *clusters* que resultan de aplicar el algoritmo de agrupación.

Para mitigar este inconveniente dentro de la formulación propuesta en este trabajo, se propone realizar una modificación en el planteamiento de [3] y utilizar únicamente los k puntos iniciales a manera de *pairwise constraints* del tipo *cannot-link* para la formación de los *clusters,* y mediante el uso de programación lineal entera binaria (PLEB) determinar la pertenencia y asignación de las instancias a los k *clusters*. De esta manera, el problema original de *clustering* con restricciones de tamaño se convierte en un problema de optimización, encontrando de una manera eficiente la solución mediante una nueva propuesta heurística.

Uno de los cambios más importantes y significativos que se presenta en esta nueva aproximación y en el que difiere del trabajo planteado por Shunzi Zhu et al. es el uso de la matriz de distancias/disimilaridades como punto de partida. La matriz de disimilaridades permitirá caracterizar a las instancias involucradas en el proceso de *clustering*, por lo que no se trabajará en el espacio geométrico convexo en el que operan algoritmos como el *K-Means* o el *MPCK-Means*.

Sabemos que el tamaño de un *cluster* c_j está representado por su cardinalidad $|c_j|$, por lo tanto para que el proceso de *clustering* se inicie el usuario deberá especificar el valor de k, para la generación de los k puntos iniciales, y el tamaño de los *clusters* $E_j = \{e_1, e_2, \dots, e_k\}$. Como se acabó de mencionar los k puntos iniciales $u = \{u_1, u_2, \dots, u_k\}$ son a la vez restricciones del tipo *instance-level cannot-link*, esto quiere decir que ninguno de ellos puede pertenecer a un mismo *cluster*.

Una vez generados los k puntos iniciales, se deberá encontrar una asignación del resto de objetos hacia alguno de los *clusters*, por lo que es necesario el cálculo de la matriz de distancias/disimilaridades entre todos los objetos del *dataset*. Con estas distancias se calculará el coste entre cada punto inicial y los demás objetos, sin tomar en cuenta los $k-1$ puntos semilla restantes, tal como lo ilustra la Figura 14.

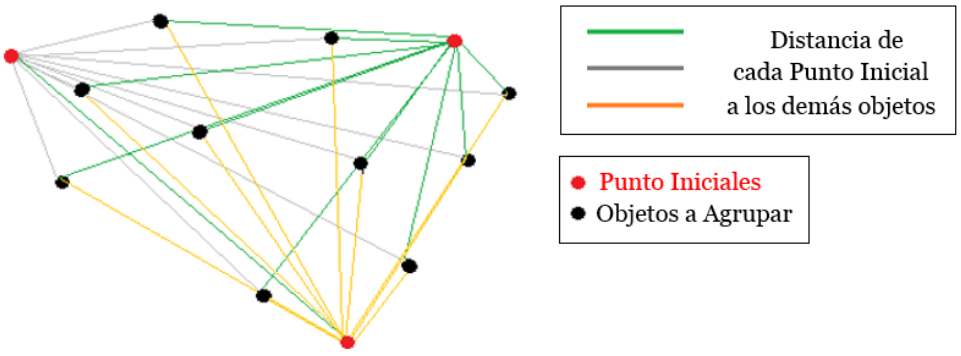

Figura 14. *Distancias entre los Objetos del Dataset y Puntos Iniciales*

Considérese un modelo matricial, donde A_{nxk} representa a una matriz booleana de pertenencia de los documentos a un *cluster* en particular. Cada fila de la matriz respresenta a un documento D_i y cada columna es un *cluster* c_j. Si un elemento de esta matriz booleana a_{ij} toma un valor $a_{ij} = 1$, implica que el documento i pertenece al *cluster* j, de manera complementaria, $a_{ij} = 0$ indica que el documento i no pertenece al *cluster* j.

$$A = \begin{array}{c} \\ D_1 \\ D_2 \\ \vdots \\ \vdots \\ D_n \end{array} \begin{array}{c} \begin{array}{ccccc} c_1 & c_2 & \dots & \dots & c_k \end{array} \\ \begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 \end{pmatrix}_{nxk} \end{array}$$

$$D_i: Documento\ i = 1, 2, \dots, n$$

$$c_j: Grupo\ j = 1,2, \dots, k$$

En la matriz A_{nxk}, la suma de las filas indica la pertenencia, única, de un objeto (documento) a un *cluster*. Mientras que la suma de las columnas indica el tamaño del *cluster*. Por tanto, existe la posibilidad de que cada columna tenga más de un valor $a_{ij} = 1$; además cada columna deberá cumplir con la restricción especificada del tamaño de grupo $E_j = \{e_1, e_2, \dots, e_k\}$.

$$\sum_{j=1}^{k} a_{ij} = 1\ ;\ \forall i = 1, 2, \dots, n \ \longrightarrow \ Pertenencia\ a\ un\ cluster$$

$$\sum_{i=1}^{n} a_{ij} = e_j\ ;\ \forall j = 1, 2, \dots, k \ \longrightarrow \ Tamaño\ del\ cluster$$

Ahora, este modelo matricial que en un inicio representaba un problema de *clustering* se podrá modelar como un problema de optimización, más específicamente como un problema de programación lineal PPL. Este problema de optimización se puede resolver mediante programación lineal entera binaria, ya que las decisiones de pertenencia de un objeto a un *cluster* únicamente pueden tomar dos valores: cero en el caso de que el objeto si pertenezca al *cluster* y uno en el caso contrario.

Para que no se viole la restricción de tamaño especificada por el usuario es necesario que al modelo de programación lineal se le incorporen ciertas restricciones en forma de ecuación. Siguiendo el esquema estándar de programación lineal, explicado en la Sección 2.6.1.1, se ha de plantear una función objetivo que deberá ser minimizada y adicionalmente los valores de las restricciones deberán ingresar en el modelo, en forma de vectores.

La función objetivo a minimizar incluirá dos variables: la distancia que proviene de la matriz de disimilaridades y la pertenencia de un objeto a un *cluster*. La pertenencia de las instancias a sus respectivos grupos será determinada por la solución del PLEB.

Función Objetivo PLEB:

$$Función\ Objetivo = min\left(\sum_{j=1}^{k}\sum_{i=1}^{n} d_{ij}p_{ij}\right)$$

Restricciones:

Tamaño del *cluster*: $\sum_{i=1}^{n} d_{ij} = e_j\ ; \forall j = 1,2, \dots . k$

Pertenencia: $\sum_{j=1}^{k} p_{ij} = 1; \forall i = 1,2, \dots . n$

El pseudocódigo de este nuevo algoritmo CSCLP – *Clustering with Size Constraints and Linear Programming*, se describe a continuación [63]:

Algoritmo: CSCLP

ENTRADA:

Matriz de Distancias/Disimilaridades *DM*
Número de *Clusters k*
Tamaño de los *Clusters* $\{e_1, e_2, \dots, e_k\}$,

SALIDA:

Resultado del Agrupamiento Etiquetado

MÉTODO:

1. Selecciona los puntos iniciales de los *clusters* $u_1, \dots, u_k$, de manera aleatoria o con cualquier otro método
2. Construir la función objetivo
3. Construir las restricciones de tamaño de *cluster* y pertenencia
4. Resolver el PLEB

Se debe notar que este nuevo procedimiento para la asignación de los tamaños de los *clusters* es sensible a la asignación de los k puntos iniciales y al orden en que se ingresan, dichos puntos, en el proceso de optimización.

5.3. Algoritmo K-Medoids con Restricción de Tamaño (K-MedoidsSC – KMedoids with Size Constraints)

En el trabajo "*Semi-supervised clustering with pairwise and size constraints*" [17] publicado por Shaohong Zhang et al., se propone un algoritmo llamado *K-MeansS* que permite realizar *clustering* con restricciones de tamaño para un algoritmo del tipo *K-Means*. Las pruebas han demostrado que este procedimiento provee evidencia empírica de que la asignación de los objetos del *dataset* puede ser una de las alternativas para la asignación de cada uno de los objetos hacia los diferentes *clusters*.

Uno de los inconvenientes y limitaciones que tiene el algoritmo *K-Means*, es que durante la actualización de los centroides dentro del proceso de refinamiento iterativo, estos son calculados en un espacio geométrico convexo donde se pueden calcular las coordenadas de los nuevos centroides. Aunque, en la mayoría de los casos, en el *clustering* de documentos se trabaja en un espacio vectorial producido por la matriz de pesados TF-IDF, no siempre se trabaja con un esquema de bolsa de palabras y un modelo vectorial, sino que en ocasiones es necesario mezclar varios criterios y unificarlos para obtener unas sola métrica que cuantifique las disimilaridades.

Una manera de solventar este tipo de inconvenientes es el uso de algoritmos que para el proceso de *clustering* tomen como entrada la matriz de disimilaridades y trabajen con diferentes métricas, como el algoritmo *K-Medoids* explicado en la Sección 3.3.2. Inspirados en el trabajo de Shaohong Zhang et al. se propone un cambio a esta formulación mediante el uso del algoritmo *K-Medoids* en el proceso de *clustering* con restricciones de tamaño.

El trabajar con una matriz de disimilaridades, al contrario de un conjunto de puntos que representan a las instancias, genera una serie de dificultades y modificaciones al modelo original de agrupación. A continuación se presenta el planteamiento de este nuevo algoritmo basado en el *K-Medoids* donde se incorporan restricciones de tamaño.

La nueva formulación se sustenta en la penalización de una función de coste. Esta función de coste J_{KMS}, toma en cuenta cuatro parámetros implícitos y fundamentales en el proceso de *clustering* con restricciones de tamaño: criterio de optimización de la

función objetivo, penalización por incumplimiento del tamaño del *cluster*, penalización porque el tamaño del *cluster* es más pequeño de lo esperado y penalización porque el tamaño del *cluster* es más grande de lo esperado. La siguiente expresión formula lo antes expuesto:

$$J_{KMS} = J_{KM} + \alpha J_A + \beta J_S + \gamma J_L$$

Donde los coeficientes α, β y γ son los parámetros no negativos, que representan a los diferentes pesos de las funciones de penalización para los tamaños de *cluster*.

Antes de definir cada uno de los elementos que componen a la función de coste, es menester conocer algunas de las definiciones involucradas en el proceso de penalización. En un espacio geométrico $\mathbb{R}^n$, donde se encuentran distribuidos los objetos a agrupar $x = \{x_1, x_2, \dots, x_n\}$, cuyos medoides son $u = \{u_1, u_2, \dots, u_k\}$ y donde los tamaños de los *clusters* especificados por el usuario son $E_j = \{e_1, e_2, \dots, e_k\}$. Si se ejecuta un proceso de *clustering*, cualquiera, se generarán k *clusters* $c = \{c_1, c_2, \dots, c_k\}$ de tamaños diferentes a los esperados, donde se puede observar las siguientes propiedades entre los objetos y los *clusters*:

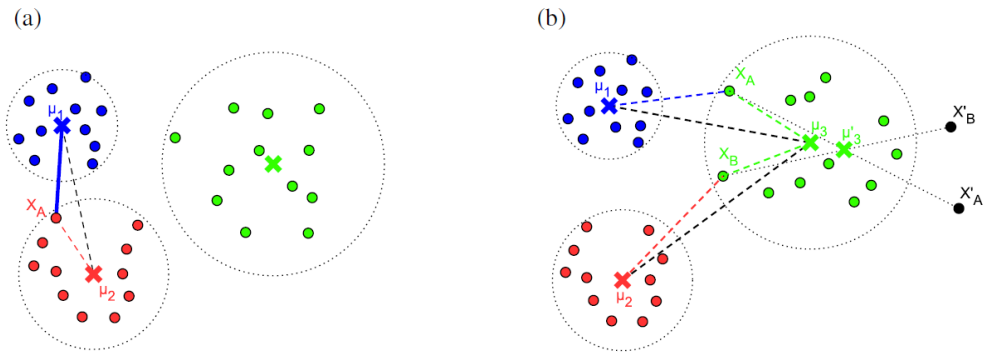

Figura 15. *Puntos NO y Puntos FI y VFI [17]*

En la Figura 15(a), considérese el siguiente supuesto: c_1 y c_2 son los dos *clusters* más cercanos entre sí, cuyos tamaños después de un primer proceso de *clustering* particional iterativo, no cumplen con la restricción de tamaño e_1 y e_2 que se la ha impuesto. Entonces se deberá realizar una reasignación de uno o más puntos de dichos *clusters* para que se cumpla con la restricción. El punto x_A es el punto más alejado del medoide u_2 en el *cluster* c_2, pero a la vez el más cercano del medoide u_1 en el *cluster* c_1, entonces se considera que x_A es un punto denominado *nearest out-class* (*NO*), que es el candidato más adecuado para ser transferido del *cluster* c_2 hacia el *cluster* c_1 con la finalidad de que se cumpla la restricción de tamaño.

Por su contraparte en la Figura 15(b), suponga que: c_1 y c_2 son los dos *clusters* más cercanos a c_3 , y el tamaño e_3 de c_3 después de un primer proceso de *clustering* particional iterativo, es más grande de lo esperado. El punto x_A es el punto más alejado del medoide u_3 en el *cluster* c_3, pero a la vez el más cercano del medoide u_1 en el *cluster* c_1, entonces se considera que x_A es un punto *farthest in-class* (FI), que es el candidato más adecuado para ser transferido del *cluster* c_3 hacia el *cluster* c_1 con la finalidad de que se cumpla la restricción de tamaño. Un ejemplo similar se presenta entre los *clusters* c_2 y c_3, donde x_B es el punto *farthest in-class* (FI). Los puntos x_A' y x_B' se denominan puntos *virtual paired FI* (*VFI*) y son únicamente una proyección geométrica de los puntos originales x_A y x_B en el sentido opuesto.

En la función J_{KMS} el término J_{KM} expresa la función de coste del algoritmo *K-Medoids*, sin ningún tipo de restricción:

$$J_{KM} = min \sum_{i=1}^{k} \sum_{x \in c_i} d(x, u_i)$$

El término J_A, será la penalización por incumplimiento del tamaño del *cluster*. El primer problema de incorporar las restricciones de tamaño es el adecuado alineamiento de las clases generadas $\{c_j\}_{j=1}^{k}$por el algoritmo, con las clases esperadas $\{E_j\}_{j=1}^{k}$. Una solución sencilla a este inconveniente es ordenar de manera ascendente a ambas distribuciones, para luego alinear las distribuciones de las clases y posteriormente calcular las diferencias entre los valores obtenidos y esperados.

Si consideramos que $p = \{p_j : p_j = E_j/n\}_{j=1}^{k}$ es la distribución a priori del tamaño del *cluster* y $q = \{q_j : q_j = c_j/n\}_{j=1}^{k}$es la distribución a posteriori, entonces la diferencia entre estas dos distribuciones puede ser calculada mediante la divergencia de Jensen-Shannon:

$$JSD(p, q) = \frac{1}{2} KL(p, q) + \frac{1}{2} KL(q, p)$$

Donde $KL(p, q)$ es la divergencia de Kullback-Leibler para p y q:

$$KL(p, q) = \sum_j p_j log \frac{p_j}{q_j} = \sum_j \frac{E_j}{n} log \frac{E_j}{c_j}$$

Por lo que, el coste global de la divergencia de tamaño se define como:

$$J_A = JSD(p, q) * n$$

El término J_S, es la penalización cuando el tamaño del *cluster* e_j es más pequeño de lo esperado. Para que se cumpla con el valor del *cluster* establecido, se debe aumentar la función de coste con la penalización de los *clusters* i con tamaño más pequeño, mediante la denominada función de coste *NO*, que utiliza el conjunto de puntos de la clase j más cercana con tamaño excedente. El punto *nearest out-class* (*NO*) x deberá minimizar la función *NO*:

$$D_{NO(x,u_i)} = \frac{d(x, u_i)\left[d(x, u_i) + d\left(x, u_j\right)\right]}{d\left(x, u_j\right) d\left(u_i, u_j\right)}$$

$$\forall x \in e_j, i \neq j$$

El conjunto de puntos *nearest out-class* (*NO*) para todas las clases, estará definido por la siguiente medida de penalización:

$$J_S = \sum_{i=1}^{k} \sum_{x \in NO(e_j)} d(x, u_i)$$

Finalmente, J_L es la penalización cuando el tamaño del *cluster* e_j es más grande de lo esperado. Al igual que con los tamaños de *cluster* más pequeños, en este caso también se deberá aumentar la función de coste con la penalización de los tamaños de *clusters* más grandes a través de la función *FI:*

$$D_{FI(x,u_i)} = \frac{d(x,u_i)\left[d(x,u_i) + d(x,u_j)\right]}{d(x,u_j)d(u_i,u_j)}$$

$$\forall x \in e_j, i \neq j$$

En contraste con la penalización por tamaños de *cluster* más pequeños, la penalización para los *clusters* de gran tamaño se traducirá en un movimiento del medoide u_j.

Nótese que la función *FI,* en términos matemáticos, es similar que la función *NO*. Por lo que la aplicación de este mismo criterio para los *clusters* con objetos excedentes hará que los medoides se vean obligados a desplazarse hacia los puntos más lejanos de la clase lo cual se contrapone al objetivo original. Para resolver este particular, se usan los puntos denominados *virtual farthest in-class* (*VFI*). Para un punto *FI* un punto *VFI* se define como:

$$VFI = 2\frac{\sum_{x\in\{e_j - FI(e_j)\}} x}{\left|\{e_j \backslash FI(e_j)\}\right|} - y$$

Donde para un punto *FI* $y \in FI(e_j)$. Al trabajar con una matriz de distancias, no se posee como parámetro de entrada los atributos de cada instancia, por lo que no es posible calcular las coordenadas de un nuevo punto VFI. Por lo tanto, se deberá replantear el algoritmo *K-MeansS*, mediante la penalización aplicada directamente sobre la distancia. Esta representaría una adaptación particular para el trabajo sobre matrices de distancias. Entonces, la penalidad para *clusters* de tamaño más grande puede ser definida como:

$$J_L = \sum_{i=1}^{k} \sum_{x\in VFI(e_j)} d(x,u_i)$$

El pseudocódigo del algoritmo K-MedoidsSC se describe a continuación [17]:

Algoritmo: KmedoidsSC

ENTRADA:
Matriz de Distancias/Disimilaridades DM
Número de *Clusters* k
Tamaño de los *Clusters* $\{e_1, e_2, \dots, e_k\}$,
Máximo Número de Iteraciones T;

SALIDA:
Resultado del Agrupamiento etiquetado como Y de X;

MÉTODO:

1. Inicializa los centroides de los *clusters* $u_1, \dots, u_k$, de manera aleatoria o con cualquier otro método;
2. **Repetir**
3. Asignar las etiquetas Y a los medoides más cercanos
4. Actualizar los medoides de acuerdo a los criterios de la ecuación: $J_{KMS} = J_{KM} + \alpha J_A + \beta J_S + \gamma J_L$
5. **Hasta**
6. Que Y converja o se cumpla el número máximo de iteraciones T
7. **Devolver** Y

Los resultados de los experimentos, empleando el modelo de programación lineal y posteriormente el algoritmo K-MedoidsSC, con *datasets* de prueba muestran que los modelos son factibles y aplicables, y los valores de los resultados tienen un buen rendimiento. La parte experimental de los dos algoritmos, se expone en el Capítulo 6 de este estudio.

6. Metodología y Experimentación

Resumen: Este capítulo tiene por objetivo llevar a cabo diferentes pruebas mediante el análisis experimental de los modelos planteados. Para el caso del *clustering* de documentos, previo a la experimentación con *datasets*, se presenta la metodología que se ha de utilizar durante todo el proceso de *clustering*. Luego se realizarán experimentos sobre *datasets* de *benchmarking*, que no corresponden a conjuntos de datos documentales, con la finalidad de validar los resultados de los dos nuevos algoritmos planteados en las secciones 5.2 y 5.3 de este trabajo. Finalmente, se realizan experimentos sobre *datasets* que permitan agrupar documentos y se realiza la interpretación y discusión, a posteriori, de los resultados obtenidos en ambos casos.

6.1. Metodología

Existen múltiples metodologías para la implementación de técnicas de minería de datos en un proyecto. Según se sugiere en el artículo [24], en muchas de las ocasiones la comunidad dedicada a la explotación masiva de datos no presta mucha atención a aspectos como: la especificación de requerimientos, identificación de técnicas de elicitación, ni se sugiere modelos sistemáticos, lo que posteriormente suele traer complicaciones a los proyectos que buscan el descubrimiento de patrones.

En el artículo "*On clustering validation techniques*" de Halkidi et al. [61] se realiza un resumen de la metodología propuesta por Fayyad et al. [64] para el tratamiento de los datos en los procesos de KDD que es extrapolable hacia los procesos de *clustering*. Los pasos básicos para desarrollar dicho proceso se presentan en la Figura 16.

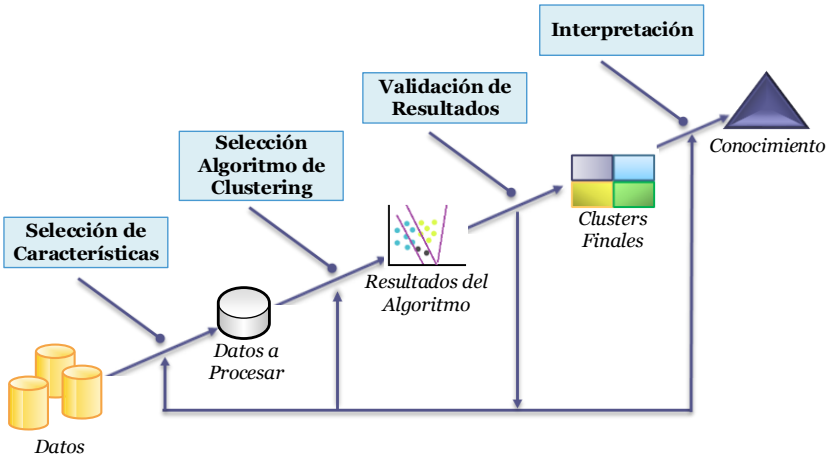

Figura 16. *Etapas del Proceso de Clustering [61]*

6.1.1. Selección de características

Esta etapa tiene por objetivo seleccionar adecuadamente los elementos sobre los que se va a realizar el *clustering* con el fin de codificar la mayor cantidad de información posible en relación con la tarea de interés [61]. El proceso de *clustering* de documentos puede dar lugar a diferentes particiones dependiendo del criterio específico de agrupación que se haya utilizado sobre el conjunto de datos. Por lo tanto, en ocasiones, es necesario

ejecutar un pre-procesamiento en el conjunto de datos antes de realizar una tarea de agrupamiento.

En la metodología para el pre-procesamiento de la información se aplicarán técnicas de procesamiento natural del lenguaje y modelos de recuperación de la información con la finalidad de obtener una matriz de disimilaridades que sirva como entrada para los algoritmos de *clustering* propuestos.

Un artículo científico suele estar compuesto de varios apartados que varían de acuerdo al estilo de la revista, el estilo del autor, la temática, etc. En el caso de los artículos orientados al *machine learning*, los artículos suelen seguir el formato IMRaD (*Introduction, Method, Results and Discussion*), y hay tres secciones que se han adoptado como un estándar de facto en todos ellos: los títulos, las palabras clave y los resúmenes. Estas secciones han sido adoptadas por las revistas científicas, ya que además de unificar en cierta medida la presentación, facilita un rápido análisis sobre el contenido del manuscrito *in extenso* [13]. De acuerdo a [9], estos tres elementos suelen aportar casi el 90% de la información relativa a la temática total del documento y por esta razón se los ha utilizado como fuente de entrada de datos a procesar.

El objetivo de aunar la información obtenida en los títulos, palabras clave y resúmenes, es construir una matriz de disimilaridades, que caracterice las diferencias/similitudes entre los documentos. El esquema de la Figura 17, muestra cuáles son las fases que han de seguirse en el pre-procesamiento de la información para estos tres elementos.

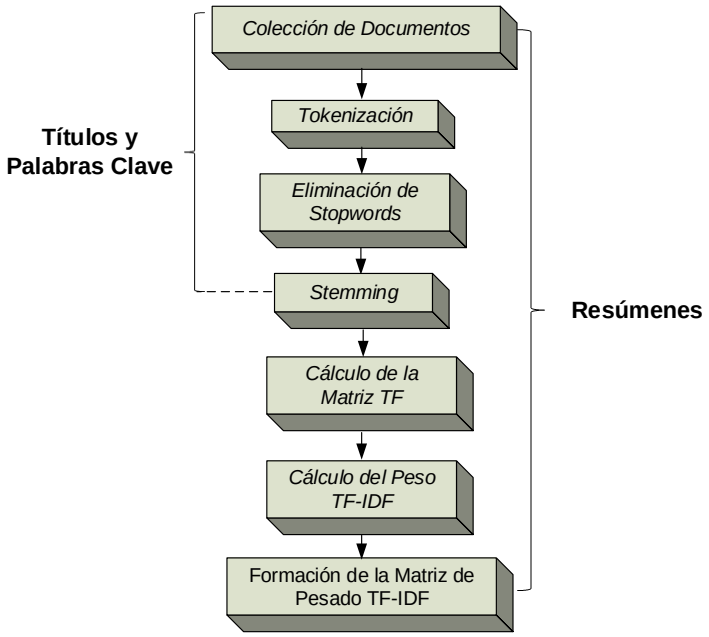

Figura 17. *Fases de Pre-Procesamiento de Información en los Documentos*

Las palabras clave son términos o frases cortas (lexemas) que permiten clasificar y direccionar las entradas en los sistemas de indexación y de recuperación de la información en las bases de datos. En la mayoría de las revistas científicas el número de palabras clave oscila entre 3 y 10 y suelen ser obtenidas de tesauros específicos a la temática [13]. Por otra parte, los títulos en los *papers* suelen tener una media de 8 a 10 palabras [9].

Para estructurar la matriz de disimilaridades de los títulos y de las palabras clave, se ha hecho uso del coeficiente de *Jaccard*, ya que como se ha mencionado estos dos factores suelen tener una menor cantidad de *tokens* y no se justifica el uso de una bolsa de palabras.

El resumen es la sección del manuscrito más consultada y leída y quizás, en algunas ocasiones la única [13]. Esta sección del *paper* no suele tener más de 250 palabras como extensión.

Para cuantificar la semejanza/diferencia entre los resúmenes de los documentos se ha hecho uso del modelo vectorial descrito en la Sección 2.5.1 y se ha usado el índice coseno sobre la matriz de pesado y normalización TF-IDF, para encontrar la matriz de disimilaridades.

Con el cálculo de estas tres matrices de disimilaridades correspondientes a títulos, palabras clave y resúmenes, y por medio de la siguiente ecuación, se han integrado los criterios para conseguir una matriz de disimilaridad total.

$$S'_{TOT} = \theta S'_T + \pi S'_K + \omega S'_A \quad ; \quad \theta + \pi + \omega = 1 \; ; \{\forall\, \theta, \pi, \omega\} \in [0,1]$$

Donde: S'_T: $Matriz\ de\ Disimilaridad\ de\ Títulos$

S'_K: $Matriz\ de\ Disimilaridad\ de\ Palabras\ Clave$

S'_A: $Matriz\ de\ Disimilaridad\ de\ Resúmenes$

De [13] se sabe que los corpus de los resúmenes en un *paper* suelen aportar una mayor cantidad de información útil que los otros dos elementos (título y palabras clave), por lo que la ponderación de este coeficiente en este trabajo será mayor.

El diagrama de bloques de la Figura 18, resume el proceso de formación de la matriz de disimilaridades antes expuesto.

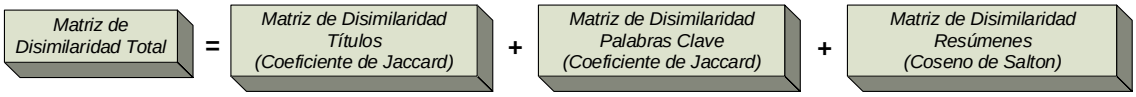

Figura 18. *Matriz de Disimilaridad Total*

6.1.2. Selección del algoritmo de clustering

Este paso hace referencia a la elección de un algoritmo que dé como resultado la definición de un buen esquema de *clustering* para un conjunto de datos. La medida de proximidad y el criterio de agrupamiento caracterizan principalmente a un algoritmo de *clustering*, pero también es importante definir un esquema de agrupación que se ajuste a las características intrínsecas del *dataset* [61].

En cuanto al algoritmo de *clustering*, se ha elegido técnicas de agrupamiento enfocadas en dos características: el manejo de espacios altamente dimensionales y la susceptibilidad a la incorporación de restricciones adicionales.

Para resolver el problema de *clustering* de documentos con restricciones de tamaño se han planteado dos nuevos algoritmos, que se describieron en las secciones 5.2 y 5.3. Los dos algoritmos poseen algo en común y es que operan directamente con la matriz de similaridad de un conjunto de datos en lugar del espacio original de características.

6.1.3. Validación de los resultados

Los resultados del algoritmo de agrupamiento deben ser verificados por medio de criterios y técnicas adecuadas. Puesto que en los algoritmos de *clustering* las agrupaciones se definen sin un conocimiento previo de los resultados, es necesario que la partición final de los datos se valide con alguna metodología que sea independiente al método de *clustering* [61].

En ausencia de información para aplicar validación externa, a priori, se podría pensar que utilizar un índice de validación relativa, es más deseable que el uso de un índice de validación interno, ya que estos primeros tratan de explotar la redundancia de los datos. Sin embargo, la mayoría de los resultados [4] han demostrado que incluso para modelos simples los índices relativos no producen una mejora sustancial y aumentan potencialmente los costes computacionales. Por esta razón para este estudio, se utilizará el coeficiente de silueta como método de validación interna de los resultados, aunque en algunos de los casos, en los que la información estuvo disponible, se ha utilizado el *ground truth* de la asignación de las clases para realizar validación externa con los índices *ARI*, *AMI* y *NMI*.

6.1.4. Interpretación de los resultados

En muchos casos, los expertos en el área de *data mining* tienen que integrar los resultados de la agrupación con otras pruebas experimentales y análisis con el fin de llegar a la conclusión correcta [61].

En las Secciones 6.2 y 6.3 de este trabajo se han realizado algunas inferencias a partir de los resultados del proceso de *clustering* y del análisis de los datos originales sobre los que se trabaja.

6.2. Experimentación para Validar los Algoritmos CSCLP y K-MedoidsSC

Previo a la experimentación sobre *datasets* de documentos, que son el objeto de este trabajo, es necesario realizar pruebas para evaluar la efectividad y rendimiento de los dos nuevos algoritmos heurísticos planteados. Para ello se han utilizado tres *datasets* de *benchmarking*, muy conocidos, de la UCI Machine Learning Repository (*University California Irvine*). Las características de los *datasets* se resumen en la Tabla 9 y sus descripciones completas pueden ser encontradas en [38][39][40].

	Iris	Wine	Seeds
Número de Instancias (Objetos)	150	178	210
Número de Atributos	4	13	7
Clases (Número de Grupos)	3	3	3

Tabla 9. *Principales Metadatos de los Datasets Iris, Wine y Seeds*

Los algoritmos propuestos se fundamentan en la restricción de los tamaños de los *k clusters*. En la Tabla 10 se puede observar una comparativa de los diferentes tamaños de los grupos, si se aplican sobre los *datasets:* algoritmos sin restricciones de tamaño (AHC – FPA y *K-Medoids*) y las dos nuevas propuestas con restricciones de tamaño (CSCLP y K-MedoidsSC) en contraste con el valor real del tamaño de los *clusters* (determinado a través de los datos propios de los *datasets*). En el caso del algoritmo AHC – FPA se ha hecho un corte a un determinado umbral para obtener el número de *clusters* deseados.

Múltiples estudios relacionados al *clustering* con restricciones de tamaño, han demostrado que el algoritmo *K-Means*, así como muchas de sus variaciones, obtienen mejores resultados si se utiliza un método diferente al aleatorio para la selección de los puntos iniciales [49]. Los puntos iniciales para los algoritmos *K-Medoids*, CSCLP y K-MedoidsSC han sido escogidos mediante la *Farthest Neighbor Technique* (algoritmo de *González*) y el algoritmo de *Buckshot*.

Siguiendo el esquema determinado por el algoritmo de *Buckshot*, para la selección de los k puntos iniciales, el algoritmo jerárquico debería ejecutarse sobre muestras de tamaño $\sqrt{kn}$ objetos, lo que implicaría tener muestras de 21 (Iris), 23 (Wine) y 25 (Seeds) objetos.

El muestreo usado por *Buckshot* se justifica para reducir la complejidad computacional y cuando los *datasets* tienen tamaños muy grandes. Debido a que los *datasets* escogidos tienen un tamaño n no superior a las 210 instancias, se ha decidido trabajar con la totalidad de n objetos y evitar ese comportamiento no determinístico que se le suele atribuir al componente muestral de esta técnica.

Algoritmo	Farthest Neighbor Technique			Algoritmo de Buckshot		
	Iris	Wine	Seeds	Iris	Wine	Seeds
AHC – FPA*	(50,29,71)	(10,32,136)	(8,42,160)	(50,29,71)	(10,32,136)	(8,42,160)
K-Medoids	(50,55,45)	(60,41,77)	(74,70,66)	(50,45,55)	(88,74,16)	(89,64,57)
CSCLP	(50,50,50)	(59,71,48)	(70,70,70)	(50,50,50)	(59,71,48)	(70,70,70)
K-MedoidsSC	(50,50,50)	(59,71,48)	(70,70,70)	(50,50,50)	(59,71,48)	(70,70,70)
Datos Reales	(50,50,50)	(59,71,48)	(70,70,70)	(50,50,50)	(59,71,48)	(70,70,70)
ID Objetos: Puntos Iniciales	[23,75,119]	[19,118,15]	[23,107,204]	[39,98,113]	[135,80,109]	[48,107,152]
* No utiliza ninguna técnica para seleccionar los puntos iniciales						

Tabla 10. *Resultados de los Tamaños de los Clusters Datasets Iris, Wine y Seeds*

Los resultados, expuestos en la Tabla 10, demuestran que tanto AHC – FPA como *K-Medoids*, incumplen con el valor esperado en el tamaño del *cluster*. De manera contraria, las dos nuevas propuestas, tal como se esperaba cumplen con las restricciones de tamaño impuestas por el usuario, coincidiendo de manera perfecta con los tamaños reales de los *clusters*.

Una vez que se ha demostrado la factibilidad de las dos nuevas propuestas en cuanto al cumplimiento de los tamaños de los *clusters* es importante validar los resultados del agrupamiento y su rendimiento. Para medir el rendimiento del proceso de *clustering*, se han incluido cuatro medidas de validación externas e internas: *Adjusted Rand Index* (ARI), *Normalized Mutual Information* (NMI), *Adjusted Mutual Information* (AMI) y el coeficiente de silueta $S(i)$.

Datos	Algoritmo	Farthest Neighbor Technique				Algoritmo de Buckshot			
		ARI	AMI	NMI	$S(i)$	ARI	AMI	NMI	$S(i)$
Iris	AHC – FPA*	0.674	0.735	0.760	0.632	0.674	0.735	0.760	0.632
	K-Medoids	0.904	0.897	0.899	0.714	0.904	0.897	0.899	0.714
	CSCLP	0.886	0.861	0.862	0.711	0.886	0.861	0.862	0.685
	K-MedoidsSC	0.628	0.626	0.630	0.451	0.851	0.829	0.831	0.651
Wine	AHC – FPA*	0.059	0.137	0.186	0.525	0.059	0.137	0.186	0.525
	K-Medoids	0.347	0.363	0.373	0.513	0.208	0.196	0.221	0.519
	CSCLP	0.215	0.207	0.215	0.359	0.300	0.288	0.296	0.334
	K-MedoidsSC	0.131	0.165	0.174	0.201	0.255	0.248	0.256	0.266
Seeds	AHC – FPA*	0.223	0.286	0.379	0.539	0.223	0.286	0.379	0.539
	K-Medoids	0.412	0.419	0.424	0.321	0.366	0.415	0.424	0.388
	CSCLP	0.429	0.389	0.394	0.264	0.483	0.455	0.460	0.322
	K-MedoidsSC	0.267	0.257	0.263	0.165	0.540	0.470	0.474	0.207
* No utiliza ninguna técnica para seleccionar los puntos iniciales									

Tabla 11. *Validación del Clustering en los Datasets Iris, Wine y Seeds*

Los resultados son notablemente adecuados en el *dataset* Iris de *Fisher*, donde la distribución de los grupos es bastante heterogénea y en la que se puede distinguir de manera clara dos grupos [50]. Pero también los resultados en los otros *datasets* Wine y Seeds han generado valores de los índices internos y externos apropiados, lo que demuestra que las propuestas algorítmicas planteadas en este trabajo son válidas.

Las Figuras 19, 20 y 21 plasman de manera visual los agrupamientos resultantes para los algoritmos con restricciones de tamaño CSCLP y K-MedoidsSC, en los conjuntos de datos Iris, Wine y Seeds.

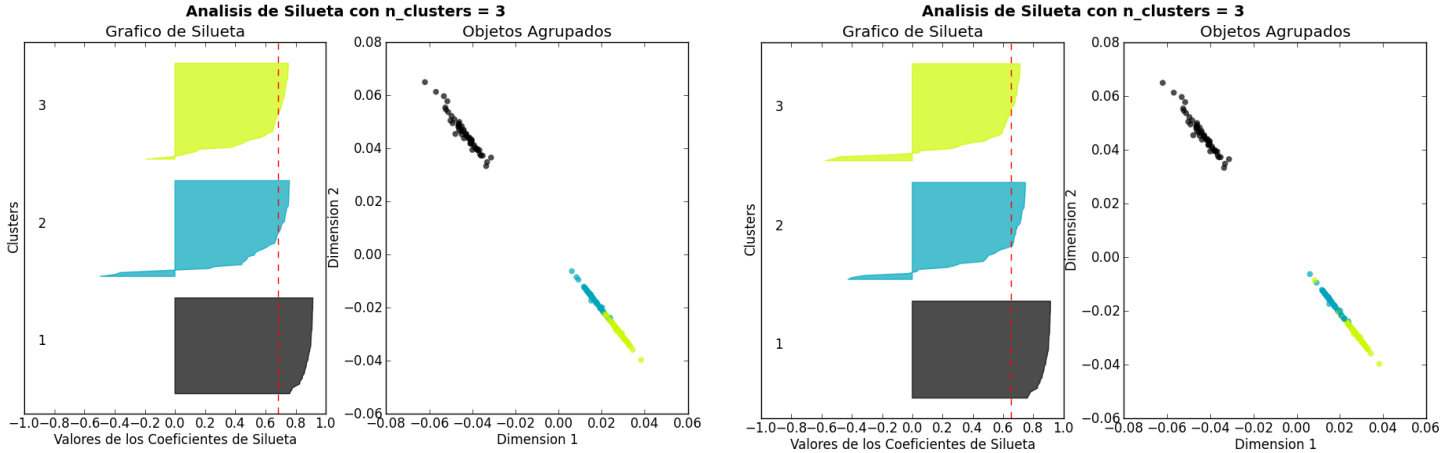

Figura 19. *Agrupamiento Dataset Iris, Algoritmos CSCLP y K-MedoidsSC, Puntos Iniciales: Buckshot*

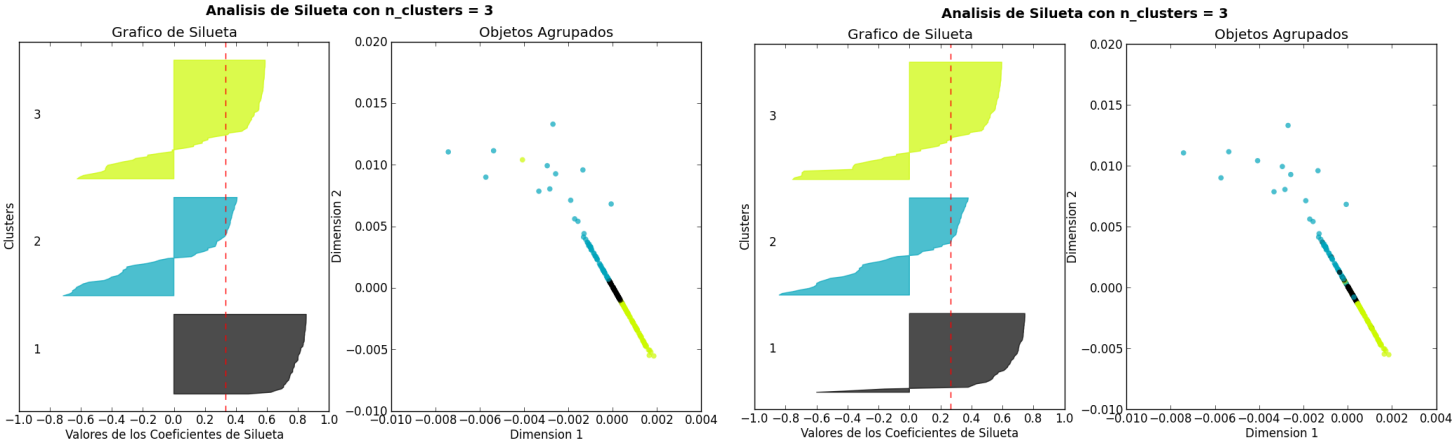

Figura 20. *Agrupamiento Dataset Wine, Algoritmos CSCLP y K-MedoidsSC, Puntos Iniciales: Buckshot*

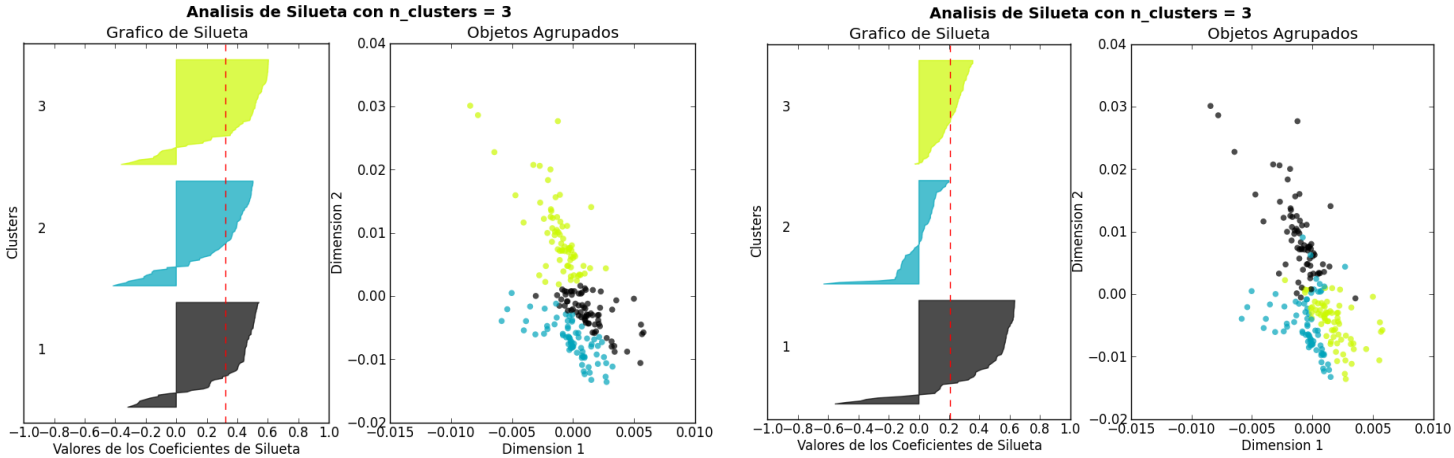

Figura 21. *Agrupamiento Dataset Seeds, Algoritmos CSCLP y K-MedoidsSC, Puntos Iniciales: Buckshot*

6.3. Experimentación en Datasets Documentales

Extendiendo el estudio de los algoritmos planteados y una vez validados los dos modelos heurísticos para *clustering* con restricciones de tamaño, se ha procedido a llevar estos dos procedimientos al campo del agrupamiento de documentos. Para poder ejecutar las pruebas correspondientes se han usado tres *datasets* de conferencias cuya temática es el *Machine Learning*.

En la etapa inicial de selección de características, donde se realiza la extracción de datos y el pre-procesamiento de los mismos, se hizo uso de Python[1] 2.7.5 mediante su distribución Anaconda 1.6.1, en el ambiente de desarrollo Spyder (*Scientific Python Development EnviRonment*) 1.2.1.

El primer *dataset* corresponde a la "*13th International Conference on Machine Learning and Applications (ICMLA-14)*" que se realizó durante los días 3 a 5 de Diciembre de 2014 en Detroit, USA. Para la construcción de este primer *dataset* se utilizó técnicas de extracción de datos desde su sitio web (*Web Scraping*) [30], mediante la librería *Beautiful Soup* de Python. Esta librería se define como una herramienta para analizar documentos y páginas web en formato HTML y XML que permite la descarga de la información contenida en ellas creando un árbol sintáctico de los elementos del documento. La versión más reciente de *Beautiful Soup* es la 4.4.1 y está disponible para las versiones de Python 2.6+ y Python 3+ [31].

Como resultado del *web scrapping* y una vez homogeneizados los datos se construyó un fichero con extensión *.csv* sobre el cual posteriormente se realizaron las tareas de filtrado de datos y extracción del conocimiento.

Los otros dos *datasets* fueron obtenidos del repositorio de la UCI [32] [33] que corresponden a la *"27th Conference on Artificial Intelligence of Association for the Advancement of Artificial Intelligence (AAAI-13)"* que se realizó durante los días 14 a 18 de Diciembre de 2013 en Bellevue, Washington, USA y la *"28th Conference on Artificial Intelligence (AAAI-14)"* desarrollada del 27 al 31 de Julio, 2014 en Quebec, Canadá, respectivamente.

Los tres *datasets* contienen ciertas características de lo los *papers* aceptados para sus respectivas conferencias. Los *datasets* ICMLA-14 y AAAI-13, incluyen solo los *papers* de las sesiones principales, mientras que el *dataset* AAAI-14 aúna a los *papers* de todas las sesiones. Al igual que con la *ICMLA-14,* para su posterior procesamiento se usó la extensión *.csv* en los *datasets* AAAI-13 y AAAI-14. La Tabla 12 muestra los principales metadatos de los tres *datasets* utilizados:

		ICMLA-14	**AAAI-13**	**AAAI-14**
Características del Dataset		Multivariante	Multivariante	Multivariante
Número de Instancias		70	150	399
Número de Atributos		5	5	6
Atributos	**Paper ID**	Numérico	N/A	N/A
	Title	Texto Libre	Texto Libre	Texto Libre
	Keywords	Texto Libre	Texto Libre	Texto Libre
	Abstract	Texto Libre	Texto Libre	Texto Libre
	Conference Session	Texto Libre	N/A	N/A
	Topics	N/A	Categórico	Categórico
	High-Level keywords	N/A	Categórico	Categórico
	Authors	N/A	N/A	Categórico
	Groups	N/A	N/A	Categórico

Tabla 12. *Principales Metadatos de los Datasets ICMLA-14, AAAI-13 y AAAI-14*

Es importante conocer que la primera fila de cualquiera de los tres *datasets* no contiene ningún documento, sino únicamente las etiquetas descriptivas de los atributos. Los

[1] **Python:** es un lenguaje de programación multiparadigma que se está consolidando como uno de los lenguajes más populares para computación científica e industrial, gracias a su naturaleza interactiva de alto nivel y su ecosistema de maduración, es una opción atractiva para el desarrollo de algoritmos y análisis exploratorio de datos. Posee licencia de código abierto *Python Software Foundation License* que ofrece compatibilidad con la licencia pública general de GNU a partir de la versión 2.1.1 [36].

atributos que han sido utilizados para el agrupamiento con restricciones de tamaño son: *Title, Keywords y Abstract*, ya que son los elementos comunes para las tres conferencias. Es menester aclarar que los corpus de las tres conferencias sobre los cuales se ha trabajado están en idioma inglés.

Antes de construir la matriz de disimilaridades que servirá como parámetro de entrada en el proceso de *clustering*, se debe realizar el procesamiento de los atributos de acuerdo a la metodología descrita en la Sección 6.1.1. Para el procesamiento de los contenidos textuales de los atributos, se ha utilizado varias librerías de Python.

En la limpieza de los datos, a través de las funciones `re` y `lower` se han removido todos los caracteres que no sean alfanuméricos y transformado todas las palabras a minúsculas, respectivamente. La tokenización se ha llevado a cabo con el uso de la función `split`, que transforma los corpus a una lista cuyo contenido serán los *tokens* de las palabras incluidas en los títulos, palabras clave y resúmenes de los documentos.

Por otra parte, mediante la función `stopwords.words('english')` de la librería NLTK, se han eliminado las *stopwords* más frecuentes del diccionario inglés, ya que per se no aportan contenido para el análisis, sino más bien todo lo contrario. El conjunto de *stopwords* suman un total de 127 términos que se los puede observar en el Anexo I.

NLTK *(Natural Language ToolKit)*, es un conjunto de programas de código abierto, que trabaja en el campo de la lingüística y el análisis de datos textuales. La librería NLTK permite aplicar las técnicas y algoritmos más comunes en NLP para, por ejemplo, generar métricas, determinar la frecuencia de términos, encontrar la polaridad positiva/negativa de frases y textos, etc. y todo esto mediante la invocación desde cualquier programa escrito en lenguaje Python [51].

Ídem que con las *stopwords,* NLTK provee una función `PorterStemmer` que implementa el Algoritmo de *Porter* para el proceso de *stemming*. Este proceso servirá de entrada para el posterior cálculo de frecuencia de términos en la matriz de bolsa de palabras, que como se mencionó en la Sección 2.5.1, suele ser muy dispersa. Python ayuda a optimizar el procesamiento de esta matriz, no solo en memoria sino también para acelerar las operaciones algebraicas de matriz/vector, mediante el paquete `scipy.sparse` de la librería *Scipy*. Debido a que la cantidad de documentos en las tres conferencias analizadas no tiene una diversidad muy amplia de palabras y el tamaño máximo de la matriz bolsa de palabras fue de: 398 x 6360 para el *dataset* AAAI-14, no fue necesario el uso de dicho paquete, ya que no implicaba un tiempo de procesamiento alto.

Scipy es una librería *open source* de herramientas numéricas y algoritmos matemáticos para Python que contiene módulos para tareas de ciencia e ingeniería. Posee, en particular, algunos paquetes entre los cuales destacan: *Numpy*, *IPython*, *Matplotlib* y *Pandas* [37].

Sobre la bolsa de palabras se ha ejecutado un proceso de normalización y pesado TF-IDF, con la ayuda de la función `TfidfTransformer` y el método `transformer.fit_transform()` de la librería *Scikit Learn*.

Scikit Learn es una librería libre de Python con licencia BSD, especializada en *machine learning*, que contiene varios modelos para tareas de clasificación, regresión, *clustering*, reducción de la dimensionalidad, selección de modelos y pre-procesamiento de los datos. Está diseñada para interoperar con otras librerías de Python de carácter numérico y científico, como *Numpy, Scipy* y también librerías de gráficos como *Matplotlib*, su última versión es la 0.17.1 [36].

Para estructurar la matriz de disimilaridades se ha hecho uso del esquema descrito en la Sección 6.1.1. En la librería *Scipy* encontramos la función `spatial.distance.cosine` que realiza el cálculo del índice coseno de manera automática entre los documentos.

Ya que la ecuación que describe a la matriz de disimilaridades total posee tres parámetros θ, π y ω, que pueden ser ajustados, se han realizado varias pruebas con diferentes valores para refinar dicha matriz que servirá como entrada en los diferentes procesos de *clustering*. Como se mencionó en la Sección 6.1.1, los corpus de los resúmenes en un *paper* suelen aportar una mayor cantidad de información útil que el título y las palabras clave, por lo que su ponderación ha sido mayor. La Tabla 13 describe tres escenarios contemplados para la experimentación.

	θ	π	ω
Escenario I (ES-1)	0.1	0.35	0.55
Escenario II (ES-2)	0.15	0.25	0.6
Escenario III (ES-3)	0.05	0.2	0.75

Tabla 13. *Valores θ, π y ω en la Matriz Total de Distancias*

En la Figura 22, se puede observar el *dataset* ICMLA-14 graficado en $\mathbb{R}^2$ mediante escalamiento multidimensional con la función `sklearn.manifold.MDS` de *Scikit Learn*, bajo el método `mds.fit_transform` y con modelo de escalamiento métrico. Para esto se ha utilizado los valores de la matriz de distancias de los 69 documentos que conformaron todas las sesiones. Esto es muy útil para observar sobre un plano a los diferentes objetos por medio de un cambio dimensional.

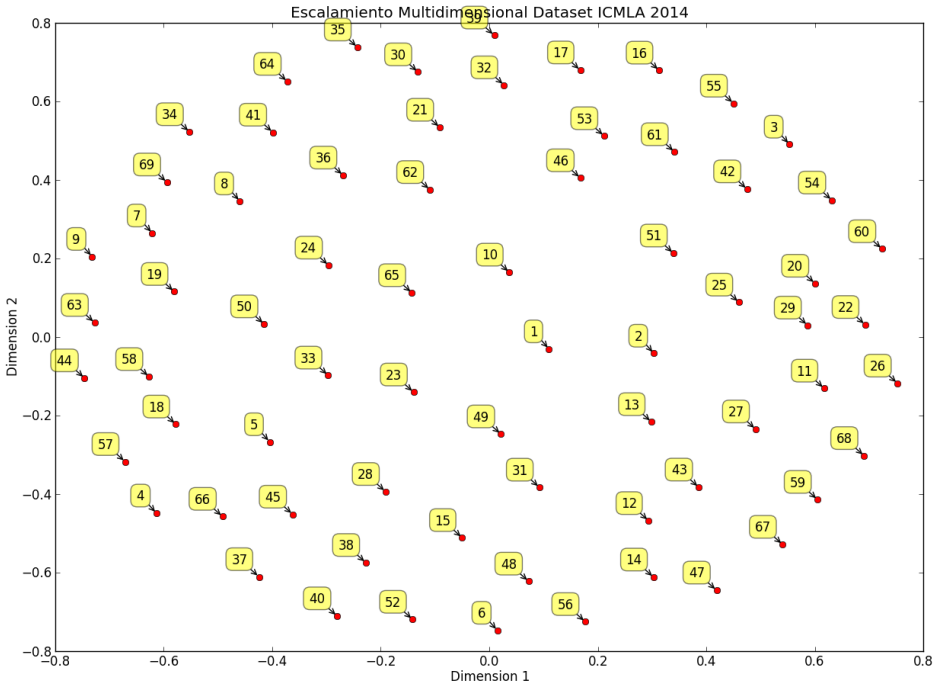

Figura 22. *Escalamiento Multidimensional de los Documentos de la Conferencia ICMLA-14 (ES-1)*

A priori, se puede observar en la Figura 22 que, no existe ningún patrón de aglutinamiento por densidad, en un área específica del MDS.

Es importante notar que los *datasets* AAAI-13 y AAAI-14, no poseen como atributos la organización de las sesiones de la conferencia por temática, ergo, se desconoce el tamaño de los *clusters*. Realizando nuevamente un *web scrapping* de las páginas oficiales de las

conferencias [34] [35], se extrajo la organización final de las sesiones temáticas de las conferencias. La estructura final de todas las conferencias y sus respectivas sesiones se pueden observar en los Anexos II, III y IV.

La disposición de los documentos en la conferencia ICMLA-14, de acuerdo a la organización final por sesiones temáticas, se la puede observar en la Figura 23. Dicha gráfica representa un mapa de calor de la matriz de disimilaridades del *dataset*, construida a través de las herramientas que proporciona la librería *Pylab*. Se puede observar que existe una concentración de los puntos sobre una sección de la diagonal de la matriz (documentos 20 a 29), lo que en principio implicaría un mayor grado de relación temática entre estos documentos. En el Anexo I se puede consultar que este rango de documentos (20-29) corresponde a las sesiones denominadas "Neural Networks I y II".

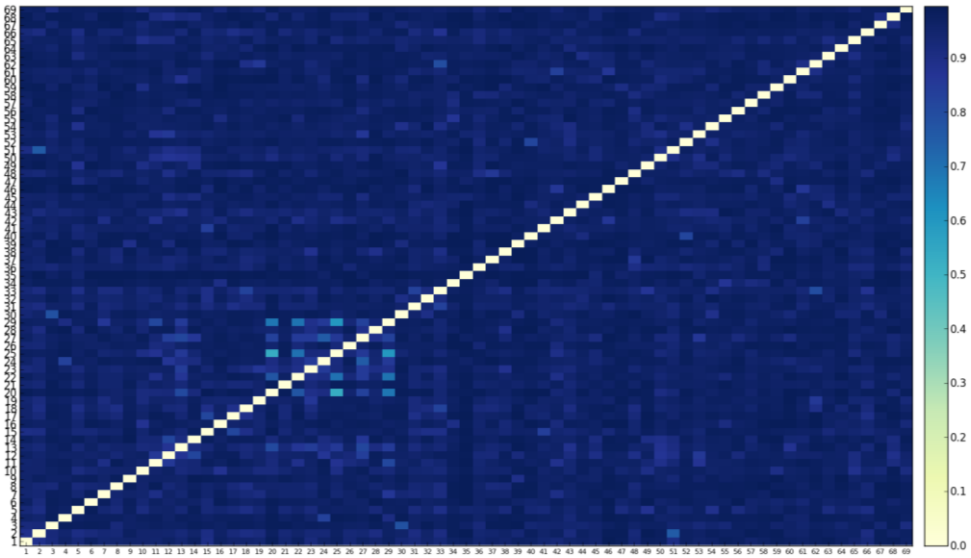

Figura 23. *Matriz de Disimilaridades Dataset ICMLA-14 (ES-1)*

Las entradas que recibe el algoritmo de *clustering* con restricciones de tamaño son: el número de total de documentos n, el número de *clusters* k y el tamaño de los *clusters* E_j. La Tabla 14 resume los valores de entrada requeridos para este proceso.

Dataset	n	k	E_j
ICMLA-14	69	14	(5, 5, 5, 4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)
AAAI-13	149	39	(4, 4, 4, 4, 4, 4, 4, 1, 3, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 4, 3, 3, 4, 4, 4, 3, 4)
AAAI-14	398	23	(34, 11, 4, 7, 14, 4, 41, 9, 2, 7, 26, 32, 18, 3, 11, 7, 77, 25, 16, 10, 21, 19)

Tabla 14. *Valores de Entrada para el Clustering, Datasets: ICMLA-14, AAAI-13 y AAAI-14*

Si se revisa la línea temática y distribución de las sesiones en la que se agruparon los documentos de la conferencia ICMLA-14, se puede observar una singularidad. Existen sesiones que tienen el mismo nombre pero han sido divididas en dos bloques y son: Neural Networks I y II, Information Retrieval I y II y Machine Learning I y II. A priori se desconoce la razón, pero se asume que obedece a cuestiones de logística ya que una sesión en general no excede de los 5 documentos. La Tabla 15, muestra los tamaños de los grupos si se unifican las sesiones con igual nombre, en el *dataset* ICMLA-14.

Dataset	n	k	E_j
ICMLA-14	69	11	(5, 5, 5, 9, 10, 5, 5, 5, 10, 5, 5)

Tabla 15. *Estructura de las Sesiones de la Conferencia ICMLA-14 (11 grupos)*

Sobre la matriz de disimilaridades y con el conocimiento previo del número de *k clusters*, se pueden ejecutar algoritmos de agrupamiento tradicionales en los que no se restringe el tamaño de los *clusters*. A manera de ejemplo, la Figura 24, muestra el dendograma formado sobre el *dataset* ICMLA-14, utilizando el algoritmo *farthest point algorithm* con la ayuda de la función `scipy.cluster.hierarchy`. Como resultado de la agrupación jerárquica y mediante una técnica de poda de árbol se ha realizado un agrupamiento en 14 sesiones, con la finalidad de observar cuáles son los tamaños que un algoritmo de este tipo formaría. Esto último, se lo hizo mediante la función `fcluster` de la librería *Scipy* de Python.

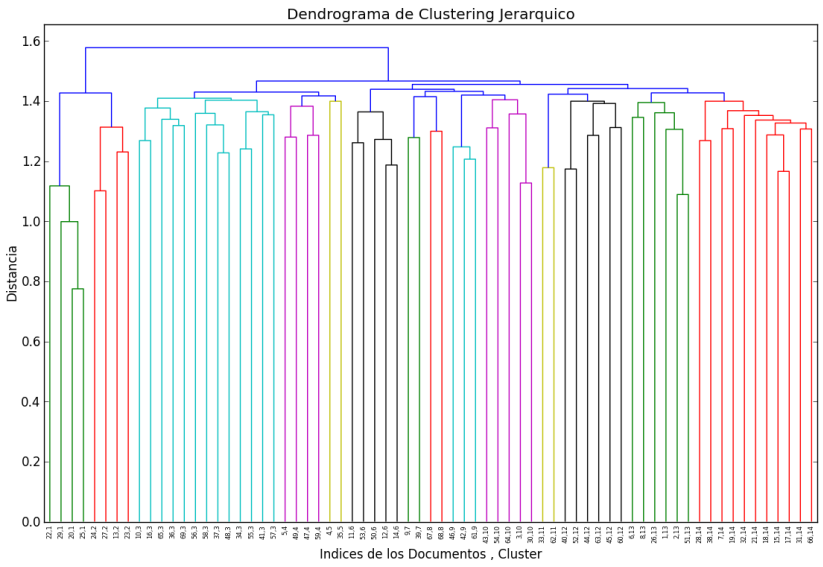

Figura 24. *Dendograma - Farthest Point Algorithm, Dataset ICMLA-14 (ES-1) (14 grupos)*

En la Figura 25, se puede observar en el mapa de calor de la matriz de disimilaridades el resultado de la agrupación de los valores correspondientes al algoritmo AHC-FPA para 14 grupos. Se observa que los tamaños de los *clusters* no necesariamente corresponden a la asignación prevista en la Tabla 14.

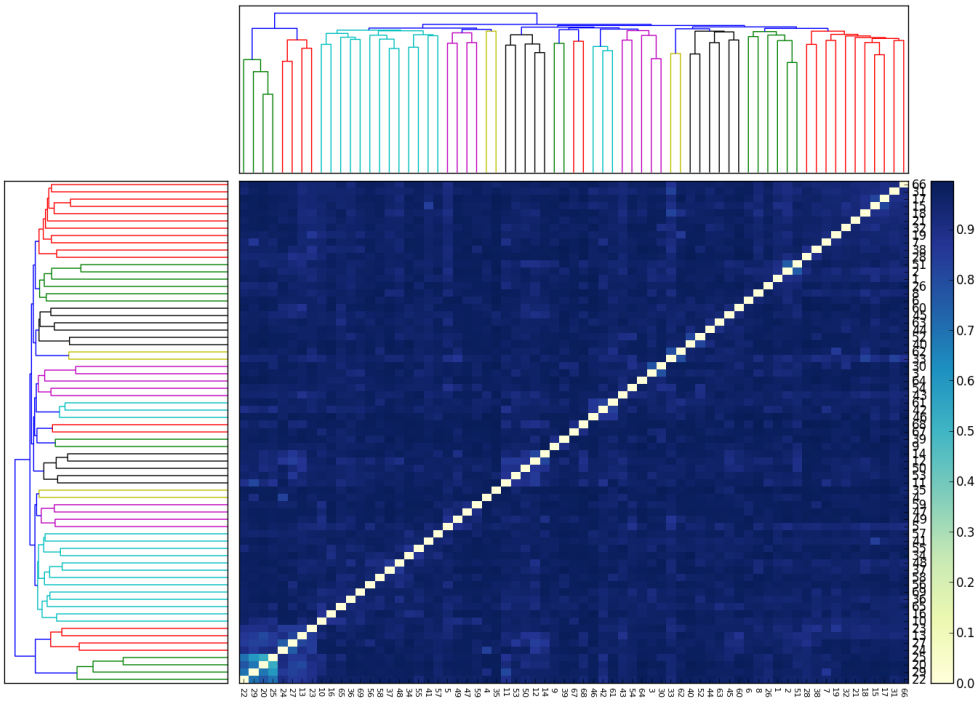

Figura 25. *Matriz Disimilaridades y Dendograma AHC-FPA, Dataset ICMLA-14 (ES-1) (14 grupos)*

Como resultado del proceso de *clustering*, nuevamente se puede ver que, existe un *cluster* que destaca por su similaridad en la diagonal de la matriz y que corresponde al grupo de documentos: 20, 22, 25 y 29. Este grupo a su vez, se encuentra próximo al *cluster* de documentos: 23, 24, 27 y 13. Ambos grupos, y como ya se mencionó anteriormente, corresponden mayoritariamente a *papers* de las sesiones "Neural Networks I y II". Por lo que se puede afirmar que este conjunto de documentos son un patrón de *clustering* en el *dataset* ICMLA-14.

Aunque el uso de algoritmos jerárquicos no es donde estriba y se centra este estudio, si es importante conocer cuál es su distribución, ya que los centros de los grupos arrojados por este algoritmo servirán de entrada para la selección de los puntos iniciales en el algoritmo de *Buckshot*. Al igual que en los experimentos realizados en la Sección 6.2, se trabajará con los puntos iniciales generados por el algoritmo de *González* y por el algoritmo *Buckshot* sin muestreo, para valores de k=11 y k=14, en los tres escenarios supuestos. Los resultados de este proceso se sumarizan en la Tabla 16:

	ID Objetos: Puntos Iniciales		
	k = 14		
	ES-1	**ES-2**	**ES-3**
González	[35, 17, 68, 62, 30, 9, 34, 38, 4, 47, 40, 64, 37, 26]	[35, 17, 68, 62, 30, 9, 38, 34, 4, 47, 40, 37, 64, 46]	[35, 17, 68, 62, 30, 37, 9, 38, 4, 47, 64, 40, 34, 20]
Buckshot	[25, 27, 48, 5, 4, 12, 9, 67, 42, 30, 33, 40, 2, 15]	[25, 67, 39, 4, 59, 40, 42, 34, 10, 27, 12, 2, 62, 15]	[25, 14, 40, 61, 3, 12, 34, 5, 9, 47, 27, 62, 63, 2]
	k = 11		
	ES-1	**ES-2**	**ES-3**
González	[35, 17, 68, 62, 30, 9, 34, 38, 4, 47, 40]	[35, 17, 68, 62, 30, 9, 38, 34, 4, 47, 40]	[35, 17, 68, 62, 30, 37, 9, 38, 4, 47, 64]
Buckshot	[25, 27, 48, 5, 12, 9, 30, 33, 40, 2, 15]	[25, 36, 4, 40, 42, 10, 27, 12, 2, 62, 15]	[25, 14, 61, 3, 12, 10, 9, 47, 27, 62, 2]

Tabla 16. *Puntos Iniciales, Dataset ICMLA-14*

Como ya se ha señalado en la Sección 5.2, el nuevo algoritmo CSCLP se basa en restricciones del tipo *cannot-link* (puntos iniciales) y su optimización se sustenta en programación lineal entera binaria PLEB. Python posee librerías para la resolución de problemas de optimización matemática a través de programación lineal como PYOMO, pero dado que la resolución del problema de *clustering* se resume en un problema de PLEB con entradas matriciales se ha decidido utilizar el paquete computacional Matlab[2] en su versión 7.10.0 (R2010a), que se especializa en el manejo de matrices. Por lo tanto, fue necesario la exportación de la matriz de disimilaridades obtenida en Python, hacia el otro ambiente de trabajo (Matlab).

Matlab posee una función llamada `bintprog`, dentro del *toolbox Optimization*, que permite resolver problemas de PLEB de manera simple. Esta función busca el mínimo de una función, dentro de un problema de programación lineal y se describe en forma canónica como:

$$\min_x f \rightarrow x \text{ sujeto a: } \begin{cases} x \in \{0,1\} \\ A.x \leq b \\ Aeq.x = beq \\ lb \leq x \leq ub \end{cases}$$

[2] **Matlab:** abreviatura de MATrix LABoratory "Laboratorio de Matrices", es una herramienta de software matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación propio (lenguaje M). Está disponible para las plataformas Unix, Windows, Mac OS X y GNU/Linux. Se define como un lenguaje de cálculo técnico de alto nivel con un entorno interactivo para el desarrollo de algoritmos, la visualización de datos, el análisis de datos, etc.

Donde:

f, x, b, beq, lb y ub son vectores, mientras que A y Aeq son matrices. Aunque f, lb y ub pueden ser declarados en forma de matrices también.

La sintaxis utilizada para la función `bintprog` es:

```
[x,fval] = bintprog(f,A,b,Aeq,beq)
```

Siendo:

- **`f:`** la función objetivo a minimizar
- **`A:`** las restricciones en forma de inecuación y matriz
- **`b:`** las restricciones en forma de inecuación y vector
- **`Aeq:`** las restricciones en forma de ecuación y matriz
- **`beq:`** las restricciones en forma de ecuación y vector

Esta función devolverá los valores de `x` que representan la pertenencia de los documentos a los *clusters* y `fval` que es la función objetivo evaluada con los coeficientes resueltos por el programa.

Se define la pertenencia de un documento (objeto) a uno u otro grupo, a través de un modelo matricial booleano, como el descrito en la Sección 5.2, donde la sumatoria de cada fila respresenta la pertenecia de un documento a un único *cluster* y la sumatoria de cada columna representa el número de documentos pertenecientes a un grupo, es decir, la restricción especificada del tamaño de grupo E_j. Para este caso en particular, las restricciones se definen únicamente en forma de ecuación, por lo que la sintaxis específica para el problema sería:

```
[x,fval] = bintprog(f,[],[],Aeq,beq)
```

Es importante acotar que las restricciones se relacionan con el valor de salida `x`, mediante la expresión $Aeq.x = beq$. Se debe notar que, la sintaxis sólo permite la entrada de una restricción, pero en el planteamiento se cuenta con dos restricciones: la de tamaño y la de pertenencia. Por esta razón, tanto para el componente matricial `Aeq` como para el componente vectorial `beq`, se debió generar un arreglo $Aeq_{(1)}x = beq_{(1)}$ que haga cumplir la restricción de tamaño y un $Aeq_{(2)}x = beq_{(2)}$ que permita incorporar la restricción de pertenencia, para luego concatenar ambas restricciones y formar un $Aeq.x = beq$.

Con los valores de `x` entregados por Matlab, ya se puede construir la matriz de pertenencia de los documentos a los diferentes *clusters*, con la singularidad de que los tamaños de los *clusters* E_j ya están restringidos.

Los resultados de la Tabla 17, han demostrado nuevamente que el enfoque CSCLP cumple con los tamaños de los *clusters* asignados, de manera inequívoca.

Algoritmo	**k = 14**					
	González			**Buckshot**		
	ES-1	**ES-2**	**ES-3**	**ES-1**	**ES-2**	**ES-3**
CSCLP	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)
Tamaño clusters (E_j)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)

Algoritmo	k = 11					
	González			Buckshot		
	ES-1	ES-2	ES-3	ES-1	ES-2	ES-3
CSCLP	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)
Tamaño clusters (E_j)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)

Tabla 17. *Resultados de los Tamaños de los Clusters, Dataset ICMLA-14, Algoritmo CSCLP*

Uno de los mayores retos a los que se expone un investigador, en el análisis *clustering*, es la subjetividad inmersa en el proceso. Al ser un tipo de aprendizaje no supervisado, existen varias maneras de encontrar una solución de agrupamiento dependiendo del criterio de *clustering* que se haya establecido [1]. Un reto aún mayor es definir un coeficiente, adecuado, que permita validar los resultados del proceso de agrupamiento.

En la Tabla 18, se muestran los valores medios del coeficiente de silueta $S(i)$ y los índices de validación externa *ARI*, *AMI* y *NMI*. Todos estos índices, son medidas que se habían propuesto, en la Sección 6.1.3, como parte de la metodología para la validación del *clustering*. Es importante aclarar que para definir los índices de validación externa, se asume que el *ground truth* de la asignación de las clases, provista por los organizadores, es la solución "ideal" o "real".

En la conferencia ICMLA-2014, así como también para las conferencias AAAI-2013 y AAAI-2014, se desconoce los criterios bajo los cuales se realizaron los agrupamientos, ya que como se ha mencionado el *clustering* es subjetivo y es posible que los organizadores hayan establecido más de un criterio para determinar la similaridad o afinidad entre *papers* de una misma sesión (grupo). Pero, dado que los tamaños de los grupos y la organización de las sesiones de las conferencias, siguen una línea temática específica, se asume que los *papers* han sido agrupados por similitud entre sus temas.

En ocasiones, los organizadores de las conferencias por factores externos tales como: número de asistentes a una sesión, limitaciones en el espacio físico del recinto, problemas de arribo de los conferencistas en los días estipulados para la presentación de los *papers*, etc. se ven obligados a modificar la estructura de las sesiones. Por lo que las etiquetas de las clases (*true labels*) del agrupamiento "ideal" se ven alteradas y esto a su vez también perturba al criterio de similitud, establecido en un principio.

El cálculo de los valores *ARI*, *AMI* y *NMI* se lo realizó con la ayuda de la función `metrics` embebida en la librería *Scikit Learn* de Python.

CSCLP	Escena	González				Buckshot			
		ARI	AMI	NMI	$S(i)$	ARI	AMI	NMI	$S(i)$
k = 14	ES-1	0.0399	0.0742	0.4937	-0.2528	0.0946	0.1457	0.5328	-0.2206
	ES-2	0.0477	0.0795	0.4966	-0.3011	0.0789	0.1265	0.5223	-0.2657
	ES-3	0.0555	0.1021	0.5090	-0.2657	0.0243	0.0463	0.4785	-0.2496
CSCLP	Escena	González				Buckshot			
		ARI	AMI	NMI	$S(i)$	ARI	AMI	NMI	$S(i)$
k = 11	ES-1	0.0687	0.1012	0.4030	-0.2301	0.0527	0.0845	0.3919	-0.2511
	ES-2	0.0740	0.1061	0.4063	-0.2283	0.0581	0.1054	0.4058	-0.2489
	ES-3	0.0847	0.1521	0.4368	-0.2273	0.1006	0.1134	0.4112	-0.2595

Tabla 18. *Validación del Clustering en el Dataset ICMLA, Algoritmo CSCLP*

De los resultados obtenidos se puede concluir que, en función de los valores del coeficiente de silueta y de los índices externos, el agrupamiento con los mejores resultados es el que utiliza los valores θ, π y ω propuestos en el escenario 3 (ES-3) y con el método de inicialización de los k puntos del algoritmo de *González*, cuando k=11. Y para k=14, la mejor *performance* se obtiene en el ES-1 con *Buckshot*.

Si se analizan los resultados se puede observar que los grupos que se han formado en algunos de los casos son especialmente heterogéneos y en la mayoría de los casos se han formado grupos de manera artificial con valores de inercia inter-*cluster* e intra-*cluster* muy bajos. Esto se debe principalmente a que el valor de k es demasiado alto en comparación con el tamaño n del *dataset*.

El mapa de calor de la matriz de disimilaridades, con mejores prestaciones, de acuerdo a los resultados obtenidos por el algoritmo de *clustering* CSCLP con k=14, aparece en la Figura 26. Aquí, se observa nuevamente el patrón de aglutinamiento definido por las sesiones "Neural Networks I y II". Este resultado heurístico, denota que a priori el método CSCLP funciona de manera adecuada para el agrupamiento de documentos.

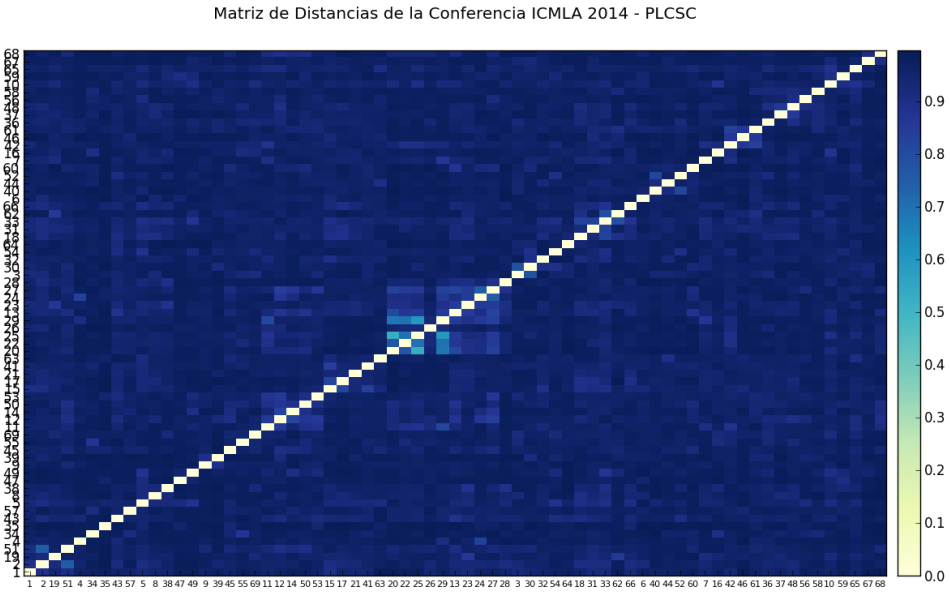

Figura 26. *Matriz de Disimilaridades Dataset ICMLA-14, Algoritmo CSCLP (ES-1), Buckshot*

La Figura 27 representa la distribución de los documentos en cada *cluster*, tomado en cuenta los índices de los documentos frente al número de *clusters*.

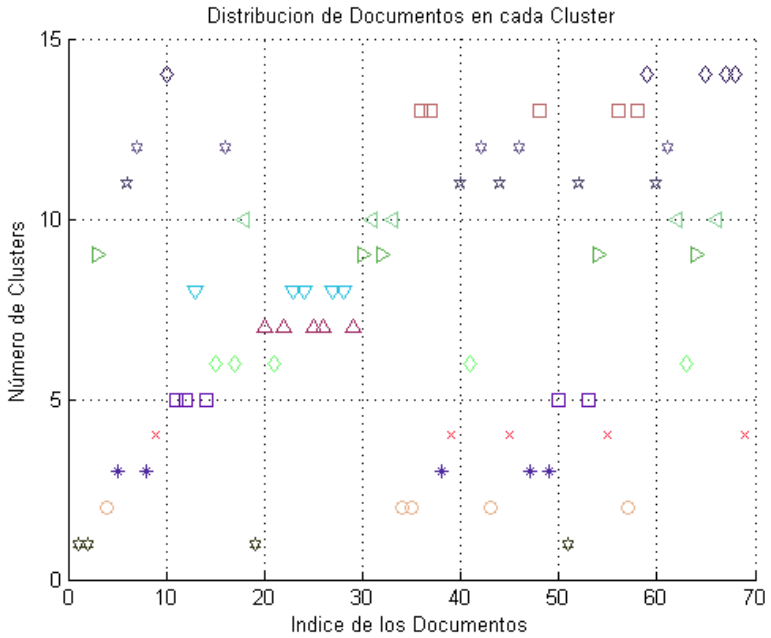

Figura 27. *Distribución de Documentos, Dataset ICMLA-14, Algoritmo CSCLP (ES-1), Buckshot*

En la Figura 28(a), se puede ver los coeficientes de silueta y la distribución de *clusters* en un espacio bidimensional para el escenario de *clustering* que mejores prestaciones ha presentado, cuando k=14. Cada color representa a un *cluster* diferente y el tamaño de la silueta indica la cantidad de documentos que conforman dicho *cluster*. La Figura 28(b) muestra los mismos resultados pero para el resultado "real" de las agrupaciones de la conferencia ICMLA-14.

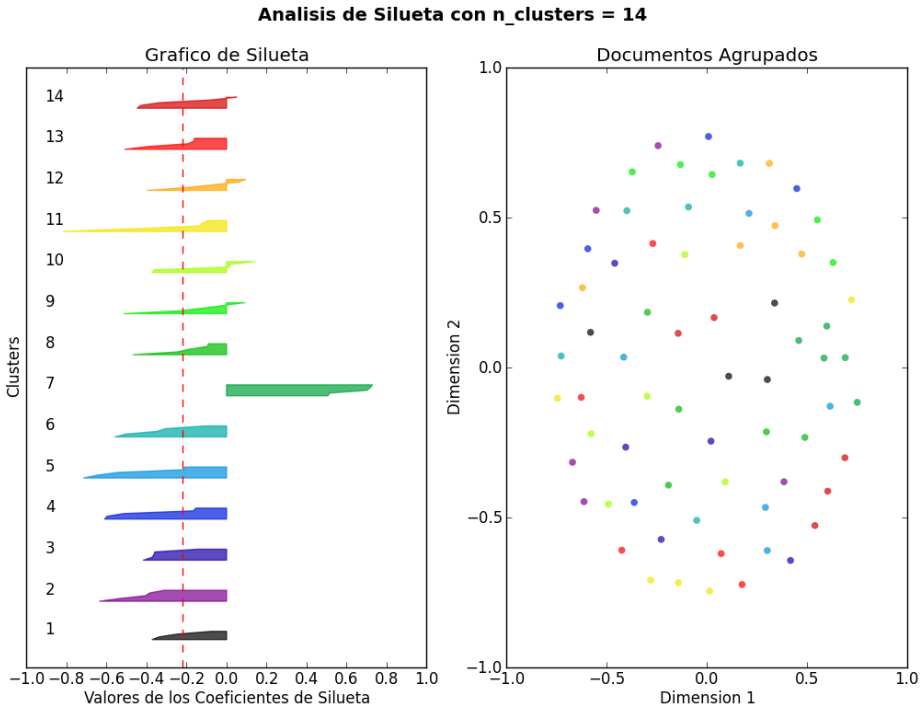

Figura 28(a). *Agrupamiento Dataset ICMLA-14, Algoritmo CSCLP (ES-1), Buckshot*

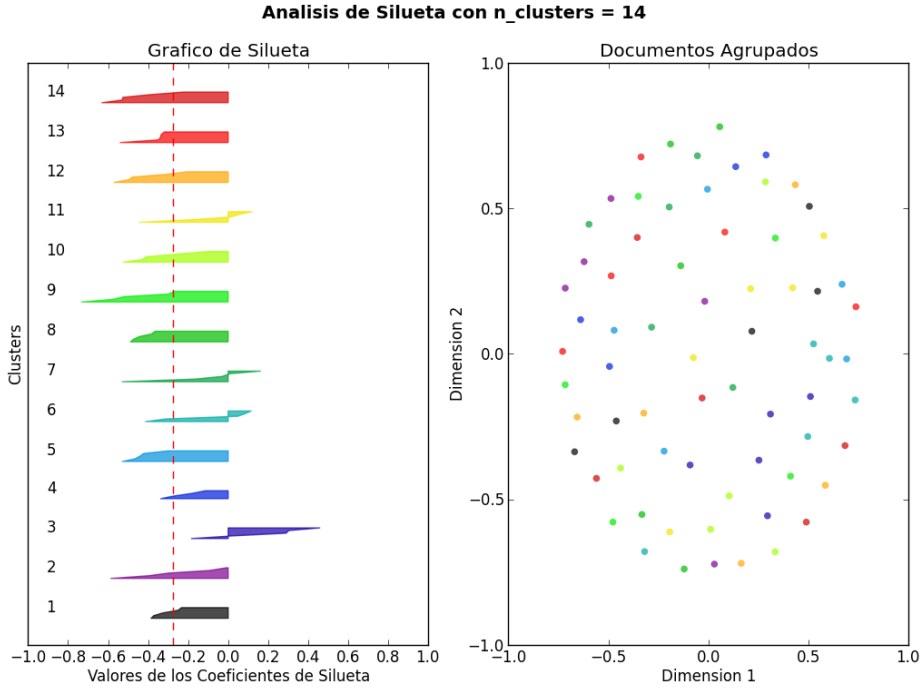

Figura 29(b). *Agrupamiento Dataset ICMLA-14, Solución de la Organización*

Analizando únicamente el componente interno de los *clusters*, expresado por el valor medio del índice $S(i)$, se puede concluir que los resultados de la agrupación con CSCLP tienen mejor rendimiento que la agrupación tomada como referencia. Así, el índice $S(i)$ para el mejor escenario es de -0.2206, mientras que el $S(i)$ para la agrupación definida por la organización de la conferencia es de -0.2768.

Ídem al método CSCLP, el algoritmo K-MedoidsSC recibe como entradas las restricciones de los tamaños para cada grupo $E_j = \{e_1, e_2, \dots, e_k\}$. El algoritmo ha sido configurado con un número máximo de iteraciones de 50, para el caso en el que el algoritmo no encuentre un punto de convergencia. El algoritmo K-MedoidsSC fue programado de manera íntegra en Python. En la Tabla 19, se resumen los resultados arrojados por el algoritmo K-MedoidsSC, en cuanto al tamaño de los grupos.

Algoritmo	**k = 14**					
	González			**Buckshot**		
	ES-1	**ES-2**	**ES-3**	**ES-1**	**ES-2**	**ES-3**
K-MedoidsSC	(4, 5, 11, 0, 0, 5, 10, 7, 5, 9, 4, 0, 0, 9)	(4, 5, 0, 12, 0, 5, 6, 5, 10, 5, 5, 7, 0, 5)	(4, 5, 7, 5, 5, 5, 6, 5, 9, 5, 5, 0, 8, 0)	(4, 7, 6, 5, 6, 0, 0, 5, 6, 0, 5, 5, 15, 5)	(4, 5, 10, 2, 0, 0, 6, 5, 5, 5, 12, 5, 5, 5)	(31, 5, 5, 3, 0, 0, 0, 5, 0, 5, 0, 10, 5, 0)
Tamaños de los clusters (E_j)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)	(4, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5, 5)
Algoritmo	**k = 11**					
	González			**Buckshot**		
	ES-1	**ES-2**	**ES-3**	**ES-1**	**ES-2**	**ES-3**
K-MedoidsSC	(5, 5, 5, 8, 0, 5, 5, 10, 9, 8, 9)	(5, 5, 5, 5, 0, 5, 5, 11, 9, 9, 10)	(5, 4, 16, 3, 5, 5, 5, 5, 7, 8, 6)	(5, 0, 5, 0, 11, 6, 5, 5, 14, 9, 9)	(5, 6, 5, 13, 6, 4, 0, 5, 8, 10, 7)	(7, 5, 5, 5, 2, 5, 5, 24, 11, 0, 0)
Tamaños de los clusters (E_j)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)	(5, 5, 5, 5, 5, 5, 5, 5, 9, 10, 10)

Tabla 19. *Resultados de los Tamaños de los Clusters, Dataset ICMLA-14, Algoritmo K-MedoidsSC*

Si bien los tamaños de los *clusters* encontrados por el algoritmo K-MedoidsSC no son diametralmente opuestos a los que habían sido definidos como parámetros de entrada, estos presentan algunos valores diferentes. Como se mencionó en la Sección 3.3.2, una de las desventajas del algoritmo *K-Means*, así como de sus variaciones *K-Medoids*, *K-Modes*, etc., es que presenta complicaciones cuando los *datasets* son especialmente homogéneos y cuando los valores de n y k son altos.

Tomando en cuenta que para el *dataset* ICMLA-14 el tamaño total del conjunto de datos es n=69 con valores de k=11 y k=14. Este número de grupos k, representan al 16% y 20% del total de objetos contenidos dentro del *dataset,* respectivamente. Para *datasets* de *benchmarking* como el Iris este porcentaje es de apenas el 2%. Esta es una de las razones por las que se obtienen valores del índice $S(i)$ tan bajos y para el caso en particular, el algoritmo no llega a cumplir con los tamaños de los *clusters* impuestos. Por lo que se podría afirmar que para *datasets* con un número bajo de n y valores altos de k, el algoritmo incumple con los tamaños de los *clusters*, para garantizar valores de cohesión adecuados dentro de los *clusters*.

La Tabla 20, resume los resultados de los índices de validación externa e interna, obtenidos con el algoritmo K-MedoidsSC.

K-MedoidsSC	**Escena**	**González**				**Buckshot**			
		ARI	**AMI**	**NMI**	$S(i)$	**ARI**	**AMI**	**NMI**	$S(i)$
k = 14	ES-1	0.0080	0.0180	0.3826	-0.2407	0.0347	0.0662	0.4358	-0.2396
	ES-2	0.0225	0.0300	0.4120	-0.2835	0.0078	0.0158	0.4168	-0.2678
	ES-3	0.0055	0.0075	0.4198	-0.2546	0.0011	0.0083	0.3164	-0.2615

K-MedoidsSC	Escena	González				Buckshot			
		ARI	AMI	NMI	$S(i)$	ARI	AMI	NMI	$S(i)$
k = 11	ES-1	0.0432	0.0715	0.3665	-0.2574	0.0465	0.0607	0.3384	-0.2317
	ES-2	0.1190	0.1745	0.4384	-0.2213	0.0850	0.1209	0.4005	-0.2184
	ES-3	0.0278	0.0521	0.3674	-0.2927	0.0209	0.0349	0.3125	-0.2966

Tabla 20. *Validación del Clustering en el Dataset ICMLA, Algoritmo K-MedoidsSC*

Del análisis de los resultados se desprende que: tanto para valores de k=14 como para k=11, el algoritmo de selección de puntos iniciales *Buckshot*, genera mejores resultados. Para 14 grupos la matriz de disimilaridades se comporta de mejor manera con los coeficientes sugeridos en el escenario 1 (ES-1) y para 11 grupos la selección recae en el escenario 2 (ES-2). Nótese como el valor del índice $S(i)$ se incrementa cuando se disminuye el valor de k.

Por el análisis gráfico de la Figura 29, se puede observar y concluir que al igual que en el caso del método CSCLP, el algoritmo K-MedoidsSC propuesto identifica de manera clara a las sesiones "Neural Networks I y II" y las agrupa en *clusters*.

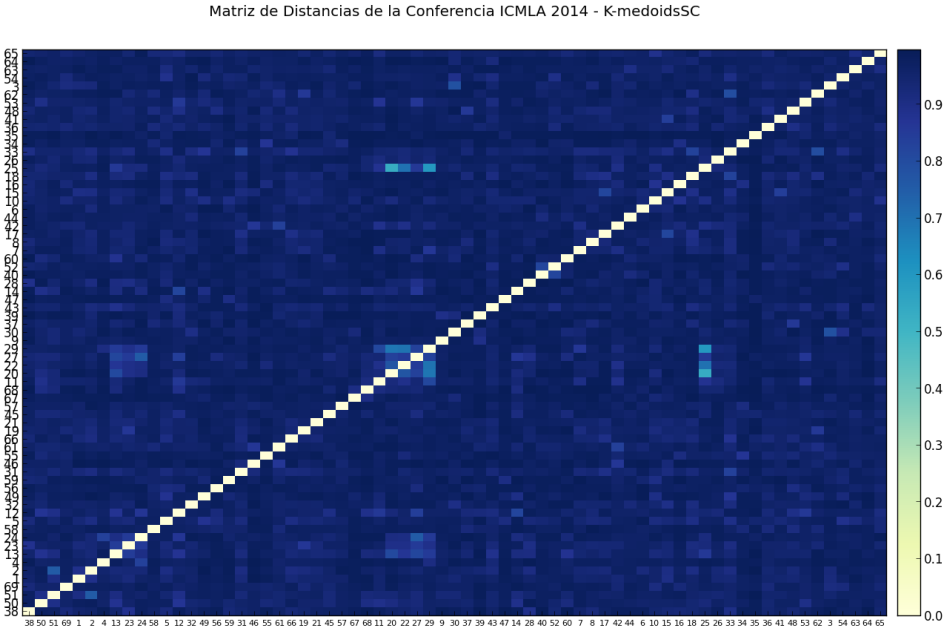

Figura 30. *Matriz de Disimilaridades Dataset ICMLA-14, Algoritmo K-MedoidsSC (ES-1), Buckshot*

En la Figura 30, se aprecia a los coeficientes de silueta y la distribución de *clusters* para el agrupamiento con mejores prestaciones, cuando k=14.

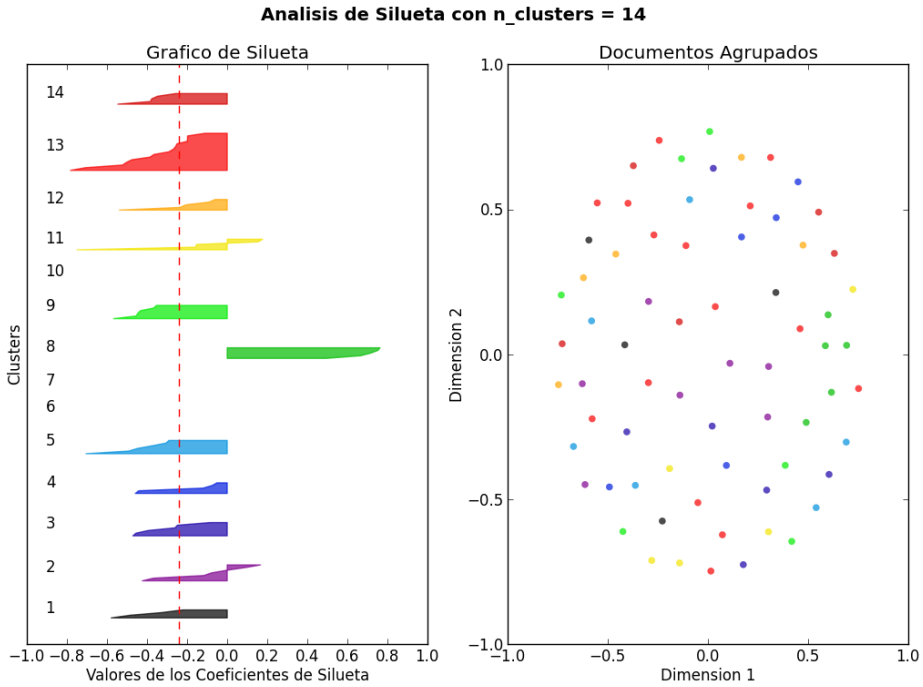

Figura 31. *Agrupamiento Dataset ICMLA-14, Algoritmo K-MedoidsSC (ES-1), Buckshot*

De las pruebas empíricas ejecutadas con los algoritmos CSCLP y K-MedoidsSC, se deduce que la combinación entre el escenario 1 (ES-1) y el algoritmo de *Buckshot*, genera una mejor performance en los agrupamientos. Por lo que este esquema será tomado como referencia para posteriores experimentos.

En la Sección 3.4, se mencionaron algunos de los métodos para seleccionar el número de *clusters*. Una forma trivial de conocer cuál es el número más adecuado de *clusters* en un *dataset*, es utilizar un algoritmo de *clustering* sin restricciones de tamaño y encontrar mediante variaciones de k en un intervalo de I_1 pruebas, el mejor valor de k en función del coeficiente de silueta. Este intervalo I_1, está definido para:

$$\{\forall k \in \mathbb{Z}^+ \wedge\ k \neq 0\ \wedge\ k > 1\}$$

El programa computacional R[3], mediante la librería `cluster` y su función `pam` permite generar un agrupamiento particional del *dataset* con el algoritmo PAM (*Partitioning Around Medoids* o *K-Medoids*). Por otra parte, la librería `fpc` y la función "pamk.best" escoge la mejor cantidad de k grupos, basado en la optimización del valor medio del coeficiente de silueta. Así, si se ejecuta este proceso sobre el *dataset* ICMLA-14, se obtiene que el valor más adecuado es un $k = 2$.

Esto quiere decir que, aunque para las necesidades de organización de las sesiones temáticas en las conferencias las restricciones de los tamaños deban tener valores de 11 o 14, el *dataset* ICMLA-14 se adecúa mejor cuando se definen solo dos grupos. La Figura 31 representa lo antes expuesto.

[3] **R:** es un entorno y lenguaje de programación con un enfoque al análisis estadístico. R es una implementación de software libre del lenguaje S pero con soporte de alcance estático y es parte del sistema GNU que se distribuye bajo la licencia GNU GPL. Está disponible para sistemas operativos Windows, Macintosh, Unix y GNU/Linux. Se trata de uno de los lenguajes más utilizados en investigación: estadística, biomédica, bioinformática, de matemáticas financieras y en el campo de la minería de datos. Permite cargar diferentes bibliotecas o paquetes con funcionalidades de cálculo o graficación.

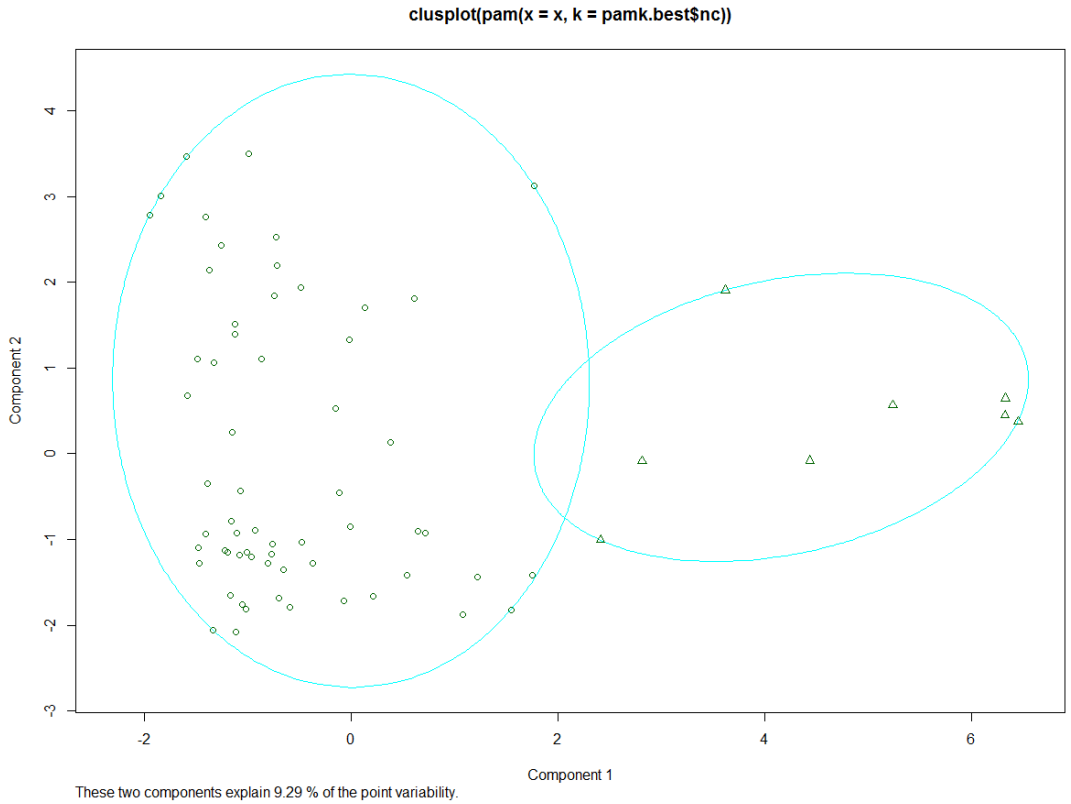

Figura 32. *Selección Número de Clusters, Dataset ICMLA-14*

Se puede observar en el gráfico de la Figura 31, que los tamaños de los dos grupos son de 8 y 61 documentos, respectivamente. Si se ejecutan las pruebas para este número de grupos, se puede observar como el índice de validación interna $S(i)$ aumenta en 0.2561 en el algoritmo CSCLP en comparación con las pruebas ejecutadas en la Tabla 18 y en 0.2755 en el algoritmo K-MedoidsSC con respecto a los ensayos de la Tabla 20. Además, el algoritmo K-MedoidsSC cumple con las restricciones de tamaño establecidas, algo que no sucedió cuando se ejecutó el algoritmo con valores de k elevados con respecto al número de instancias n.

Valores	Algoritmo	E_j	c_j	$S(i)$	ID Papers en el Dataset
k=2 **Puntos Iniciales Buckshot: [25, 27]**	CSCLP	(8, 61)	(8, 61)	0.0355	c_1 = [10, 11, 20, 22, 25, 29, 30, 42] c_2 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24, 26, 27, 28, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]
	K-MedoidsSC	(8, 61)	(8, 61)	0.0359	c_1 = [7, 10, 20, 22, 25, 26, 29, 42] c_2 = [1, 2, 3, 4, 5, 6, 8, 9, 11, 12, 13, 14, 15, 16, 17, 18, 19, 21, 23, 24, 27, 28, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69]

Tabla 21. *Clustering en el Dataset ICMLA, Algoritmos CSCLP y K-MedoidsSC, k=2*

Las Figuras 32(a) y 32(b) plasman, respectivamente: las matrices de disimilaridades y los análisis de silueta con la distribución de los *clusters* para ambos algoritmos.

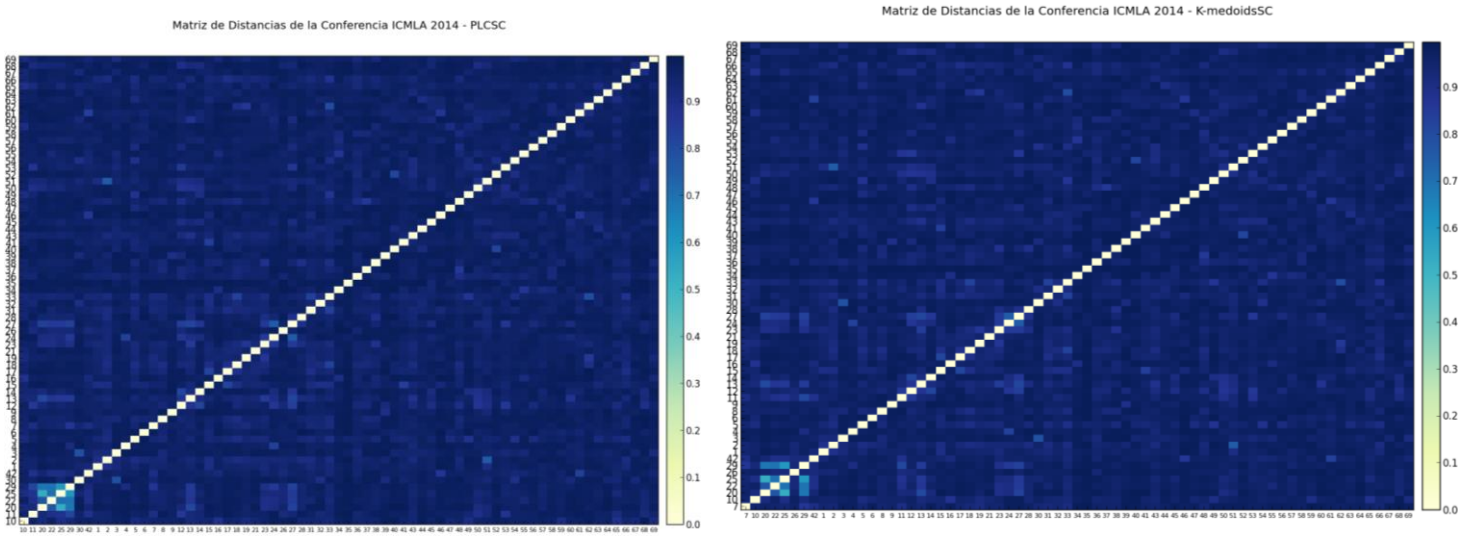

Figura 33(a). *Matriz de Disimilaridades Dataset ICMLA-14, Algoritmos CSCLP y K-MedoidsSC, k=2*

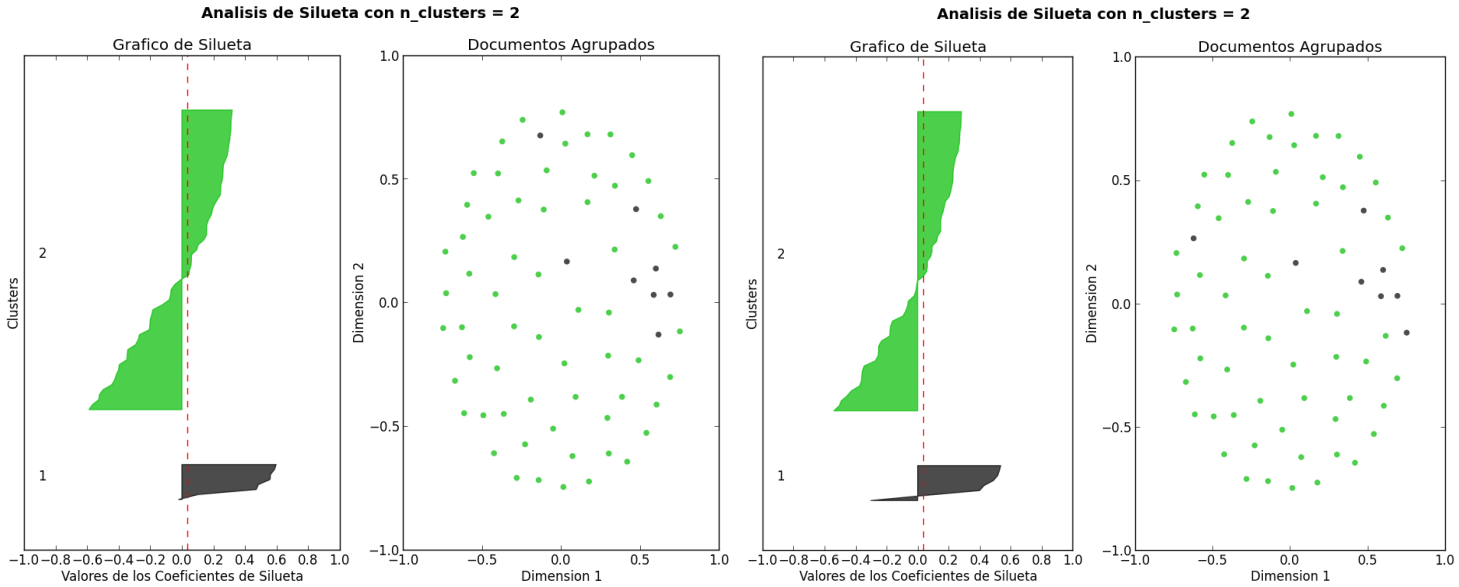

Figura 34(b). *Agrupamiento Dataset ICMLA-14, Algoritmos CSCLP y K-MedoidsSC, k=2*

Para los dos algoritmos, CSCLP y K-MedoidsSC, el patrón de *papers* con temática "Neural Networks I y II" ha sido detectado y agrupado. Se nota también una pequeña mejoría en el valor de $S(i)$ del algoritmo K-MedoidsSC con respecto al algoritmo CSCLP.

Si se revisa la estructura del *dataset* ICMLA-14, expuesta en el Anexo II, se puede apreciar que los *papers* a presentarse en la conferencia fueron distribuidos en tres días. Por lo que otra posible variación del agrupamiento, para dar solución al problema de *clustering* de documentos dentro de sesiones temáticas, puede ser con una estructura por días, como la que se presenta en la Tabla 22. Es imperativo destacar que, nuevamente, los dos algoritmos cumplen con los valores establecidos como restricciones de tamaño para los tres grupos. Por lo que, y en base a estos resultados, se puede concluir que los dos algoritmos son opciones factibles dentro de la propuesta "*Clustering de documentos con restricciones de tamaño*".

Valores	Algoritmo	E_j	c_j	ID Papers en el *Dataset*
k=3 **Es-1** **Puntos Iniciales Buckshot: [10, 25, 33]**	CSCLP	(15,20,34)	(15,20,34)	c_1 = [6, 9, 10, 14, 16, 19, 28, 36, 47, 56, 59, 64, 65, 67, 69] c_2 = [11, 12, 13, 20, 22, 24, 25, 26, 27, 29, 30, 39, 42, 43, 45, 50, 51, 53, 57, 68] c_3 = [1, 2, 3, 4, 5, 7, 8, 15, 17, 18, 21, 23, 31, 32, 33, 34, 35, 37, 38, 40, 41, 44, 46, 48, 49, 52, 54, 55, 58, 60, 61, 62, 63, 66]
	K-MedoidsSC	(15,20,34)	(15,20,34)	c_1 = [3, 9, 12, 14, 20, 24, 27, 38, 45, 47, 49, 50, 51, 60, 64] c_2 = [1, 6, 11, 13, 16, 22, 23, 25, 26, 29, 30, 36, 39, 40, 42, 53, 57, 59, 68, 69] c_3 = [2, 4, 5, 7, 8, 10, 15, 17, 18, 19, 21, 28, 31, 32, 33, 34, 35, 37, 41, 43, 44, 46, 48, 52, 54, 55, 56, 58, 61, 62, 63, 65, 66, 67]

Tabla 22. *Resultados Tamaños de Clusters, Dataset ICMLA-14, Algoritmos CSCLP y K-MedoidsSC, k=3*

La Tabla 23, por otra parte, recopila los resultados de los ensayos ejecutados para poder validar a los algoritmos. La validación interna, como era de esperarse, ha disminuido cuantitativamente con respecto a la configuración de $k = 2$, ya que como se acabó de mencionar el *dataset* ICMLA-14 se adecúa mejor cuando se definen solo dos grupos.

Algoritmo	ARI	AMI	NMI	$S(i)$
CSCLP	0.0399	0.0441	0.0724	-0.0094
K-MedoidsSC	0.0969	0.0542	0.0821	-0.0371

Tabla 23. *Validación del Clustering en Dataset ICMLA-14, Algoritmos CSCLP y K-MedoidsSC, k=3*

Al igual que con las otras prueba experimentales, las Figuras 33(a) y 33(b) muestran las matrices de disimilaridades y los análisis de silueta con la distribución de los *clusters* para los dos algoritmos en análisis.

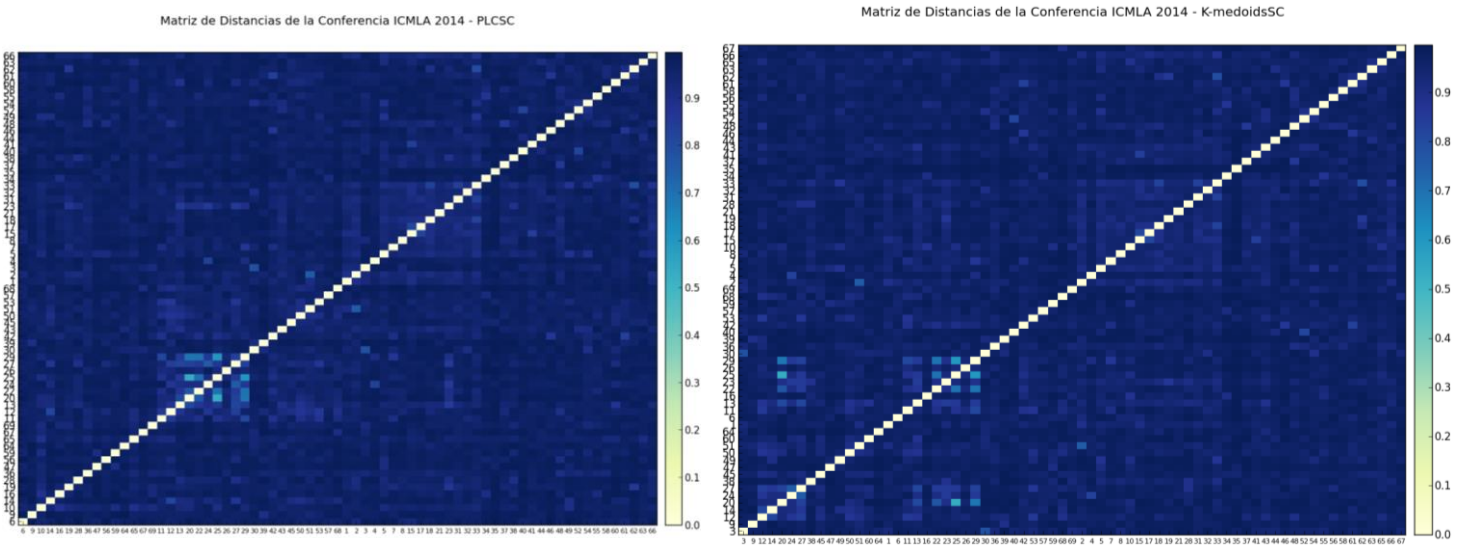

Figura 35(a). *Matriz de Disimilaridades Dataset ICMLA-14, Algoritmo K-MedoidsSC (ES-1), Buckshot*

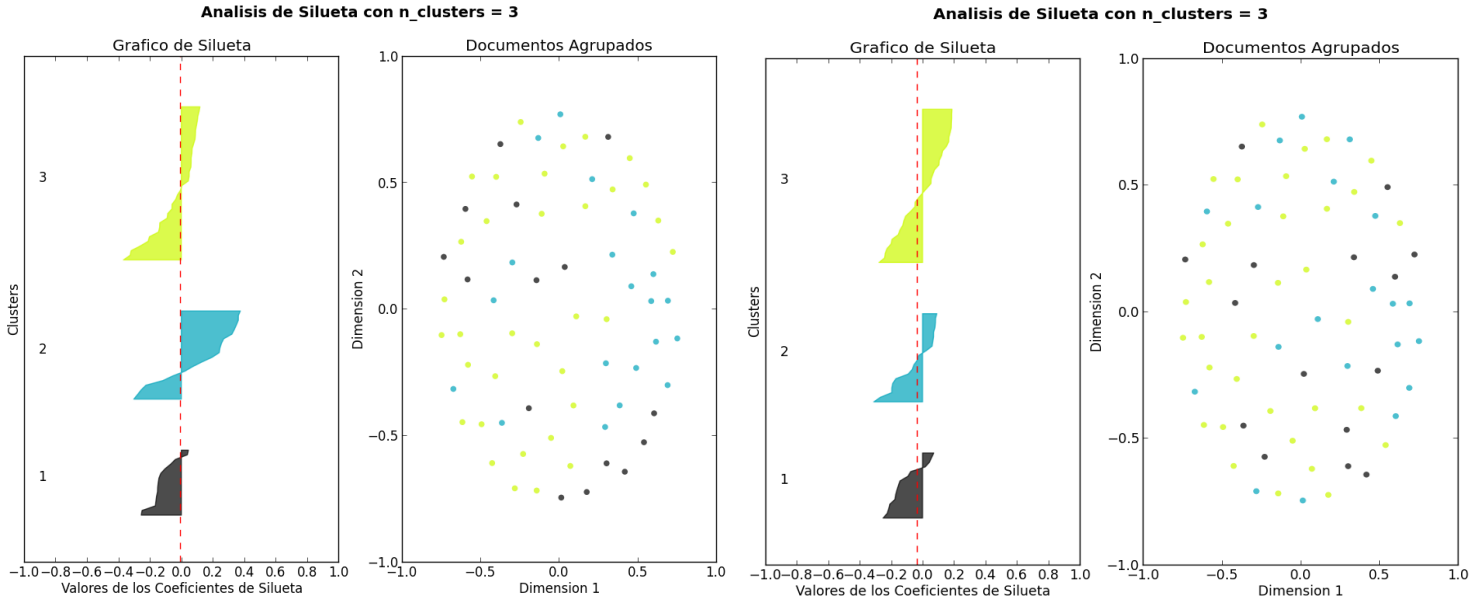

Figura 36(b). *Agrupamiento Dataset ICMLA-14, Algoritmos CSCLP y K-MedoidsSC, k=2*

En la sección 3.6.1.2, se explicó que los valores de silueta oscilaban entre el rango de [-1,1] y que valores cercanos a 1 implicaban grupos con mayor homogeneidad intra-*cluster* y heterogeneidad inter-*cluster*. Los valores resultantes del coeficiente de silueta, tanto para CSCLP como para el K-MedoidsSC, tienen una media de -0.02325 lo que a priori nos indicaría que los grupos han sido creados de manera forzada, pero en comparación con el valor del coeficiente de silueta "real" de -0.04113 en dónde el valor de *k* número de grupos para cada sesión es fijado de acuerdo al criterio de los organizadores de la conferencia, se observa que el valor presenta mejores prestaciones.

Extrapolando todo el procedimiento antes descrito para el *dataset* ICMLA-14, se han ejecutado pruebas similares con los *datasets* AAAI-13 y AAAI-14. El resumen de los principales resultados que se obtuvieron se presenta en la Tabla 24 y la Tabla 25. En el Anexo V y VI se pueden encontrar las pruebas de procedimiento realizadas para las conferencias AAAI-13 y AAAI-14, respectivamente.

Dataset	Algoritmo	E_j	c_j
AAAI-13 **k=3** **Puntos Iniciales Buckshot: [42, 96, 118]**	CSCLP	(44, 52, 53)	(44, 52, 53)
	K-MedoidsSC	(44, 52, 53)	(44, 52, 53)
AAAI-14 **k=11** **Puntos Iniciales Buckshot: [4, 113, 145, 190, 195, 222, 256, 266, 268, 300, 304]**	CSCLP	(10, 11, 18, 19, 21, 25, 30, 42, 45, 57, 120)	(10, 11, 18, 19, 21, 25, 30, 42, 45, 57, 120)
	K-MedoidsSC	(10, 11, 18, 19, 21, 25, 30, 42, 45, 57, 120)	(11, 21, 239, 19, 77, 13, 0, 0, 18, 0, 0)

Tabla 24. *Resultados Tamaños de Clusters, Datasets AAAI-13 y AAAI-14, Algoritmos CSCLP y K-MedoidsSC*

Dataset	Algoritmo	ARI	AMI	NMI	$S(i)$
AAAI-13 k=3	CSCLP	0.0080	0.0071	0.0195	-0.0336
	K-MedoidsSC	0.0126	0.0115	0.0239	-0.0306
AAAI-14 k=11	CSCLP	0.0487	0.1204	0.1782	-0.2341
	K-MedoidsSC	0.0138	0.0312	0.0894	-0.1458

Tabla 25. *Validación del Clustering, Datasets AAAI-13 y AAAI-14, Algoritmos CSCLP y K-MedoidsSC*

7. Conclusiones y Recomendaciones

Este estudio ha presentado dos algoritmos novedosos de aprendizaje semi-supervisado, que permiten restringir los tamaños de los *clusters*. Las pruebas empíricas tanto en *datasets* de *benchmarking*, como en *datasets* documentales han demostrado que los dos algoritmos desarrollados, así como los procedimientos y metodologías planteadas, son opciones viables para resolver problemas de *clustering* con restricciones de tamaño,. Además, se han presentado una serie de gráficas para que los resultados intermedios sean inteligibles y replicables.

El primer algoritmo CSCLP se basa en restricciones del tipo *cannot-link* entre los puntos iniciales, que han de determinarse por algún método o algoritmo externo. La optimización para garantizar las restricciones de tamaño y la cohesión intra-*cluster* y separación inter-*cluster*, se sustenta en un modelo de programación lineal entera binaria PLEB. CSCLP ha demostrado ser robusto ante: variaciones de tamaño del *cluster* y del *dataset*, y también a valores aleatorios o atípicos del conjunto de datos. Aun así, hay que tomar en cuenta que sigue siendo un modelo heurístico.

Las pruebas empíricas ejecutadas sobre los *datasets* han demostrado que el algoritmo K-MedoidsSC, al ser una variante del *K-Means*, es especialmente sensible a valores altos de k y n o cuando se tiene una combinación de k alto y n bajo. Para efectos prácticos, en el caso de estudio concerniente, si el número de k *clusters* deseados es alto (en comparación al tamaño n del *dataset*) ya que el problema así lo requiere, se puede buscar transitoriamente un valor k', tal que $k' < k$, con la finalidad de que el algoritmo converja y encuentre un agrupamiento optimizado. Luego, y en cada una de las particiones resultantes de k', se ejecutaría nuevamente el algoritmo K-MedoidsSC hasta encontrar un número de *clusters* que cumpla que la sumatoria de todos los grupos sea igual al número de *clusters* deseado. Resumiendo lo antes expuesto, se haría un proceso de sub-agrupamiento para que se cumpla con los valores de k estipulados.

Basados en la experiencia empírica, como resultado de realizar pruebas y ensayos sobre diferentes conjuntos de datos, se puede concluir que el algoritmo CSCLP es más estable que el algoritmo K-MedoidsSC y bajo las mismas condiciones de entrada (puntos iniciales y tamaños de los *clusters*) en la mayoría de las pruebas ha alcanzado un mejor rendimiento desde la perspectiva de valoración interna con el coeficiente de silueta. Sin embargo, una de las desventajas de CSCLP en contraposición con el algoritmo K-MedoidsSC es su variabilidad y dependencia de los puntos iniciales, algo que no ocurre con este último algoritmo ya que los puntos semilla son únicamente usados como guía para el agrupamiento y no suponen restricciones del tipo *cannot-link*. Así también podemos llegar a la conjetura de que cuando el *dataset* tiene grupos, notablemente, bien definidos y separados el comportamiento de los dos algoritmos es óptimo.

En la etapa de preparación de los datos, se debe dar un especial énfasis en la limpieza y detección de valores anómalos (*outliers*) ya que el algoritmo particional planteado K-MedoidsSC es sensible a estos objetos, y se puede dar el caso que el algoritmo no encuentre un punto de convergencia. Una forma de resolver esto último es especificando un criterio de parada basado en el número de iteraciones.

El escalamiento multidimensional, que en este artículo ha sido utilizado para representar de manera gráfica a las instancias (documentos), se podría combinar con otras técnicas como el análisis de componentes principales PCA para que los resultados gráficos se ajusten a conjuntos de datos de mayor escala.

Un elemento común y fundamental para los dos algoritmos de *clustering* planteados, es la matriz de distancias. Este es un parámetro de entrada que caracteriza las semejanzas o diferencias entre los diferentes documentos. Si bien se han probado algunos escenarios para la configuración de la matriz de distancias, existen muchas más variaciones que se podrían examinar con la finalidad de refinar la matriz y así obtener un mejor rendimiento. También, se podría pensar en el uso de n-gramas en la bolsa de palabras, usar una métrica de similaridad diferente al coeficiente coseno o realizar pruebas en las que se trabaje con un modelo vectorial íntegro para todo el corpus de los *papers*, incluyendo títulos y palabras clave.

Existen otras metodologías de agrupamiento, como el *spectral clustering,* que también utilizan como parámetro de entrada a la matriz de distancias/disimilaridades. Por lo que, como trabajo futuro se podría pensar en realizar variaciones de los algoritmos CSCLP y K-MedoidsSC con sustento en estas metodologías de *clustering*.

Otra prospectiva para trabajos futuros, se encamina en la escalabilidad de las dos metodologías para grandes bases de datos, ya que los dos algoritmos están inicialmente pensados para trabajar con *datasets* considerados de pequeña escala. Esto a la vez abriría la puerta a la incorporación de nuevas técnicas de agrupamiento de textos como *topic modeling*, que ha demostrado ser una solución viable para conjuntos de datos grandes y homogéneos o modelos estadísticos como el *Latent Dirichlet Allocation* (LDA) que es un modelo generativo que permite que conjuntos de observaciones puedan ser explicados por grupos no observados y que explican por qué algunas partes de los datos son similares.

8. Referencias

[1] JAIN, Anil K.; MURTY, M. Narasimha; FLYNN, Patrick J. Data clustering: a review. ACM computing surveys (CSUR), vol. 31, no 3, p. 264-323, 1999. http://dl.acm.org/citation.cfm?id=331504

[2] CHEN, Ming-Syan; HAN, Jiawei; YU, Philip S. Data mining: an overview from a database perspective. Knowledge and data Engineering, IEEE Transactions, vol. 8, no 6, p. 866-883, 1996. http://www.nyu.edu/classes/jcf/g22.3033-002/handouts/chen96data.pdf

[3] ZHU, Shunzhi; WANG, Dingding; LI, Tao. Data clustering with size constraints. Knowledge-Based Systems, vol. 23, no 8, p. 883-889, 2010. http://www.sciencedirect.com/science/article/pii/S095070511000095X

[4] DALTON, Lori; BALLARIN, Virginia; BRUN, Marcel. Clustering algorithms: on learning, validation, performance, and applications to genomics. Current Genomics, vol. 10, no 6, p. 430-445, 2009. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC2766793/

[5] ROUSSEEUW, Peter J. Silhouettes: a graphical aid to the interpretation and validation of cluster analysis. Journal of computational and applied mathematics, vol. 20, p. 53-65, 1987. http://www.sciencedirect.com/science/article/pii/0377042787901257

[6] DAVIDSON, Ian; RAVI, S. S. Clustering with Constraints: Feasibility Issues and the k-Means Algorithm. SDM, p. 201-211, 2005 http://www.cs.albany.edu/~davidson/Publications/SDM2005Final.pdf

[7] MARTÍNEZ, Fernando. Árboles de estimación estocástica de probabilidades: Newton Trees. Universidad Politécnica de Valencia – Máster en Ingeniería del Software, Métodos Formales y Sistemas de Información, 2010. https://riunet.upv.es/bitstream/handle/10251/14537/Tesis_fmartinez.pdf?sequence=1&isAllowed=y

[8] VENKATESAN, Rajkumar. Cluster analysis for segmentation. Universidad de Virginia – Darden Business Publishing, 2007. http://faculty.darden.virginia.edu/GBUS8630/doc/M-0748.pdf

[9] HARTLEY, James. New ways of making academic articles easier to read. International Journal of Clinical and Health Psychology, vol. 12, no 1, p. 143-160, 2012. http://www.aepc.es/ijchp/articulos_pdf/ijchp-405.pdf

[10] KOKASH, Natallia. An introduction to heuristic algorithms. University of Trento –Department of Informatics and Telecommunications, p. 1-8, 2005. http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.105.8050&rep=rep1&type=pdf

[11] GANGANATH, Nuwan; CHENG, Chi-Tsun; TSE, Chi K. Data clustering with cluster size constraints using a modified k-means algorithm. Cyber-Enabled Distributed Computing and Knowledge Discovery (CyberC), International Conference IEEE, p. 158-161, 2014. http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6984299&tag=1

[12] REBOLLO-MONEDERO, David; SOLÉ, Marc; NIN, Jordi; FORNÉ, Jordi. A modification of the k-means method for quasi-unsupervised learning. Knowledge-Based Systems, vol. 37, p. 176-185, 2013. http://www.sciencedirect.com/science/article/pii/S0950705112002122

[13] MATTAR, Salim; GONZÁLEZ, Marco. The keys of the key words in scientific articles. Revista MVZ Córdoba, vol. 17, no 2, p. 2955-2956, 2011. http://revistas.unicordoba.edu.co/revistamvz/mvz-172/V17N2A1.pdf

[14] WAGSTAFF, Kiri; CARDIE, Claire; ROGERS, Seth; SCHRÖDL, Stefan. Constrained k-means clustering with background knowledge. Proceedings of

the Eighteenth International Conference on Machine Learning ICML, vol. 1, p. 577-584, 2001.
http://nichol.as/papers/Wagstaff/Constrained%20k-means%20*clustering*%20with%20background.pdf

[15] WAGSTAFF, Kiri; CARDIE, Claire. Clustering with instance-level constraints. Proceedings of the Seventeenth International Conference on Machine Learning, p. 1103-1110, 2000.
http://www.litech.org/~wkiri/Papers/wagstaff-constraints-00.pdf

[16] ARAUJO, Rodrigo. A semi-Supervised approach for kernel-based temporal clustering. University of Waterloo – Tesis Doctoral, 2015
https://uwspace.uwaterloo.ca/bitstream/handle/10012/9265/Araujo_Rodrigo.pdf?sequence=1&isAllowed=y

[17] ZHANG, Shaohong; WONG, Hau-San; XIE, Dongqing. Semi-supervised clustering with pairwise and size constraints. Neural Networks (IJCNN), International Joint Conference IEEE, p. 2450-2457, 2014.
http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6889553

[18] LIN, Jianyi; BERTONI, Alberto; NALDI, Giovanni. Exact algorithms for size constrained clustering. Universita degli Studi di Milano – Tesis Doctoral. Librería Ledi Srl, 2013.
https://air.unimi.it/retrieve/handle/2434/172513/173785/phd_unimi_R07628.pdf

[19] GILPIN, Sean; DAVIDSON, Ian. A flexible ILP formulation for hierarchical clustering. Artificial Intelligence, 2015.
http://www.sciencedirect.com/science/article/pii/S0004370215000831

[20] DAVIDSON, Ian; RAVI, S. Using instance-level constraints in agglomerative hierarchical clustering: theoretical and empirical results. Data mining and knowledge discovery, vol. 18, no 2, p. 257-282, 2009.
http://link.springer.com/article/10.1007/s10618-008-0103-4

[21] BILENKO, Mikhail; BASU, Sugato; MOONEY, Raymond J. Integrating constraints and metric learning in semi-supervised clustering. Proceedings of the twenty-first international conference on Machine learning. ACM, p. 11, 2004.
http://research.microsoft.com/pubs/63997/04-semi-icml.pdf

[22] HÖPPNER, Frank; KLAWONN, Frank. Clustering with size constraints. Computational Intelligence Paradigms. Springer Berlin Heidelberg, p. 167-180, 2008.
https://public.fh-wolfenbuettel.de/~klawonn/Papers/hoeppnerklawonn08.pdf

[23] HUBERT, Lawrence; ARABIE, Phipps. Comparing partitions. Journal of classification, vol. 2, no 1, p. 193-218, 1985.
http://link.springer.com/article/10.1007/BF01908075

[24] POLLO CATTANEO, María Florencia, et al. Metodología para especificación de requisitos en proyectos de explotación de información. XI Workshop de Investigadores en Ciencias de la Computación, 2009.
http://sedici.unlp.edu.ar/bitstream/handle/10915/19797/Documento_completo.pdf?sequence=1

[25] BECKER, Jörg; KUROPKA, Dominik. Topic-based vector space model. Proceedings of the 6th international conference on business information systems, p. 7-12, 2003
http://udoo.uni-muenster.de/downloads/publications/1433.pdf

[26] GONÇALVES, Alexandre; ZHU, Jianhan; SONG, Dawei; UREN, Victoria; PACHECO, Roberto. LRD: Latent relation discovery for vector space expansion and information retrieval. Advances in Web-Age Information Management. Springer Berlin Heidelberg, p. 122-133, 2006.
http://link.springer.com/chapter/10.1007%2F11775300_11

[27] RAO, Singiresu S.;Engineering optimization: theory and practice. John Wiley & Sons, ISBN: 9780470183526, cuarta edición, 2009. http://eu.wiley.com/WileyCDA/WileyTitle/productCd-0470183527.html
[28] DESALE, Sachin; RASOOl, Akhtar; ANDHALE, Sushil; RANE, Priti. Heuristic and meta-heuristic algorithms and their relevance to the real world: a survey. International Journal of Computer Engineering in Research Trends, ISSN: 2349-7084, vol. 2, no 5, p. 296-304, 2015. http://www.ijcert.org/V2I55.pdf
[29] MARCHENA WILLIAMS, Ornelas Carlos; GONZÁLEZ-LONGATT, Francisco. Optimización y la programación lineal: una introducción. Fuentes Alternas de Energía y Generación Distribuida, 2007. http://sauce.pntic.mec.es/~jpeo0002/Archivos/PDF/T08.pdf
[30] International Conference on Machine Learning and Applications, 2014. Fecha de consulta: 2015/12/10. http://www.icmla-conference.org/icmla14/
[31] RICHARDSON, Leonard. Beautiful Soup Documentation 4.0.0, 2015. https://media.readthedocs.org/pdf/beautifulsoup-korean/latest/beautifulsoup-korean.pdf
[32] BRODLEY, Carla. UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2014. Fecha de consulta: 2015/12/10. http://archive.ics.uci.edu/ml/datasets/AAAI+2013+Accepted+Papers
[33] BRODLEY, Carla. UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2014. Fecha de consulta: 2015/12/10. http://archive.ics.uci.edu/ml/datasets/AAAI+2014+Accepted+Papers
[34] Conference on Artificial Intelligence of Association for the Advancement of Artificial Intelligence, 2013. Fecha de consulta: 2015/12/10. http://www.aaai.org/Conferences/AAAI/aaai13.php
[35] Conference on Artificial Intelligence of Association for the Advancement of Artificial Intelligence, 2014. Fecha de consulta: 2015/12/10. http://www.aaai.org/Conferences/AAAI/aaai14.php
[36] PEDREGOSA, Fabian, et al. Scikit-Learn: machine learning in python. The Journal of Machine Learning Research, vol. 12, p. 2825-2830, 2001. http://www.jmlr.org/papers/volume12/pedregosa11a/pedregosa11a.pdf
[37] JONES, Eric; OLIPHANT, Travis; PETERSON, Pearu. SciPy: open source scientific tools for python, vol. 73, p. 86, 2015. http://www.scipy.org
[38] FISHER, Ronald; UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2014. Fecha de consulta: 2015/12/10. https://archive.ics.uci.edu/ml/datasets/Iris
[39] FORINA, M. et al ; UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2014. Fecha de consulta: 2015/12/10. https://archive.ics.uci.edu/ml/datasets/Wine
[40] CHARYTANOWICZ, Małgorzata; NIEWCZAS, Jerzy; KOWALSKI, Piotr A.; KULCZYCKI, Piotr; ŁUKASIK, Szymon; ZAK, Sławomir. UCI Machine Learning Repository, University of California, School of Information and Computer Science, Irvine, CA, 2014. Fecha de consulta: 2015/12/10. https://archive.ics.uci.edu/ml/datasets/seeds
[41] ROMANO, Simone; BAILEY, James; XUAN VINH, Nguyen; VERSPOOR, Karin. Standardized mutual information for clustering comparisons: one step further in adjustment for chance. Proceedings of the 31st International Conference on Machine Learning ICML-14, pp. 1143-1151, 2014 http://jmlr.org/proceedings/papers/v32/romano14.pdf

[42] WU, Junjie; XIONG, Hui; CHEN, Jian. Adapting the right measures for k-means clustering. Proceedings of the 15th ACM SIGKDD international conference on Knowledge Discovery and Data Mining. ACM, pp. 877-886, 2009
http://dl.acm.org/citation.cfm?id=1557115

[43] YAN, Mingjin. Methods of determining the number of clusters in a data set and a new clustering criterion. Virginia Polytechnic Institute and State University – Tesis Doctoral, 2005.
https://theses.lib.vt.edu/theses/available/etd-12062005-153906/unrestricted/Proposal-Face.pdf

[44] JHA, Monica. Document clustering using k-medoids. International Journal on Advanced Computer Theory and Engineering (IJACTE). ISSN: 2319-2526, vol. 4, no 1, 2015.
https://arxiv.org/ftp/arxiv/papers/1504/1504.01183.pdf

[45] GORDON, A. D. Classification. Chapman & Hall/CRC, London, ISBN: 9781584880134, 1999.
https://www.crcpress.com/Classification-2nd-Edition/Gordon/p/book/9781584880134

[46] HU, Guobiao; ZHOU, Shuigeng; GUAN, Jihong; HU, Xiaohua. Towards effective document clustering: a constrained k-means based approach. Information Processing & Management, vol. 44, no 4, p. 1397-1409, 2008.
http://read.pudn.com/downloads143/doc/622615/semi01/Towards%20effective%20document%20clusterin%20A%20constrained%20K-means%20based%20approach.pdf

[47] CUTTING, Douglass R.; KARGER, David R.; PEDERSEN, Jan O.; TUKEY, John W. Scatter/gather: a cluster-based approach to browsing large document collections. Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, p. 318-329, 1992
http://dl.acm.org/citation.cfm?id=133214

[48] PANTEL, Patrick; LIN, Dekang. Document clustering with committees. Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, p. 199-206, 2002.
http://dl.acm.org/citation.cfm?id=564412

[49] FAYYAD, Usama M.; REINA, Cory; BRADLEY, Paul S. Initialization of iterative refinement clustering algorithms. Proceedings of ACMSIGKDD, p. 194-198, 1998.
https://www.aaai.org/Papers/KDD/1998/KDD98-032.pdf

[50] CHAVENT, Marie. A monothetic clustering method. Pattern Recognition Letters, vol. 19, no 11, p. 989-996, 1998.
http://www.sciencedirect.com/science/article/pii/S0167865598000877

[51] BIRD, Steven; LOPER, Edward. NLTK: the natural language toolkit. Proceedings of the COLING/ACL on Interactive presentation sessions. Association for Computational Linguistics, p. 69-72, 2006.
http://dl.acm.org/citation.cfm?id=1225421

[52] GUERRERO CASAS, Flor María; RAMÍREZ HURTADO, José Manuel; El Análisis de Escalamiento Multidimensional: una alternativa y un complemento a otras técnicas multivariantes. La sociología en sus escenarios, no 25, 2012.
http://147.156.1.4/asepuma/X/K11C.pdf

[53] CUADRAS, Carles M. Distancias Estadísticas. Revista de Estadística Española, vol.30, no 119, p. 295-378, 1989.
http://goo.gl/LUrTxI

[54] CUADRAS, Carles M. Nuevos Métodos de Análisis Multivariante. CMC Editions, 2014.

http://www.ub.edu/stat/personal/cuadras/metodos.pdf
[55] MANNING, Christopher D.; RAGHAVAN, Prabhakar; SCHÜTZE, Hinrich; An introduction to information retrieval. Cambridge University Press, ISBN: 97805218, vol. 1, no 1, 2008.
http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf
[56] CARRILLO RUIZ, Maya; LÓPEZ LÓPEZ, Aurelio; Una representación vectorial para contenido de textos en tratamiento de información. Coordinación de Ciencias Computacionales INAOE, 2008.
http://ccc.inaoep.mx/Reportes/CCC-08-004.pdf
[57] MORAN, Kelly H.; WALLACE, Byron C.; BRODLEY, Carla E.; Discovering better AAAI keywords via clustering with community-sourced constraints. Twenty-Eighth AAAI Conference on Artificial Intelligence, 2014.
http://www.aaai.org/ocs/index.php/AAAI/AAAI14/paper/view/8536/8567
[58] JOSHI, Aravind K., et al.; Natural Language Processing; Science, vol. 253, no 5025, p. 1242-1249, 1991.
http://search.proquest.com/openview/306a821ef3e8e15770fc115f0db69e64/1.pdf?pq-origsite=gscholar&cbl=1256
[59] WALLACH, Hanna M.; Topic modeling: beyond bag-of-words. Proceedings of the 23rd international conference on Machine learning. ACM, p. 977-984, 2006.
http://dl.acm.org/citation.cfm?id=1143967
[60] VIDAL, Erica. Algoritmo divisivo de *clustering* con determinación automática de componentes. Universidad Nacional del Rosario – Facultad de Ciencias Exactas, Ingeniería y Agrimensura, 2014.
http://www.fceia.unr.edu.ar/lcc/t523/uploads/63.pdf#cite.jain2
[61] HALKIDI, Maria; BATISTAKIS, Yannis; VAZIRGIANNIS, Michalis. On clustering validation techniques. Journal of Intelligent Information Systems, vol. 17, no 2, p. 107-145, 2001.
http://link.springer.com/article/10.1023/A:1012801612483
[62] KAUFMAN, Leonard; ROUSSEEUW, Peter J. Clustering by means of medoids. Statistical Data Analysis Based on the L1–Norm and Related Methods, edited by Y. Dodge, North-Holland, p. 405–416, 1987.
https://wis.kuleuven.be/stat/robust/papers/publications-1987/kaufmanrousseeuw-*clustering*bymedoids-l1norm-1987.pdf
[63] BHAT, Aruna. K-Mediods clustering using partitioning around medoids for performing face recognition. Int. J. Soft Comp. Mat. Cont, vol. 3, no 3, p. 1-12, 2014.
http://wireilla.com/ns/maths/Papers/3314ijscmc01.pdf
[64] FAYYAD, Usama; PIATETSKY-SHAPIRO, Gregory; SMYTH, Padhraic. From data mining to knowledge discovery in databases. AI magazine, vol. 17, no 3, p. 37, 1996.
https://www.aaai.org/ojs/index.php/aimagazine/article/viewFile/1230/1131
[65] GROSSI, Valerio; MONREALE, Anna; NANNI, Mirco; PEDRESCHI, Dino; TURINI, Franco. Clustering formulation using constraint optimization. Software Engineering and Formal Methods. Springer Berlin Heidelberg, p. 93-107, 2015
http://www.di.unipi.it/mokmasd/symposium-2015/preproceedings/MoKMaSD_2015_GrossiEtAl_Clustering_formulation.pdf
[66] MITCHELL, John E. Branch-and-cut algorithms for combinatorial optimization problems. Handbook of applied optimization, p. 65-77, 2002.
http://homepages.rpi.edu/~mitchj/papers/bc_hao.pdf
[67] CASTILLO, Enrique, et al. Formulación y Resolución de Modelos de Programación Matemática en Ingeniería y Ciencia, 2002.
http://eco.mdp.edu.ar/cendocu/repositorio/00216.pdf

[68] GONZALEZ, Teofilo F. Clustering to minimize the maximum intercluster distance. Theoretical Computer Science, vol. 38, p. 293-306, 1985. http://www.sciencedirect.com/science/article/pii/0304397585902245

[69] WILLETT, Peter. Recent trends in hierarchic document clustering: a critical review. Information Processing & Management, vol. 24, no 5, p. 577-597, 1988. http://www.sciencedirect.com/science/article/pii/0306457388900271

[70] JI, Xiang; XU, Wei. Document clustering with prior knowledge. Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval. ACM, p. 405-412, 2006. http://dl.acm.org/citation.cfm?id=1148241

[71] STEINBACH, Michael; KARYPIS, George; KUMAR, Vipin. A comparison of document clustering techniques. KDD workshop on text mining, vol. 400, no. 1, p. 525-526, 2000. http://www.cs.cmu.edu/~dunja/KDDpapers/Steinbach_IR.pdf

[72] SEGARAN, Toby. Programming collective intelligence: building smart web 2.0 applications. O'Reilly Media, Inc., ISBN: 9780596529321, 2007. http://shop.oreilly.com/product/9780596529321.do

9. Anexos

Anexo I

Stopwords
'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now'

Tabla 26. *Conjunto de Stopwords*

Anexo II

ID Sesión	Nombre de la Sesión	Número de Papers	ID Papers en el Dataset
1	Ensemble Methods	5	[1, 2, 3, 4, 5]
2	Applications in security	5	[6, 7, 8, 9, 10]
3	Feature Extraction and Selection	5	[11, 12, 13, 14, 15]
4	Machine Learning I	4	[16, 17, 18, 19]
5	Neural Networks I	5	[20, 21, 22, 23, 24]
6	Neural Networks II	5	[25, 26, 27, 28, 29]
7	Semi-Supervised Learning	5	[30, 31, 32, 33, 34]
8	Medicine and Bioinformatics	5	[35, 36, 37, 38, 39]
9	Real-time Systems and Industry	5	[40, 41, 42, 43, 44]
10	Information Retrieval I	5	[45, 46, 47, 48, 49]
11	Information Retrieval II	5	[50, 51, 52, 53, 54]
12	Science and Industry	5	[55, 56, 57, 58, 59]
13	Machine Learning II	5	[60, 61, 62, 63, 64]
14	Medicine, Science and Music	5	[65, 66, 67, 68, 69]
Día 1 – Sesiones: 1, 3, 10, 11			
Día 2 – Sesiones: 5, 6, 8			
Día 3 – Sesiones: 2, 4, 7, 9, 12, 13, 14			

Tabla 27. *Estructura de las Sesiones de la Conferencia ICMLA - 2014*

Anexo III

ID Sesión	Nombre de la Sesión	Número de Papers	ID Papers en el Dataset
1	Situated Interaction	1	[62]
2	Matrices	3	[17, 45, 94]
3	Vision	3	[42, 73, 115]
4	Bayesian Inference	3	[24, 91, 119]
5	Game Playing / Plan Reuse	3	[63, 99, 106]
6	Game Theory	4	[3, 40, 87, 96]
7	Security and Network Games	4	[21, 122, 137, 141]
8	Nearest Neighbors and Hierarchical Clustering	4	[10, 22, 102, 111]
9	Clustering	4	[23, 41, 46, 52]
10	Regression	4	[37, 103, 105, 144]
11	Markets and Preferences	4	[9, 29, 88, 126]
12	Bayesian Inference and Causality	4	[16, 27, 36, 86]
13	Reinforcement Learning	4	[2, 28, 107, 118]
14	Recognition and Detection	4	[6, 12, 34, 57]
15	Options/Pricing	4	[1, 5, 19, 146]
16	Search	4	[4, 31, 44, 101]
17	Logic and Answer Set Programming	4	[49, 75, 117, 132]
18	Sentiment and Recommendation	4	[65, 116, 145, 147]
19	Language Processing	4	[79, 131, 142, 148]
20	AI in Science	4	[47, 84, 98, 100]
21	Teamwork	4	[51, 53, 97, 127]
22	Bribery / Voting	4	[18, 32, 56, 136]
23	Planning	4	[72, 83, 123, 130]
24	Situation Calculus and STRIPS	4	[8, 35, 39, 60]
25	Constraints	4	[7, 48, 67, 89]
26	Tensors and Manifolds	4	[43, 55, 64, 110]
27	Equilibrium and Negotiation	4	[25, 30, 68, 124]
28	Planning Under Uncertainty	4	[59, 121, 134, 140]
29	Logic and Knowledge Representation	4	[78, 82, 93, 113]
30	Kernels and Density Estimation	4	[13, 70, 77, 108]
31	Privacy and Social Media	4	[11, 38, 61, 71]
32	Mechanism Design and Aggregation	4	[20, 50, 69, 80]
33	Multi-* Machine Learning	4	[66, 81, 90, 104]
34	Temporal Reasoning	4	[54, 74, 109, 125]
35	NLP Generation and Translation	4	[58, 76, 120, 149]
36	Satisfiability	4	[14, 114, 128, 135]
37	Task Learning	4	[33, 129, 133, 143]
38	Classification	4	[26, 92, 95, 112]
39	Sample Complexity / Anomaly Detection	4	[15, 85, 138, 139]
Día 1 – Sesiones: 1, 2, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18			
Día 2 – Sesiones: 8, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30			
Día 3 – Sesiones: 3, 4, 5, 6, 7, 31, 32, 33, 34, 35, 36, 37, 38, 39			

Tabla 28. *Estructura de las Sesiones de la Conferencia AAAI - 2013*

Anexo IV

ID Sesión	Nombre de la Sesión	Número de Papers	ID Papers en el Dataset
1	Human-Computation and Crowd Sourcing	2	[175, 221]
2	NLP and Knowledge Representation	3	[163, 210, 297]
3	Game Playing and Interactive Entertainment	4	[100, 118, 321, 326]
4	Cognitive Modeling	4	[21, 229, 241, 362]
5	Cognitive Systems	7	[120, 123, 141, 217, 227, 384, 385]
6	Humans and AI	7	[49, 94, 121, 160, 211, 246, 287]
7	NLP and Text Mining	7	[4, 82, 117, 187, 204, 242, 397]
8	Heuristic Search and Optimization	9	[96, 147, 231, 236, 284, 293, 311, 388, 390]
9	Robotics	10	[26, 59, 88, 128, 143, 153, 207, 215, 224, 392]
10	Applications	11	[30, 53, 69, 75, 89, 114, 235, 240, 247, 361, 374]
11	NLP and Machine Learning	11	[54, 140, 159, 195, 203, 214, 281, 373, 380, 386, 393]
12	Computational Sustainability and AI	14	[18, 27, 64, 76, 81, 92, 146, 166, 188, 198, 216, 228, 239, 368]
13	Reasoning under Uncertainty	16	[12, 46, 55, 105, 149, 180, 196, 213, 226, 301, 324, 327, 342, 346, 363, 381]
14	Multiagent Systems	18	[17, 23, 31, 61, 98, 111, 115, 136, 154, 170, 179, 182, 219, 237, 296, 320, 331, 371]
15	Vision	19	[6, 20, 44, 79, 84, 97, 126, 138, 152, 184, 208, 209, 254, 258, 264, 350, 351, 370, 382]
16	Search and Constraint Satisfaction	21	[11, 28, 72, 106, 112, 165, 178, 186, 193, 201, 206, 251, 278, 285, 292, 315, 334, 335, 339, 372, 387]
17	Planning and Scheduling	25	[15, 32, 50, 93, 108, 116, 124, 139, 156, 158, 169, 173, 197, 202, 238, 248, 257, 267, 282, 309, 313, 316, 347, 358, 378]
18	Knowledge Representation and Reasoning	26	[22, 36, 57, 67, 70, 77, 101, 110, 130, 132, 171, 174, 181, 185, 280, 288, 295, 322, 325, 341, 349, 353, 355, 357, 375, 396]
19	Machine Learning Applications	32	[5, 7, 16, 85, 90, 95, 99, 103, 113, 137, 177, 183, 222, 223, 232, 252, 260, 261, 272, 279, 299, 300, 302, 307, 310, 323, 336, 340, 354, 364, 376, 379]
20	AI and the Web	34	[2, 9, 14, 19, 29, 39, 42, 51, 52, 56, 63, 65, 83, 86, 102, 104, 157, 168, 205, 220, 245, 256, 263, 269, 270, 277, 290, 291, 298, 312, 369, 377, 394, 395]

21	Game Theory and Economic Paradigms	41	[3, 35, 38, 40, 45, 58, 66, 68, 71, 78, 80, 125, 135, 150, 162, 167, 172, 189, 190, 191, 192, 225, 234, 253, 262, 266, 271, 273, 304, 305, 308, 314, 318, 337, 338, 343, 344, 348, 360, 367, 391]
22	Novel Machine Learning Algorithms	77	[1, 8, 10, 13, 24, 25, 33, 34, 37, 41, 43, 47, 48, 60, 62, 73, 74, 87, 91, 107, 109, 119, 122, 127, 129, 131, 133, 134, 142, 144, 145, 148, 151, 155, 161, 164, 176, 194, 199, 200, 212, 218, 230, 233, 243, 244, 249, 250, 255, 259, 265, 268, 274, 275, 276, 283, 286, 289, 294, 303, 306, 317, 319, 328, 329, 330, 332, 333, 345, 352, 356, 359, 365, 366, 383, 389, 398]
Subsesión 1 – Sesiones: 1, 6, 12, 20			
Subsesión 2 – Sesiones: 2, 7, 11			
Subsesión 3 – Sesiones: 3, 21			
Subsesión 4 – Sesiones: 4, 5			
Subsesión 5 – Sesiones: 8, 16			
Subsesión 6 – Sesión: 9			
Subsesión 7 – Sesiones: 10, 19, 22			
Subsesión 8 – Sesiones: 13, 18			
Subsesión 9 – Sesiones: 14			
Subsesión 10 – Sesión: 15			
Subsesión 11 – Sesión: 17			

Tabla 29. *Estructura de las Sesiones de la Conferencia AAAI – 2014*

Anexo V

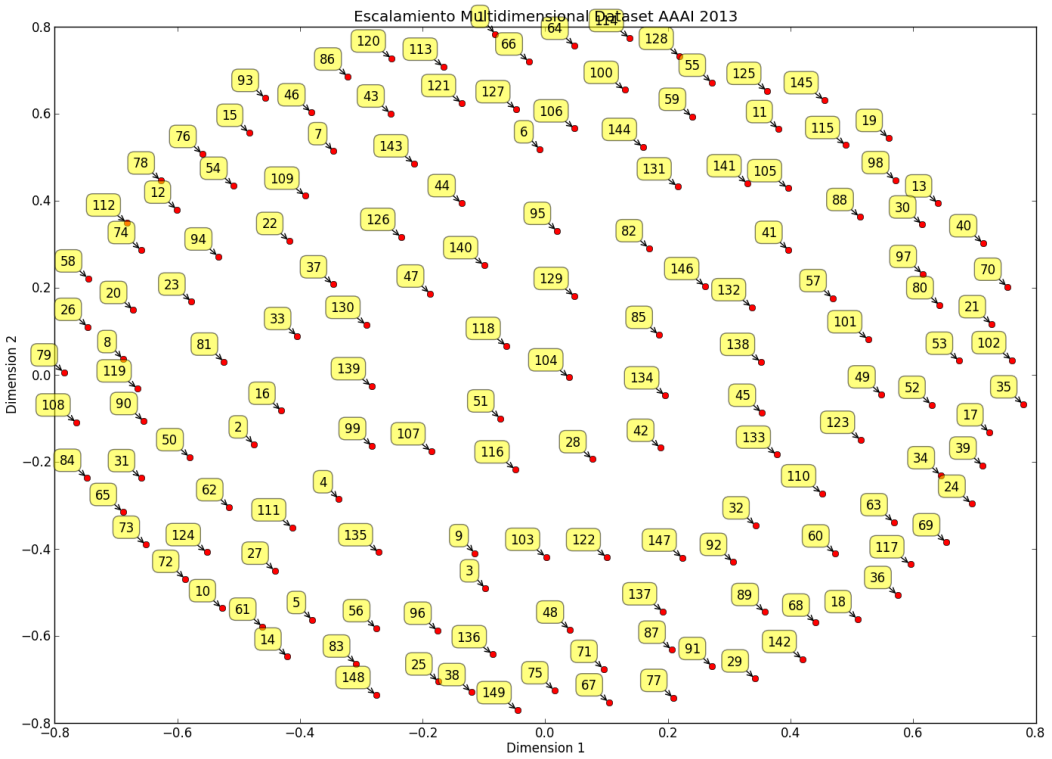

Figura 37. *Escalamiento Multidimensional de los Documentos de la Conferencia AAAI-13 (ES-1)*

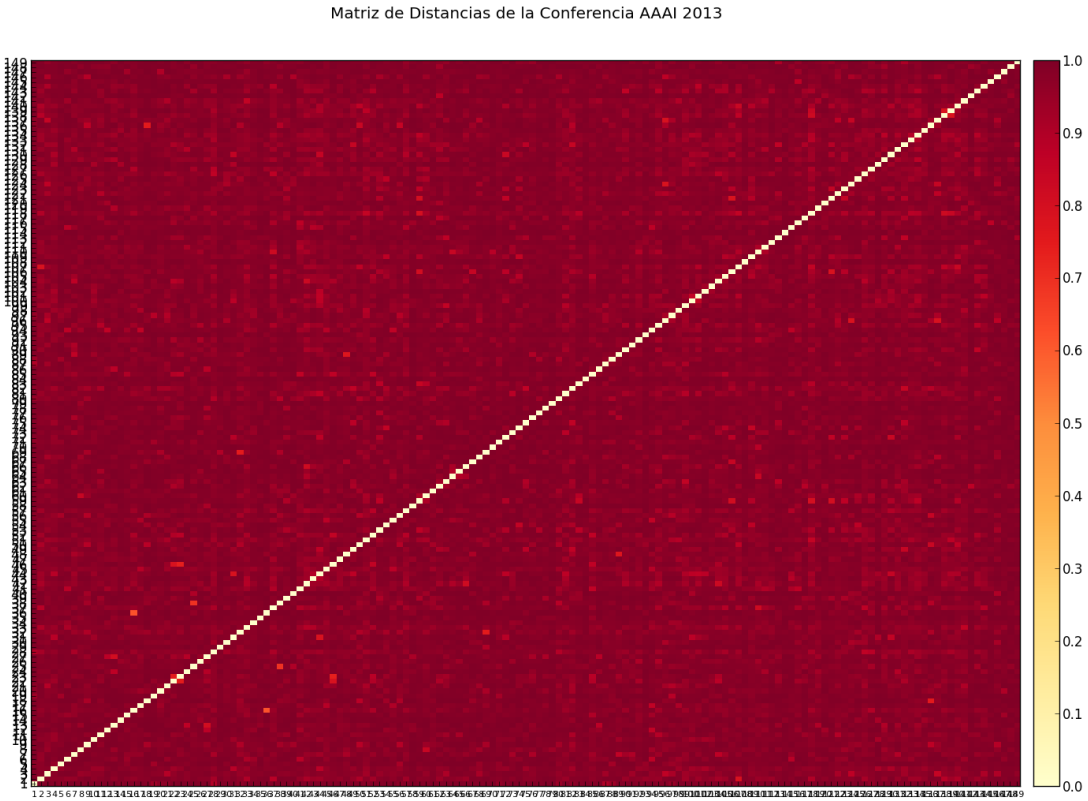

Figura 38. *Matriz de Disimilaridades, Dataset AAAI-13 (ES-1)*

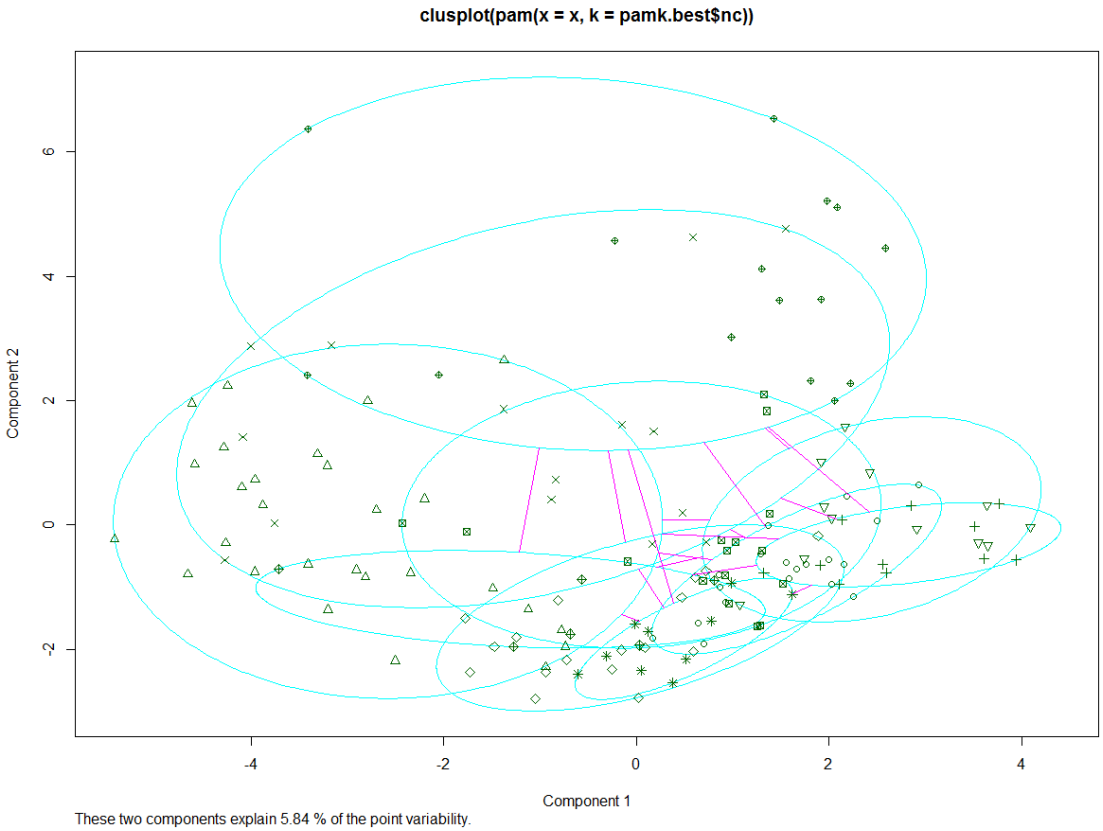

Figura 39. *Selección Número de Clusters, Dataset AAAI-13, k=10*

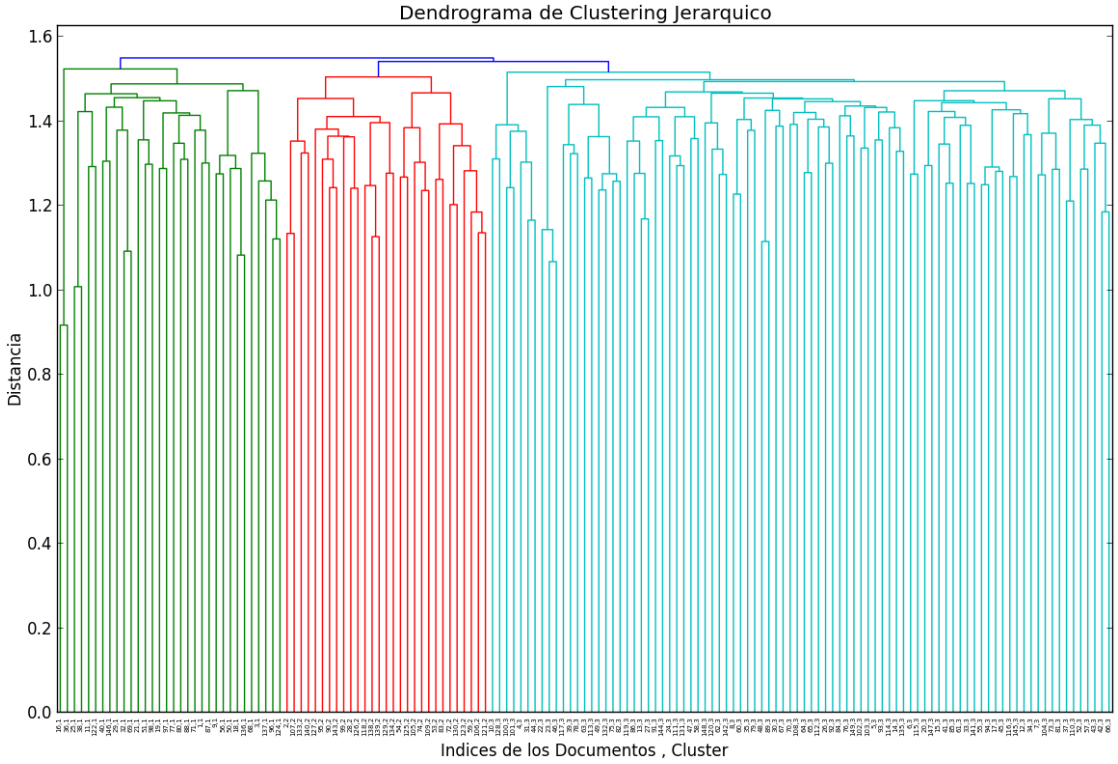

Figura 40. *Dendograma - Farthest Point Algorithm, Dataset AAAI-13 (ES-1) (3 grupos)*

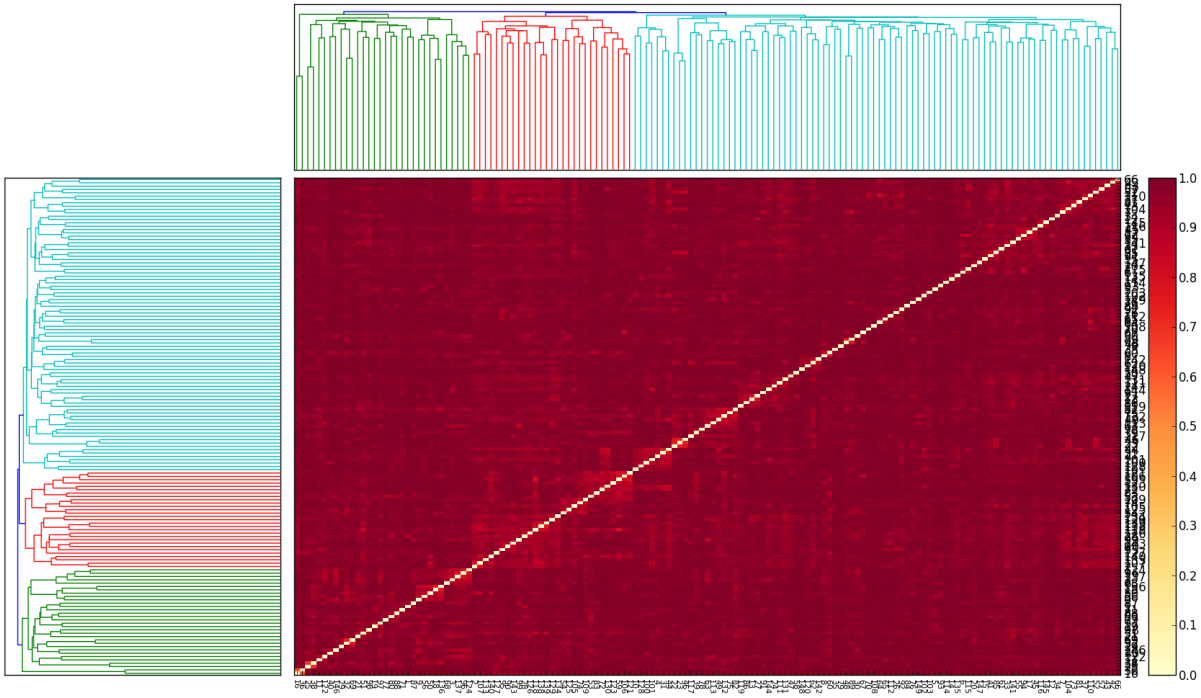

Figura 41. *Matriz Disimilaridades y Dendograma AHC-FPA, Dataset AAAI-13 (ES-1) (3 grupos)*

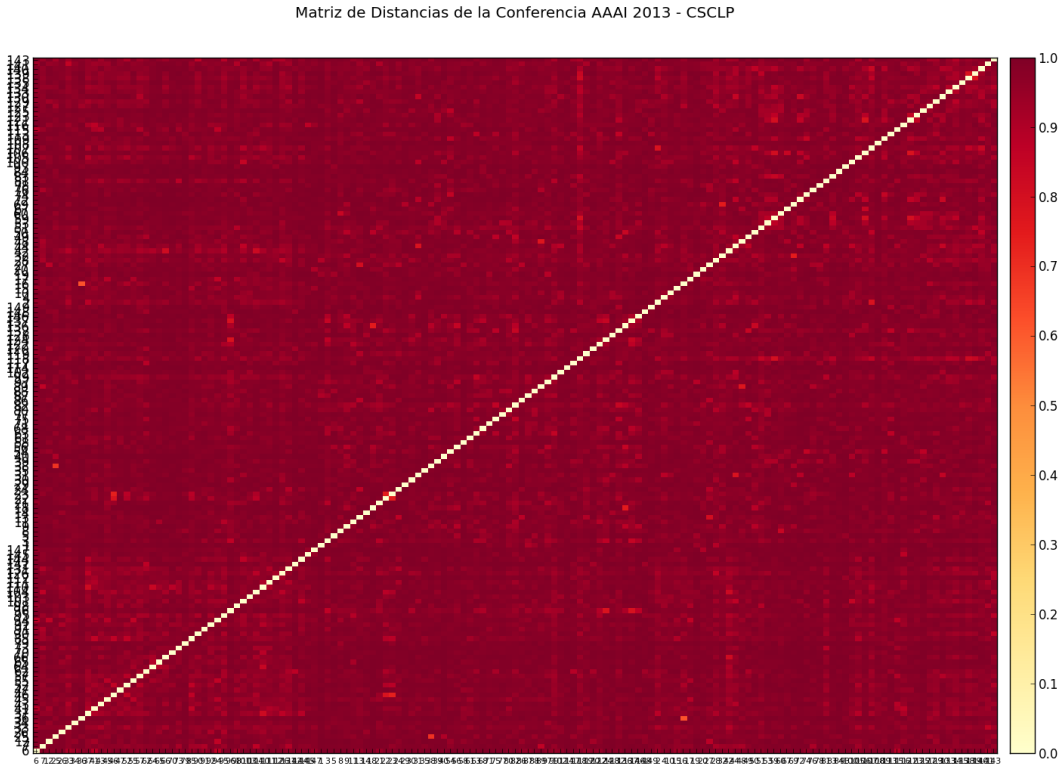

Figura 42. *Matriz de Disimilaridades Dataset AAAI-13, Algoritmo CSCLP (ES-1), Buckshot*

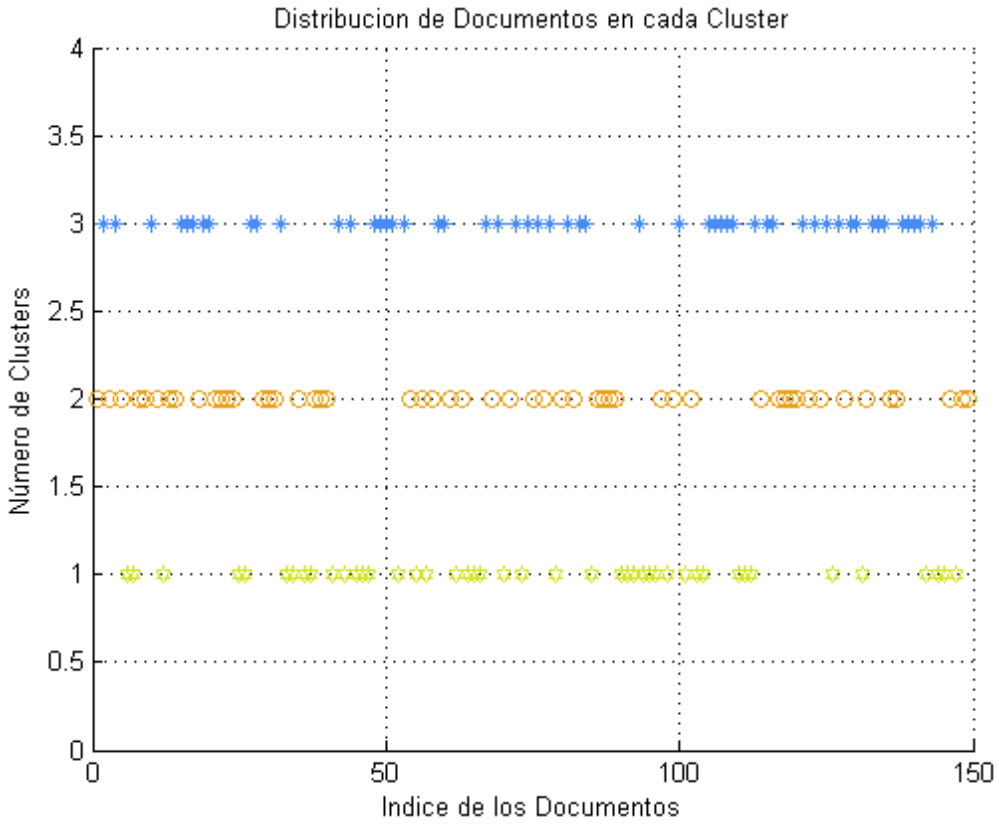

Figura 43. *Distribución de Documentos, Dataset AAAI-13, Algoritmo CSCLP (ES-1), Buckshot*

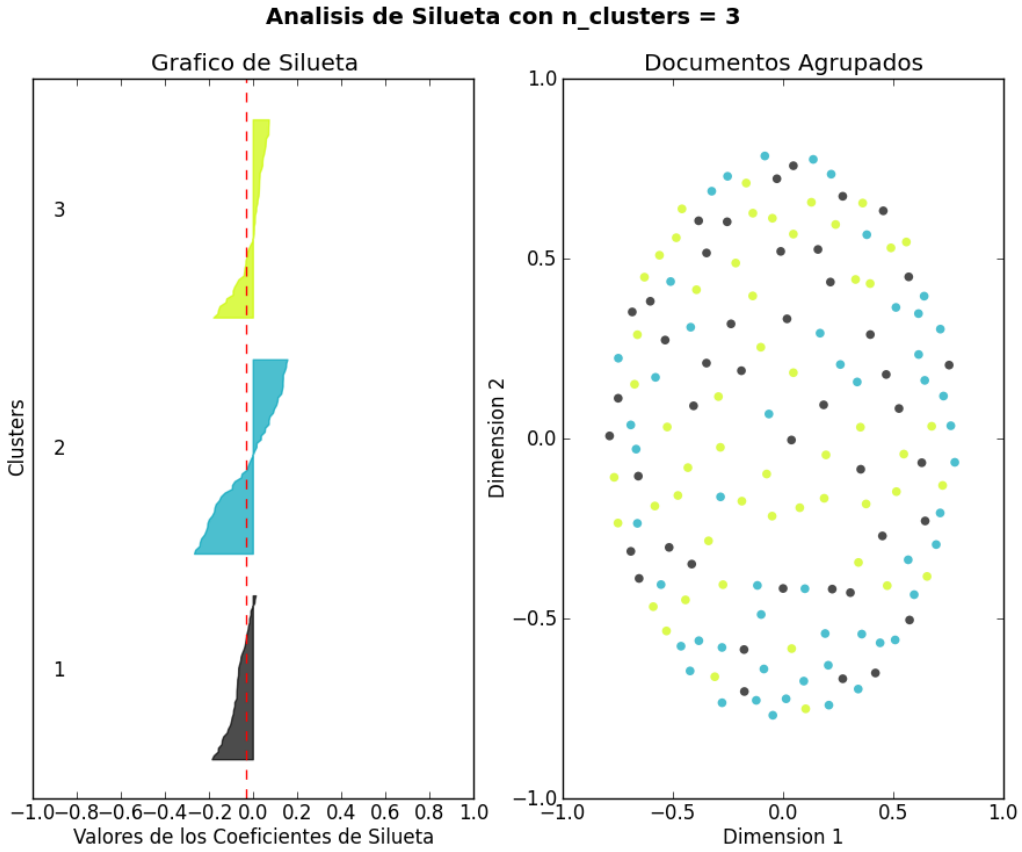

Figura 44. *Agrupamiento Dataset AAAI-13, Algoritmo CSCLP (ES-1), Buckshot*

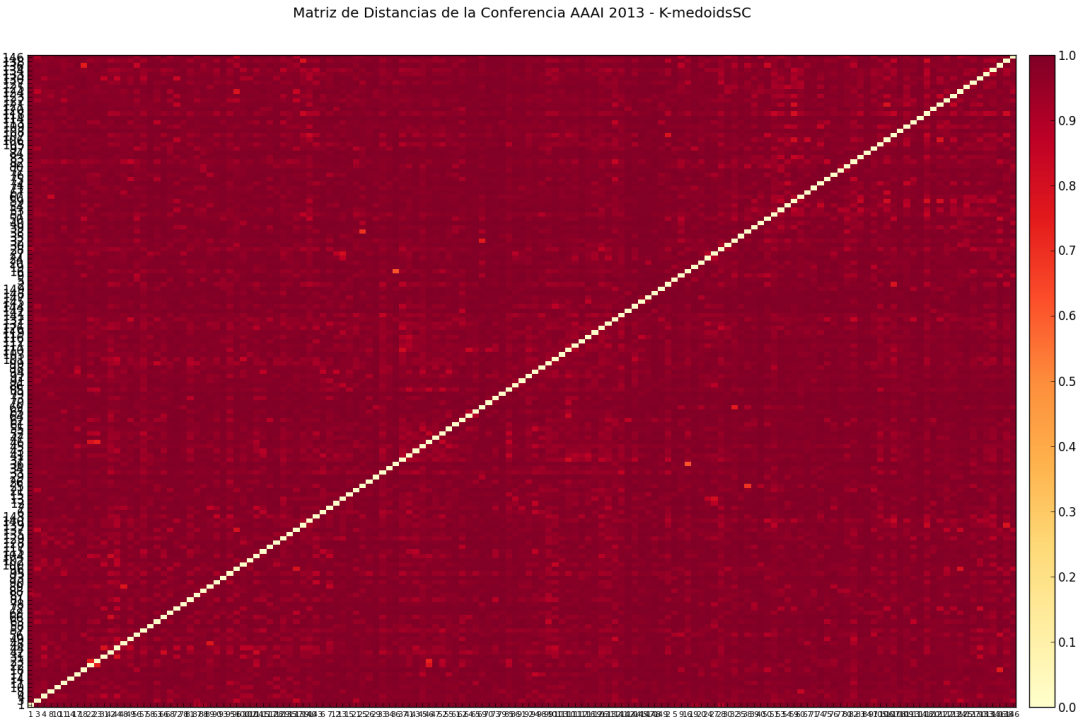

Figura 45. *Matriz de Disimilaridades Dataset AAAI-13, Algoritmo K-MedoidsSC (ES-1), Buckshot*

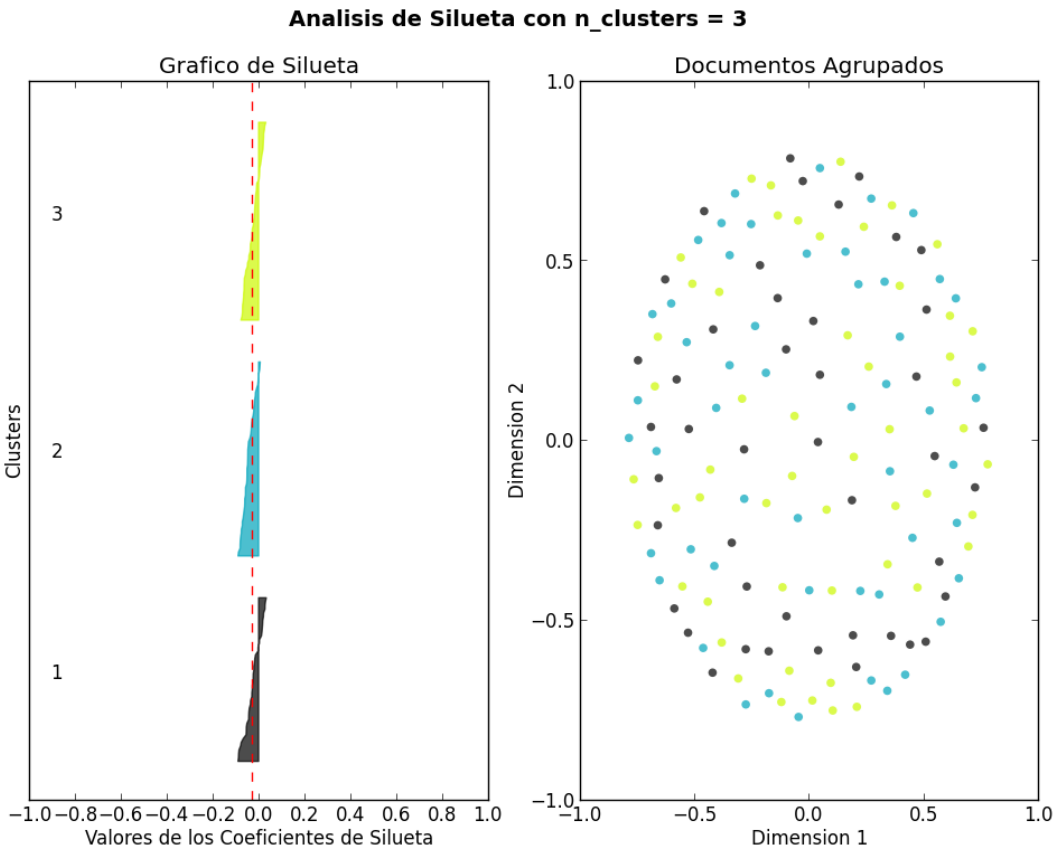

Figura 46. *Agrupamiento Dataset AAAI-13, Algoritmo K-MedoidsSC (ES-1), Buckshot*

Anexo VI

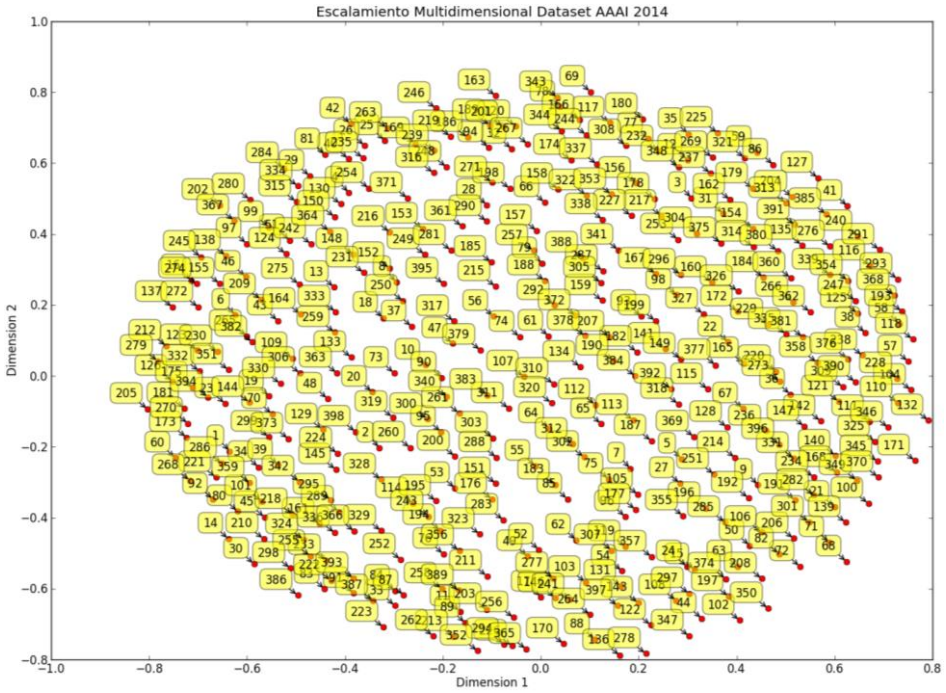

Figura 47. *Escalamiento Multidimensional de los Documentos de la Conferencia AAAI-14 (ES-1)*

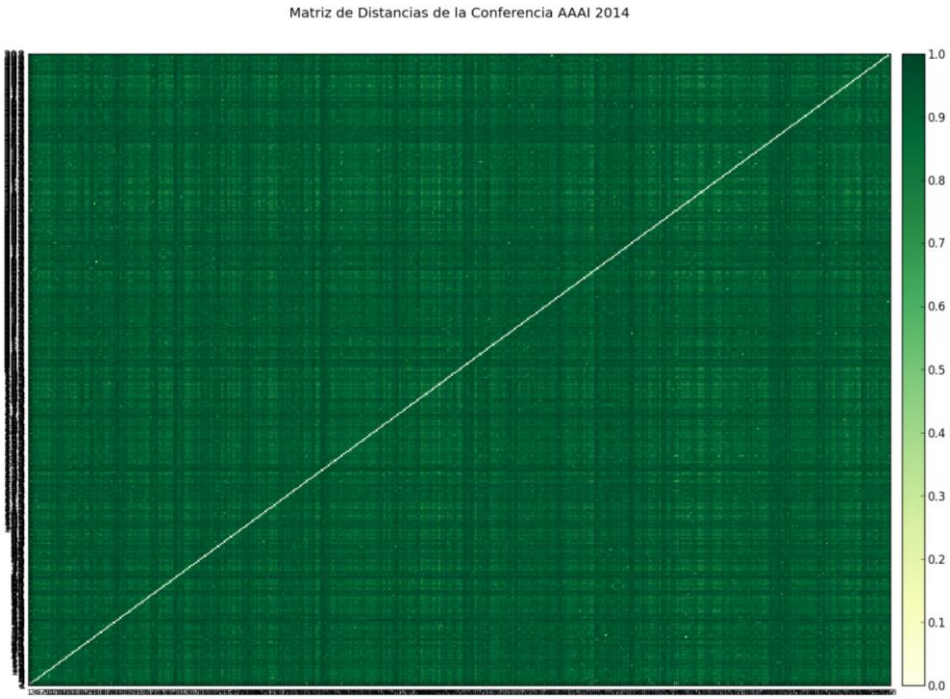

Figura 48. *Matriz de Disimilaridades, Dataset AAAI-14 (ES-1)*

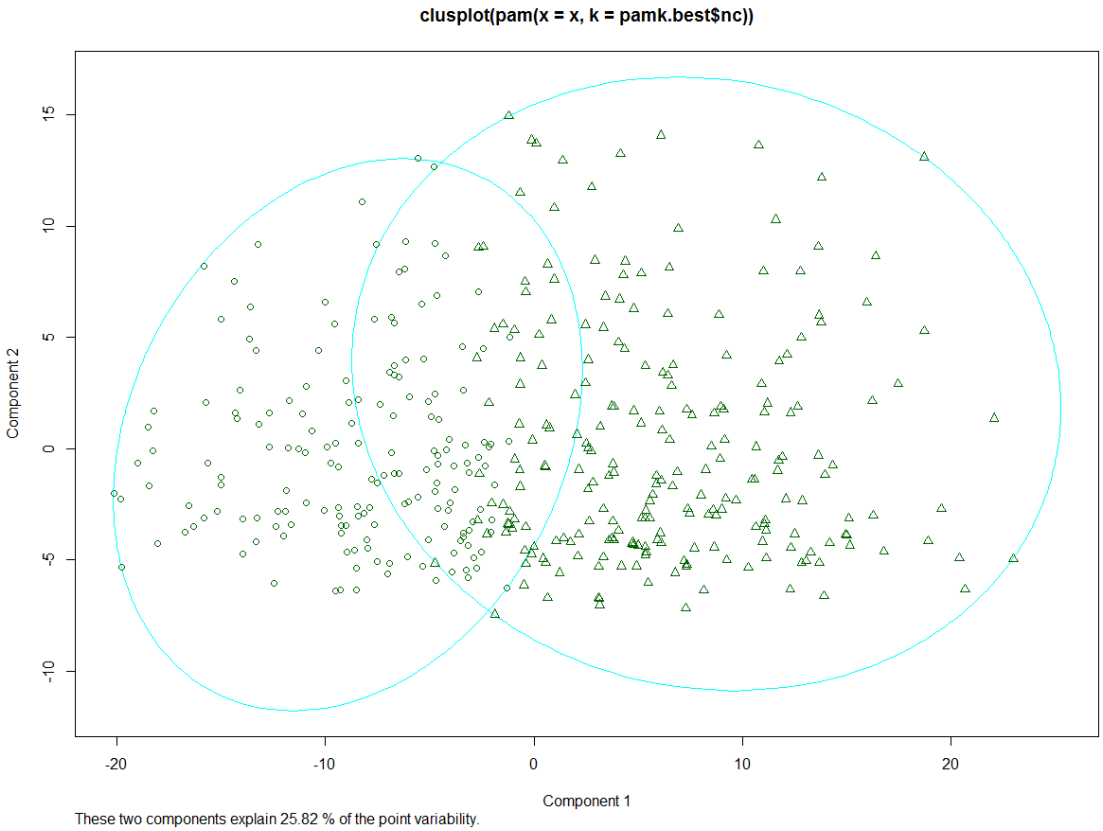

Figura 49. *Selección Número de Clusters, Dataset AAAI-14, k=2*

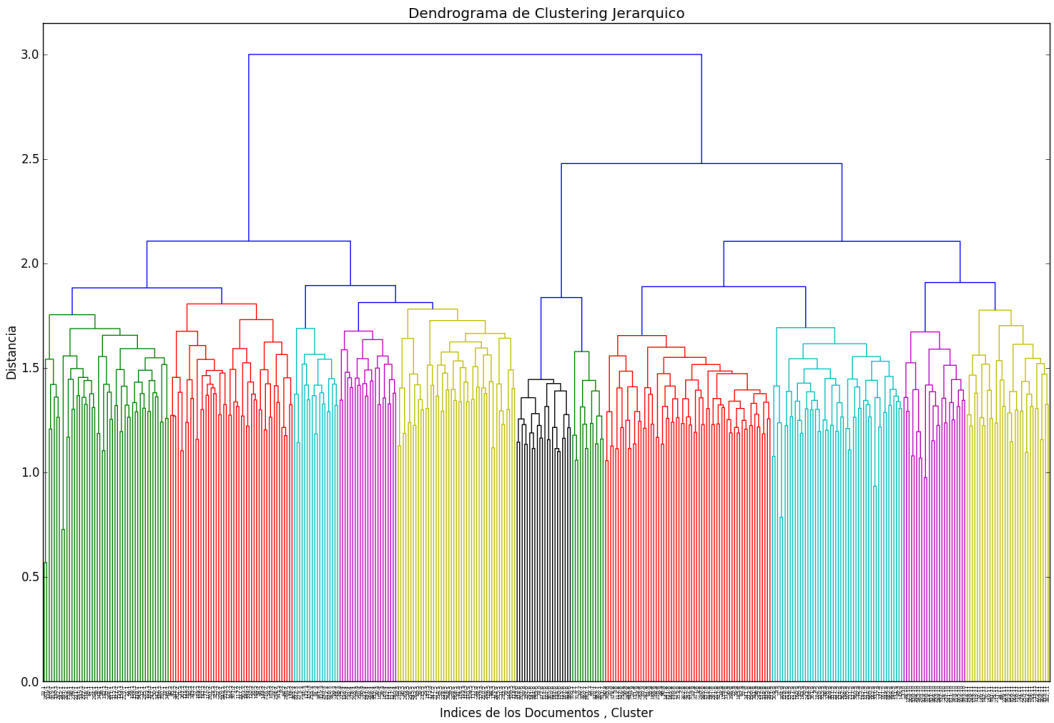

Figura 50. *Dendograma - Farthest Point Algorithm, Dataset AAAI-14 (ES-1) (11 grupos)*

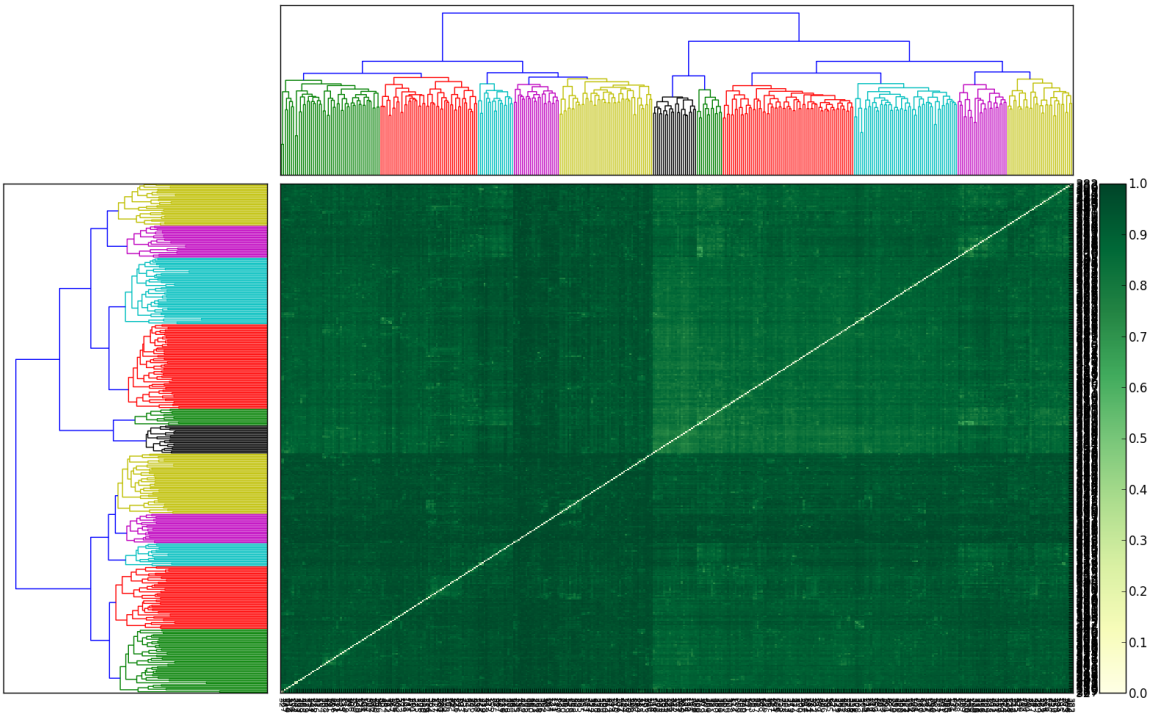

Figura 51. *Matriz Disimilaridades y Dendograma AHC-FPA, Dataset AAAI-14 (ES-1) (11 grupos)*

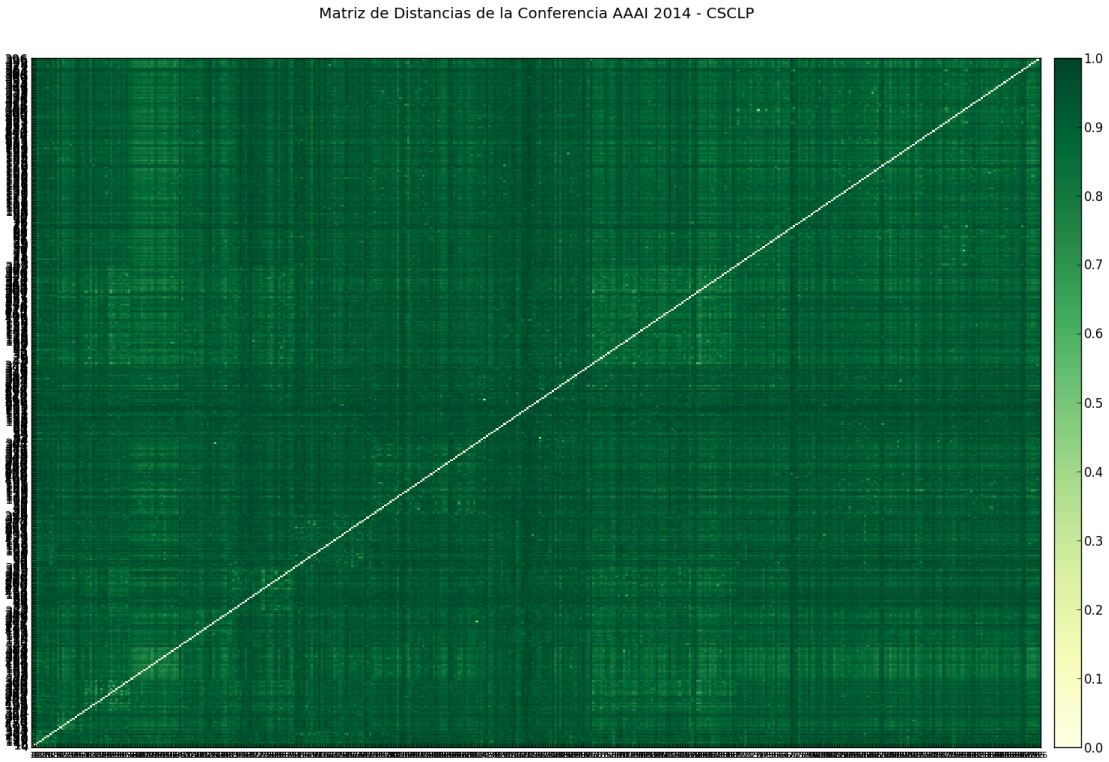

Figura 52. *Matriz de Disimilaridades Dataset AAAI-14, Algoritmo CSCLP (ES-1), Buckshot*

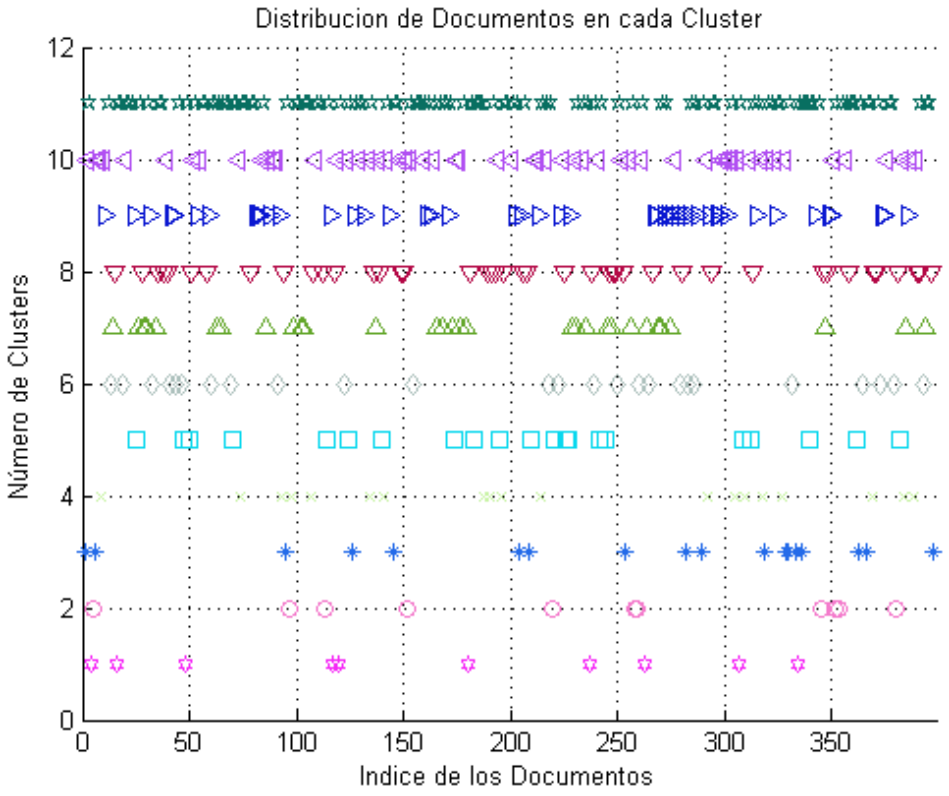

Figura 53. *Distribución de Documentos, Dataset AAAI-14, Algoritmo CSCLP (ES-1), Buckshot*

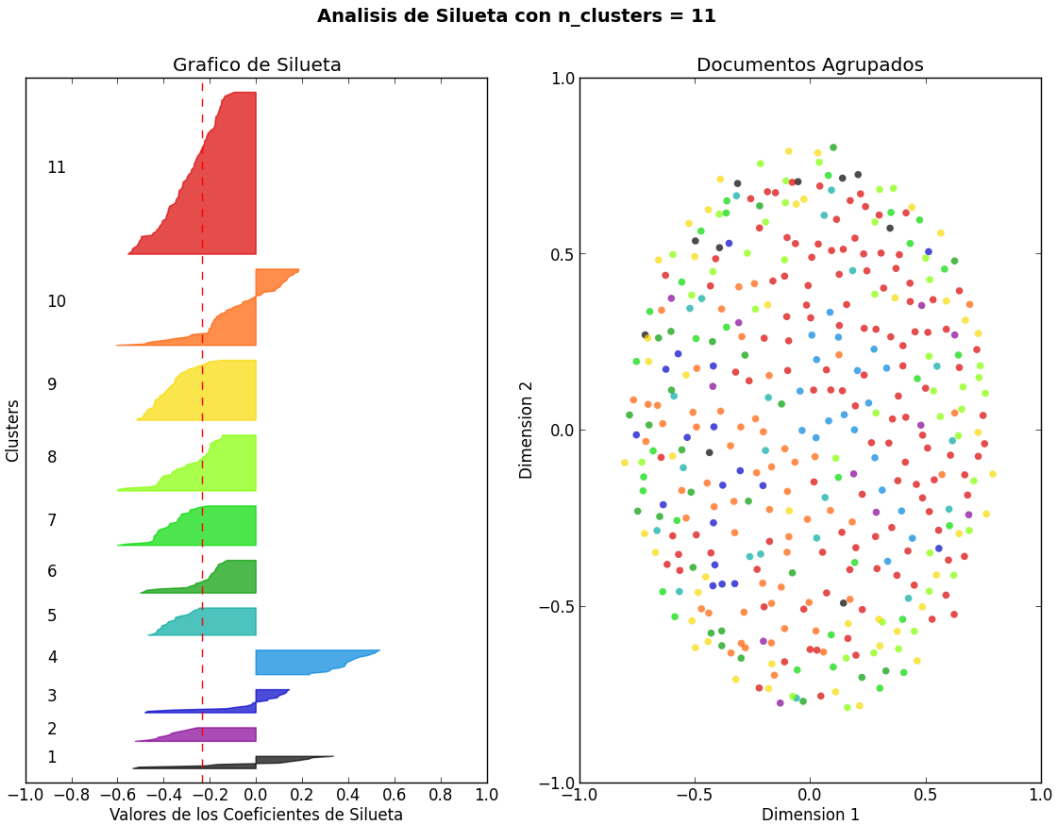

Figura 54. *Agrupamiento Dataset AAAI-14, Algoritmo CSCLP (ES-1), Buckshot*

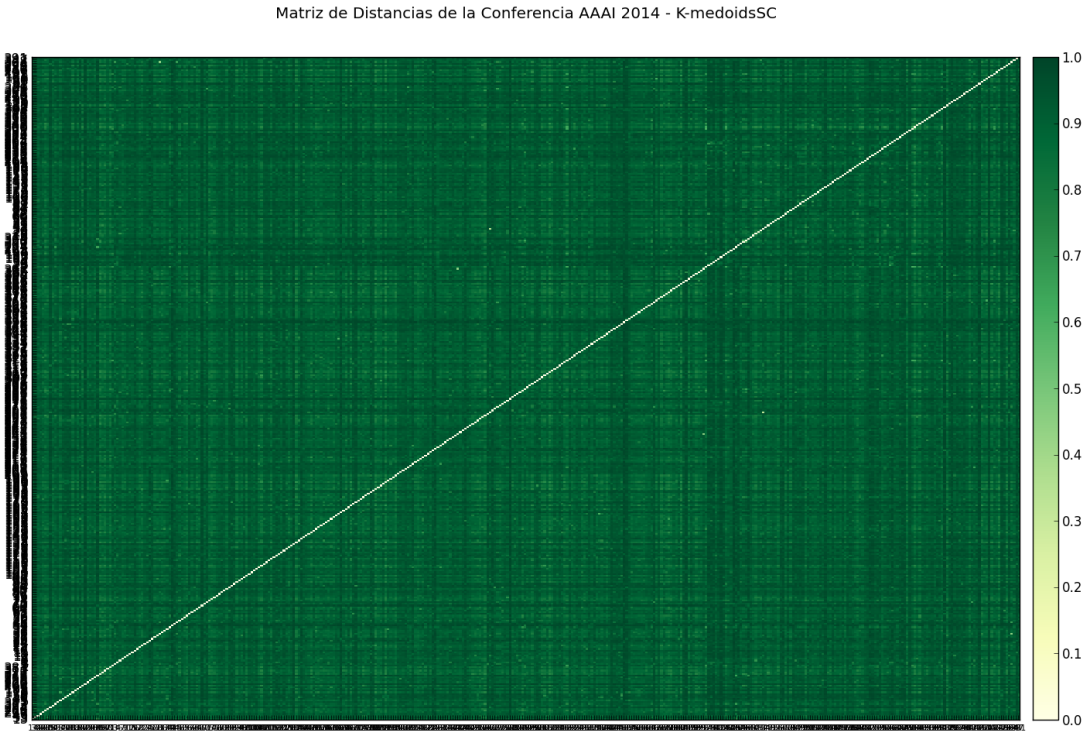

Figura 55. *Matriz de Disimilaridades Dataset AAAI-14, Algoritmo K-MedoidsSC (ES-1), Buckshot*

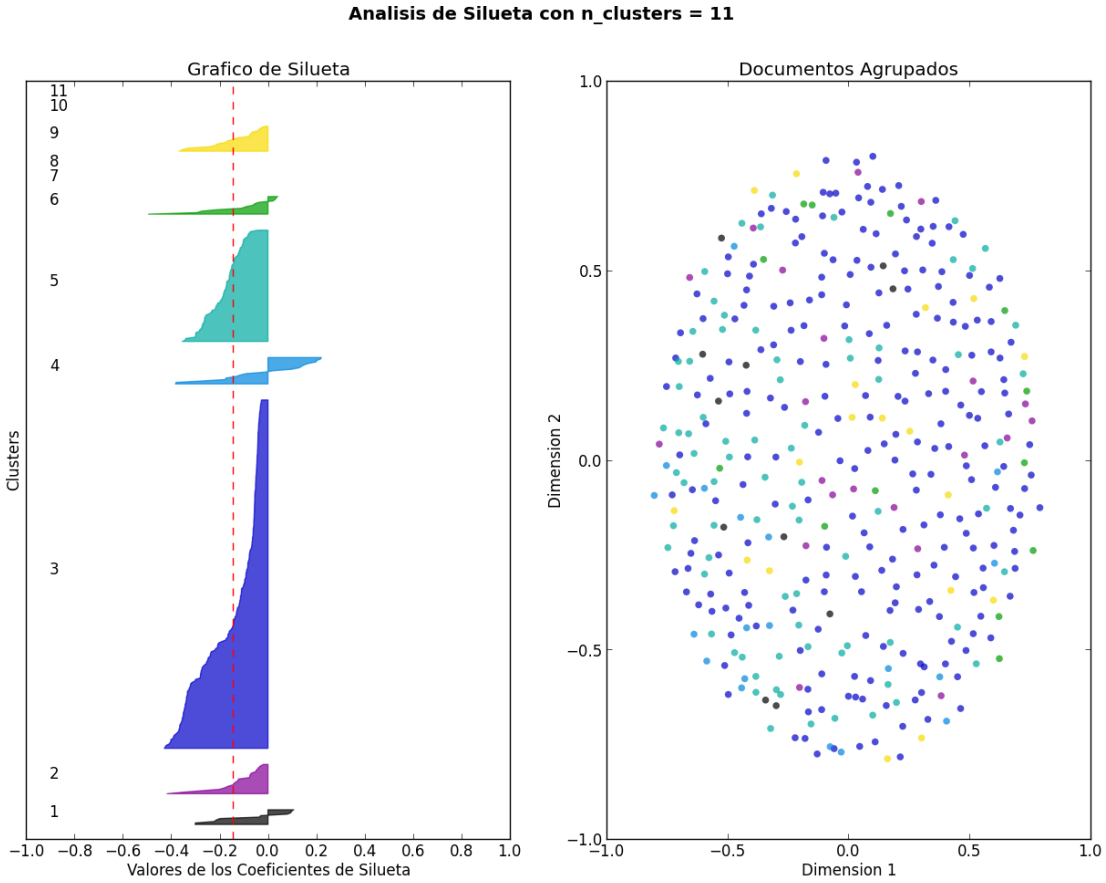

Figura 56. *Agrupamiento Dataset AAAI-14, Algoritmo K-MedoidsSC (ES-1), Buckshot*