Máster en Ingeniería de Sistemas y Control

Clustering de documentos de texto

Estudiante: Xabier Ganzábal García
Directoras: Raquel Dormido Canto y Natividad Duro Carralero
Curso académico: 2014-2015
Convocatoria: Septiembre

Máster en Ingeniería de Sistemas y Control

Clustering de documentos de texto

Proyecto tipo A
Estudiante: Xabier Ganzábal García
Directoras: Raquel Dormido Canto y Natividad Duro Carralero

Autorización

Autorizamos a la Universidad Complutense y a la UNED a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado: Xabier Ganzábal García

Firma del alumno

Resumen del proyecto

Este proyecto trata sobre *clustering* o agrupamiento de documentos de texto. El agrupamiento también es conocido como clasificación no supervisada. Dado un conjunto de elementos, los algoritmos de agrupamiento intentan encontrar una estructura interna que permita agrupar los elementos de manera coherente. Se habla de clasificación no supervisada porque no hay una serie de ejemplos ya clasificados que sirvan como punto de partida al proceso, como si ocurre en los problemas de clasificación supervisada.

En el caso de agrupamiento de documentos de texto surgen problemas añadidos. Los documentos de texto son en general no estructurados y además el procesamiento del lenguaje natural es un campo complejo con sus propias dificultades y técnicas. Es necesario preprocesar los documentos para limpiarlos y facilitar su tratamiento. Posteriormente, los documentos se deben transformar a una estructura de datos adecuada para los algoritmos de agrupamiento. Una característica del agrupamiento de documentos de texto es que se trabaja con datos de gran dimensionalidad, vectores de varios miles de componentes, lo que requiere la utilización de técnicas específicas.

El trabajo trata sobre cómo preprocesar los documentos y transformarlos a una representación útil. También se ocupa de los principales algoritmos de agrupamiento, de reducción de la dimensionalidad y de cómo validar los resultados obtenidos.

El proyecto incluye una aplicación que permite realizar experimentos de agrupamiento con un conjunto de datos bastante conocido, *20newsgroups*. Mediante una interfaz gráfica sencilla de utilizar es posible realizar agrupamientos con diferentes opciones de preprocesamiento, de reducción de la dimensionalidad y algoritmos de agrupamiento. La aplicación genera gráficas con los grupos resultantes y valora los resultados obtenidos.

Utilizando esta aplicación se ha realizado un experimento en el que se comparan algunas de las opciones de preprocesamiento, de reducción de la dimensionalidad y algoritmos de agrupamiento disponibles en la aplicación.

Palabras clave

Agrupamiento, agrupamiento espectral, agrupamiento jerárquico, agrupamiento por consenso, bolsa de palabras, *c-means*, Calinski-Harabasz, centroide, *clustering*, coeficiente de Jaccard, completitud, Cox-Lewis, Davies-Bouldin, Dunn, DBSCAN, disimilitud, distancia, etiquetado gramatical, Fowlkes y Mallow, homogeneidad, Hopkins, índice de Rand ajustado, información mutua, k-means, lematización, medida-F, medida-V, métrica, minado de texto, OPTICS, PCA, propagación de afinidad, proyección aleatoria, reducción de la dimensionalidad, silueta, similitud, *stemming*, *stopwords*, tendencia, tokenización, umbral de varianza, validación, VAT, Ward.

Índice

Índice de figuras

Índice de tablas

1. Introducción

El tema de este trabajo es el *clustering* o agrupamiento de documentos de texto. El *clustering* también se conoce como clasificación no supervisada. Esto quiere decir que no se parte de una serie de ejemplos ya clasificados que sirvan de guía para este proceso como ocurre en la clasificación supervisada. Los algoritmos de agrupamiento intentan encontrar una estructura dentro de los datos que permita repartirlos en varios grupos, de manera que los elementos de un mismo grupo sean más parecidos entre sí que los de grupos diferente.

En el caso de los documentos de textos se pretende, partiendo de una colección de documentos, agruparlos de manera coherente. Los documentos pueden ser, entre otros, correos electrónicos, páginas web, artículos de prensa, publicaciones científicas o comentarios en foros o redes sociales. Se trata de un área con diversas aplicaciones, útil desde el punto de vista científico y económico y en la que se investiga activamente.

1.1. Descripción del problema

La primera dificultad consiste en llegar a una representación adecuada y manejable de los documentos. En la mayoría de los casos de trata de texto no estructurado o semiestructurado escrito en el lenguaje natural. El lenguaje natural se presta a ambigüedades en varios niveles que dificultan su comprensión y entendimiento por parte de una aplicación informática.

Además, en algunas aplicaciones se utilizan textos recogidos en foros de internet o fuentes similares en las que los autores escriben de manera informal. Las abreviaturas, palabras con grafías no registradas y símbolos específicos son habituales. También es posible que el texto no aparezca solo en el documento, sino mezclado con imágenes, gráficos o anuncios. Por lo tanto es necesario realizar una primera fase de limpieza y simplificación.

Una vez transformados los documentos, la llamada *maldición de la dimensionalidad* se presenta con intensidad, ya no es raro que los documentos se representen con un vector de varios miles de dimensiones. Afortunadamente hay herramientas para seleccionar solo las que más información contienen.

Para el agrupamiento de documentos de texto se utilizan los algoritmos habituales de agrupamiento. Como veremos, hay una gran cantidad de opciones, cada una con puntos

fuertes y débiles. Según el caso, hay que indicar el número de grupos que se desea obtener y otros parámetros de cuya correcta elección depende que se obtenga un buen resultado. Hay que tener la precaución de comprobar que los datos realmente tengan una estructura de grupos para no llegar a conclusiones erróneas. Esto último se conoce como análisis de tendencia y se puede realizar antes del agrupamiento.

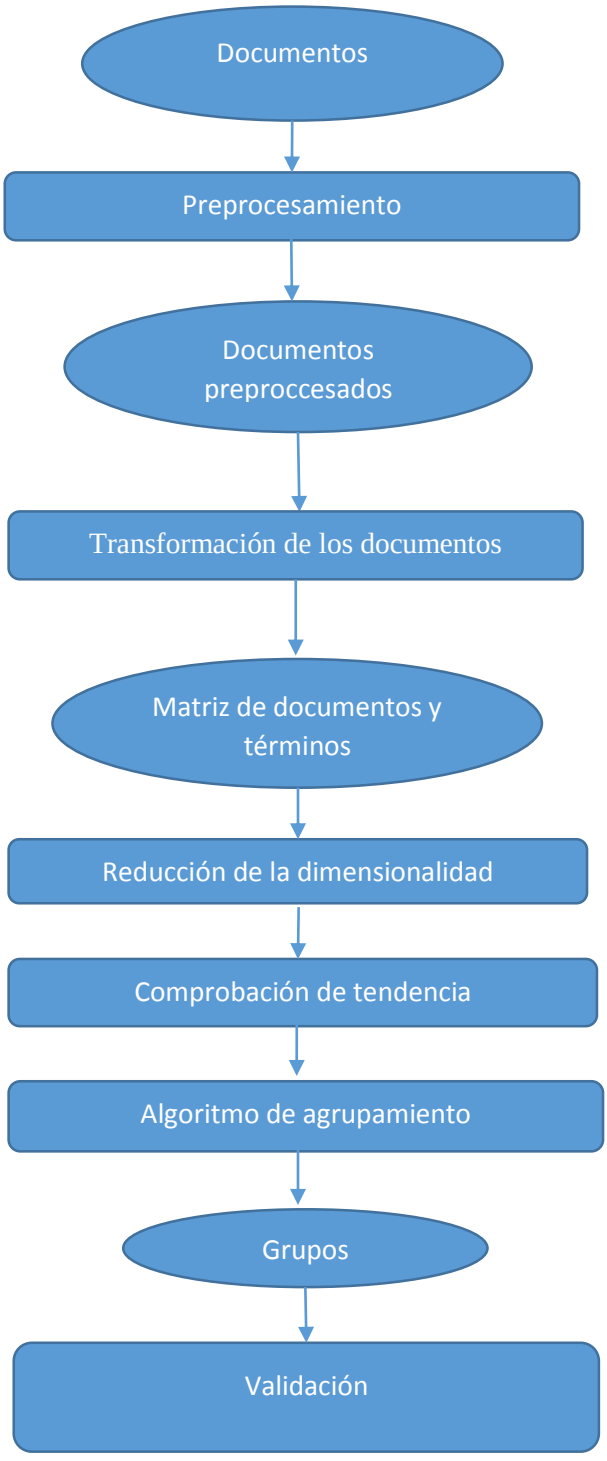

Ilustración 1. 1. Esquema del proceso de agrupamiento de documentos de texto.

Finalmente, cuando se ha obtenido un agrupamiento hay que evaluar su calidad. Se puede hacer utilizando como referencia un agrupamiento que se considere correcto si es que existe, pero en la mayoría de las aplicaciones de clasificación no supervisada no lo hay. En esos casos hay que utilizar índices que permitan evaluar los resultados obtenidos sin referencias externas, o que permitan comparar dos posibles agrupamientos. La ilustración 1.1 muestra todas las fases del proceso de agrupamiento.

1.2. Organización de la memoria

El capítulo 2 trata sobre los fundamentos del minado de texto. Además de definir el concepto y presentar sus aplicaciones más habituales, se explica el proceso mediante el que una colección de documentos de texto se preprocesa y se transforma a una estructura de datos apropiada para los algoritmos de agrupamiento. En concreto, se explica cómo transformar cada documento en un vector cuyas componentes representan la frecuencia de aparición de los términos del diccionario que se genera a partir de la colección de documentos.

El capítulo 3 se centra en el problema del agrupamiento, principalmente en los algoritmos. En la sección 3.1 se define el agrupamiento de datos. En la sección 3.2 se trata sobre las métricas habituales para representar la proximidad entre vectores y grupos de vectores. La sección 3.3 se ocupa de los algoritmos de agrupamiento más habituales. En la sección 3.4 se explica el agrupamiento por consenso, que consiste en tomar varios agrupamientos diferentes y generar otro a partir de ellos.

El capítulo 4 trata sobre el análisis de la tendencia y la validación de los resultados, para los que se explican varios índices. También se presentan varios algoritmos sobre la reducción de la dimensionalidad, algo fundamental en el caso del agrupamiento de documentos de texto.

En el capítulo 5 se describe la aplicación que se ha desarrollado como parte del proyecto. Se ha desarrollado en Python con una interfaz gráfica que permite realizar los análisis de manera sencilla. Esta aplicación trabaja con el conjunto de datos *20newsgroups*, bastante extendido como conjunto de prueba en el campo del procesamiento de lenguaje natural.

En el capítulo 6 se explica una serie de experimentos realizados con la aplicación desarrollada. El objetivo es poder establecer una comparación sobre la utilidad de diferentes opciones de preprocesamiento, técnicas de reducción de la dimensionalidad y algoritmos de agrupamiento.

Para terminar, el capítulo 7 recoge las conclusiones de la memoria y plantea las líneas abiertas para profundizar en el trabajo.

2. Preprocesamiento y transformación de los documentos

En este capítulo se presentan las técnicas básicas para el preprocesamiento de texto y cómo transformar los documentos a una estructura apropiada para los algoritmos de agrupamiento.

El primer paso es limpiar los documentos de texto. Pueden incluir imágenes, anuncios u otros elementos de los que hay que prescindir. También es habitual eliminar números, signos de puntuación y en general caracteres que no formen palabras. Posteriormente se pueden aplicar diversas opciones de preprocesamiento lingüístico.

Después se transforman los documentos a un modelo de representación adecuado. En este capítulo veremos el modelo bolsa de palabras, uno de los más extendidos.

2.1. Concepto de minado de texto

El minado de texto (*text mining*) también conocido como minado de datos de texto (*text data mining*), análisis de texto inteligente o descubrimiento de conocimiento en texto se refiere a la extracción de información útil de textos en leguaje natural.

El texto se puede encontrar en periódicos, correos electrónicos, críticas de productos o informes médicos. En la mayoría de los casos se trata de texto no estructurado, lo que dificulta la aplicación de las técnicas habituales en minería de datos.

El minado de texto es un campo multidisplicinar que incorpora [Vallinaku] minado de datos, minado web, extracción y recuperación de información y procesado del lenguaje natural.

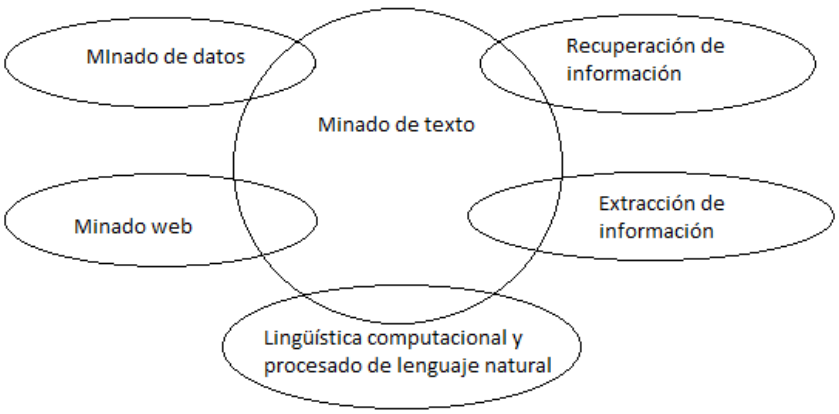

Ilustración 2. 1. Relación del minado de datos otras áreas.

- Minería de datos. La minería de datos es el proceso de encontrar relaciones entre datos estructurados. Coincide con la minería de texto en que procesan gran cantidad de datos para encontrar información útil. La diferencia está en que el minado de datos trabaja con datos estructurados procedentes de una base datos y el de texto con datos no estructurados procedentes de periódicos, páginas de internet y chats entre otras fuentes.
- Minería web. Consiste en aplicar la minería de datos para encontrar patrones y tendencias útiles en la web, a partir de las páginas web. Es muy utilizado en el comercio electrónico y los métodos de aprendizaje en línea. Se puede dividir en tres tipos según qué se analice: uso, estructura y contenido de la web. La minería de uso web analiza el comportamiento de los usuarios a través de los registros del servidor y el historial en el cliente. El minado de estructura web analiza la estructura de las conexiones dentro de un sitio web. El minado de contenido web extrae información relevante del contenido de las páginas.
- Recuperación de información. En general, la recuperación de información consiste en responder a preguntas. Se trata de encontrar la información apropiada a la consulta del usuario. La aplicación más importante la constituyen los buscadores web, que se ocupan de encontrar un subconjunto de páginas de la web según el tema de interés del usuario.
- Extracción de información. Se refiere a la identificación de palabras o términos que permitan identificar un documento. Hay que procesar el lenguaje natural para identificar elementos relevantes de información, como nombre de personas, organizaciones o lugares.
- Procesado de lenguaje natural. Esta área incluye conceptos lingüísticos como:
    - Clasificar las palabras según su categoría gramatical: nombre, verbo, adjetivo…Se conoce como POS, *part-of-speech*.
    - Stemming. Consiste en encontrar la raíz a una palabra. Es decir, la parte común a las palabras de una misma familia. No coincide necesariamente con el lexema.
    - Lematización. En este caso no se busca la raíz de las palabras, sino una forma canónica de las mismas, el lema. Por ejemplo, el lema para ‘nuevos’, ‘nuevas’, ‘nuevo’ y ‘nueva’ sería ‘nuevo’.

- Desambiguación. Escoger el significado apropiado de una palabra que puede tener varios significados o sentidos.

2.1.1. Aplicaciones del minado de texto

Entre las aplicaciones más destacadas están:

- Buscadores web. Los buscadores web indexan una gran cantidad de páginas que deben clasificar por su categoría o relevancia. Es una de las aplicaciones más importantes, ya que son una herramienta básica para todos los usuarios. Un caso especial se da al buscar noticias, en las que los buscadores ofrecen agrupadas las noticias de diferentes medios referidas a un mismo hecho.
- Análisis de sentimiento. En este caso se pretende saber si los consumidores o usuarios de ciertos productos tienen buena o mala opinión de él a partir de sus comentarios, generalmente recogidos de blogs o foros de internet. Es habitual usarlo para películas.
- Detección de spam. Esta aplicación es habitual en los servidores de correo web. Se trata de identificar los mensajes tipo spam para filtrarlos al usuario. Aunque en parte se hace a través del estudio del origen de los mensajes, también se utiliza el minado de texto para identificarlos mediante su contenido.
- Información de fondos bibliográficos. Es una aplicación especialmente importante en el caso de organizaciones que tienen una gran colección de documentos. Se trata de generar automáticamente una lista de palabras clave que describan la categoría o utilidad de un documento de cara a su clasificación o indexación, para así poder buscarlos de manera efectiva. Es muy habitual en el caso de publicaciones científicas, sobre todo para artículos antiguos que no fueron clasificados con criterios actuales en el momento de su creación.
- Resumen de documentos. Esta aplicación se puede ver como un avance sobre la anterior. En lugar de extraer solo las palabras clave necesarias para clasificar el texto, el objetivo es generar un resumen del mismo seleccionando la información más relevante, también en lenguaje natural de manera que una persona lo pueda leer sin problemas.
- Seguridad. Se puede utilizar el minado de texto para monitorizar blogs, foros y otras páginas web con el fin de detectar posibles amenazas a la seguridad de un país o al orden público.
- Bioinformática. El minado de texto se utiliza para identificar información disponible en la literatura biomédica. Ayuda a analizar y encontrar tendencias en la interacción

entre genes, enfermedades y drogas y permite encontrar asociaciones indirectas entre entidades diferentes. Por ejemplo, qué enfermedades están relacionadas con un gen en particular o qué compuestos químicos son relevantes para cierta enfermedad.

La siguiente tabla muestra algunas herramientas:

Herramienta	Función
Text Analyst	Agrupa documentos usando una red semántica
SAS Textminer	Realiza un análisis estadístico de los documentos para descubrir temas y conceptos
Sysomos	Análisis del sentimiento en redes sociales
TextFinder	Búsqueda de rápido texto para aplicaciones con datos genómicos
Intelligent Miner	Agrupamiento y clasificación de documentos

Tabla 2. 1. Algunas aplicaciones de minado de texto.

2.1.2. Técnicas de minado de texto

La clasificación y el agrupamiento, técnicas clásicas en el minado de datos, son también habituales en el caso del minado de texto.

La clasificación asigna los documentos a clases predefinidas. Funciona de manera supervisada, es decir, a partir de un conjunto de documentos de ejemplo ya asignados a una categoría. Se puede encontrar un ejemplo de clasificación de mensajes SMS como spam en [Almeida].

El agrupamiento, por el contrario, no tiene clases predefinidas ni ejemplos que sirvan como guía al proceso. Funciona de manera no supervisada. El objetivo es encontrar diferentes grupos significativos dentro del conjunto de documentos con el que trabaja. Es la técnica utilizada, por ejemplo, para agrupar noticias obtenidas de varias fuentes según el hecho al que se refieran. En el capítulo 3 se estudia este tema en detalle.

2.2. Preprocesamiento de documentos

Como ya se ha comentado, es habitual que se intente agrupar una colección de documentos de texto semiestructurado o no estructurado. Antes de convertir la colección a una estructura de datos apropiada para los algoritmos de agrupamiento se realizado un preprocesamiento de los documentos que incluye los siguientes los pasos[Kwale].

- Limpieza del texto.

- Tokenización.
- Filtrado.
- Stemming y lemmatization.
- Preprocesado lingüístico.

2.2.1. Limpieza del texto

El primer paso consiste en eliminar del documento las partes que no son relevantes para interpretar el contenido. Por ejemplo, eliminar anuncios de una página web, tablas, gráficos y etiquetas.

2.2.2. Tokenización

Después se eliminan los signos de puntuación, como comas, puntos y guiones y los caracteres no textuales como los tabuladores, sustituyéndolos por espacios. Según la aplicación también se pueden eliminar los números.

Ejemplo: El texto ‘Carlos, te ha llamado tu hermana.’ se convertiría en ‘Carlos te ha llamado tu hermana’.

Tras este paso, el conjunto de palabras resultante forma el diccionario de la colección de documentos.

2.2.3. Filtrado

El objetivo de este paso es reducir el tamaño del diccionario, que normalmente es de miles de palabras. Se eliminan las palabras menos importantes:

- Las palabras vacías (*stop words*). Se trata de palabra que no tienen significado, como las preposiciones o los artículos.

  Ejemplo: Utilizando el conjunto de palabras vacías en español del paquete nltk la frase ‘Las patatas de Mallorca me gustan mucho’ se transforma en 'patatas mallorca gustan’.

  Ejemplo: Los mensajes del grupo ‘comp.graphics’ del conjunto de datos *20newsgroups* tienen en total 9297 palabras diferentes. Si se filtran utilizando el conjunto de palabras vacías en inglés del paquete nltk quedan solo 9177, una reducción del 1.29%.

  En el capítulo 4 se explica este conjunto de datos en profundidad.

- Palabras muy habituales. Si una palabra es muy habitual, ya sea en general o dentro del conjunto de documentos estudiado, es poco útil para clasificar, agrupar o relacionar los documentos en cuestión.
- Palabras muy poco habituales. Igual que en el caso anterior, normalmente estas palabras no resultan muy relevantes.

2.2.4. Stemming y lematización

En este caso se pretende reducir el tamaño del diccionario unificando varias palabras en una a través de una forma común.

2.2.4.1. Stemming

En el *stemming* se busca una raíz común para una familia de palabras. Pretende mejorar la recuperación de la información al unificar términos similares en uno solo. Así, al buscar documentos con la palabra 'comida', se devolverían también los que contengan 'comidas' o 'comer'. No es una técnica válida para todos los idiomas. Por ejemplo, no tiene sentido aplicarlo en chino.

Esta raíz no tiene por qué ser el lexema de la palabra, basta con que palabras relacionadas sean representadas por la misma raíz, que no tiene porqué ser una palabra válida en sí misma.

Hay varios enfoques para los algoritmos de *stemming*:

- Tabla de consulta (*lookup table*). Este método tiene como ventajas que es sencillo y admite excepciones. La desventaja es que todas las inflexiones de una palabra deben aparecer en las tablas. Según el idioma, puede haber muchas posibilidades para una palabra. Para ayudar en la enorme tarea de generar la tabla se puede utilizar una herramienta automática, aunque pueda dar lugar a errores.
- Eliminación de sufijos (o prefijos). Estos algoritmos utilizan una serie de reglas fijas para encontrar la raíz. En español estas reglas podrían consistir en eliminar las terminaciones 'o', 'a', 'mente' o 'aba'. Puede ser útil identificar primero la categoría gramatical de la palabra para poder aplicar reglas pertinentes.

Se han definido muchos algoritmos de *stemming* diferentes. El más citado es el de Porter [Porter], para inglés. En el experimento del capítulo 4 se comparará con otro algoritmo habitual, el de Lancaster. El lenguaje Snowball se creó para facilitar la creación de stemmers.

Ejemplo: Snowball incorpora *stemmers* para varios idiomas. Este pequeño *script* utiliza el de español. La salida en todos los casos es la misma, 'com'.

```
from nltk.stem import SnowballStemmer
stemmer = SnowballStemmer("spanish")
print stemmer.stem("comer")
print stemmer.stem("comida")
print stemmer.stem("comidas")
print stemmer.stem("comieron")
```

Aunque en un primer momento pueda parecer una buena idea para reducir el diccionario sin perder información, no siempre es así. En [Potts] se muestra como los *stemmers* de Porter y Lancaster no resultan útiles para el análisis de sentimiento, una de las áreas más importantes dentro del minado de texto. Esto es así porque reduce a una forma común palabras con matices positivos y negativos, que es precisamente lo que se intenta diferenciar en este tipo de análisis. La siguiente tabla contiene algunos de estos ejemplos contraproducentes.

Palabra con sentido positivo	**Palabra con sentido positivo**	**Forma reducida común**
captivation	captive	captiv
common	commoner	common
defend	defendant	defend
defense	defensive	defens
dependability	dependent	depend
dependable	dependent	depend
desirable	desire	desir
dominance	dominate	domin
dominance	domination	domin

En este ejemplo se determina si una palabra tiene connotación negativa o positiva utiliza el *Harvard General Inquirer* [Inquirer].

Algortimo de Porter

Como ejemplo, vamos a ver en detalle cómo funciona el *stemmer* de Porter. Se trata de un algoritmo basado en la eliminación de sufijos. Lo primero es dar algunas definiciones. Cada palabra o parte de una palabra se puede representar de una de estas formas:

CVCV ... C

CVCV ... V

VCVC ... C

VCVC ... V

donde C es una cadena de consonantes de longitud mayor que 0, y V una cadena de vocales de longitud mayor o igual que 0. Las cuatro formas anteriores se pueden representar como

[C]VCVC ... [V]

donde los corchetes indican que el término es opcional. Si se utiliza (VC){m} para representar que el par VC se repite m veces, la expresión anterior será

[C](VC){m}[V]

m recibe el nombre de medida de una palabra o parte de una palabra. Algunos ejemplos son:

m=0 TR, EE, TREE, Y, BY.

m=1 TROUBLE, OATS, TREES, IVY.

m=2 TROUBLES, PRIVATE, OATEN, ORRERY.

Según el contexto, la 'y' se considera vocal o consonante.

En el algoritmo se usan reglas de la forma:

(condición) S1 -> S2

Quiere decir que si la palabra termina con el sufijo S1 y la parte de la palabra anterior al sufijo cumple la condición, S1 se reemplaza por S2. Es habitual que se exprese en función de m, por ejemplo la regla

(m > 1) EMENT ->

se aplica a las palabras que acaban en EMENT y en las que m es mayor que 1; en este caso S2 es la cadena vacía, así que el sufijo EMENT simplemente se suprime. Por ejemplo la palabra 'REPLACEMENT' se transformaría en 'REPLACE'. En cambio la palabra 'ELEMENT' no cumple la condición y no se modificaría.

La condición puede incluir

*S – la raíz acaba en S (lo mismo para otras letras).

v - la raíz contiene una vocal.

*d - la raíz acaba en una consonante doble.

*o - la raíz acaba en consonante-vocal-consonante, y la segunda consonanten no es W, X or Y.

Además, hay condiciones compuestas con los operadores 'and', 'or' y not'.

En un conjunto de reglas escritas una debajo de otra solo se aplicará una, la que tenga la parte S1 más larga concordante con la palabra en cuestión. Por ejemplo, para este conjunto de reglas (la condición es nula en todas ellas):

SSES -> SS

IES -> I

SS -> SS

S ->

la palabra 'CARESSES' se mapea a 'CARESS' aplicando la primera regla, ya que 'SSES' es la parte coincidente más larga. Igualmente, 'CARESS' se mapea a 'CARESS, sin cambios, aplicando la tercera regla y 'CARES' a 'CARE', aplicando la última.

Introducidos los conceptos básicos, ya podemos pasar a ver las reglas concretas del algoritmo. Las palabras se van transformando siguiendo una serie de pasos sucesivos. Para cada paso se muestra una tabla con las reglas en la primera columna y uno o más ejemplos de aplicación en la segunda.

El primer paso se ocupa de plurales y participios. Es el más complejo.

Paso 1A

REGLA	EJEMPLO(S)
SSES -> SS	caresses -> caress
IES -> I	ponies -> poni ties -> ti
SS -> SS	caress -> caress
S ->	cats -> cat

Paso 1B

REGLA	EJEMPLO(S)
(m>0) EED -> EE	feed -> feed agreed -> agree
(*v*) ED ->	plastered -> plaster bled -> bled
(*v*) ING ->	motoring -> motor sing -> sing

Si se aplican la segunda o tercera de las reglas del paso 1b (entre paréntesis aparece el sufijo eliminado por esa regla), se hace lo siguiente:

REGLA	EJEMPLO(S)
AT -> ATE	conflat(ed) -> conflate
BL -> BLE	troubl(ed) -> trouble
IZ -> IZE	siz(ed) -> size
(*d and not (*L or *S or *Z)) -> una sola letra *Si la palabra acaba en consonante doble que no sea L, S o Z se deja una sola consonante*	hopp(ing) -> hop tann(ed) -> tan fall(ing) -> fall hiss(ing) -> hiss fizz(ed) -> fizz
(m=1 and *o) -> E	fail(ing) -> fail fil(ing) -> file

Esta es la parte más confusa del algoritmo. Las tres primeras reglas vuelven a poner la 'E' que se quitó en el paso anterior, de manera que las terminaciones correspondientes se reconozcan en el paso 2. La cuarta regla se ocupa de dobles consonantes al final, eliminando una de ellas según la letra de la que se trate. La última regla añade una 'E' que no se había eliminado en los pasos anteriores.

Paso 2

(m>0) ATIONAL -> ATE	relational -> relate
(m>0) TIONAL -> TION	conditional -> condition rational -> rational
(m>0) ENCI -> ENCE	valenci -> valence
(m>0) ANCI -> ANCE	hesitanci -> hesitance
(m>0) IZER -> IZE	digitizer -> digitize
(m>0) ABLI -> ABLE	conformabli -> conformable
(m>0) ALLI -> AL	radicalli -> radical
(m>0) ENTLI -> ENT	differentli -> different
(m>0) ELI -> E	vileli - > vile
(m>0) OUSLI -> OUS	analogousli -> analogous
(m>0) IZATION -> IZE	vietnamization -> vietnamize
(m>0) ATION -> ATE	predication -> predicate
(m>0) ATOR -> ATE	operator -> operate
(m>0) ALISM -> AL	feudalism -> feudal
(m>0) IVENESS -> IVE	decisiveness -> decisive
(m>0) FULNESS -> FUL	hopefulness -> hopeful
(m>0) OUSNESS -> OUS	callousness -> callous
(m>0) ALITI -> AL	formaliti -> formal
(m>0) IVITI -> IVE	sensitiviti -> sensitive
(m>0) BILITI -> BLE	sensibiliti -> sensible

Paso 3

(m>0) ICATE -> IC	triplicate -> triplic
(m>0) ATIVE ->	formative -> form
(m>0) ALIZE -> AL	formalize -> formal

(m>0) ICITI -> IC	electriciti -> electric
(m>0) ICAL -> IC	electrical -> electric
(m>0) FUL ->	hopeful -> hope
(m>0) NESS ->	goodness -> good

Paso 4

(m>1) AL ->	revival -> reviv
(m>1) ANCE ->	allowance -> allow
(m>1) ENCE ->	inference -> infer
(m>1) ER ->	airliner -> airlin
(m>1) IC ->	gyroscopic -> gyroscop
(m>1) ABLE ->	adjustable -> adjust
(m>1) IBLE ->	defensible -> defens
(m>1) ANT ->	irritant -> irrit
(m>1) EMENT ->	replacement -> replac
(m>1) MENT ->	adjustment -> adjust
(m>1) ENT ->	dependent -> depend
(m>1 and (*S or *T)) ION ->	adoption -> adopt
(m>1) OU ->	homologou -> homolog
(m>1) ISM ->	communism -> commun
(m>1) ATE ->	activate -> activ
(m>1) ITI ->	angulariti -> angular
(m>1) OUS ->	homologous -> homolog
(m>1) IVE ->	effective -> effect
(m>1) IZE ->	bowdlerize -> bowdler

Tras este paso los sufijos ya están eliminados. Solo faltan unos retoques.

Paso 5A

(m>1) E ->	probate -> probat rate -> rate

(m=1 and not *o) E ->	cease -> ceas

Paso 5B

(m > 1 and *d and *L) -> single letter	controll -> control roll -> roll

Las palabras con varios sufijos se recortan en varios pasos, aunque aplicando solo una regla de cada grupo. Por ejemplo:
GENERALIZATIONS -> GENERALIZATION (paso 1) -> GENERALIZE (paso 2) -> GENERAL (paso 3) -> GENER (paso 4).

2.2.4.2. Lematización

La lematización sigue un enfoque diferente. En lugar de buscar la raíz de las palabras busca una forma base para las diferentes inflexiones, el lema. Por ejemplo, para las palabras 'hermano', 'hermanos', 'hermana' y 'hermanas' el lema podría ser 'hermano'.

No hay tanta diversidad de *lematizadores* como de *stemmers*. Una posibilidad es usar Wordnet [Wordnet], que proporciona una función para encontrar el lema asociado a una palabra. Solo está disponible en inglés.

Wordnet

Wordnet es una gran base de datos léxica para el idioma inglés. La relación principal entre las palabras en Wordnet es la sinonimía, palabras que tienen el mismo o parecido significado y son intercambiables en muchos contextos. Nombres, verbos, adjetivos y adverbios se agrupan en conjuntos de sinónimos, *synsets*, que expresan un mismo concepto.

Estos conjuntos están vinculados entre sí mediante relaciones semánticas, conceptuales y léxicas. Esto da lugar a una red de palabras y conceptos que se puede recorrer con un navegador. Esta estructura hace que se una herramienta útil para las aplicaciones de lingüística computacional y procesamiento de leguaje natural.

Además de poder consultarla a través de la página de la universidad de Princeton, la base de datos Wordnet está disponible para su descarga y se han desarrollado paquetes para usarla desde varios lenguajes de programación. En la aplicación desarrollada como parte del proyecto se ha utilizado a través del paquete ntlk de Python.

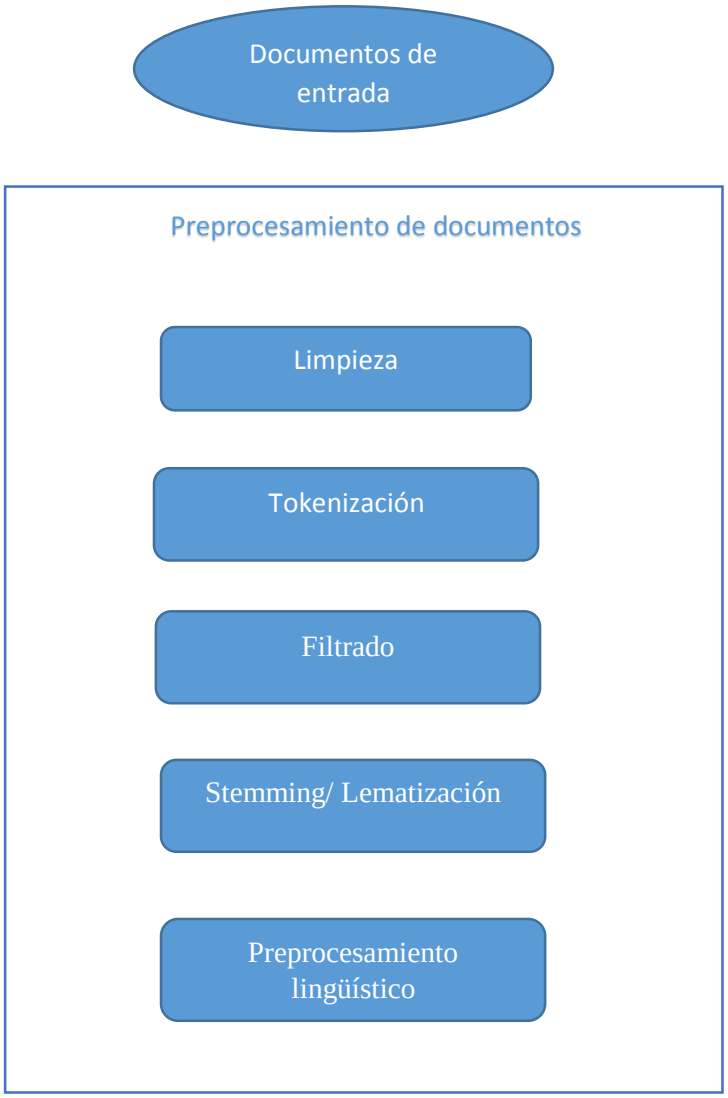

Ilustración 2. 2. Pasos en el preprocesamiento de documentos.

2.2.5. Preprocesado lingüístico

Además del *stemming* y la lematización hay varios métodos de preprocesado lingüístico que permiten mejorar la información disponible, entre otros:

Etiquetado gramatical

Consiste en asignar a cada palabra su categoría gramatical, es decir, nombre, verbo, adjetivo…, aunque los etiquetadores en procesamiento de lenguaje natural utilizan un mayor grado de detalle. No basta con tener una lista de palabras, hay que analizar también el contexto en que aparece para poder determinar la categoría gramatical. Por ejemplo, la palabra 'dado' puede ser un nombre o un participio. Esta información es útil para desambiguar el sentido de una palabra o para realizar un análisis sintáctico que permita comprender mejor una oración.

Fragmentación de texto (*Text chunking*)

Consiste en agrupar varias palabras en trozos o bloques de texto. Por ejemplo, el *NP-chunking* se realiza después del etiquetado gramatical para identificar sintagmas nominales.

Desambiguación

Una misma palabra puede tener diferentes significados. Por ejemplo, 'dado' puede ser un nombre o un verbo, con significados muy diferentes. Se puede intentar deducir el significado apropiado según el contexto o la función gramatical que realice la palabra. La base de datos léxica Wordnet, descrita en el apartado 2.2.4 se puede utilizar con este fin.

2.3. Representación de los documentos

Una vez preprocesados los documentos el siguiente paso es obtener una representación formal de los mismos apropiada para los algoritmos de agrupamiento. La representación más habitual es el modelo de espacio vectorial. En este modelo cada documento viene representado por un vector. Las componentes de este vector representan la frecuencia en el documento de los términos del diccionario que se obtuvo con el procedimiento de preprocesado. Esta representación también es conocida como 'bolsa de palabras' (*bag of words*), ya que cada documento pasa a representarse como un multiconjunto de palabras, ignorando otros aspectos como el orden en el que aparecían en el documento originalmente.

La colección de documentos se representa mediante una matriz llamada *Document Term Matrix* (DTM). Hay varias formas de crearla.

Modelo booleano

Es la representación más sencilla dentro del modelo de espacio vectorial. En este caso simplemente se tiene en cuenta si una palabra aparece o no el documento. Por tanto, el vector tendrá solo ceros y unos. Sea p el número de palabras en el diccionario y d el número de documentos. La colección se representará mediante una matriz $M_{p \times d}$. Si en el documento i-ésimo de la colección aparece la palabra j-ésima del diccionario tendremos M(i, j) = 1. Si no aparece, M(i, j) = 0.

Ejemplo: vamos a suponer que tenemos una colección de tres documentos que, tras preprocesarla, queda así:

- Documento 1: 'gustar patata frita patata cocida'

- Documento 2: ‘cosecha patata abundante’
- Documento 3: ‘gustar comida abundante’

El diccionario de la colección está formado siete por términos: abundante, cocida, comida, cosecha, frita, gustar, patata.

Se formará una matriz de seis filas por tres columnas para representar la colección, $M_{7\times3}$.

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{pmatrix}$$

Uso de frecuencias

La primera mejora que se puede introducir consiste en tener en cuenta el número de apariciones de cada término en el documento. De esta forma habrá más información sobre los documentos que en modelo booleano.

Ejemplo: Siguiendo con el caso anterior, el primer documento tiene dos ocurrencias de la palabra ‘patata’. Es la única palabra que aparece más de una vez en un mismo documento. La matriz quedaría así:

$$M = \begin{pmatrix} 0 & 1 & 1 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 0 & 1 \\ 2 & 1 & 0 \end{pmatrix}$$

Ponderación de términos

Para mejorar la calidad de la representación de los documentos se pueden utilizar estrategias que tienen en cuenta que asignan un peso a los términos mediante una fórmula que tiene tres factores:

- Factor local. Este factor tiene en cuenta el número de apariciones del término en un documento. Cuanto más aparezca un término dentro de un documento, más peso se le dará.

- Factor global. Este factor tiene en cuenta la frecuencia del término en toda la colección. La idea es que cuanto más frecuente sea una palabra dentro de la colección menos relevante es que aparezca en un documento.
- Factor de normalización. Tiene en cuenta la longitud de cada documento. Si un documento es muy largo es normal aparezcan más palabras y que se repitan en más ocasiones.

Se pueden elegir varias fórmulas para cada uno de los términos y para combinarlos. Una elección habitual es la llamada frecuencia de término – frecuencia inversa de documentos, o por sus siglas en inglés tf-idf (*Term frequency – Inverse document frequency*), una medida numérica que se obtiene multiplicando dos factores: el de la frecuencia del término (*term frequency*), que incluye el factor local y el de normalización, y la frecuencia inversa del documento (*Inverse document frequency*), que es el factor global. Estos términos se pueden definir de varias formas.

La siguiente tabla muestra las opciones más habituales para ambos factores. La columna de la izquierda resume las posibilidades para el factor de frecuencia del término. Las dos primeras opciones son las que se comentaron en los apartados anteriores. f(t, d) representa el número de apariciones del término t en el documento d. Las expresiones de las dos últimas filas incluyen un factor de normalización prevenir un sesgo hacia los documentos más largos. El número de ocurrencias del término t en el documento se divide por el número máximo de ocurrencias del término más frecuente en ese documento.

Opciones habituales para tf	**Opciones habituales para idf**
0, 1	1
f(t,d)	$log\frac{N}{n_t}$
log(1+f(t, d))	$\log\left(1+\frac{N}{n_t}\right)$
$0.5+0.5\frac{f(t,d)}{\max(f(t,d):t\in d)}$	$\log\left(1+\frac{max_t n_t}{n_t}\right)$
$K+(1-k)\frac{f(t,d)}{\max(f(t,d):t\in d)}$	$\log\left(\frac{N-n_t}{n_t}\right)$

Tabla 2. 2. Opciones para los factores de frecuencia de término y frecuencia inversa.

En la columna de la derecha se recogen las opciones para el término de frecuencia inversa. Reducirá la relevancia de un término dentro del documento en función de lo frecuente

que sea en toda la colección. La primera opción equivale a no tener en cuenta la frecuencia inversa. N es el número de documentos en la colección, n_t es el número de documentos en los que aparece el término t. $max_t n_t$ es el número de documentos en que aparece el término que aparece en más documentos.

Cuando se han escogido las opciones para el factor de frecuencia del término y el de frecuencia inversa, ya se tiene:

$$tf\_idf(t,d,D) = tf(t,d) \times idf(t,D)$$

donde D representa la colección documentos.

Ejemplo: aplicando esta transformación (según la implementación de scikit [SCIKIT]) al ejemplo anterior, se obtiene

$$M = \begin{pmatrix} 0 & 0.52 & 0.52 \\ 0.45 & 0 & 0 \\ 0 & 0 & 0.69 \\ 0 & 0.69 & 0 \\ 0.45 & 0 & 0 \\ 0.34 & 0 & 0.52 \\ 0.69 & 0.52 & 0 \end{pmatrix}$$

Podemos fijarnos en la primera columna, que representa el primer documento. Las filas segunda, quinta y sexta corresponden a las palabras ‘cocida’, ‘frita’ y ‘gustar’ respectivamente, que aparecen una vez cada una. En el caso de las dos primeras palabras, que solo aparecen en el primer documento el coeficiente es 0.45. Para la tercera, que aparece en dos documentos se reduce a 0.34. Por ser más habitual que las otras en el resto de documentos, se considera menos relevante.

3. Técnicas de clasificación no supervisada

En este capítulo se tratan los algoritmos y técnicas más habituales en el proceso de agrupamiento. Son solo aplicables a todo tipo de datos, no solo texto. Como hemos visto en el capítulo anterior, los documentos de texto se pueden representan como un vector de números. Lo mismo podría hacerse con una imagen, que puede representarse con un vector de números, uno por píxel. Son estos vectores que representan los elementos los que se agrupan.

3.1. Introducción

Supongamos que se desea repartir en varios grupos a una serie de países:

España, Luxemburgo, Bélgica, Sudáfrica, Túnez, China, India, Rusia, Argentina, Estados Unidos y Singapur.

Si le pedimos a una persona que las agrupe sin darle ninguna indicación, una opción sería agruparlos según el continente en que estén. En este caso, se obtendrían cuatro grupos:

España, Luxemburgo, Bélgica	Sudáfrica, Túnez	China, India, Rusia, Singapur	Estados Unidos, Argentina

Algunos pondrían a Rusia en Europa, o tal vez en Europa y Asia a la vez.

Pero esta no es la única posibilidad. Otra persona podrá repartir los según la población que tienen y, por ejemplo diferenciar entre países con una población enorme, países con población grande, media y baja. El resultado podría ser.

China, India	Estados Unidos, Rusia	España, Sudáfrica, Argentina	Bélgica, Singapur, Luxemburgo, Túnez

Como en el caso anterior, las fronteras entre los grupos no están del todo claras y puede que algún país se encuentre a medio camino entre dos de ellos.

Una tercera opción sería agruparlos según su renta per cápita: alta, media y baja. Considerando alta una renta per cápita mayor de 30,000 dólares, baja una menor de 10,000 y media en otro caso, el resultado sería:

Estados Unidos, España, Bélgica, Luxemburgo, Singapur	Rusia, Argentina, China, Túnez, Sudáfrica	India

Donde de nuevo se pueden realizar las mismas matizaciones que antes.

Las tres opciones son sensatas y razonables. Este ejemplo pone de manifiesto que el número de grupos y la pertenencia de un elemento a uno u otro depende del criterio utilizado. Al tratarse de una clasificación no supervisada, no hay un conjunto de elementos ya clasificados que sirvan de guía al proceso. También muestra que un elemento no pertenece necesariamente a un solo grupo.

Los elementos se representan mediante una serie de características. En el ejemplo anterior las características de cada país serían, al menos: continente, población y renta per cápita. En el caso general, si hay l características cada elemento está representado por un vector l-dimensional.

Para llevar a cabo el proceso de agrupamiento hay que realizar una serie de pasos y definir varios elementos.

- Selección de características. De entre todas las características disponibles, hay que elegir las características que se usaran en el proceso pensando en las que proporcionarán más información para la aplicación de interés. En general es necesario preprocesar las características para normalizar los datos, eliminar casos extremos y tratar los elementos incompletos.
- Tendencia al agrupamiento. Antes de empezar a intentar agrupar los datos conviene investigar si realmente los datos tienen una estructura subyacente.
- Medida de proximidad. Esta medida cuantifica como de parecidos o diferentes son dos elementos a partir de sus vectores de características.
- Criterio de agrupamiento. El criterio depende qué se considere como un agrupamiento razonable. Por ejemplo, un grupo compacto en el espacio l-dimensional se puede considerar razonable según cierto criterio, pero según otro podría serlo un grupo alargado. El criterio de agrupamiento se puede expresar mediante una función de coste o mediante otro tipo de reglas.
- Algoritmo de agrupamiento. Se han definido muchos algoritmos para intentar encontrar la estructura subyacente de los datos. Según los detalles de cada caso unos serán más útiles que otros.

- Validación de los resultados. Una vez realizado el agrupamiento, hay que evaluar la calidad de los resultados obtenidos.
- Interpretación de los resultados. En muchos casos será necesario que los resultados sean analizados por un experto en la materia para obtener información relevante.

A lo largo de este capítulo iremos tratando estos temas en detalle.

3.1.1. Definición de agrupamiento

Aunque grupo y agrupamiento son conceptos bastante intuitivos no es fácil definirlos con precisión.

Empecemos por una definición de grupo intuitiva y cercana a la percepción visual. Viendo un vector l-dimensional como un punto en un espacio de l-dimensiones, es posible definir un grupo como una región continua del espacio con una densidad de puntos relativamente alta, separada de otras regiones similares por regiones de baja densidad.

Ahora, veamos una definición formal de agrupamiento bastante general, pero que puede no ser válida para todos los casos.

Partiendo de un conjunto de vectores $X = \{\mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_N\}$ se define como m-agrupamiento de X la partición de X en m conjuntos (grupos), C1, …, Cm, de manera que:

- $C_i \neq \emptyset, i = 1, \ldots, m$
- $\bigcup_{i=1}^{m} C_i = X$
- $C_i \cap C_j = \emptyset, i \neq j, i, j = 1, \ldots, m$

Además los vectores de un grupo C_i son más similares a los vectores de ese mismo grupo, y menos similares a los vectores de otros grupos. Cuantificar más y menos similar depende del tipo de grupo. Para grupos compactos serán necesarias ciertas medidas, otras en el caso de grupos alargados y otras para grupos circulares o elípticos.

Esta definición de agrupamiento exige que cada vector pertenezca a un solo grupo. También es posible pensar en un agrupamiento difuso (*fuzzy*) en el que un vector pertenezca a uno o varios grupos en cierta medida.

Un agrupamiento difuso de X en m grupos está caracterizado por m funciones de pertenencia u_j donde

$$u_j: X \to [0,1], j = 1, \dots, m$$

y

$$\sum_{j=1}^{m} u_j(\boldsymbol{x}_i) = 1, i = 1, \dots, N \quad 0 < \sum_{i=1}^{N} u_j(\boldsymbol{x}_i) < N, j = 1, \dots, m$$

Aquí, la pertenencia de un vector a un conjunto (grupo) no es una cuestión de sí o no. Está caracterizada por un número en el intervalo [0, 1]. Los valores cercanos a uno indican una alto grado de pertenencia, y los cercanos a cero un grado bajo.

3.1.2. Tipos de características

No todas las características de un vector son del mismo tipo. Una primera división es que sean discretas, es decir, que solo pueden tomar un número finito de valores, o continuas, que pueden tomar cualquier valor dentro de un subcobjunto de R.

Otra división se refiere a su nivel de medida, que es el significado relativo de los valores que puede tomar la variable. Hay cuatro tipos de características: nominales, ordinales, de intervalo o de razón.

Una característica nominal describe una variable que solo puede tomar algunos valores. Los valores designan categorías, pero no se puede hablar de orden entre dos de estos valores. Por ejemplo, si hablamos del sexo de una persona las posibilidades son 'Hombre' y 'Mujer'. Es posible codificarlas asignando un número a cada valor posible, pero en cualquier caso las comparaciones entre estos números no tendrán ningún significado. No tiene sentido decir que 'Mujer' va antes que 'Hombre' o viceversa. Tampoco se puede sumar, restar o en general realizar operaciones matemáticas con este tipo de características (aunque sí se puede hablar de moda).

En una característica ordinal, los valores que puede tomar indican un orden. Por ejemplo, la calificación en una asignatura puede ser 'Suspenso', 'Aprobado', 'Notable', 'Sobresaliente'. Está claro que son valores ordenados peor a mejor nota, pero operaciones como suma o diferencia no tienen sentido.

Si para una característica la diferencia entre dos valores tiene sentido pero su razón no, se habla de una característica de intervalo. Un ejemplo podría ser la temperatura en grados centígrados. Tiene sentido decir 'hoy la temperatura máxima será cinco grados centígrados mayor que la de ayer', pero no 'hoy hace el doble de calor que ayer'.

La cuarta categoría es la para las características de razón. En estas si tiene sentido hablar de la razón entre dos medidas. Por ejemplo, se puede decir que una persona pesa el doble que otra.

3.1.3. Aplicaciones del agrupamiento

El agrupamiento se realiza con el objetivo de obtener información sobre el conjunto de datos mediante el análisis de los grupos resultantes. Estas son las aplicaciones más destacadas:

- Reducción de datos. En muchas ocasiones la cantidad de datos es demasiado grande para poder manejarlos de manera eficiente. Si se agrupan N vectores en m grupos se puede procesar cada grupo como un único elemento. De esta manera, todos los vectores de un solo grupo pueden pasar a estar representados por un único representante del mismo.
- Generación de hipótesis. El agrupamiento puede ayudar a entender la naturaleza y estructura de los datos. A partir de los grupos obtenidos se pueden idear hipótesis sobre los datos, que habrá que verificar por otros medios.
- Verificación de hipótesis. También puede ser útil para comprobar si una hipótesis es cierta. Por ejemplo, 'los jugadores de baloncesto son más altos que la media'. Se puede comprobar su veracidad a partir de un conjunto grande de personas. Supongamos que cada persona se caracteriza por su altura, su profesión y su cociente intelectual. Si tras el agrupamiento se forma un grupo de personas altas y que se dedican al baloncesto, independientemente de su cociente intelectual, se puede decir que la hipótesis está apoyada por el análisis de grupos.
- Predicción basada en grupos. En este caso, se aplica el análisis de grupos a los datos disponibles y los grupos obtenidos se caracterizan según los vectores por los que se han formado. En el futuro, cuando llegue un nuevo vector, se buscará el grupo al que sea más probable que pertenezca.

3.2. Medidas de proximidad

Ya se ha comentado que para agrupar vectores se debe calcular como de parecidos o diferentes son. Además, también será necesario medir la similitud o disimilitud entre grupos de vectores y entre un vector y un grupo. Empezamos con algunas definiciones.

Sea X el conjunto de vectores que queremos agrupar. Una medida de disimilitud d sobre X es una función que asocia un número real a cada par de vectores de X

$$d: X \times X \rightarrow R$$

tal que

$$\exists\, d_0 \in R:\ -\infty < d_0 \leq d(\boldsymbol{x},\boldsymbol{y}) < +\infty, \forall\, \boldsymbol{x},\boldsymbol{y} \in X \quad (3.1)$$

$$d(\boldsymbol{x},\boldsymbol{x}) = d_0, \forall\, \boldsymbol{x} \in X \quad (3.2)$$

y

$$d(\boldsymbol{x},\boldsymbol{y}) = d(\boldsymbol{y},\boldsymbol{x})\ \forall\, \boldsymbol{x},\boldsymbol{y} \in X \quad (3.3)$$

Si además cumple

$$d(\boldsymbol{x},\boldsymbol{y}) = d_0\ si\ y\ solo\ si\ \boldsymbol{x} = \boldsymbol{y} \quad (3.4)$$

y

$$d(\boldsymbol{x},\boldsymbol{z}) \leq d(\boldsymbol{x},\boldsymbol{y}) + d(\boldsymbol{y},\boldsymbol{z})\forall\, \boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in X \quad (3.5)$$

se dice que es una métrica de disimilitud. La ecuación (3.2) dice que hay un valor mínimo posible para la disimilitud, d_0, que se alcanza cuando los dos vectores son idénticos. Es habitual llamar distancia a esta métrica, aunque el término no sea matemáticamente preciso.

De la misma manera, una medida de similitud s sobre X se define como

$$s: X \times X \rightarrow R$$

tal que

$$\exists\, s_0 \in R:\ -\infty < s(\boldsymbol{x},\boldsymbol{y}) \leq s_0 < +\infty, \forall\, \boldsymbol{x},\boldsymbol{y} \in X \quad (3.6)$$

$$s(\boldsymbol{x},\boldsymbol{x}) = s_0, \forall\, \boldsymbol{x} \in X \quad (3.7)$$

y

$$s(\boldsymbol{x},\boldsymbol{y}) = s(\boldsymbol{y},\boldsymbol{x})\ \forall\, \boldsymbol{x},\boldsymbol{y} \in X \quad (3.8)$$

Si además cumple

$$s(\boldsymbol{x},\boldsymbol{y}) = s_0\ si\ y\ solo\ si\ \boldsymbol{x} = \boldsymbol{y} \quad (3.9)$$

y

$$s(\boldsymbol{x},\boldsymbol{y})s(\boldsymbol{y},\boldsymbol{z}) \leq [s(\boldsymbol{x},\boldsymbol{y}) + s(\boldsymbol{y},\boldsymbol{z})]s(\boldsymbol{x},\boldsymbol{z})\ \forall\, \boldsymbol{x},\boldsymbol{y},\boldsymbol{z} \in X \quad (3.10)$$

se dice que se es una métrica de similitud. Para estas medidas, s_0 es un valor máximo.

Por ejemplo, la distancia euclídea cumple las condiciones para ser una métrica de disimilitud.

En el siguiente apartado utilizaremos b_{max}, el valor máximo que toma una medida para un conjunto X.

3.2.1. Medidas de proximidad entre vectores

Vectores de números reales

Medidas de disimilitud

Entre las más habituales se encuentran las métricas de disimilitud ponderadas L_p,

$$d_p(\boldsymbol{x}, \boldsymbol{y}) = \left(\sum_{i=1}^{l} w_i |x_i - y_i|^p \right)^{1/p} \quad (3.11)$$

Donde x_i, y_i, son las i-ésimas coordenadas de x e y, i = 1,.., l y w_i es el coeficiente de peso. Si $w_i = 1$ para todo i, se obtienen la llamada distancia de Minkowski de orden p. Si además p = 2, se trata de la distancia euclídea.

Algunas métricas L_p habituales son la L1, o distancia Manhattan,

$$d_1(\boldsymbol{x}, \boldsymbol{y}) = \sum_{i=1}^{l} w_i |x_i - y_i| \quad (3.12)$$

y la L_∞, o distancia de Chebyshev

$$d_\infty(\boldsymbol{x}, \boldsymbol{y}) = max_{1 \le i \le l} w_i |x_i - y_i| \quad (3.13)$$

La métrica L_2 se puede generalizar de la siguiente manera:

$$d(\boldsymbol{x}, \boldsymbol{y}) = \sqrt{(\boldsymbol{x} - \boldsymbol{y})^T B (\boldsymbol{x} - \boldsymbol{y})} \quad (3.14)$$

donde B es una matriz simétrica definida positiva.

Medidas de similitud

- Producto escalar. Definido como

$$s_{\text{escalar}}(\mathbf{x}, \mathbf{y}) = \boldsymbol{x}^T \boldsymbol{y} = \sum_{i=1}^{l} x_i y_i \quad (3.15)$$

  donde $\mathbf{x}^T$ representa el vector traspuesto de **x**.

Normalmente esta medida se utiliza con vectores de igual longitud, pero se pueden normalizar como paso previo al cálculo. Si los dos vectores tienen longitud a, esta medida estará acotada entre $+a^2$ y $-a^2$.

A partir de esta se puede obtener una medida de disimilitud como

$d_{escalar}(\mathbf{x}, \mathbf{y}) = b_{max} - s_{escalar}(\mathbf{x}, \mathbf{y})$

Una medida relacionada es

$$s_{\text{coseno}}(\mathbf{x}, \mathbf{y}) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\mathbf{x}_d\| \|\mathbf{y}_d\|}$$

donde $\|\mathbf{x}_d\|$ y $\|\mathbf{y}_d\|$ son las longitudes de los vectores **x** e **y**. En este caso, como el coseno, estará acotada entre +1 y -1,

- Coeficiente de correlación de Pearson.

$$r_{pearson}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x}_d^T \mathbf{y}_d}{\|\mathbf{x}_d\| \|\mathbf{y}_d\|} \quad (3.16)$$

donde $\boldsymbol{x}_d = [x_1 - \bar{x}, \dots, x_l - \bar{x}\,]$, $\boldsymbol{y}_d = [y_1 - \bar{y}, \dots, y_l - \bar{y}\,]$, x_i, y_i, son las i-ésimas coordenadas de **x** e **y**, i = 1, .., l, $\bar{x} = \frac{1}{l}\sum_{i=1}^{l} x_i$ y $\bar{y} = \frac{1}{l}\sum_{i=1}^{l} y_i$. x_d e y_d se conocen como vectores diferencia.

La diferencia del coeficiente de Pearson con el producto escalar es que se calcula usando los vectores diferencia en lugar de los originales. Está acotado entre +1 y -1. Una medida de disimilitud relacionada es

$$D(\mathbf{x}, \mathbf{y}) = \frac{1 - r_{pearson}(\mathbf{x}, \mathbf{y})}{2} \quad (3.17)$$

que toma valores entre 0 y 1.

- Otra medida de similitud habitual es la distancia de Tanimoto. Se define como

$$s_T(\mathbf{x}, \mathbf{y}) = \frac{\boldsymbol{x}^T \boldsymbol{y}}{\|\boldsymbol{x}\|^2 + \|\boldsymbol{y}\|^2 - \boldsymbol{x}^T \boldsymbol{y}} = \frac{1}{1 + \frac{(\boldsymbol{x} - \boldsymbol{y})^T (\boldsymbol{x} - \boldsymbol{y})}{\boldsymbol{x}^T \boldsymbol{y}}} \quad (3.18)$$

Como el producto escalar se puede considerar una medida de la correlación entre los dos vectores, la distancia de Tanimoto es inversamente proporcional al cuadrado de la distancia euclídea entre los dos vectores, dividida por su correlación.

Si los dos vectores están normalizados para tener la misma longitud a, se obtiene

$$s_T(\mathbf{x}, \mathbf{y}) = \frac{1}{-1 + 2\frac{a^2}{\boldsymbol{x}^T \boldsymbol{y}}}$$

En este caso s_T es inversamente proporcional a $\frac{a^2}{\boldsymbol{x}^T\boldsymbol{y}}$, de manera que cuanto más correlacionados están los vectores, mayor es s_T.

Vectores con valores discretos

No es rato tratar con vectores cuyas componentes solo pueden tomar un número finito de valores. Aunque es posible utilizar las medidas del apartado anterior, también hay algunas distancias específicas. Muchas de ellas se pueden expresar a partir de la tabla de contingencia de los dos vectores, que se define a continuación.

Si tenemos vectores de l coordenadas que solo pueden tomar k valores, habrá k^l vectores diferentes. A partir de dos de estos vectores, **x** e **y**, se puede definir la matriz k × k

$$A(\mathbf{x},\mathbf{y}) = [a_{ij}],\ i, j = 0, 1, \ldots, k\text{-}1 \quad (3.19)$$

en la que el elemento a_{ij} representa el número de posiciones en las que el primer vector tiene el valor i y la posición correspondiente del segundo vector tiene el valor j. Esta matriz se conoce como tabla de contingencia.

Por ejemplo, si las coordenadas solo pueden tomar tres valores, [0, 1, 2], la tabla de contingencia será una matriz 3 × 3. Para los vectores **x** = [0, 1, 2, 1, 2, 1], **y** = [1, 0, 2, 1, 0, 1] la matriz A(**x**, **y**) será

$$A(\boldsymbol{x},\boldsymbol{y}) = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 2 & 0 \\ 1 & 0 & 1 \end{bmatrix}$$

Se puede comprobar que $\sum_{i=0}^{k-1}\sum_{j=0}^{k-1} a_{ij} = l$.

Medidas de disimilitud

- La distancia de Hamming es el número de posiciones en las que dos vectores difieren. A partir de la matriz A, se puede definir como la suma de los elementos fuera de la diagonal.

$$d_H(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=0}^{k-1}\sum_{j=0,j\neq i}^{k-1} a_{ij} \quad (3.20)$$

Cuando los vectores son binarios, es decir, k = 2 y los valores posibles son [0, 1], la distancia de Hamming es

$$d_H(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{l}(x_i - y_i)^2 \quad (3.21)$$

A partir de esta medida, se puede obtener una de similitud como

$$s_H(\boldsymbol{x},\boldsymbol{y}) = b_{max} - d_H(\boldsymbol{x},\boldsymbol{y})$$

- La distancia L_1. Definida como en el caso de los valores reales.

$$d_1(\boldsymbol{x},\boldsymbol{y}) = \sum_{i=1}^{l} w_i|x_i - y_i| \quad (3.22)$$

Coincide con la de Hamming para vectores binarios.

Medidas de similitud

Una medida de similitud muy utilizada para vectores con valores discretos es la de Tanimoto. Se basa en la comparación de conjuntos. Si X e Y son dos conjuntos y n_X, n_Y y $n_{X\cap Y}$ son las cardinalidades, de X, Y y la intersección de ambos, respectivamente, la medida de Tanimoto entre los dos conjuntos es

$$\frac{n_{X\cap Y}}{n_X + n_Y - n_{X\cap Y}} = \frac{n_{X\cap Y}}{n_{X\cup Y}}$$

Es el ratio entre el número de elementos comunes y el total.

En el caso de dos vectores, **x** e **y**, se definen $n_{\boldsymbol{x}} = \sum_{i=1}^{k-1}\sum_{j=0}^{k-1} a_{ij}$ y $n_{\boldsymbol{y}} = \sum_{i=0}^{k-1}\sum_{j=1}^{k-1} a_{ij}$. $n_{\boldsymbol{x}}$ y $n_{\boldsymbol{y}}$ son el número de coordenadas distintas de 0 de **x** e **y**. La medida de Tanimoto se define como

$$s_T(\boldsymbol{x},\boldsymbol{y}) = \frac{\sum_{i=1}^{k-1} a_{ii}}{n_{\boldsymbol{x}} + n_{\boldsymbol{y}} - \sum_{i=1}^{k-1}\sum_{j=1}^{k-1} a_{ij}} \quad (3.23)$$

Otras medidas de similitud tienen en cuenta solo as posiciones en la que los dos vectores coinciden, por ejemplo

$$\frac{\sum_{i=1}^{k-1} a_{ii}}{l}, \frac{\sum_{i=1}^{k-1} a_{ii}}{l} \; y \; \frac{\sum_{i=0}^{k-1} a_{ii}}{l} \quad (3.24)$$

Otros tipos de medidas

Medidas dinámicas

Las medidas que hemos visto hasta ahora se aplican a vectores de la misma dimensión. En algunas aplicaciones, este no es el caso y las medidas anteriores no son aplicables. Hay que utilizar medidas dinámicas, como las distancias de edición (*edit distance*).

Una distancia de edición es una medida de disimilitud entre cadenas de texto, basada en contar el número mínimo de cambios necesarios para transformar una cadena en otra. Hay varias definiciones para la distancia de edición, con diferentes operaciones de transformación disponibles.

Se aplican, por ejemplo, para la corrección de errores ortográficos o tipográficos. Si se encuentra una palabra que no está en el diccionario de la aplicación, para sustituirla se buscan las más cercanas a la palabra en cuestión.

Vectores con valores mezclados

Es habitual que un vector de características tenga componentes con valores reales y también algunas con valores discretos.

Una opción es usar medidas para vectores con componentes reales, ya que las componentes discretas se pueden tratar como reales sin problemas.

Una segunda opción es discretizar los valores reales y usar medidas para vectores con componentes discretos.

También hay medidas que pueden tratar vectores con componentes de ambos tipos sin hacer cambios, como el coeficiente de similitud de Gower:

$$s(\boldsymbol{x}, \boldsymbol{y}) = \frac{\sum_{q=1}^{l} s_q(\boldsymbol{x}, \boldsymbol{y})}{\sum_{q=1}^{l} w_q} \quad (3.25)$$

en la que $s_q(\boldsymbol{x}, \boldsymbol{y})$ la similitud entre las coordenadas q-ésimas de ambos vectores y w_q es un factor de peso para la coordenada q-ésima. La forma que toma $s_q(\boldsymbol{x}, \boldsymbol{y})$ y el valor de w_q dependen del tipo y el valor de cada coordenada.

Datos incompletos

En muchos casos se encuentran algunos vectores incompletos, de los que no se conocen todas sus coordenadas. Hay varias opciones:

- Descartar los vectores incompletos. Esta opción es viable si los vectores incompletos son una pequeña parte del total.
- Sustituir los valores que faltan con la media que tiene esa componente en los vectores en los que sí está disponible.
- También se pueden seguir estrategias más sofisticadas que permiten medir la proximidad usando solo las coordenadas disponibles en ambos vectores, como la

que sigue a continuación. Para empezar, para todos los pares de componentes x_i e y_i de los vectores x e y, se define b_i como

$$b_i = \begin{matrix} 0 \; si \; x_i \; e \; y_i \; están \; disponibles \\ 1 \; en \; otro \; caso \end{matrix} \quad (3.26)$$

Después, se define la proximidad entre x e y como

$$\rho(\boldsymbol{x}, \boldsymbol{y}) = \frac{l}{l - \sum_{i=1}^{l} b_i} \sum_{\forall i: b_i = 0} \varphi(x_i, y_i) \quad (3.27)$$

donde $\varphi(x_i, y_i)$ es la proximidad entre los dos escalares x_i e y_i. Si se trata de una medida de disimilitud, es habitual escoger $\varphi(x_i, y_i) = |x_i - y_i|$. Analizando la expresión 3.27, si todas la componentes están disponibles en ambos vectores el resultado es la suma de las proximidades entre componentes. Si, por ejemplo, en dos vectores de 10 componentes hay tres que no están disponibles en alguno de los dos, se suma la proximidad entre las otras siete y el resultado se multiplica por 10/7. Es decir, de esta manera se supone que la proximidad entre las componentes que faltan es similar a las presentes, y la proximidad total se calcula teniendo en cuenta la proporción de componentes ausentes.

3.2.2. Medidas de proximidad entre un punto y un conjunto

En algunos algoritmos agrupamiento hay que calcular la proximidad entre un vector y un grupo de vectores ya formado, para ver si el vector se asigna a ese grupo. Es decir, la proximidad entre un punto **x** (de un espacio de l dimensiones), y un conjunto de puntos, C. Nos referimos a esta proximidad como $\rho^{\mathrm{pC}}(\boldsymbol{x}, C)$.

Algunas opciones sencillas son:

- Calcular la proximidad de x con todos los puntos de C y escoger el mayor valor:

$$\rho_{max}^{\mathrm{pC}}(\boldsymbol{x}, C) = max_{y \in C} \; \rho(\boldsymbol{x}, \boldsymbol{y}) \quad (3.28)$$

- Calcular la proximidad de x con todos los puntos de C y escoger el menor valor:

$$\rho_{min}^{\mathrm{pC}}(\boldsymbol{x}, C) = min_{y \in C} \; \rho(\boldsymbol{x}, \boldsymbol{y}) \quad (3.29)$$

- Calcular la proximidad de x con todos los puntos de C y hallar el valor medio:

$$\rho_{avg}^{\mathrm{pC}}(\boldsymbol{x}, C) = \frac{1}{n_c} \sum_{y \in C} \rho(\boldsymbol{x}, \boldsymbol{y}) \quad (3.30)$$

donde n_c es la cardinalidad del conjunto C.

En estas definiciones $\rho(\boldsymbol{x}, \boldsymbol{y})$ puede ser cualquier medida de proximidad.

Otro enfoque consiste en encontrar un representante del conjunto C y calcular su proximidad con **x**. Es habitual escoger como representante de un conjunto un punto, un hiperplano o una hiperesfera, según la forma del grupo. Los puntos representantes son apropiados para grupos compactos, los hiperplanos para grupos de forma lineal y las hiperesfereas para grupos de forma hiperesférica.

Ilustración 3. 1. De izquierda a derecha, grupo compacto, grupo lineal y grupo esférico con sus representantes. Tomado de [Theo].

Punto representante

Algunas opciones típicas son:

- Punto (o vector) medio:

$$\boldsymbol{m}_p = \frac{1}{n_c} \sum_{\boldsymbol{y} \in C} \boldsymbol{y} \quad (3.31)$$

  donde n_c es la cardinalidad de C. Es la opción más habitual cuando se trata de valores reales. No es tan buena opción cuando se trata de valores discretos, ya que es posible que $\mathbf{m}_p$ no tenga uno de esos valores discretos. Para no tener este problemas, se puede usar el centro medio, $\mathbf{m}_c$.

- El centro medio $\mathbf{m}_c \in C$ es el punto para el que se cumple

$$\sum_{\boldsymbol{y} \in C} d(\boldsymbol{m}_C, \boldsymbol{y}) \leq \sum_{\boldsymbol{y} \in C} d(\boldsymbol{z}, \boldsymbol{y}), \forall\, \boldsymbol{z} \in C \quad (3.32)$$

  donde d es una medida de disimilitud entre dos puntos. Cuando se trata de medidas de similitud, se invierte la desigualdad.

- El centro mediano $\mathbf{m}_{med} \in C$ es el punto para el que se cumple

$$med(d(\boldsymbol{m}_{med}, \boldsymbol{y})|\boldsymbol{y} \in \boldsymbol{C}) \leq med(d(\boldsymbol{z}, \boldsymbol{y})|\boldsymbol{y} \in C), \forall\, \boldsymbol{z} \in C \quad (3.33)$$

donde d es una medida de disimilitud entre dos puntos. La mediana de un conjunto de q escalares es el número más pequeño del conjunto que mayor o igual a [(q+1)/2] números del conjunto.

Hiperplano representante

En los grupos alargados, en los que los puntos se agrupan a lo largo de una línea, un punto no es buen representante, es mejor usar una recta. En el caso más general, con vectores de k >2 dimensiones, se trataría de más de grupos que se distribuyen alrededor de un hiperplano, que es lo que se usaría como representante (ver ilustración 3.1).

Una vez escogido un hiperplano H representante del conjunto C, la distancia de **x** a C será la distancia mínima de **x** al a uno de los puntos H:

$$d(\boldsymbol{x}, H) = min_{\boldsymbol{z} \in H} d(\boldsymbol{x}, \boldsymbol{z}) \quad (3.34)$$

Hiperesfera representante

Otro tipo de grupos son los que se agrupan en torno a una esfera, o una hiperesfera para dimensiones mayores. Como en el caso anterior, como distancia de **x** a una hiperesfera Q es la distancia más pequeña entre **x** y cualquier punto de Q.

3.2.3. Funciones de proximidad entre dos conjuntos

En algunos algoritmos de agrupamiento también es necesario calcular la distancia entre dos conjuntos. Si D_i y D_j son don conjuntos de vectores, las funciones de proximidad más comunes son:

- Función de proximidad máxima:

$$\rho_{max}^{CC}(D_i, D_j) = max_{\boldsymbol{x} \in D_i, \boldsymbol{y} \in D_j}\ \rho(\boldsymbol{x}, \boldsymbol{y}) \quad (3.35)$$

  Si ρ es una medida de disimilitud, no es una medida. Si ρ_{max}^{CC} es una medida de similitud, sí es una medida, pero no una métrica. En el primer caso ρ_{max}^{CC} queda determinada por los dos vectores más diferentes (lejanos), en el segundo, por los más parecidos (próximos).

- Función de proximidad mínima:

$$\rho_{min}^{CC}(D_i, D_j) = min_{\boldsymbol{x} \in D_i, \boldsymbol{y} \in D_j}\ \rho(\boldsymbol{x}, \boldsymbol{y}) \quad (3.36)$$

  Si ρ es una medida de similitud, no es una medida. Si ρ_{min}^{CC} es una medida de disimilitud, sí es una medida, pero no una métrica. En el primer caso ρ_{min}^{CC} queda

determinada por los dos vectores más parecidos (próximos), en el segundo, por los más diferentes (lejanos).

- Función de proximidad media:

$$\rho^{\mathrm{CC}}_{media}(D_i, D_j) = \rho\left(\boldsymbol{m}_{D_i}, \boldsymbol{m}_{D_j}\right) \quad (3.37)$$

donde $\boldsymbol{m}_{D_i}$es el punto representante de D_i. Por ejemplo, puede ser el punto medio, el centro medio, o el centro mediano de D_i. Si ρ es una medida, $\rho^{\mathrm{CC}}_{media}$ también lo es.

- Basada en la anterior, se puede definir otra función de proximidad que tiene en cuenta las cardinalidades de ambos conjuntos:

$$\rho^{\mathrm{CC}}_{car}(D_i, D_j) = \sqrt{\frac{n_{D_i} n_{D_j}}{n_{D_i} + n_{D_j}}}\rho\left(\boldsymbol{m}_{D_i}, \boldsymbol{m}_{D_j}\right) \quad (3.38)$$

3.3. Algoritmos de agrupamiento

Los algoritmos de agrupamiento asignan a diferentes grupos los vectores de un conjunto X. Se han definido muchos algoritmos de agrupamiento, con diferentes características y casos de aplicación. En algunos de ellos, hay que especificar el número de grupos que se quiere obtener (o el número máximo) y en otros no se hace ninguna indicación al respecto. Un análisis en profundidad de todos los algoritmos sería muy extenso, así que en este apartado, tras describir las principales categorías de algoritmos, explicaremos en detalles algunos de los más extendidos.

3.3.1. Tipos de algoritmos de agrupamiento

La siguiente clasificación es orientativa, no exhaustiva y puede haber solapamiento entre las categorías. En cualquier caso, sirve como un primer acercamiento a los algoritmos que se presentarán posteriormente.

- Algoritmos basados en conectividad o jerárquicos. Con estos algoritmos se obtiene una jerarquía de grupos. Hay dos enfoques básicos:
    - Aglomerativos: estos algoritmos proceden de 'abajo hacia arriba'. Al principio cada vector forma su propio grupo, y en cada paso se va uniendo dos grupos en uno.

  - Divisivos: en este caso, se procede de 'arriba hacia abajo'. Al principio todos los vectores los vectores están en un único grupo, que se va dividiendo sucesivamente en cada paso.

Es habitual mostrar el resultado de estos algoritmos en forma de dendrograma.

- Algoritmos basados en densidad. En estos algoritmos los grupos se definen como zonas donde la densidad de vectores es mayor que en el resto del conjunto de vectores. Los grupos están separados por zonas de menor de densidad. Los vectores situados en estas zonas de menor densidad suelen considerarse ruido y no se asignan a ningún grupo.
- Algoritmos secuenciales. En cada paso, asignan un vector a un grupo. En la mayoría de los casos, cada vector pasa una vez (o unas pocas veces) por la fase de asignación. Normalmente el resultado depende del orden en que se procesen los vectores.
- Basados en la optimización de funciones. En estos algoritmos la calidad de un agrupamiento viene dada por una función de coste. Normalmente producen agrupamientos sucesivos mientras intentan optimizar la función de coste y se detienen cuando se alcanza un mínimo local. También es habitual que se fije el número de grupos. Podemos distinguir algunas subcategorías:
  - Agrupamientos duro. Cuando un vector solo puede pertenecer o no a un grupo se habla de agrupamiento duro.
  - Agrupamientos blandos. Si por el contrario se permite que un vector pertenezca a un grupo en un cierto grado, se habla de agrupamiento blando o difuso.
  - Algoritmos probabilísticos. Son un tipo de algoritmos de agrupamiento duro que siguen una enfoque bayesiano. Cada vector se asigna al grupo Ci para el que la probabilidad $P(C_i \mid x)$ es máxima. Esta probabilidad se estima mediante un problema de optimización.
- Algoritmos basados en subespacios. En muchas aplicaciones los vectores tienen una dimensión muy alta. En el casos de los documentos de texto no es raro trabajar con varios miles de dimensiones. En estos casos hay que enfrentarse a la *maldición de la dimensionalidad* con algoritmos específicos.

3.3.2. Algoritmos secuenciales

Como primer ejemplo de algoritmo de agrupamiento, vamos a empezar con un algoritmo sencillo llamado BSAS, siglas de esquema algoritmo secuencial básico (*basic sequential algorithmic scheme*).

Recibe como parámetros el número máximo de grupos y un umbral de disimilitud. También es necesario especificar una medida de disimilitud entre un punto y un grupo. La idea es ir recorriendo el conjunto de vectores y asignar cada nuevo vector a un grupo ya existente o crear uno nuevo, según la distancia entre el vector y los grupos existentes. Veamos el pseudocódigo. La variable m representa el número de grupos ya creados.

```
BSAS(umbral, maxGrupos)
m = 1
Cm ={x1} //inicializar el primer grupo
Para i = 2 a N
        Encontrar el Ck tal que d(xi, Ck) = min i ≤ j≤m d(xi, Cj)
        Si d(xi, Ck) > umbral y m < maxGrupos
                m = m+1       //crear un nuevo grupo con el vector
                Cm = { xi }
        Si no
                Ck = Ck +{ xi } //añadir el vector al grupo más cercano
                Actualizar el representante de Ck si es necesario
```

Observaciones:

- También es posible no especificar maxGrupos y dejar que el algoritmo fije el número de grupos.
- Se puede usar una medida de similitud cambiando el operador min por max.
- El orden en que se recorren los vectores afecta al resultado final.
- Es muy importante escoger bien el valor del umbral. Dependerá del conjunto de datos. Si es demasiado grande, creará menos grupos de los deseados, y si es demasiado pequeño, creará muchos grupos.
- Es apropiado para grupos compactos.

Una posible mejora, llamada MBSAS (la M es de modificado), consiste en hacer dos pasadas. Una sirve para determinar los grupos y otra para asignar los vectores:

- En la primera pasada, solo se asigna un vector a cada grupo. Cuando un vector está a una distancia menor que el umbral de un grupo ya creado no se asigna al mismo, sino que se deja para la segunda pasada.
- En la segunda, no se crean nuevos grupos ni se modifican los ya creados. Los vectores no asignados en la primera se asignan al grupo más cercano.

La ventaja es que en la segunda pasada la asignación de los vectores se produce con todos los grupos ya formados, no como en la versión básica del algoritmo. La desventaja es, obviamente, que hay que hacer dos pasadas. De cualquier manera, el orden en que se recorren los vectores sigue afectando al resultado final.

Número de grupos

Un problema que surge con cualquier algoritmo de agrupamiento que necesite el número de grupos como parámetro de entrada es que una mala elección del mismo dará lugar a un agrupamiento pobre. Es posible utilizar BSAS para determinarlo. El primer paso es encontrar las distancias mínima y máxima entre dos puntos de X, que servirán de valores mínimo y máximo para el umbral.

$$a = min_{i,j=1,\dots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j) y\ b = max_{i,j=1,\dots,N} d(\boldsymbol{x}_i, \boldsymbol{x}_j)$$

La idea es ejecutar múltiples veces BSAS para un mismo umbral sin especificar el número de grupos y ver qué número de grupos se obtiene más veces. En cada ejecución los vectores de X se ordenan de una forma diferente, para obtener resultados diferentes. Este proceso se repite para diferentes valores del umbral, comprendidos entre a y b.

El pseudocódigo sería:

```
Para umbral de a hasta b con paso c
        Ejecutar n veces BSAS(umbral), cada vez con los vectores en un orden diferente.
        Hallar el número de grupos obtenidos más repetido, que llamaremos m(umbral)
```

Tras esto, se representa el umbral contra el número de grupos obtenidos. En la gráfica se buscarán las regiones planas, es decir, intervalos en los que el número de grupos se mantiene constante. El número de grupos correspondiente al mayor intervalo en que la función esta plana se tomará como número óptimo de grupos.

En la siguiente ilustración se muestra un ejemplo. A la izquierda están los datos que se pretende agrupar, que forman dos grupos claros. A la derecha, el gráfico del umbral contra

m(umbral), obtenido según el procedimiento anterior. Se puede observar que para un gran intervalo de valores del umbral el número de grupos se mantiene constante.

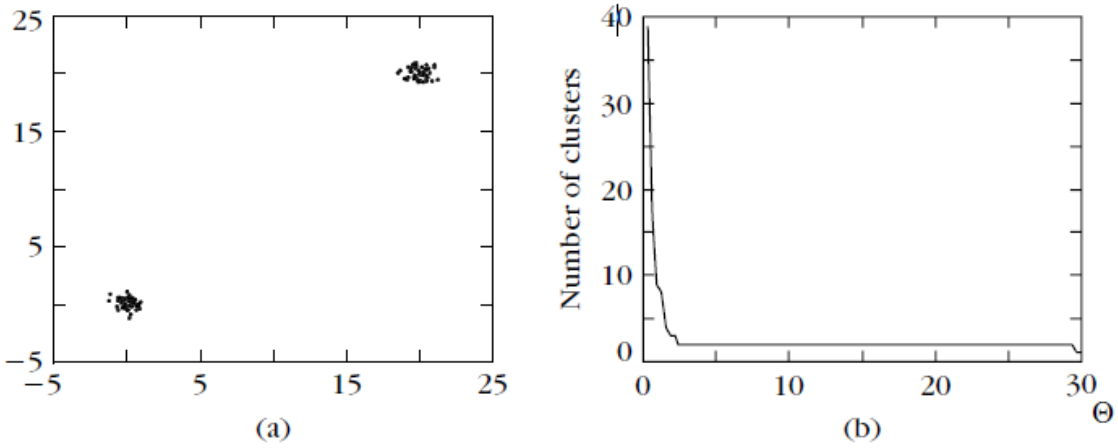

Ilustración 3. 2. Determinación del número de grupos con BSAS. Tomado de [Theo].

3.3.3. Algoritmos jerárquicos

En este tipo de algoritmos no se produce un agrupamiento, sino una jerarquía de agrupamientos anidados. Es habitual utilizarlos en la taxonomía biológica y las ciencias sociales, aunque se han utilizado también en otras muchas áreas.

Se dice que un agrupamiento, R_1 con k grupos, está anidado en otro agrupamiento R_2, con $h < k$ grupos si cada grupo en R_1 es un subconjunto de algún grupo en R_2.

Estos algoritmos requieren N pasos, tantos como puntos haya que agrupar. En cada paso se obtiene un nuevo agrupamiento basado en el agrupamiento del paso anterior. Hay dos tipos de algoritmos de agrupamiento jerárquico, aglomerativos y divisivos.

En el agrupamiento aglomerativo se empieza con N grupos, cada uno con un solo punto. En cada paso, se reduce el número de grupos en 1, de manera que el agrupamiento anterior esté anidado dentro del nuevo. Tras N pasos se obtendrá un único grupo con todos los puntos.

El agrupamiento divisivo procede de manera inversa. Empieza con un único grupo con los N elementos. En cada paso, uno de los grupos del paso anterior se divide en dos nuevos grupos. Tras N pasos se llega a tener N grupos, cada uno con un solo punto.

3.3.3.1. Clustering aglomerativo

A continuación se presenta el pseudocódigo un algoritmo genérico de clustering jerárquico aglomerativo similar al descrito en [Ward]. Se basa en una función $g(C_i, C_j)$ que mide la proximidad entre dos grupos. t representa el nivel actual de la jerarquía.

Clustering aglomerativo

t = 0

Agrupamiento inicial es R0 = {C_i = {$\mathbf{x}_i$}. i = 1, …, N}

Repetir hasta que todos los vectores estén en un solo grupo

t = t + 1

De entre las parejas (C_r, C_s) de grupos de R_{t-1}, encontrar i, j tales que

$$g(C_i, C_j) = \begin{cases} min_{r,s} g(C_r, C_s) si\ g\ es\ una\ función\ de\ similitud \\ max_{r,s} g g(C_r, C_s) si\ g\ es\ una\ función\ de\ disimilitud \end{cases}$$

Crear el grupo C_{ij} = (C_i U C_j) y el nuevo agrupamiento $R_t = R_{t-1} – \{C_i, C_j\}$ U C_{ij}

En cada paso se parten de n grupos y se obtienen n-1 de forma que se minimice cierta función. Si se desea llegar a un número concreto de grupos, se corta el proceso en el momento adecuado.

En [Ward] se trataba de minimizar la varianza dentro de cada grupo. Cuando se usa este criterio es cuando se habla de método de Ward, pero también es habitual usar el mismo procedimiento con otra función, como se hace en los casos de enlace completo (*complete linkage*) y enlace medio (*average linkage*).

Al pasar de m a m-1 grupos hay m(m-1)/2 posibilidades. Se evalúan todas ellas con el criterio escogido y se escoge la mejor. Al ir uniendo unos grupos se establece una jerarquía, ya que dos grupos se unen en uno mayor, que los contiene a ambos. El número de parejas que hay que evaluar a lo largo de todo el proceso es proporcional a N^3, pero la complejidad exacta depende de g.

Método de Ward

Vamos a ilustrar este procedimiento con un ejemplo tomado de [Ward]. Es un ejemplo sencillo con cinco elementos, cada uno con una sola característica. La función a minimizar, F, es la suma de varianzas de esa característica en cada grupo.

Sea X = [1, 7, 2, 9, 12] el conjunto de vectores unidimensionales que queremos ordenar. Al principio partimos de cinco grupos cada uno con un solo elemento y por tanto F = 0. El primer paso consiste en buscar la pareja de elementos que formen el grupo con menor

varianza. Claramente hay que unir el primer y tercer elemento, que son los que tienen las características más parecidas. Pasamos a tener 4 conjuntos: [1, 2], [7], [9], [12]. El primero de estos conjuntos tiene una varianza de 0.5, los otros tres de 0, al tener un solo elemento. Por tanto, F = 0.5.

En el segundo paso hay que decidir si añadimos alguno de los elementos originales al grupo de dos elementos o si es mejor unir dos de esos elementos en un nuevo grupo. Hay que calcular la suma de las varianzas dentro de cada grupo en todos los casos y escoger la mejor:

- [1, 2, 7], [9], [12]. La varianza del primer grupo es 1.33, en los otros 0.
- [1, 2, 9], [7], [12]. La varianza del primer grupo es 38, en los otros 0.
- [1, 2, 12], [7], [9]. La varianza del primer grupo es 74, en los otros 0.
- [1, 2], [7, 9], [12]. La varianza del segundo grupo es 2. Unida a la del primero, que se calculó antes, la suma de varianzas es 2.5.
- [1, 2], [7, 12], [9]. La varianza del segundo grupo es 12.5. Unida a la del primero, que se calculó antes, la suma de varianzas es 13.
- [1, 2], [9, 12], [7]. La varianza del segundo grupo es 4.5. Unida a la del primero, que se calculó antes, la suma de varianzas es 5.

A la vista de estos resultados, hay que escoger la cuarta opción, los nuevos grupos serán [1, 2], [7, 9] y [12].

En el siguiente paso, habrá que escoger entre unir los dos primeros grupos o unir el elemento suelto a uno de ellos. Es decir, hay tres posibilidades.

- [1, 2, 7, 9], [12]. La suma de varianzas es 44.75.
- [1, 2, 12], [7, 9]. La suma de varianzas es 76.
- [1, 2], [7, 9, 12]. La suma de varianzas es 13.1667.

Por lo tanto, se escoge la tercera opción. En el siguiente paso, solo hay dos grupos para unir así que no hay decisiones que tomar.

Una representación gráfica del proceso podría ser:

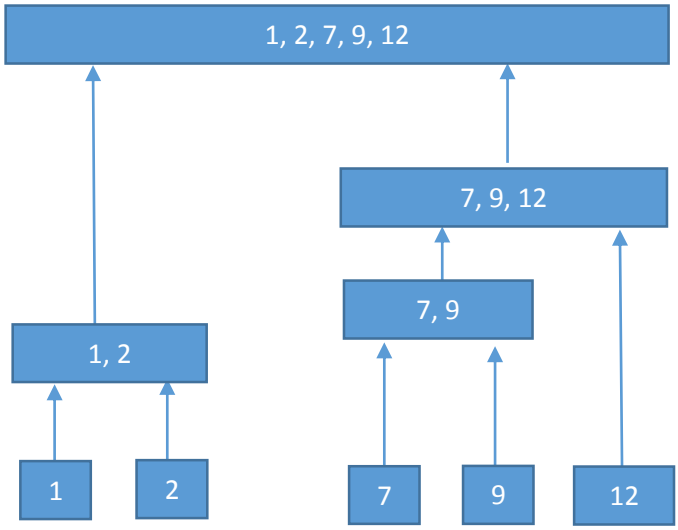

Ilustración 3. 3. Ejemplo de agrupamiento jerárquico aglomerativo.

Enlace completo

Se trata de minimizar la distancia máxima entre dos elementos del grupo. Es posible utilizar diferentes definiciones de distancia, no como en el caso anterior.

Vamos a realizar el agrupamiento de los datos del ejemplo anterior, X = [1, 7, 2, 9, 12], usando este método. Utilizaremos también la distancia euclídea.

Empezando con cinco grupos, hay diez formas posibles de pasar a cuatro, pero como se trata de minimizar la distancia máxima entre dos elementos del grupo está claro que la mejor opción es unir los dos más parecidos 1 y 2, en un solo grupo. Por tanto, tendremos [1, 2], [7], [9], [12].

Para el segundo paso las opciones son:

- [1, 2, 7], [9], [12]. Solo podemos hablar de distancia entre miembros en el primer grupo. La máxima es 6.
- [1, 2, 9], [7], [12]. La distancia máxima es 8.
- [1, 2, 12], [7], [9]. La distancia máxima es 11.
- [1, 2], [7, 9], [12]. La distancia máxima es 2, en el segundo grupo.
- [1, 2], [7], [9, 12]. La distancia máxima es 3, en el tercer grupo.
- [1, 2], [7, 12], [9]. La distancia máxima es 5, en el segundo grupo.

Por tanto, la mejor opción es la cuarta: [1, 2], [7, 9], [12]. Para pasar a dos grupos las opciones son:

- [1, 2, 7, 9], [12]. Distancia máxima 8.
- [1, 2], [7, 9, 12]. Distancia máxima 5.
- [1, 2, 12], [7, 9]. Distancia máxima 11.

La mejor opción es la segunda: [1, 2], [7, 9, 12]. Solo falta el último paso, en que se tendrá un solo grupo con todos los elementos. El resultado ha sido el mismo que en el caso anterior, así que no hace falta repetir el diagrama.

Enlace único y enlace medio

Otros criterios habituales son el enlace único y el enlace medio. El primero minimiza la distancia mínima entre dos elementos del grupo y el segundo la distancia media.

3.3.2.2. Algoritmos divisivos

En los algoritmos divisivos se sigue el procedimiento inverso. Al principio todos los elementos forman un solo grupo. En cada paso, uno de los grupos se parte en dos, también de manera que se minimice cierta función (o se maximice, según el tipo de función). Se repite el procedimiento hasta que se tengan N grupos de un único elemento.

El problema de este tipo de algoritmos es su elevado coste computacional. Tienen que evaluar muchas más posibilidades que en caso aglomerativo. Partiendo de N elementos en el primer paso hay $2^{N-1} - 1$ posibilidades. Para que sea viable utilizar algoritmos divisivos hay que utilizar algún tipo de criterio que permita descartar muchas de las posibilidades sin evaluarlas.

3.3.4. Algoritmos basados en la optimización de funciones

3.3.4.1. k-means

Este método es uno de los más estudiados y utilizados. Divide N elementos en k grupos. Cada elemento se asigna al grupo con la media más cercana. La media de los elementos de un grupo es el representante del mismo, y se denomina centroide. El número de grupos en que se quieren dividir los elementos es un parámetro de entrada del algoritmo.

Aunque el problema es NP-duro, hay heurísticas eficientes, como el algoritmo de Lloyd o de Lloyd-Forgy, que es el que se usa habitualmente. El algoritmo es el siguiente:

1. Escoger k centroides como punto de partida para los representantes de los k grupos.
2. Asignar cada elemento a uno de los grupos: aquel cuyo representante es más cercano.
3. Cuando todos los elementos están asignados a un grupo, se calcula el nuevo representante del grupo como la media de todos sus elementos (centroide).
4. Los pasos 2 y 3 se repiten hasta que los centroides no cambian.

El algoritmo siempre llega a un punto final, pero no se puede garantizar que llegue a un mínimo global. Por este motivo es habitual ejecutarlo varias veces con diferentes inicializaciones y quedarse con la mejor solución obtenida.

Formalmente, podríamos describir el algoritmo de la siguiente manera. Dados un conjunto de puntos X = {x1,…, xn}, el objetivo es encontrar una partición de X en k conjuntos ($k \leq n$) S = {S1, …, Sn} de manera que se minimice

$$\sum_{i=1}^{k} \sum_{\boldsymbol{x} \in \boldsymbol{S_i}} \|\boldsymbol{x} - \boldsymbol{\mu}_i\|^2$$

donde $\boldsymbol{\mu}_i$ es la media de los vectores en S_i.

Al final del proceso el espacio de las observaciones se puede representar como un diagrama de Voronoi, como el que se puede ver en la siguiente figura, tomada de [Voronoi]. Los puntos son los centroides obtenidos al final del proceso, y cada región coloreada es la zona que está más cerca de ese centroide que de cualquier otro.

Ilustración 3. 4. Diagrama de Voronoi.

Las características principales del algoritmo k –means son:

- Es apropiado para grupos compactos.
- Es sencillo conceptualmente y rápido de ejecutar. La complejidad es O(nkq), donde q es el número de iteraciones necesarias para alcanzar a convergencia. En la práctica q << n, así que es una buena opción para conjuntos grandes.
- Recibe el número de grupos como parámetro de entrada. Si el número de grupos no es apropiado, el algoritmo no obtendrá buenos resultados.
- Es aplicable a vectores con valores reales, pero en principio no se puede usar si las características son categorías.

Todas las observaciones son asignadas a algún grupo. Esto quiere decir que los valores atípicos y el ruido influenciarán los grupos formados. Se han desarrollado versiones de este algoritmo menos sensibles a los valores atípicos o para datos con valores discretos.

Inicialización de los centroides

Para la inicialización de los centroides, originalmente había dos opciones básicas:

- Escoger aleatoriamente k observaciones como centroides.
- Asignar aleatoriamente cada observación a un grupo y calcular la media de cada grupo para obtener los centroides iniciales.

En 1997, Athur y Vassilvitskii [Arthur] propusieron el método k-means++. El algoritmo es el siguiente:

1. Se escoge un centroide aleatoriamente entre los puntos de X.
2. Para el resto de los puntos, se calcula D(x), la distancia entre x y el centroide más cercano de los que ya se han escogido.
3. Se escoge un nuevo punto aleatoriamente, usando una distribución de probabilidad en la que la probabilidad de escoger un punto es proporcional a $D(x)^2$.
4. Los puntos 2 y 3 se repiten hasta que se hayan escogido k centroides.

Este método consigue una mejoría apreciable en el error final. Aunque la selección inicial toma un tiempo adicional, el algortimo k-means converge rápidamente y el tiempo total

es menor. Los autores encontraron que en general el tiempo de ejecución se reduce a la mitad.

Ejemplo

Finalizamos la explicación de este algoritmo con un ejemplo de ejecución realizado con una *applet* programada por E.M. Mirkes [Mirkes]. En el ejemplo, se agrupan 20 puntos distribuidos aleatoriamente en dos grupos. Los dos centroides iniciales se escogen aleatoriamente entre los puntos y el resto de puntos se asigna a uno u otro grupo según el centroide más próximo. En la ilustración se muestran los cuatro pasos que requiere este ejemplo para converger. Cada paso incluye actualización de centroides y asignación de puntos. Se puede observar que en los pasos 2-4 el centroide no es uno de los puntos. Del tercer al cuarto paso no hay cambio en los centroides, por lo que algoritmo termina.

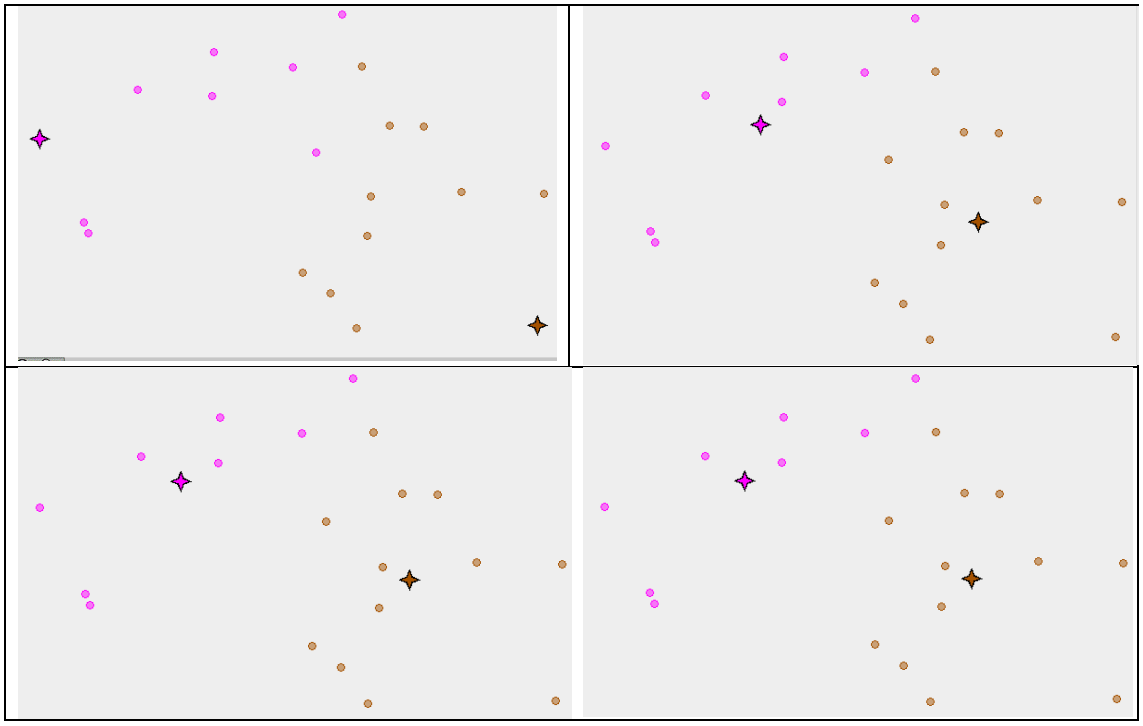

Ilustración 3. 5. Ejecución del algoritmo k-means.

Procesamiento por lotes (Mini batch k-means)

Con el fin de mejorar la velocidad de k-means se definió el algoritmo *mini batch k-means*. En lugar de utilizar todos los puntos, utiliza un pequeño subconjunto de los mismos, escogido aleatoriamente en cada iteración para minimizar la misma función que k-means. A cambio de una pequeña pérdida de calidad en la solución final, la velocidad de ejecución mejora drásticamente.

3.3.4.2. K-medioides

En el algoritmo k-means cada grupo está representado por un centroide, que es la media de los vectores del grupo. En esta versión, el representante debe ser necesariamente uno de los vectores. En concreto, aquel cuya distancia media con el resto de vectores del grupo es mínima, que se llamará medioide (*medoid*). Cada grupo contiene a su medioide y a todos los vectores que no son medioides y están más cerca del medioide de ese grupo que de cualquier otro.

Además, puede utilizar cualquier distancia entre pares de puntos, en lugar del cuadrado de la distancia euclídea como k-means. La propuesta original [Kaufman] usaba la distancia Manhattan, pero también es habitual usar la euclídea o la de Minkowski.

Hay varios algoritmos basados en el método de los k-medioides, el más utilizado es PAM (*Partitioning Around Medioids*):

1. Seleccionar aleatoriamente k elementos como medioides inciales.
2. Asociar cada punto (que no sea un medioide) al medioide más cercano.
3. Para cada medioide m,
    1. Para cada punto que no sea un medioide, o
    2. Intercambiar m y o y calcular la función de coste
4. Seleccionar la configuración con el menor coste.
5. Repetir los pasos 2-4 hasta que no haya cambios en los medioides.

En la siguiente ilustración, también obtenida con el applet de [Mirkes], se muestran tanto los centroides como los medioides obtenidos al final de los algoritmos k-means y k-medoids, respectivamente. Los centroides son las estrellas de cuatro puntas y no coinciden con ninguno de los puntos.

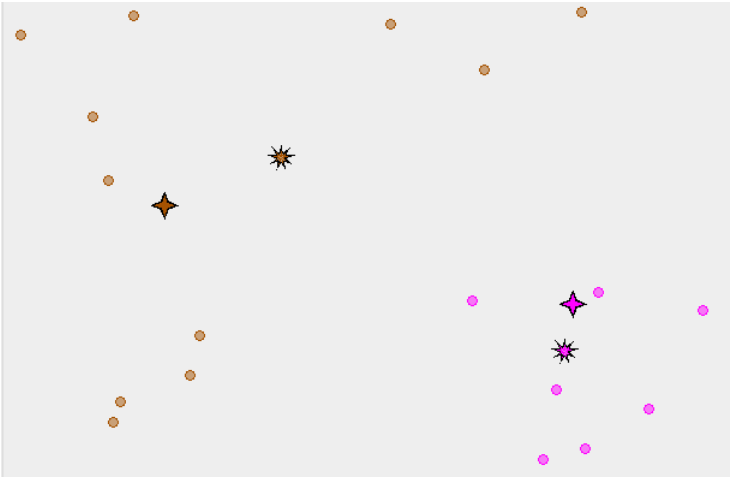

Ilustración 3. 6. Centroides (cuatro puntas) y mediodes para un agrupamiento en dos grupos.

Este algoritmo tiene dos ventajas sobre k-means:

- Se puede aplicar a datos con características nominales o discretas, en las que el valor medio de un conjunto de vectores puede no estar en el dominio de los vectores.
- Suele ser más robusto frente al ruido y los valores atípicos.

Como contrapartida, también tiene algunas desventajas:

- La media de un grupo tiene un significado estadístico y geométrico claro. Con los medioides no es necesariamente así.
- Calcular los medioides es computacionalmente más exigente.

3.3.4.3. Clustering difuso: Fuzzy c-means

En los algoritmos de agrupamiento difuso los vectores no se asignan a un solo grupo, sino que pertenecen a varios grupos, a cada uno de ellos con un grado de pertenencia. El algoritmo fue propuesto en 1973 por Dunn y modificado en 1981 por Bezdek.

Se trata de minimizar la siguiente función objetivo:

$$J_m = \sum_{i=1}^{N}\sum_{j=1}^{C} u_{ij}^{m} \left\|x_i - c_j\right\|^2, 1 \le m < \infty$$

donde C es el número de grupos, m es un número real llamado factor de borrosidad, u_{ij} es el grado de pertenencia del vector x_i al grupo j, cuyo representante es c_j y $\|\cdot\|$ puede ser cualquier medida de similitud. La suma de los grados de pertenencia de un mismo vector a los C grupos debe sumar uno, es decir

$$\sum_{k=1}^{C} u_{ik} = 1, \qquad \forall\, i$$

El algoritmo recibe como entrada el número de grupos, el factor de borrosidad y el umbral y procede iterativamente hasta que la diferencia entre los grados de pertenencia entre dos pasos consecutivos cae por debajo de cierto umbral. El pseudocódigo podría ser el siguiente:

1. Inicializar aleatoriamente los grados de pertenencia u_{ij}.
2. Calcular los centroides según

$$c_j = \frac{\sum_{i=1}^{N} u_{ij}^m \cdot x_i}{\sum_{i=1}^{N} u_{ij}^m}$$

3. Calcular los grados de pertenencia según

$$u_{ij} = \frac{1}{\sum_{k=1}^{C} \left(\frac{\|x_i - c_j\|}{\|x_i - c_k\|} \right)^{\frac{2}{m-1}}}$$

4. Si no se ha alcanzado la condición de convergencia, volver al punto 2. Si se ha alcanzado, el algoritmo termina.

La condición de convergencia se alcanza cuando la diferencia entre los factores de pertenencia entre dos pasos consecutivos cae por debajo de cierto umbral predeterminado. En concreto, se calcula la diferencia entre los factores u_{ij} en el paso p y en el paso p + 1. Si para todos los valores de i y j el valor de la diferencia está por debajo de este umbral, el algoritmo finaliza. Es decir

$$\left| u_{ij}^{(p+1)} - u_{ij}^{(p)} \right| < \delta, \forall\, i, j$$

Como en el caso de k-means, el algoritmo encuentra un mínimo local de la función.

3.3.5. Algoritmos basados en la densidad de puntos

Estos algoritmos consideran que los grupos son las regiones del espacio l-dimensional densas en puntos. No imponen ninguna restricción sobre la forma de los grupos, así que pueden encontrar grupos de forma arbitraria. Además son robustos frente a los valores atípicos. La complejidad en el tiempo es menor que $O(n^2)$, por lo que son una opción para

conjuntos grandes. Algunos algoritmos de este tipo son DBSCAN, OPTICS, DBCLASD y DENCLUE.

3.3.5.1. DBSCAN

El algoritmo DBSCAN es uno de los más habituales en la actualidad. Fue propuesto por Ester, Kroegel, Sander y Xu en 1996 [Ester]. Las siglas significan *Density-based spatial clustering of applications with noise* y, como su nombre indica, es un algoritmo basado en densidad para aplicaciones con ruido. Los grupos se forman en las zonas de mayor densidad de puntos, donde los puntos tienen muchos vecinos. Los puntos que están en zonas de baja densidad se consideran valores atípicos y no se asignan a ningún grupo.

DBSCAN recibe dos parámetnros, minPts y ε. A la hora de agrupar un conjunto de puntos, X, DBSCAN los clasifica en puntos núcleo, puntos alcanzables y atípicos:

- Puntos núcleo. Un punto p es núcleo si tiene al menos minPts a una distancia ε. Se dice que esos puntos son directamente alcanzables desde p.
- Un punto q es alcanzable desde p si existe un camino desde p hasta q pasando por una serie de puntos pi de manera que cada pi+1 es directamente alcanzable desde pi. Todos los puntos del camino deben ser puntos núcleo, salvo q.
- Los puntos que no son alcanzables desde ningún otro punto son atípicos.

Los grupos se forman con al menos un punto núcleo y todos los puntos que son alcanzables desde él, sean estos puntos núcleo o no.

El pseudocódigo del algoritmo es:

```
DBSCAN(X, minPts, ε)
        C = 0
        para cada punto p de X
                si p está visitado
                        pasar al siguiente punto
                marcar p como visitado
                puntosVecinos = buscarVecinos(P, ε)
                si len(puntosVecinos) <minPts
                        marcar p como ruido
                sino
```

```
                    C = siguiente grupo
                    expandirGrupo(p, puntosVecinos, C, ε, minPts)
expandirGrupo(p, puntosVecinos, C, ε, minPts)
    añadir p a C
    para cada punto v en puntosVecinos
        si v no está visitado
            marcar v como visitado
            puntosVecinos2 =buscarVecinos(v, eps)
            si len(puntosVecinos2) >= minPts
                puntosVecinos = puntosVecinos + puntosVecinos2
        si v no está en ningún grupo
            añadir v a C
buscarVecinos(p, ε)
    devolver todos los puntos a distancia menor que ε de p, p incluido
```

Ventajas:

- No hay que indicar el número de grupos.
- Es robusto frente a los valores atípicos, que no afectan a la formación de los grupos.
- Puede encontrar grupos de forma arbitraria.
- No importa como estén ordenados los puntos, salvo para los que están en el borde de grupos.

Desventajas:

- No es enteramente determinista. Los puntos que son alcanzables desde dos grupos pueden ser asignados a cualquiera de ellos. De cualquier manera, la asignación de puntos como ruido o punto núcleo sí es determinista.
- No funciona bien si los grupos tienen densidades distintas porque es difícil fijar parámetros que valgan para todos los casos.
- Fijar los valores adecuados de los parámetros puede ser complicado si no se entienden bien los datos.

3.3.5.2. OPTICS

OPTICS (*Ordering Points To Identify Clustering Structure*). Es un algoritmo similar a DBSCAN propuesto en 1999 [Ankerst]. Intenta mejorar una de los puntos problemáticos de DBSCAN: la identificación de grupos con diferentes densidades.

Los puntos del conjunto se ordenan de manera que los puntos más cercanos espacialmente sean vecinos en la ordenación. Además, para cada punto se almacena también una distancia que representa la densidad que hay que aceptar en un grupo para que los dos puntos pertenezcan al mismo grupo.

Al igual que DBSCAN recibe dos parámetnros, minPts y ε. A cada punto se le asigna una distancia núcleo (*core distance*) que describe la distancia al vecino minPts-ésimo. Es decir, si minPts = 5, la distancia al quinto punto más cercano.

$$core - dist_{\varepsilon,minPts}(p) = \begin{cases} Sin\ definir & si\ |N_\varepsilon(p)| < minPts \\ \text{distancia al vecino minPts} - \text{ésimo} & en\ otro\ caso \end{cases}$$

$|N_\varepsilon(p)|$ representa el número de puntos que hay la ε–vecindad de p. De esta manera OPTICS identifica puntos que forman parte de un grupo más compacto.

También se define la distancia de accesibilidad (*reachability distance*) entre un punto y su vecino más cercano.

$$reachability - dist_{\varepsilon,minPts}(o,p) = \begin{cases} Sin\ definir & si\ |N_\varepsilon(p)| < minPts \\ \max(core - dist_{\varepsilon,minPts}(p), dist(o,p) & en\ otro\ caso \end{cases}$$

Está es el ε'< ε que hay que asumir para que p y o pertenezcan al mismo grupo.

La ilustración 3.7 uestra cómo funciona el algoritmo. En la parte superior izquierda está el conjunto de datos. En la parte superior derecha, el árbol de expansión generado por OPTICS y en la parte de abajo el gráfico de accesibilidad.

Este gráfico es un tipo de dendrograma. Tiene en el eje x la ordenación de los puntos de X y en el eje y la distancia de accesibilidad de cada punto. Como los puntos que están en el mismo grupo tienen distancias de accesibilidad pequeñas, los grupos están representado por valles en el gráfico (los colores de la ilustración son etiquetas, no los genera OPTICS). Cuanto más profundo sea el valle, más denso será el grupo. Las zonas amarillas se corresponden con los puntos atípicos, se puede observar que en ellas no hay valles. Estos valles se pueden detectar visualmente o mediante algún algoritmo, usando en cualquier

caso un valor umbral para los valores del eje y. Con diferentes valores del umbral se obtiene una jerarquía de agrupamientos.

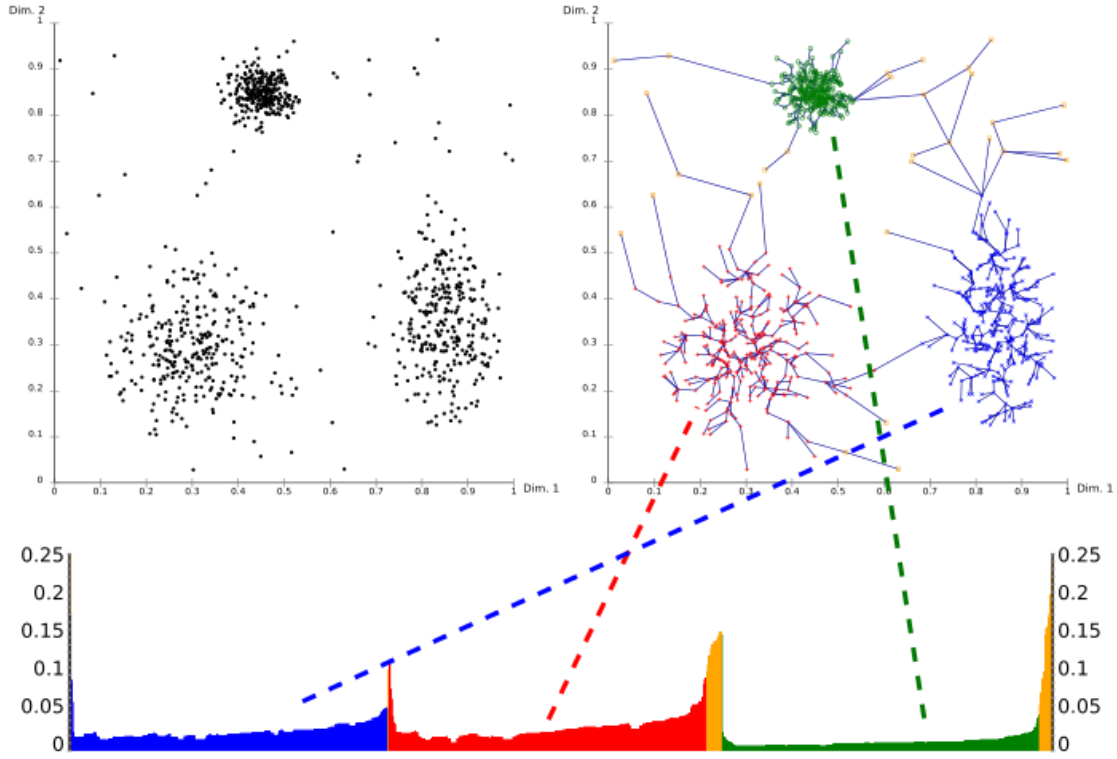

Ilustración 3. 7. Resultado de OPTICS. Tomado de [OPTICS].

3.3.6. Algoritmos basados en grafos

Estos algoritmos comienzan construyendo un grafo a partir del conjunto de puntos que se quiere agrupar. Una de sus principales ventajas es que pueden detectar grupos de diferentes formas, característica que tienen pocos algoritmos.

Antes de empezar, conviene recordar algunos conceptos básicos de teoría de grafos que se utilizarán durante la explicación.

Un gráfo G = (V, E) es un conjunto de objetos en el que algunas parejas de objetos están conectadas por vínculos. Los objetos interconectados se llaman vértices y los vínculos que conectan algunos de estos vértices aristas. Se dice que dos vértices unidos por una arista están conectados.

Según las características que cumpla, un grafo puede ser:

- Completo, si todas las posibles parejas de vértices están conectadas.
- Denso, si el número de aristas está próximo al máximo posible.
- Ponderado, si las aristas llevan asociadas un valor, llamado peso.

- Conexo, si existe un camino entre cualquier par de vértices. Hay un camino entre dos vértices v_i y v_j si existe una sucesión de vértices conectados que empieza en vi y termina en v_j. Un grafo completo es claramente conexo.
- No dirigido, si las aristas no tienen orientación.

3.3.6.1. Algoritmos basados en el árbol mínimo recubridor

Dado un grafo conexo, no dirigido y ponderado, un árbol recubridor del mismo es un subgrafo que contiene todos los vértices y además es un árbol [Grygo]. De entre todos los árboles recubridores de un grafo, el árbol recubridor mínimo (en adelante MST, de *Minimum Spanning Tree*) es aquel que menos pesa, es decir, aquel en el que la suma de los pesos asociados a las aristas es menor.

A partir del conjunto de datos X se construye el grafo completo y ponderado G(V, E) en el que cada punto de X es un vértice de G. A cada arista se le asigna un peso igual a la distancia correspondiente entre los puntos (vértices) que une, $w_e = d(\mathbf{x}_i, \mathbf{x}_j)$. Por tanto el peso de una arista representa en este caso una distancia. Cuanto mayor sea, más alejados (menos similares) están los vértices (puntos) que une.

Una vez determinado el MST de G, hay dos aproximaciones. Si el número de grupos, k, que se desea obtener está determinado, basta con eliminar las k – 1 aristas más pesadas del MST. Eso dará lugar a k componentes conexas, que son los k grupos buscados. Si el número de grupos no está determinado se eliminan las aristas consideradas inconsistentes.

Se pueden definir diferente criterios para considerar a una arista inconsistente. En general se trata de aristas con un peso sustancialmente mayor que las que le rodean. Esto indicaría que esa arista separa dos conjuntos de vértices de manera que los vértices de cada conjunto están más próximos entre sí (aristas cortas) que de los vértices del otro conjunto.

Una manera sencilla de definir las aristas inconsistentes sería la siguiente:

- Para cada arista, se consideran todas las aristas que están, como mucho, a p pasos de la misma y se calcula la media y la desviación estándar de sus pesos, m_e y w_e. Dos aristas e_1 y e_2 están a p pasos la una de la otra si el camino mínimo que conecta un vértice de e_1 con de e_2 contiene p-1 aristas.
- Las aristas cuyo peso esté más de q desviaciones estándar por encima de m_e se consideran inconsistentes.

Usando este método los resultados dependen de los valores elegidos para p y q. Un valor habitual para q es 2.

Este algoritmo está recomendado para casos en que los grupos se tocan o tienen densidades diferentes. El resultado no depende de la manera en la que se ordenen los puntos como en otros casos.

Sobre el tiempo de ejecución, es posible determinar el MST de un grafo denso en un tiempo del orden O(m), donde m es el número de aristas. Como se empieza con un grafo completo, y por tanto denso, se tendrá $O(n^2)$, donde n es el número de vértices. El tiempo para eliminar las aristas depende del método usado.

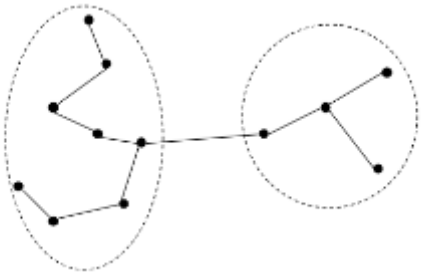

Ilustración 3. 8. Determinación de los grupos a partir del MST.

3.3.6.2. Agrupamiento espectral (spectral clustering)

Esto algoritmos [Luxburg] han recibido mucha atención recientemente debido a sus buenos resultados en aplicaciones en las que los algoritmos clásicos no funcionan bien. Realiza una primera fase de reducción de la dimensionalidad antes de proceder al agrupamiento utilizando la descomposición espectral (autovectores y autovalores) de cierta matriz. Para poder explicar este algoritmo es necesario definir varios conceptos previos.

Dados un conjunto de puntos X y una medida de similitud $s_{ij} \geq 0$ entre todos los pares de datos x_i y x_j, se construye el grafo de similitud, no dirigido y conexo, G = (V, E), en el que cada vértice representa un punto. Se asigna a cada arista un peso, w_{ij}, que representa la similitud entre v_i y v_j. Estos pesos se pueden agrupar en la matriz de proximidad o afinidad, W = [W(i, j)], i, j = 1, …, N, donde N es el número de vértices. En lo que sigue, se asume que se una matriz simétrica. Al contrario que en el caso anterior, se trata de una medida de similitud, no de disimilitud. Cuanto mayor sea s_{ij} más parecidos son x_i y x_j. Una elección habitual es

$$W(i,j) = \begin{cases} exp\left(\frac{\left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\|^2}{2\sigma^2}\right), si\ \left\|\boldsymbol{x}_i - \boldsymbol{x}_j\right\| < \varepsilon \\ 0, \qquad en\ otro\ caso \end{cases}$$

La constante ε la fija el usuario, y es un umbral por encima del cual dos vértices no están conectados. σ también la fija el usuario y su elección influenciará el resultado del agrupamiento.

Con la intención de simplificar la explicación, vamos a suponer que se pretende dividir X en solo dos grupos A y B. Hay que elegir un criterio apropiado para partir el grafo en dos y encontrar un algoritmo eficiente para hacerlo.

Un criterio asociado es el llamado corte. Si A y B son los grupos resultantes, el corte es

$$corte(A,B) = \sum_{i \in A, j \in B} W(i,j)$$

Es decir, seleccionar A y B de manera que la suma de los pesos entre las parejas formadas por un elemento de A y uno de B sea mínima. De esta manera, los puntos de A y B son menos similares que en cualquier otra partición de V. Este criterio tiene como desventaja que tiende a crear grupos pequeños y aislados.

Para solucionarlo es habitual el corte normalizado (aunque también se utilizan otros criterios, cada uno con sus ventajas e inconvenientes). El corte normalizado entre dos grupos A y B es

$$Ncorte(A,B) = \frac{corte(A,B)}{V(A)} + \frac{corte(A,B)}{V(B)}$$

donde V(A) es el grado de A, definido como

$$V(A) = \sum_{i \in A} D_{ii} = \sum_{i \in A, j \in V} W(i,j)$$

El índice

$$D_{ii} = \sum_{j \in V} W(i,j)$$

mide la similitud de del vértice v_i con el resto de nodos. Una valor bajo representa un nodo aislado.

Como minimizar Ncorte(A,b) es un problema NP-duro se busca una solución aproximada que se pueda encontrar de manera más eficiente. En concreto, se calcula el autovector asociado al segundo menor autovalor, **v,** de la matriz la laplacina normalizada.

$L_{norm} = I- D^{-1/2}WD^{-1/2}$

También es posible utilizar otras definiciones de matriz laplaciana normalizada, como por ejemplo $L = D^{-1}W$. En ese caso se utiliza el autovector asociado al mayor autovalor.

Para realizar la asignación de los puntos se puede proceder de varias maneras. Por ejemplo, se calcula la mediana, m, de las componentes de **v**. Los puntos se asignan al grupo A o B según su componente en **v** sea mayor o menor que m. El proceso se puede repetir dentro de cada grupo si se desea obtener más de dos grupos.

Otro enfoque permite obtener k grupos directamente. Consiste en calcular k autovectores en lugar de uno solo y utilizar otro algoritmo, como k-means, para agrupar los puntos en función de sus k componentes en esos vectores.

Respecto de la complejidad, calcular los autovalores y autovectores de una matriz N × N es un problema $O(N^3)$. De cualquier manera, en la mayoría de los casos se trata de una matriz de dispersa. Por otro lado, solo requiere el menor autovalor y su autovector asociado, sin que sea necesario calcularlos con gran exactitud. En estas condiciones, se puede usar el algoritmo de Lanczos y el tiempo de ejecución cae $O(N^{3/2})$.

3.3.7. Algoritmos basados en el intercambio de mensajes

Dentro de este apartado vamos a ver único algoritmo, definido recientemente [Frey], llamado propagación de afinidad.

3.3.7.1. Propagación de afinidad

Este algoritmo crea los grupos mediante intercambio de mensajes entre pares de puntos hasta que identifica a algunos de ellos como representantes (*exemplars*) del resto. La diferencia con otros algoritmos que utilizan representantes es que inicialmente considera a todos los puntos como posibles candidatos.

En el apartado sobre el algoritmo k-means vimos que la elección de los representantes (llamados centroides en ese algoritmo) influenciaba en gran medida el resultado final. Para solucionarlo se ejecutaba varias veces el algoritmo, con inicializaciones diferentes.

Esta opción solo funciona bien cuando el número de grupos es bajo y hay una probabilidad razonable de conseguir un buen conjunto de representantes.

En el algoritmo de propagación de afinidad, cada punto se interpreta como un nodo en una red y los nodos se intercambian mensajes recursivamente hasta que se encuentran los representantes, y con ellos los grupos. Los mensajes toman valores reales y su magnitud representa la afinidad que en ese momento tiene un nodo para elegir a otro como su representante. De ahí el nombre *propagación de afinidad*. Los valores de los mensajes se utilizan buscando minimizar cierta función.

La entrada del algoritmo es una serie de valores s(i, k), que indica como de apropiado es el punto k para ser representante del punto i. Cuando el objetivo es minimizar el error cuadrático, la similitud entre dos puntos se inicializa al error cuadrático negativo (distancia euclídea): para $\mathbf{x}_i$ y $\mathbf{x}_k$, $s(i, k) = |\mathbf{x}_i - \mathbf{x}_k|^2$, pero se pueden utilizar otros criterios.

No recibe el número de grupos como entrada. En su lugar, se indican un valor de preferencia para cada punto, s(k, k), de manera que los puntos con mayor preferencia tienen más posibilidades de ser elegidos como representantes.

El número final de representantes, y por tanto de grupos, está influenciado por estas preferencias, pero se determina a través del proceso de intercambio de mensajes y también está influenciado por el facto de amortiguación (*damping factor*). Si en un principio no hay ningún punto especialmente indicado para ser candidato, todos los puntos deberían tener la misma preferencia. Este valor común se puede variar para obtener un número de grupos diferente. Si es el mínimo de las similitudes de entrada se obtendrá un número reducido de grupos, si es la mediana, un número moderado.

Los mensajes intercambiados son de dos tipos: responsabilidad y disponiblidad (*responsibility* y *availability*). La responsabilidad r(i, k) es la evidencia acumulada a favor de que el punto k sea el representante del punto i. La disponibilidad a(i, k) es la evidencia acumulada a favor de que el punto i elija como representante al punto k, teniendo en cuenta los valores de los demás puntos a favor de que k sea un representante. De esta manera, se escoge como representante a los números suficientemente similares a muchos puntos y escogidos por muchos puntos para ser su propio representante.

Ambas cantidades se inicializan a cero y en cada iteración del algoritmo se actualizan. Primero se actualizan las responsabilidades según

$$r(i,k) \leftarrow s(i,k) - max_{k' \neq k}\{a(i,k') + s(i,k')\}$$

y luego las disponibilidades,

$$a(i,k) \leftarrow min\left(0, r(k,k) + \sum \max\left(0, r(i',k)\right)\right) para\ i\ \neq k$$

$$a(k,k) \leftarrow \sum_{i' \neq k} max(0, r(i',k))$$

Se puede observar que a la hora de calcular la disponibilidad solo se tienen en cuenta las responsabilidades positivas que recibe el punto k. Es así porque un punto candidato solo debe ser un buen representante (responsabilidad positiva) para algunos puntos, sin importar como de malo sea para otros (responsabilidad negativa).

Esta actualización de valores es el intercambio de mensajes. Solo requiere cálculos locales, y solo hay que intercambiar mensajes entre los puntos con similitud conocida. Al actualizar los mensajes los valores se amortiguan con un factor λ (0.5 en [Frey]) para evitar oscilaciones. Cada mensaje se multiplica por λ veces el valor de la previa iteración más 1 - λ veces el valor de actualización.

Se puede finalizar tras un número fijo de iteraciones, cuando los cambios en los mensajes caen por debajo de cierto umbral, o cuando los representantes no cambian tras un cierto número de iteraciones. En [Frey] se finalizaba la ejecución cuando los representantes no cambiaban tras diez iteraciones.

En cualquier momento es posible identificar los representantes combinando responsabilidad y disponibilidad. Para el punto i, el valor de k que maximiza a(i, k) + r(i, k) identifica al punto i como representante si k = i, o al punto que representa a i si k ≠ i.

La siguiente ilustración muestra el progreso del algoritmo utilizando el error cuadrático negativo como similitud. A medida que se va acumulando evidencia a favor de que un punto sea representante su color va cambiando.

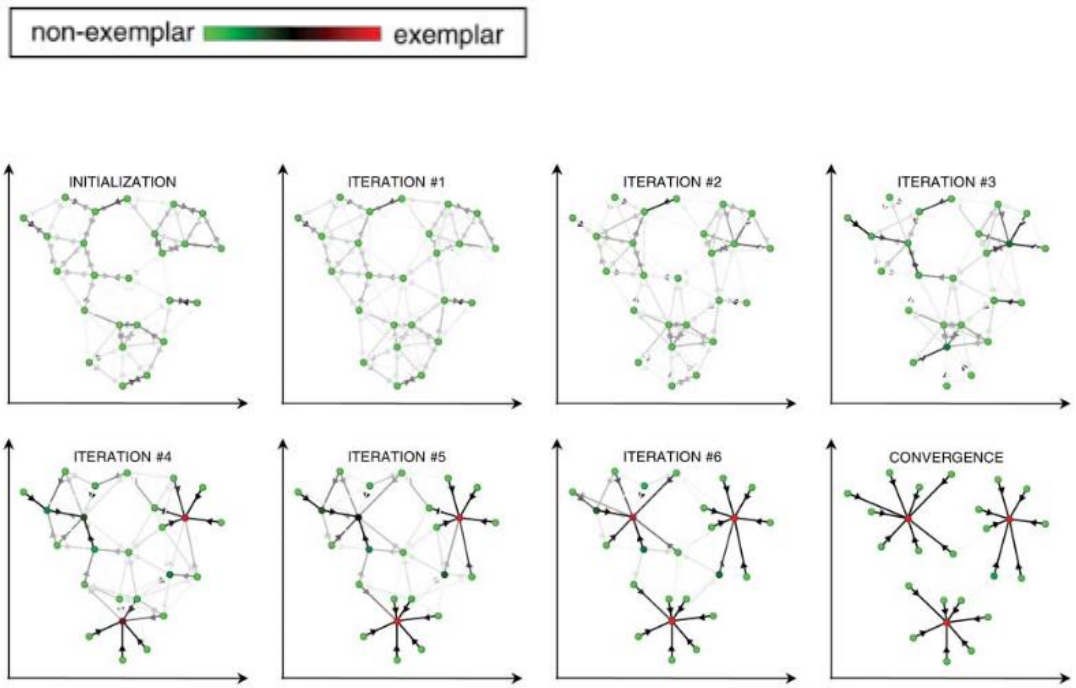

Ilustración 3. 9. Propagación de afinidad. Tomado de [Frey].

El mayor problema de este algoritmo es su complejidad. La complejidad temporal es $O(N^2T)$, donde N es el número de puntos y T el número de iteraciones hasta la convergencia. La complejidad de la memoria es $O(N^2)$ si se usa una matriz de similitud densa, pero reducible si es una matriz dispersa. Por lo tanto, no es una buena opción para conjuntos grandes.

3.4. Agrupamiento por consenso

El agrupamiento por consenso no es un algoritmo de agrupamiento como los anteriores. Se trata de partir de varios agrupamientos diferentes para obtener uno que sea mejor en algún sentido que los agrupamientos de entrada. Es un planteamiento análogo a utilizar un conjunto de clasificadores en los problemas de clasificación supervisada.

Una de los motivos es para utilizar este enfoque es que, como hemos visto a lo largo de este capítulo no todos los algoritmos de agrupamiento son igual de efectivos en todos los casos. Para un cierto conjunto de datos un algoritmos puede obtener resultados pobres mientras que otro puede obtenerlos muy buenos. Por ejemplo algunos algoritmos están más indicados para grupos compactos mientras que otros son más apropiados para grupos alargados, pero en general no se conoce la forma de los grupos a priori.

3.4.1. Obtención de un conjunto de agrupamientos

Para generar los diferentes agrupamientos se pueden seguir varias estrategias:

- Todos los agrupamientos se realizan con todos los elementos y con todas sus características. Los diferentes agrupamientos se pueden obtener con diferentes algoritmos o con diferentes parámetros de un solo algoritmo.
- Todos los agrupamientos se realizan con todos los elementos y pero solo con algunas de sus características. Cada agrupamiento se realiza con todos los elementos disponibles pero no con todas sus dimensiones. Se selecciona, mediante proyecciones aleatorias o de otro modo, un subconjunto de características diferente para cada agrupamiento. Los diferentes agrupamientos se pueden obtener con diferentes algoritmos o con diferentes parámetros de un solo algoritmo.
- Todos los agrupamientos se realizan con todas las características, pero solo parte de los elementos. Los elementos que no se utilicen para generar un agrupamiento pueden asignarse al grupo más cercano. Normalmente se utiliza el mismo algoritmo en todos los agrupamientos.

3.4.2. Agrupamiento de consenso

Una vez se han obtenido varios agrupamientos hay que decidir cómo combinarlos para conseguir el agrupamiento de consenso. Vamos a ver tres formas eficientes de hacerlo [Strehl], todas basadas en la representación de un conjunto de agrupamientos como hipergrafo.

3.4.2.1. Representación de un conjunto de agrupamientos mediante un hipergrafo

Un grafo está formado por vértices y aristas. Las aristas conectan dos vértices. En un hipergrafo, en lugar de aristas hay hiperaristas, que puede conectar cualquier conjunto de vértices.

Cada agrupamiento se puede representar como un vector de etiquetas. Por ejemplo, si tenemos siete elementos que agrupar el vector (1, 1, 1, 2, 2, 3, 1) significa que los elementos primero, segundo, terceo y séptimo perteneces al mismo grupo. Los vectores cuarto y quinto a un segundo grupo. El sexto elemento está solo en un tercer grupo.

Si tenemos r agrupamientos de N elementos en k grupos (k no tiene por qué ser el mismo en todos los agrupamientos) disponibles, para cada vector de etiquetas $\boldsymbol{\lambda}^{(q)} \in N^N$, se

construye la matriz binaria de pertenencia $\mathbf{H}^{(q)} \in N^{N \times k^{(q)}}$ en la que cada grupo está representado como una hiperarista (columna). En una matriz de pertenencia las entradas de una fila suman 1 si la fila corresponde a un objeto con etiqueta conocida, es decir, un objeto asignado a algún grupo. Si el objeto no tiene etiqueta conocida, toda la fila correspondiente tiene ceros.

La siguiente ilustración sirve de ejemplo. En la primera tabla se representan los agrupamientos disponibles. Hay siete elementos y cuatro agrupamientos diferentes. En la cuarta columna se puede observar que el cuarto agrupamientosdeja los elementos tercero, sexto y séptimo sin asignar.

	$\lambda^{(1)}$	$\lambda^{(2)}$	$\lambda^{(3)}$	$\lambda^{(4)}$
x_1	1	2	1	1
x_2	1	2	1	2
x_3	1	2	2	?
x_4	2	3	2	1
x_5	2	3	3	2
x_6	3	1	3	?
x_7	3	1	3	?

⇔

	$\mathbf{H}^{(1)}$			$\mathbf{H}^{(2)}$			$\mathbf{H}^{(3)}$			$\mathbf{H}^{(4)}$	
	$\mathbf{h}_1$	$\mathbf{h}_2$	$\mathbf{h}_3$	$\mathbf{h}_4$	$\mathbf{h}_5$	$\mathbf{h}_6$	$\mathbf{h}_7$	$\mathbf{h}_8$	$\mathbf{h}_9$	$\mathbf{h}_{10}$	$\mathbf{h}_{11}$
v_1	1	0	0	0	1	0	1	0	0	1	0
v_2	1	0	0	0	1	0	1	0	0	0	1
v_3	1	0	0	0	1	0	0	1	0	0	0
v_4	0	1	0	0	0	1	0	1	0	1	0
v_5	0	1	0	0	0	1	0	0	1	0	1
v_6	0	0	1	1	0	0	0	0	1	0	0
v_7	0	0	1	1	0	0	0	0	1	0	0

Ilustración 3. 10. Representación de un conjunto de agrupamientos como hipergrafo.

En la segunda tabla se muestran las matrices de pertenencia, una por cada agrupamiento. Cada grupo se convierte en una hiperarista, representada en una columna de la matriz. Por eso cada matriz tiene tantas columnas como grupos. Todas tienen tres columnas menos la última, que tiene dos, porque el agrupamiento correspondiente solo dio lugar a dos grupos. Si en la posición correspondiente a la fila i y la columna h_j hay un 1 quiere decir que el vértice v_j forma parte de la hiperarista h_j, es decir que el elemento x_i pertenece al grupo j-ésimo en el agrupamiento correspondiente.

La matriz concatenada $\mathbf{H} = (\mathbf{H}^{(1)} \ldots \mathbf{H}^{(r)})$ es la matriz de adyacencia de un hipergrafo con N vértice y $\sum_{q=1}^{r} k^{(q)}$ hiperaristas. De esta manera se mapea el conjunto de agrupamientos a un hipergrafo.

3.4.2.2. Algoritmo de particionado basado en la similitud de los grupos

Este algoritmo (*Cluster-based Similarity Partitioning Algorithm*, CSPA) sigue un enfoque muy sencillo. Si dos elementos están en el mismo grupo, se consideran similares. Si no, se consideran diferentes. De esta manera es posible pasar de un agrupamiento a una matriz binaria de similitud N × N en la que la posición (i, j) tiene un 1 si los elementos x_i y x_j pertenecen al mismo grupo y un 0 en otro caso. Si hacemos esto para los r agrupamientos y se calcula la media posición a posición de las matrices resultantes se obtiene una matriz de similitud promedio. Se puede obtener directamente mediante la expresión

$$\boldsymbol{S} = \frac{1}{r}\boldsymbol{H}\boldsymbol{H}^{*}$$

Partiendo de esta matriz se puede realizar el agrupamiento con un algoritmo de agrupamiento estándar.

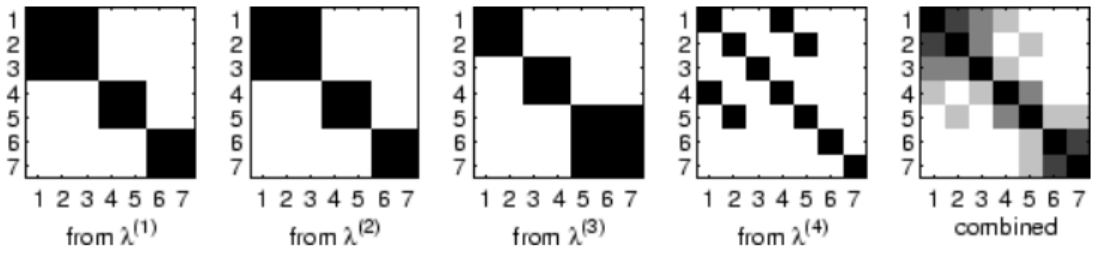

Ilustración 3. 11. Algoritmo CSPA.

3.4.2.3. Algoritmo de partición del hipergrafo

Este algortimo (HyperGraf Partitioning Algorithm) reparticiona los datos usando los grupos de entrada como indicaciones de vínculos fuertes. Se trata de encontrar una partición del hipergrafo eliminando un número mínimo de hiperaristas.

3.4.2.4. Algoritmo de Meta-Clustering

Este algoritmo (*Meta-Clustering Algorithm*, MCLA) se basa en agrupar grupos. Además, los elementos quedan asignados a un grupo con un cierto grado de confianza.

La idea de MCLA consiste en agrupar y colapsar hiperaristas relacionadas y asignar cada objeto a la hiperarista en la que más participa. Se realiza un agrupamiento de las hiperaristas para determinar cuáles están relacionadas. Al colapsar las hiperaristas el número de las mismas se reduce de $\sum_{q=1}^{r} k^{(q)}$ a k.

Tiene cuatro pasos básicos:

- Construir el hipergrafo. Las hiperaristas de H serán los vértices de otro grafo, regular y no dirigido, el meta-grafo. Los pesos de sus aristas son proporcionales a la similitud entre vértices. Una medida apropiada es la medida binaria de Jaccard, que es el ratio entre la intersección y la unión de los conjuntos de objetos correspondientes a las dos hiperaristas.
- Agrupar las hiperaristas. Se particiona el meta-grafo en k meta-grupos balanceados para encontrar las etiquetas equivalentes. Cada vértice tiene un peso proporcional al tamaño del grupo correspondiente. El balaceo asegura que la suma de los pesos de los vértices es aproximadamente la misma en cada meta-grupo. Como cada vértice en el meta-grafo representa una etiqueta de grupo diferente, un meta-grupo representa varias etiquetas correspondientes.
- Colapsar los meta-grupos. Para cada uno de los k meta-grupos se colapsan las hiperaristas en una única meta-hiperarista. Cada meta-hiperarista tiene un vector de asociación que contiene una posición por cada objeto, con un valor que indica su nivel de asociación con el meta-grupo correspondiente. El nivel de asociación va de 0 a 1, con 0 indicando la asociación más débil y 1 la más fuerte. Se calcula promediando todos los vectores **h** de un meta-grupo.
- Competir por los objetos. Cada objeto se asigna a su meta-grupo más asociado. En concreto, al meta-grupo con la entrada más alta en el vector de asociación, decidiendo los empates al azar. El nivel de confianza de una asignación viene dado por el ratio entre el nivel de asociación con el grupo al que sea asignado y la suma de los todos los niveles de asociación. No se garantiza que todos los grupos reciban al menos un elemento.

La siguiente ilustración muestra el resultado de aplicar el algoritmo al conjunto de agrupamientos de la ilustración 3.10. Como había un total de 11 grupos (tres agrupamientos con tres grupos cada uno y uno con dos grupos), el meta-grafo tiene 11 vértices.

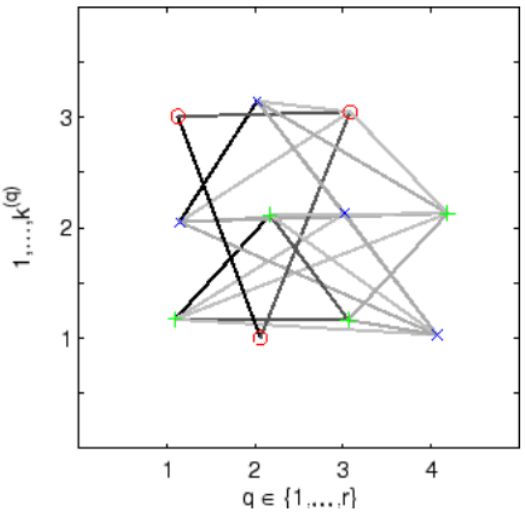

Ilustración 3. 12. Algoritmo MCLA.

Los tres meta-obtenidos se muestran respectivamente en rojo (círculos), verde (cruces) y azul (aspas). El primer meta-grupo es $C_1^{(M)} = \{h_3, h_4, h_9\}$. Colapsando las hiperaristas se obtiene la meta-hiperarista $h_1^{(M)} = \{v_5, v_6, v_7\}$ con vector de asociación (0, 0, 0, 0, 1/3, 1, 1). Por lo tanto, el meta-grupo es $C_1^{(M)}$ ganará la competición por los vértices v_6 y v_7 y por tanto se obtendrá el grupo $C_1 = \{x_6, x_7\}$ en el agrupamiento de consenso.

4. Tendencia, validación y reducción de la dimensionalidad

4.1. Tendencia al agrupamiento

Antes de empezar el proceso de agrupamiento es conveniente comprobar si de verdad los datos presentan una estructura de grupos o se corre el riesgo de malinterpretar los resultados del algoritmo de agrupamiento.

Dado un conjunto de datos, hay tres posibilidades:

- Los datos forman grupos.
- Los datos están repartidos de manera uniforme.
- Los datos están repartidos de forma aleatoria.

Un concepto importante para comprobar la tendencia al agrupamiento es la ventana de muestreo. El siguiente dibujo ilustra la idea. Las cruces representan los puntos que se pretende agrupar. Si se toma como referencia el círculo interior, los datos están distribuidos uniformemente. En cambio, si se usa el círculo exterior los datos parecen agrupados en el centro del mismo.

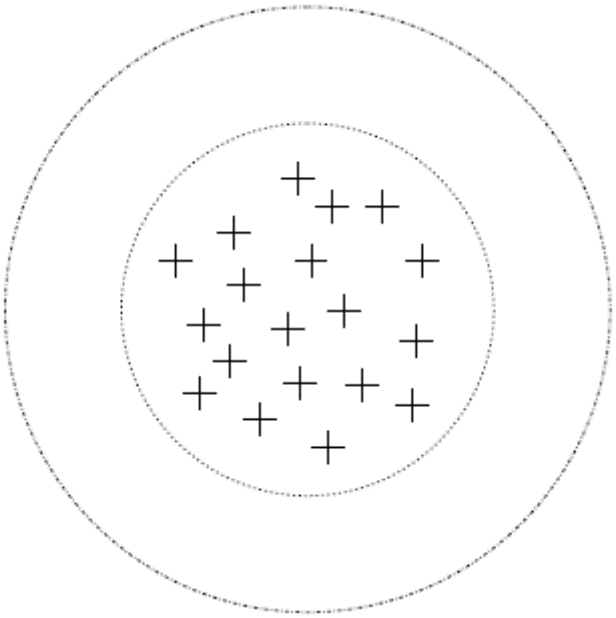

Ilustración 4. 1. Ventana de muestreo. Tomado de [Theo].

4.1.1. Test de Hopkins

Este test sirve para comprobar la distribución espacial de los puntos. Se toma un conjunto $X' = \{\mathbf{y}_i, i=1,\ldots, M\}$ de M vectores distribuidos aleatoriamente en la ventana de muestreo usando la distribución uniforme. Habitualmente, $M = 0.1N$. Se toma también un

subconjunto de M vectores de X, X_1. Sea d_j la distancia de $\mathbf{y}_j$ al vector más cercano de X_1, $\mathbf{x}_j$. Sea δ_j la distancia desde $\mathbf{x}_j$ hasta el vector más cercano en $X_1 - \{\mathbf{x}_j\}$. El estadístico de Hopkins se define como

$$h = \frac{\sum_{j=1}^{M} d_j^l}{\sum_{j=1}^{M} d_j^l + \sum_{j=1}^{M} \delta_j^l}$$

Compara la distribución de los vecinos más cercanos en X_1 con la de X'. Si X contiene grupos se espera que las distancias entre los puntos de X_1 sean pequeñas y por tanto el valor de h grande. Es más, los valores grandes de h indican una estructura de grupos en X. Cuando los puntos de X están distribuidos de manera uniforme en la ventana de muestreo el término $\sum_{j=1}^{M} d_j^l$ será menor que $\sum_{j=1}^{M} \delta_j^l$ y h tendrá un valor pequeño. Igualmente, los valores pequeños de h indican la presencia de puntos repartidos regularmente. Un valor cercano a 0.5 indica que los puntos están repartidos aleatoriamente.

4.1.2. Test de Cox-Lewis

El test original de Cox-Lewis se aplica en el caso de dos dimensiones y fue generalizado al caso l-dimensional por Panayirci [Pana]. Se define X' como en el caso anterior. Sea d_j la distancia de $\mathbf{y}_j$ al vector más cercano de X, $\mathbf{x}_j$. Sea δ_j la distancia desde $\mathbf{x}_j$ hasta el vector más cercano en $X - \{\mathbf{x}_j\}$, que llamaremos $\mathbf{x}_i$. Se consideran solo los $\mathbf{y}_j$ para los que $2dj/\ \delta j$. Sea M' el número de esos $\mathbf{y}_j$. Se define también una función apropiada $R_j(2d_j/\ \delta_j)$ para esos $\mathbf{y}_j$. A partir de estos elementos se define el estadístico

$$R = \frac{1}{M'} \sum_{j=1}^{M'} R_j$$

Los valores pequeños de R indican la presencia de grupos en X y valores grandes una estructura regular. Los valores alrededor de ½ significan que los vectores de X están repartidos aleatoriamente en la ventana de muestreo.

4.1.3. Algoritmos de valoración visual

Se trata de una familia de algoritmos que reordenan la matriz de disimilitud de los datos de forma al representarla sea posible encontrar la estructura subyacente. El algoritmo original VAT (*Visual Assesment of cluster Tendency*) fue propuesto por Bezdek y Hathaway [Bezdek] y a partir de él se han desarrollado varias versiones mejoradas.

Si se parte de un conjunto original de N datos, se puede construir una matriz de disimilitud, D, N×N en la que la posición (i, j) es el valor de la medida de disimilitud escogida entre los elementos i y j. La matriz se normaliza para que los valores estén entre 0 y 1.

VAT reordena las filas y columnas de D con una versión modificada del algoritmo de Prim para el árbol de recubrimiento mínimo y muestra la matriz reordenada, D', como una imagen en escala de grises. El objetivo es que D' se aproxime a una matriz diagonal por bloques. Si un elemento es miembro de un grupo, debería ser parte de una submatriz con valores de disimilitud bajos, que aparece como uno de los bloques oscuro a lo largo de la diagonal de la imagen VAT, I(D'), cada uno de los cuales se corresponde con un posible grupo.

En la siguiente ilustración se muestra el funcionamiento del algoritmo. En la imagen (a), a la izquierda, se puede ver el conjunto de elementos que se quiere agrupar, con cinco grupos claros. La imagen (b) es la representación de la matriz de disimilitud de los datos sin reordenar. Esta imagen no aporta ninguna información. La imagen (c) representa la matriz de similitud reordenada. En ella se observa claramente los cinco grupos como cinco cuadrados oscuros situados en la diagonal.

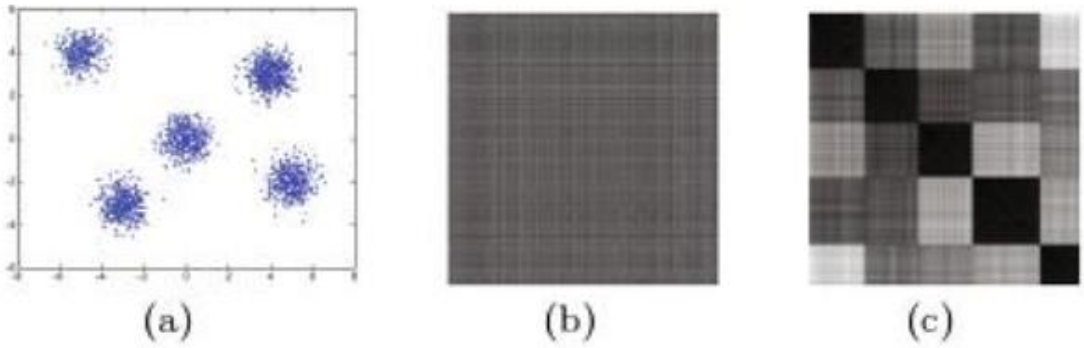

Ilustración 4. 2. Algoritmo VAT. Tomado de [Wang].

El algoritmo VAT funciona bien con grupos compactos bien diferenciados. Hay versiones mejoradas para estructuras de grupos complejas (iVAT, reVAT), conjuntos de datos grandes (bigVAT) o matrices de disimilitud rectangulares (coVAT).

4.2. Índices de validación

Una vez se ha realizado un agrupamiento hay que comprobar la calidad de los grupos obtenidos. Es habitual distinguir entre criterios internos y externos [Rendón].

- Externos. Estos criterios comparan una posible agrupación con una que se considere correcta. Obviamente, no tiene porqué existir, así que no siempre son aplicables.
- Internos. En este caso el objetivo el resultado usando solo cantidades y características presentes en el conjunto de datos.

4.2.1. Criterios externos

Estas métricas se calculan tomando como referencia una clasificación considerada como correcta de los documentos. Por ejemplo, partiendo de la colección [Almeida] se puede aplicar k-means con k = 2 para ver si realmente los dos grupos que se obtienen se corresponden con mensajes que son spam y mensajes que no. Hay que tener en cuenta que el resultado de un algoritmo de clasificación no supervisado no asigna etiquetas predeterminadas a los grupos. Se obtendrán dos grupos, pero el algoritmo no dirá que uno es de mensajes de spam y otro no, porque de hecho nunca se le han presentado esas etiquetas. Por tanto no hay que tener en cuenta el valor absoluto de las etiquetas de los grupos sino si la agrupación separa los datos de manera similar a la de referencia.

También conviene usar medidas ajustadas frente al azar. Las no ajustadas muestran dependencia entre el número de grupos y el de elemento. Por ejemplo, en el caso de la medida-V, explicada a continuación, la media de un agrupamiento aleatoria crece a medida que el número de grupos se acerca al de elementos. En el caso de las medidas ajustadas frente al azar, los agrupamientos aleatorios obtienen valores con media 0.

4.2.1.1. Homogeneidad, completitud y medida-V

Las tres siguientes métricas están muy relacionadas entre sí:

- Homogeneidad: un agrupamiento es homogéneo si cada grupo contiene solo miembro de una sola clase. El resultado estará entre 0, el peor resultado posible, y 1, el mejor.
- Completitud: un agrupamiento es completo si todos los elementos de una misma clase están en el mismo grupo. El resultado estará entre 0, el pero resultado posible, y 1, el mejor.

La completitud y la homogeneidad están relacionadas según:

homogeneidad(a, b) = completitud(b, a)

La homogeneidad y la completitud se calcula según

$$h = 1 - \frac{H(C|K)}{H(C)}$$

$$c = 1 - \frac{H(K|C)}{H(K)}$$

donde H(C | K) es la entropía condicional de las clases dado cierto agrupamiento

$$H(C|K) = -\sum_{c=1}^{|C|}\sum_{k=1}^{|K|}\frac{n_{c,k}}{n}\cdot log\left(\frac{n_{c,k}}{n_k}\right)$$

y H(C) es la entropía de las clases

$$H(C) = -\sum_{c=1}^{|C|}\frac{n_c}{n}\cdot log\left(\frac{n_c}{n}\right)$$

donde n es el número total de elementos, n_c y n_k son respectivamente las muestras que pertenecen a la clase c y al grupo k y $n_{c,k}$ es el número de elementos de la clase c asignados al grupo k. H (K|C) y H(K) se definen de manera análoga.

- Medida-V (*V-measure*): es la media armonizada entre homogeneidad y completitud.

$$V = 2 \cdot \frac{h \cdot c}{h + c}$$

Al contrario que las otras dos medidas, esta es simétrica. Se obtiene el mismo resultado si se intercambian las etiquetas reales y las resultantes de la agrupación. Se puede usar para ver el nivel de coincidencia entre dos agrupamientos.

El resultado estará entre 0, el peor resultado posible, y 1, el mejor. Es equivalente a la Información mutua normalizada (*Normalized Mutual Information* ,NMI).

Ventajas:

1. Los valores están acotados y son intuitivos.
2. No hace suposiciones sobre la estructura de los grupos.

Desventajas:

- No están normalizadas con respecto a un etiquetado aleatorio.
- Hay que conocer la clasificación real.

4.2.1.2. Índice de Rand Ajustado

El índice de Rand ajustado (*Adjusted Rand Index*, ARI) mide la similitud entre dos agrupamientos ignorando permutaciones y de manera ajustada ante el azar. Además, es simétrico. Si se intercambian las etiquetas del agrupamiento y las de referencia el resultado es el mismo.

Si llamando C al agrupamiento obtenido y W al de referencia, se definen:

- a, el número de parejas de elementos que están en el mismo grupo en C y en el mismo grupo en R.
- b, el número de parejas de elementos que están en grupos diferentes en C y en grupos diferentes en R.

A partir de estos valores se define el índice no ajustado de Rand como

$$RI = \frac{a + b}{C_2^N}$$

donde C_2^N es el número de posibles parejas formadas a partir de N elementos, sin importar el orden. N es el número de elementos que se están agrupando.

Esta definición no garantiza que un etiquetado aleatorio obtenga valores próximos a 0. Para conseguirlo se puede introducir el valor esperado de los etiquetados aleatorios. De esta manera se obtiene el índice de Rand ajustado:

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

Ventajas:

- Un etiquetado aleatorio uniforme obtendrá un valor próximo a 0.
- El valor del índice está en el intervalo [-1, 1]. Un mal agrupamiento obtendrá un valor negativo. El 1 indica una correspondencia exacta.
- No hace suposiciones sobre la estructura de los grupos.

Desventajas:

- Hay que conocer la clasificación real.

4.2.1.3. Coeficiente de Jaccard

Partiendo de las definiciones de C, R, a y b del apartado anterior, y añadiendo

- c, el número de parejas de elementos que están en el mismo grupo en C y en diferente grupo en R,

se define el coeficiente de Jaccard como

$$J = \frac{a}{a + b + c}$$

El valor está entre 0 y 1. Cuanto mayor es el valor del índice, mayor coincidencia con el agrupamiento de

4.2.1.4. Índice de Fowlkes y Mallows

$$FM = \sqrt{\frac{a}{a+b}\frac{a}{a+c}}$$

Cuanto mayor es el valor del índice, mayor coincidencia con el agrupamiento de referencia. Funciona bien para datos con ruido.

4.2.1 5. Medida-F

La medida-F (también conocida como medida-F1, F1-measure o F1-score) es una métrica habitual en problemas de clasificación, reconocimiento de patrones y recuperación de información. Se calcula a partir de otros dos, la precisión y la exhaustividad (precision y recall).

$$Precisión = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ positivos}$$

$$Exhaustividad = \frac{verdaderos\ positivos}{verdaderos\ positivos + falsos\ negativos}$$

$$Medida - F = 2 \cdot \frac{Precisión \cdot Exhaustividad}{Precisión + Exhaustividad}$$

Hay una medida-F (y una precisión y una exhaustividad) para cada uno de los grupos/categorías. Para explicar los elementos de las fórmulas, llamaremos C al agrupamiento que se quiere evaluar y R al de referencia. Para un grupo dado, se habla de verdadero positivo cuando un elemento se asigna tanto en C como en R. Si, por el contrario, en C el elemento está asignado al grupo pero en R no, se trata de un falso positivo. Finalmente, si en C no se asigna el elemento al grupo pero en F sí, es un falso negativo.

4.2.1.6. Índices de información mutua

Hay dos índices de este tipo: el de información mutua normalizada *(Normalized Mutual Information*, NMI) y el de información mutua ajustada (*Adjusted Mutual Information*, AMI). El NMI está más extendido y el AMI es una versión más reciente ajustada respecto al azar. Ambos ignoran las permutaciones en las etiquetas y son simétricos.

Partiendo de dos agrupamientos, U y V, se define la entropía de los mismos como:

$$H(U) = \sum_{i=1}^{|U|} P(i) \log\big(P(i)\big)$$

donde P(i) = |Ui|/N es la probabilidad de que un elemento escogido aleatoriamente de U pertenezca a Ui. Igualmente, para V:

$$H(V) = \sum_{i=1}^{|V|} P'(i) \log\big(P'(i)\big)$$

donde P'(i) = |Vi|/N es la probabilidad de que un elemento escogido aleatoriamente de V pertenezca a Vi. El índice de información mutua (sin normalizar) es

$$MI(U,V) = \sum_{i=1}^{|U|} \sum_{j=1}^{|V|} P(i,j) log\left(\frac{P(i,j)}{P(i)P'(j)}\right)$$

donde P(i, j) = |Ui∩Vj|/N es la probabilidad de que un elemento escogido aleatoriamente pertenezca tanto a Ui como a Vj.

El índice de información mutua normalizado es

$$NMI(U,V) = \frac{MI(U,V)}{\sqrt{H(U)H(V)}}$$

Ninguno de estos índices está ajustado respecto al azar. Como en el caso del índice de Rand ajustado, se puede incluir el valor esperado del índice de información mutua y llegar al índice de información mutua ajustado

$$AMI = \frac{MI - E[MI]}{max(H(U), H(V)) - E[MI]}$$

Ventajas

- Un etiquetado aleatorio uniforme obtendrá un valor próximo a 0.
- El valor del índice está en el intervalo [0, 1]. Un mal agrupamiento obtendrá un valor próximo a 0. Un valor de exactamente 0 indica etiquetados puramente independientes y uno de exactamente 1 una correspondencia exacta.
- No hace suposiciones sobre la estructura de los grupos.

Desventajas:

- Hay que conocer la clasificación real.
- Los índices de información mutua e información muta normalizada no está ajustados contra el azar.

4.2.2. Criterios internos

4.2.2.1. Índice de Dunn

Identifica como un buen agrupamiento aquel con grupos compactos y bien separados entre sí. Es decir, grupos con pocas varianza cuyas medias son suficientemente diferentes en comparación con la varianza entre grupos.

El tamaño o diámetro de un vector se puede calcular de cualquiera de estas formas

- $\Delta_i = max_{x,y \in C_i} d(x,y)$, usando la distancia máxima.
- $\Delta_i = \frac{1}{|C_i||C_i-1|} \sum_{x,y \in C_i,} d(x,y)$, que calcula la media entre todos los pares posibles.
- $\Delta_i = \frac{\sum_{x \in C_i} d(x,\mu)}{|C_i|}$, que utiliza la distancia de todos los puntos con la media.

Igualmente, se pueden utilizar diferente medidas para la distancia entre dos grupos (ver 3. 2. 3.), $\delta(C_i, C_j)$. Combinando una medida de cada tipo se obtiene la familia de índices Dunn.

$$DI_m = \frac{min_{1 \le i < j \le m} \delta(C_i, C_j)}{max_{1 \le k \le m} \Delta_k}$$

donde m es el número de grupos. Cuanto mayor sea el índice, mejor es el agrupamiento.

4.2.2.2. Índice Davies-Bouldin

Este índice mide la similitud entre los grupos. En concreto, para todos los grupos mide la similitud con el más similar y se calcula la media. Como la idea es que los grupos deben ser diferentes entre sí, se buscan agrupamientos que minimicen el valor del índice.

Se define [Cárdenas] como

$$DB = \frac{1}{k} \sum_{i=1}^{k} max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

donde k denota el número de grupos, σ_x la distancia media de los elementos de los elementos del grupo x a su centroide y $d(c_i, c_j)$ es la distancia entre los centroides de los grupos i y j. El valor máximo

$$max_{i \neq j} \left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)} \right)$$

representa el peor caso para el grupo i. Para cada grupo i, se busca el grupo j más similar, es decir, aquel grupo para el que el ratio entre la suma de la distancias dentro del grupo y la distancia entre grupos es mayor.

Es posible utilizar otra definición para la distancia entre grupos que la distancia entre sus centroides. Para calcular la distancia entre dos elementos hay que utilizar la misma medida que en el algoritmo de agrupación o no se obtendrán resultados coherentes.

4.2.2.3. Silueta

Para cada elemento i se definen:

- a(i), la distancia media de i con los demás elementos del mismo grupo. Se puede interpretar como una medida de lo bien asignado que está un elemento a su grupo. Un valor pequeño indica una buena asignación.
- b(i), la distancia media entre i y los puntos del grupo más cercano.

El coeficiente de silueta para un único elemento i es:

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

El valor de s(i) está acotado entre -1 y +1. Un valor cercano a +1 indica que el elemento está bien asignado. Un valor cercano a -1 significa que estaría mejor asignado al grupo más cercano. Si el valor es cercano a 0, el elemento está en la frontera entre dos grupos.

El coeficiente de silueta de un conjunto de elementos es la media del coeficiente de silueta de cada uno de los elementos. Aplicado a un todo el conjunto de datos se obtiene una medida de la calidad del agrupamiento. Se puede utilizar el coeficiente de silueta para determinar el número óptimo de grupos.

4.2.2.4. Índice de Calinski-Harabasz

El índice se calcula como

$$CH = \frac{traza(S_B)}{traza(S_w)} \cdot \frac{n_p - 1}{n_p - K}$$

donde S_B es la matriz de dispersión entre grupos, S_w la matriz de dispersión interna, n_p el número de muestras asignadas a algún grupos y K el número de grupos. Las matrices de dispersión son

$$\boldsymbol{S_w} = \sum_{k=1}^{K} \sum_{\boldsymbol{x} \epsilon C_k} (\boldsymbol{x} - \boldsymbol{\mu}_k)\,(\boldsymbol{x} - \boldsymbol{\mu}_k)^T$$

$$\boldsymbol{S_B} = \sum_{k=1}^{K} N_K (\boldsymbol{\mu}_k - \boldsymbol{\mu})\,(\boldsymbol{\mu}_k - \boldsymbol{\mu})^T$$

donde N_k es el número de elementos en el grupo C_k, $\boldsymbol{\mu}_k$ es el centroide de C_k y $\boldsymbol{\mu}$ es la media de todo el conjunto.

4.3. Reducción de la dimensionalidad

Un problema común en las tareas de agrupamiento es la llamada 'maldición de la dimensionalidad', que se da cuando los datos tienen un número de dimensiones excesivamente grande. Por ejemplo, cuando se trabaja con documentos de texto es habitual que haya miles de dimensiones. No hace falta señalar la complejidad computacional que esto implica. Además, es de esperar que muchas de esas dimensiones sean poco o nada relevantes de cara a obtener un buen agrupamiento y sirvan solo para causar problemas a los algoritmos.

En este apartado veremos algunas técnicas para seleccionar solo las variables más interesantes y obtener un conjunto de datos más manejable. Las hay supervisadas y no supervisadas. Obviamente estas últimas son las más interesantes para el agrupamiento.

4.3.1. Análisis de las componentes principales

El método de análisis de las componentes principales (PCA, *Principal Components Analysis*) transforma un conjunto de vectores l-dimensional en otro con m dimensiones, $m \leq l$. Estas m dimensiones son ortogonales y son las que explican la mayor parte de la varianza.

La primera componente principal (la primera de esas dimensiones) tiene la mayor varianza posible. La segunda y siguientes se seleccionan de manera que sigan teniendo la mayor varianza posible con la restricción de que sean ortogonales a las componentes anteriores. De esta manera, las componentes están ordenadas según su capacidad para

explicar la variabilidad de los datos. La idea es que usando solo alguna de estas componentes se puede explicar la mayor parte de esta variabialidad.

Se puede interpretar como realizar un ajuste de los datos a un elipsoide de l dimensiones. Si alguno de los ejes del elipsoide es pequeño, la varianza a lo largo de ese eje es pequeña. Si se prescinde de ese eje se pierde poca información.

Las componentes principales son los autovectores de la matriz de covarianza. Al ser autovectores, son ortogonales entre ellas. La primera componente principal será la correspondiente al mayor autovalor, y así sucesivamente.

Una aplicación habitual es la representación de datos multidimensionales. Utilizando las dos primeras componentes principales se pueden visualizar en las dos dimensiones en las que los datos están más separados, de manera que puede ser más fácil identificar posibles grupos. Utilizar dos componentes originales de los datos elegidas al azar puede llevar a que todos los puntos aparezcan más o menos superpuestos o, más en general, a una representación poco útil.

También es útil en modelos de regresión, donde desprenderse de algunas variables puede ayudar a que no se produzca un sobreajuste. Además, si se utilizan datos con ruido (uniformemente distribuido) este será proporcionalmente menos relevante en las primeras componentes principales, ya que su magnitud es mayor.

4.3.3.1. Criterios para elegir el número de componentes

Una cuestión fundamental cuando se hace PCA es cuántas variables hay que seleccionar. Si partimos de vectores de datos con 1000 características ¿el número apropiado es 2, 10 o 100? Se han propuesto varios criterios [Dean]:

- Regla de Kaiser. Utilizar solo las componentes cuyo autovalor asociado sea mayor que uno. Es una opción un tanto arbitraria, ya que incluye componentes con autovalor 1.01 y deja fuera a las de autovalor 0.99.
- Test de ladera (*scree test*). Se utiliza un gráfico en el que en el eje X está el número de componente en el eje Y el autovalor asociado, ordenado de mayo a menor autovalor. En el gráfico normalmente habrá una pendiente pronunciada seguida de un giro y una línea casi horizontal. Se escogen las componentes anteriores al comienzo de la línea horizontal.

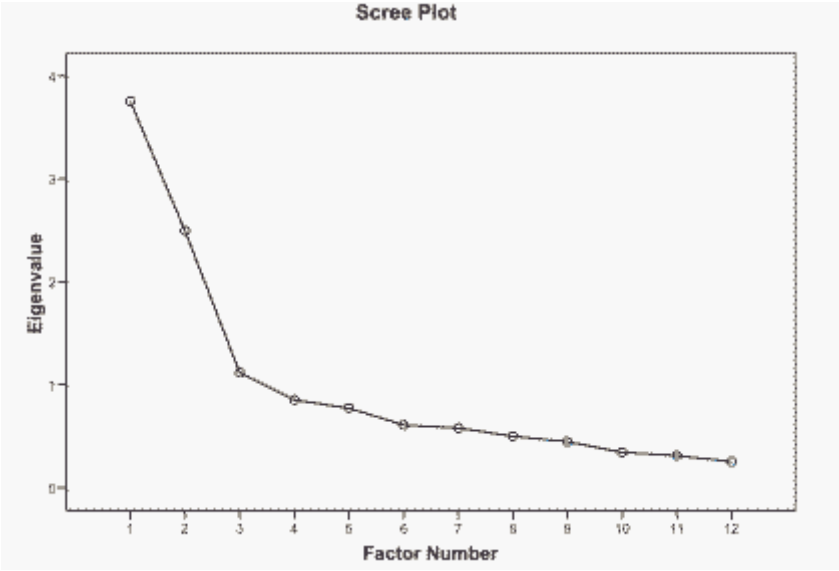

Ilustración 4. 3. Selección de las componentes principales más relevantes. Tomado de [Dean].

- Porcentaje de varianza acumulada. Esta estrategia consiste en quedarse con el menor número de variables que explica cierto nivel varianza acumulada. El nivel concreto depende de la aplicación.

4.3.2. Aglomeración de características (*feature agglomeration*)

Este algoritmo funciona de manera similar al agrupamiento jerárquico. En lugar de unir vectores/grupos parecidos, une características que se comportan de manera similar. Recibe el número de características que se quiere obtener y va agrupando las originales para reducir su número al indicado. Es posible que falle al unir características similares si hay diferencias de escala o tienen propiedades estadísticas diferentes.

4.3.3. Proyecciones aleatorias

Los métodos de proyecciones aleatorias se sustentan en lema de Johnson-Lindenstrauss, que establece que cualquier conjunto de datos puede ser proyectado en un espacio de menos dimensiones de manera que la distancia relativa entre pares de puntos quede en gran medida preservada. Es por tanto posible reducir la dimensionalidad de los datos a la vez que se preserva su estructura.

Dado $0 < \varepsilon < 1$, un conjunto N de puntos en R^l, y un número $l' > 8\ \ln(N)/\ \varepsilon^2$, existe una aplicación lineal f: $R^l \rightarrow R^{l'}$ tal que

$$(1-\varepsilon)\|u-v\|^2 \leq \|f(u)-f(v)\|^2 \leq (1+\varepsilon)\|u-v\|^2$$

para todo u, v ϵ X. El parámetro ε sirve para ajustar la calidad de la proyección. Cuanto más pequeño sea mejor se conservan las distancias relativas y mayor debe ser l'. Con este parámetro se puede balancear la pérdida de información y la compresión obtenida.

La proyección se realiza mediante una matriz P de dimensión l' × l generada aleatoriamente de manera que la longitud de las columnas sea la unidad.

$$X_{l' \times N}^{proy} = P_{l' \times l} X_{l \times N}$$

Crear la matriz aleatoriamente P y realizar la proyección es rápido, O(ll'N), y si la matriz de datos X es dispersa se reduce a O(cl'N), donde c es el número de entradas diferentes de cero por columna.

4.3.3.1. Proyección aleatoria gausiana

Los componentes de la matriz se toman de distribución N(0, 1/l').

4.3.3.2. Proyección aleatoria dispersa

En este caso se usa una matriz aleatoria dispersa. En comparación con la proyección gausiana, ofrece una calidad similar pero requiere menos memoria y permite acelerar los cálculos. Si se define s = 1/densidad, los elementos de la matriz se eligen según

$$\begin{cases} -\sqrt{\frac{s}{l'}} & con\ probabilidad\ \frac{1}{2s} \\ 0\ con\ probabilidad\ 1 - 1/s & \\ +\sqrt{\frac{s}{l'}}\ con\ probabilidad\ \frac{1}{2s} & \end{cases}$$

El valor mínimo de la densidad es $\frac{1}{\sqrt{l'}}$.

4.3.4. Umbral de varianza

Esta es una aproximación muy básica a la reducción de la dimensionalidad. Consiste simplemente en eliminar las características cuya varianza esté por debajo de cierto valor.

5. Aplicación desarrollada

Como parte del proyecto se ha desarrollado una aplicación que para probar algunos de los algoritmos explicados durante la memoria. El objetivo es proporcionar una interfaz gráfica para realizar pruebas con diversas opciones de preprocesamiento, algoritmos de agrupamiento y reducción de la dimensionalidad de manera sencilla y eficiente. La aplicación genera gráficos, realiza la validación de los grupos y permite almacenar los resultados.

Se ha realizado en Python [Python] y utiliza, entre otras librerías, scikit-learn [Scikit], con herramientas y funciones útiles para análisis de datos y nltk (*Natural Language Toolkit*) [Nltk] que proporciona *stemmers* y lematización a través de Wordnet. La interfaz se ha realizado con Qt Designer [Qt].

5.1. Conjunto de datos

El conjunto de datos utilizado es '20 Newsgroups', un conjunto muy utilizado para experimentos de clasificación y agrupamiento de texto. Se puede obtener en [20nNewsgroups]. También está disponible dentro del paquete scikit-learn para Python, que es como se ha usado en la aplicación. Se trata de una recopilación de 20.000 documentos pertenecientes a 20 grupos de noticias. Los grupos tienen aproximadamente la misma cantidad de documentos. No se trata de noticias de prensa, sino de discusiones sobre un tema.

Como ejemplo, un par de documentos del grupo sci.space:

```
This was on "That's Incredible" several years ago. The volume of liquid

the rat had to breath was considerably smaller than what a human would have

to breath, so maybe it is possible for a rat but not a human.
```

```
The other week I saw a TV program about the american space industry and NASA.

It said that in the 60's they developed a rocket that used ions or nuclear

particles for propolsion.
```

The government however, didn't give them $1billion for the developement

of a full scale rocket.

Did anybody see this program?

If not, has anybody heard of the particle propolsion system?

Thanx. 8-)

Los 20 grupos de noticias se muestran en la tabla de abajo. Algunos de ellos están muy relacionados entre sí. Por ejemplo, hay varios relacionados con ordenadores. Uno de ellos sobre hardware para Mac y otro sobre hardware para PC. También hay dos relacionados con el motor, uno de coche y otro de motos. Dos relacionados con deportes, uno para béisbol y otro para hockey.

comp.graphics comp.os.ms-windows.misc comp.sys.ibm.pc.hardware comp.sys.mac.hardware comp.windows.x
rec.autos rec.motorcycles
rec.sport.baseball rec.sport.hockey
sci.crypt sci.electronics sci.med sci.space
talk.politics.misc talk.politics.guns talk.politics.mideast
talk.religion.misc alt.atheism soc.religion.christian
misc.forsale

Tabla 5. 1. Los 20 grupos de noticias de 20Newsgroups.

La aplicación permite escoger dos de estos grupos, crear una matriz DTM a partir de ellos, aplicar algoritmos de reducción de la dimensionalidad y de agrupamiento. Para cada agrupamiento realizado muestra una gráfica con los grupos obtenidos u calcula varios índices de validación.

Las opciones de preprocesamiento disponibles son:

- Eliminar las palabras vacías (ver sección 2.2.3).
- Eliminar las palabras poco habituales o demasiado habituales (ver sección 2.2.3).
- Utilizar el stemming (Porter o Lancaster) o lematización (ver sección 2.2.4).
- Utilizar la ponderación tf-idf o simplemente la frecuencia de los términos (ver sección 2.3).

Los algoritmos de reducción de la dimensionalidad disponibles son:

- PCA (ver sección 4.3.1).
- Aglomeración de características (ver sección 4.3.2).
- Proyecciones aleatorias gaussianas y dispersas (ver sección 4.3.3).
- Umbral de varianza (ver sección 4.3.4).

Los algoritmos de agrupamiento disponibles son:

- k-means (ver sección 3.3.4.1).
- Algoritmo de Ward (ver sección 3.3.3).
- DBSCAN (ver sección 3.3.5.1).
- Agrupamiento espectral (ver sección 3.3.6.2).

Cuando se realiza un agrupamiento se calculan las siguientes métricas de validación:

- Homogeneidad, completitud, medida-V (ver sección 4.2.1.1).
- Índice de Rand ajustado (ver sección 4.2.1.2).
- Información mutua ajustada (ver sección 4.2.1.6).
- Coeficiente de silueta (ver sección 4.2.2.3).

5. 2. Descripción de la interfaz gráfica

La interfaz tiene dos zonas bien diferenciadas. En la superior hay tres pestañas llamadas ‘Crear Matriz’, ‘Reducir’ y ‘Agrupar’. Desde ellas el usuario puede ejecutar las operaciones y algoritmos implementados y modificar los parámetros pertinentes. En la

zona inferior está la zona de salida, donde se muestran los resultados. La ilustración 5.1 muestra un pantallazo de la aplicación sobre el que se han señalado las dos zonas.

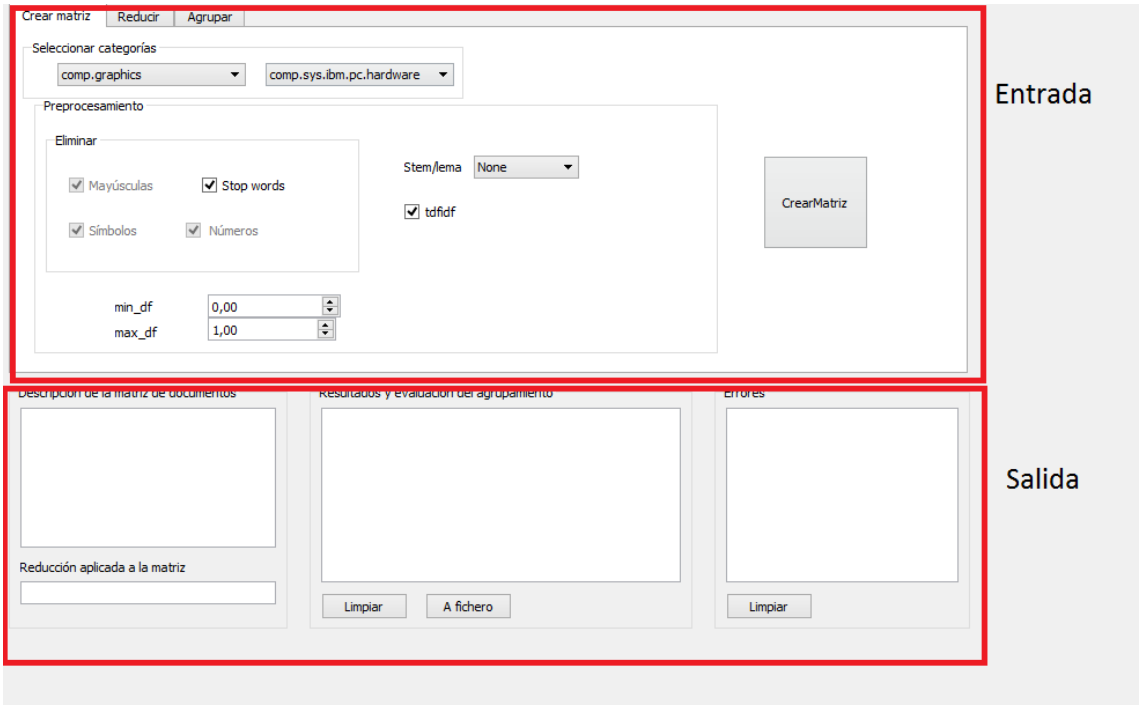

Ilustración 5. 1. Zonas de entrada y salida de la aplicación.

5. 2. 1. Pestaña 'Crear Matriz'

Desde esta pestaña se seleccionan los dos grupos de noticias que se van a utilizar y las opciones de preprocesamiento. Tiene los siguientes elementos.

- Caja 'Seleccionar categorías'. Aquí hay dos listas desplegables para seleccionar los dos grupos de noticias que se van a utilizar para crear la matriz.
- Caja 'Preprocesamiento'. Aquí se seleccionan las opciones de preprocesamiento para los documentos.
    - Las casillas de verificación 'Mayúsculas', 'Símbolos' y 'Números' están siempre seleccionadas. Se muestran con finalidad puramente informativa, para que el usuario tenga claro que se están realizando esos cambios.
    - La lista desplegable 'Stem/lema' sirve para realizar *stemming* o lematización. Tiene cuatro opciones: 'None', si no se quiere realizar *stemming* ni lematización; 'Porter' para utilizar el *stemmer* de Porter; 'Lancaster' para utilizar el *stemmer* de Lancaster; 'Wordnet' para realizar la lematización usando Wordnet. Estos conceptos se explican en la sección 2.2.4.

  - La casilla de verificación tdfidf sirve para crear la matriz usando la ponderación tf-tdf. Se explica en la sección 2.3. Si no está marcada se usa simplemente la frecuencia de los términos en cada documento.
  - Con 'max_df' se eliminan los términos que aparezcan en una proporción de documentos mayor que la indicada.
  - Con 'min_df' se eliminan los términos que aparezcan en una proporción de documentos menor que la indicada.
- Botón 'CrearMatriz'. Al pulsar este botón se crea la matriz DTM con las categorías de documentos seleccionadas en las listas desplegable y las opciones de preprocesamiento indicadas. Las cajas de texto de la zona de salida se actualizan en consonancia.

5. 2. 2. Pestaña 'Reducir'

Desde esta pestaña se pueden ejecutar algoritmos de reducción de la dimensionalidad sobre la matriz que esté cargada en memoria. Las reducciones se ejecutan siempre sobre la matriz original. Si se hacen varias consecutivas, las realizadas previamente no cuentan. Esto es así para que no sea necesario volver a crear la matriz, que lleva un cierto tiempo, cada vez que se quiere probar una nueva reducción.

La ilustración 5. 2 muestra esta pestaña. Contiene cuatro grupos diferenciados, cada uno para un algoritmo. Veámoslos uno a uno:

- Grupo PCA. Al pulsa el botón 'Reducir' de este grupo se ejecuta el algoritmo PCA sobre los datos cargados. El número de componente principales que se quiere obtener se indica en la caja etiquetada como 'N. componentes'. El algoritmo se explica en la sección 4.3.1.
- Grupo 'Feature agglomeration'. Al pulsa el botón 'Reducir' de este grupo se ejecuta el algoritmo de aglomeración de características sobre los datos cargados. El número de dimesiones que se quiere obtener se indica en la caja etiquetada como 'Número de clusters'. El algoritmo se explica en la sección 4.3.2.
- Grupo 'Proyecciones aleatorias'. En este grupo hay dos botones, según el tipo de proyección que se desee realizar. Ambos algoritmos requieren el parámetro *epsilon*. Se explican en la sección 4.3.3.
- Grupo 'Umbral de varianza'. Al pulsa el botón 'Reducir' de este grupo se ejecuta el algoritmo de umbral de varianza sobre los datos cargados. El umbral de

varianza se indica en la caja correspondiente. El algoritmo se explica en la sección 4.3.4.

- Botón ‘Eliminar reducción’. Se elimina la reducción actualmente aplicada y se vuelve a tener la matriz DTM original cargada en memoria.

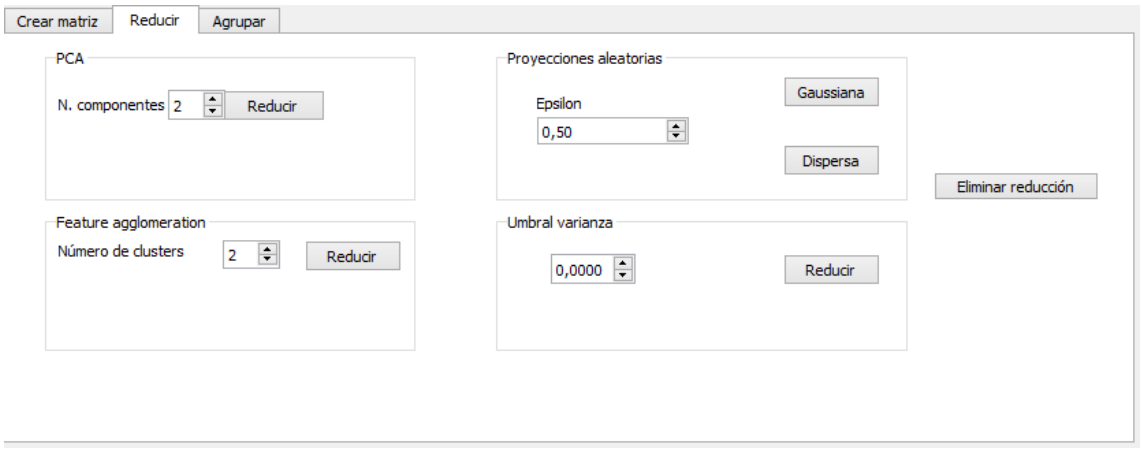

Ilustración 5. 2. Pestaña 'Reducir'.

5. 2. 3. Pestaña ‘Agrupar’

Desde esta pestaña se ejecutan los algoritmos de agrupamiento. Tiene cuatro grupos diferenciados, uno por cada algoritmo disponible. Cuando se pulsa el botón ‘Agrupar’ de uno de los grupos, se ejecuta el algoritmo, se muestra un gráfico con los grupos obtenidos y los índices de validación en la caja de texto correspondiente, en la zona de salida.

- Grupo k-means. Hay que indicar el número de grupos que se quiere obtener y cuantas veces se inicializa el algoritmo. El algoritmo se explica en la sección 3.3.4.1.
- Grupo jerárquico. Ejecuta el algoritmo de Ward. Hay que indicar el número de grupos que se quiere obtener. El algoritmo se explica en la sección 3.3.3.
- DBSCAN. Hay que indicar los parámetros épsilon y n_samples. El algoritmo se explica en la sección 3.3.5.1.
- Agrupamiento espectral. Hay que indicar el número de grupos que se quiere obtener y cuantas veces se inicializa el algoritmo. El algoritmo se explica en la sección 3.3.6.2.

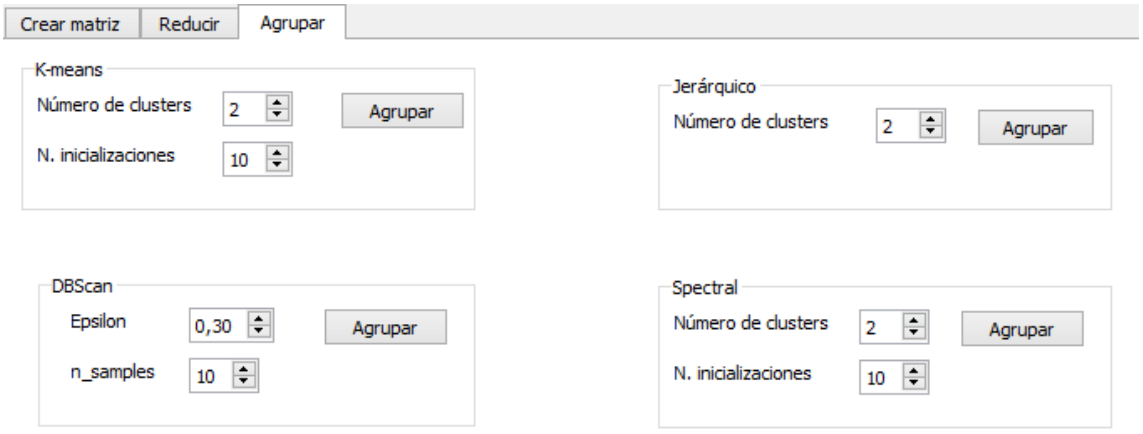

Ilustración 5. 3. Pestaña agrupar.

5. 2. 4. Zona de salida

La ilustración 5. 4 muestra en detalle la parte inferior de la interfaz.

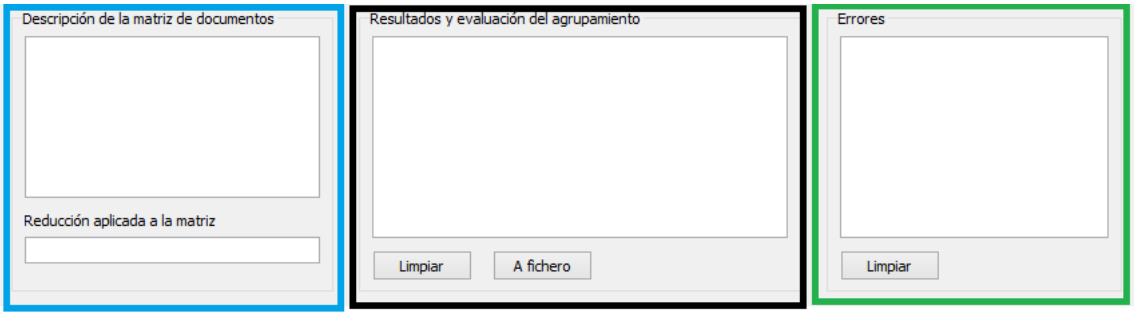

Ilustración 5. 4. Zona de salida.

- A la izquierda (recuadro azul) hay dos cajas de texto con la información de la matriz cargada en memoria actualmente (caja superior) y la reducción de la dimensionalidad que se le ha aplicado, si es que se ha aplicado alguna (caja inferior).

  Cada vez que se pulsa el botón 'CrearMatriz', el contenido de la caja superior se sustituye por la información de la matriz recién creada, y el de la inferior se borra. Cuando se realiza una reducción, el contenido de la caja inferior se sustituye por la información de la nueva reducción.

  De esta manera se sabe sobre qué datos se ejecutan los algoritmos de agrupamiento.
- En el centro (caja negra) está la caja en la que se almacenan los resultados que se van obteniendo. Cuando se crea una nueva matriz o se realiza una reducción, la información correspondiente se añade a esta caja de texto. Igualmente, cuando se

ejecuta un algoritmo de agrupamiento, la validación de los grupos obtenidos se añade a esta caja.

En todos los casos la información nueva se añade a la caja sin borrar la anterior. De esta manera se va generando un resumen de la sesión de trabajo. El usuario puede guardar una copia del resumen pulsando el botón 'A fichero', que abre un cuadro de diálogo para que elija la ruta.

En cualquier momento, el usuario puede borrar el contenido de la caja pulsando 'Limpiar' o editarlo manualmente.

- A la derecha (caja verde) hay un espacio para mostrar los errores que se puedan producir. Por ejemplo, que no haya memoria suficiente para ejecutar cierto algoritmo. Cada mensaje de error se añade a los anteriores.

  En cualquier momento, el usuario puede borrar el contenido de la caja pulsando 'Limpiar' o editarlo manualmente.

5. 3. Ciclo de trabajo

La aplicación está pensada para trabajar de la siguiente forma:

1. El primer paso es crear una matriz escogiendo dos categorías y las opciones de preprocesamiento que se deseen.
2. Se aplica, si desea una reducción de dimensionalidad.
3. Se realizan uno o más agrupamientos.
4. Se repiten los pasos 2 y 3 con todas las opciones que se consideren necesarias.
5. Si se desea, se vuelve al paso 1 para cambiar las categorías o las opciones de procesamiento.
6. Cuando se hayan realizado todos los experimentos deseados se guardan las gráficas y el resumen de texto, si se desea. También es posible ir guardando los resultados antes.

Cada vez que se carga un nuevo conjunto de datos, se realiza una reducción de la dimensionalidad o un agrupamiento las operaciones realizadas y los resultados obtenidos se añaden a la caja de texto. De esta manera la caja sirve para reconstruir la sesión de trabajo sin perder información.

6. Experimentos realizados

6. 1. Descripción del expermiento

El objetivo del experimento es comparar algoritmos de agrupamiento, de reducción de la dimensionalidad y opciones de preprocesamiento.

Se toman documentos de dos de las categorías del conjunto *20newsgroups* y se agrupan usando diferentes opciones. Los agrupamientos resultantes se evalúan usando como referencia la categoría de la que se obtuvieron. También se utiliza el coeficiente de silueta como métrica interna.

6.1.1. Conjunto de datos

Se han utilizado las categorías 'sci.electronics' (393 documentos) y 'soc.religion.christian' (391 documentos), es decir electrónica y religión cristiana. En principio son categorías bien diferenciadas y por tanto se pueden esperar buenos resultados. Se utiliza solo un subconjunto de algo menos de 400 elementos en cada categoría para mantener los tiempos de ejecución bajos.

En todos los casos, los documentos se preprocesan mediante:

- Eliminación de todos los caracteres que no sean letras.
- Paso a minúsculas.
- Eliminación de las palabras más habituales (*stop words*).
- tf–idf.

6.1.2. Pruebas realizadas

Se han utilizado los siguientes algoritmos

- K-means (sección 3.3.4.1).
- Ward (sección 3.3.3).
- Agrupamiento espectral (sección 3.3.6.2).

Para cada algoritmo se prueban:

- Cuatro opciones de preprocesamiento adicionales (sección 2.2.4):
  - Ninguna operación adicional.
  - Porter *stemming*.
  - Lancaster *stemming*.
  - Lematización con Wordnet.

- Tres opciones de reducción de la dimensionalidad (sección 4.3.1):
    - No reducir la dimensionalidad.
    - PCA de 10 componentes.
    - PCA de 2 componentes.

Por lo tanto, para cada algoritmo hay 12 combinaciones.

Las siguientes ilustraciones muestran los conjuntos de datos usando las etiquetas de referencia para las cuatro opciones de preprocesamiento adicionales. Se representa en dos dimensiones mediante el análisis de componente principales (de dos componentes). En los cuatro casos se observa que hay una zona en la que se mezclan documentos de ambas categorías, sin que haya una zona de baja densidad de puntos entre ellas. Fuera de esa zona las categorías apenas se mezclan.

Sin procesamiento adicional:

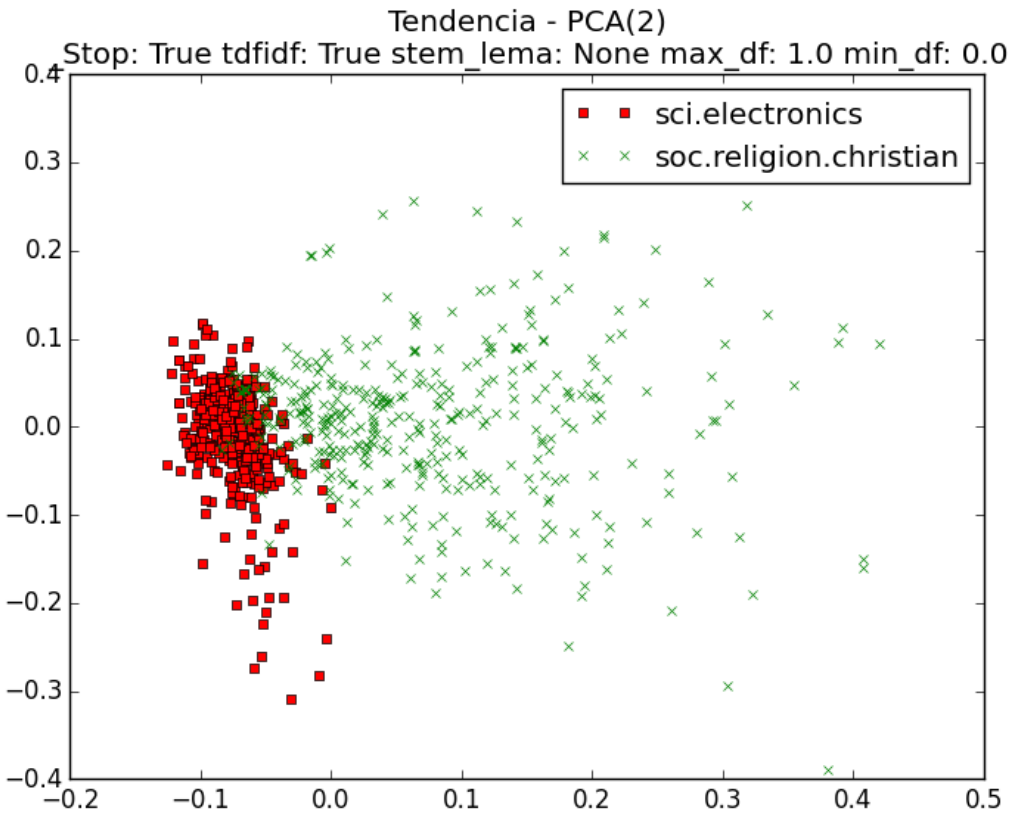

Ilustración 6. 1. Documentos con etiquetas de referencia (sin procesamientos adicional).

Con el *stemmer* de Porter:

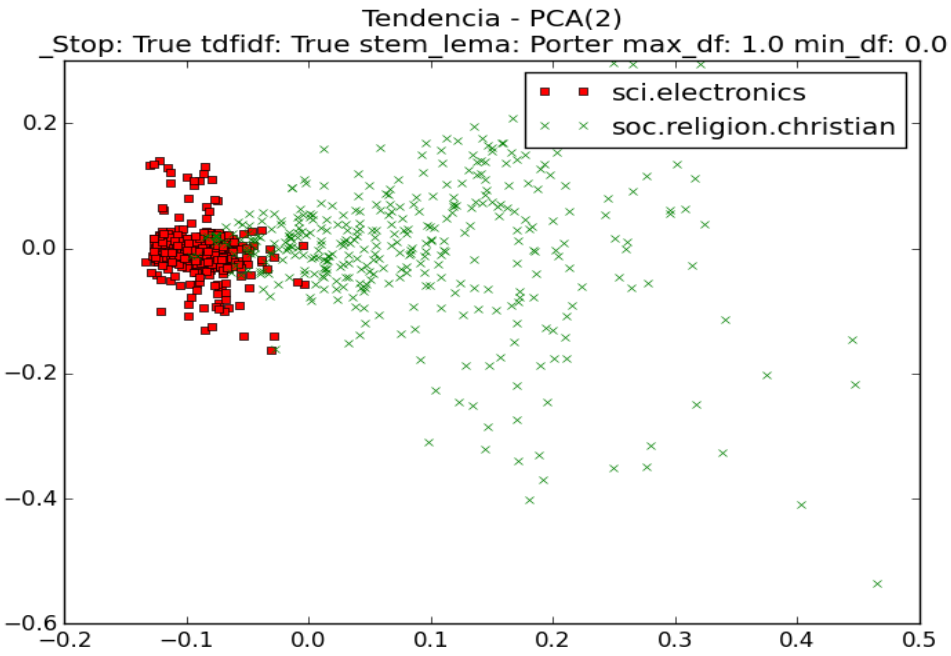

Ilustración 6. 2.Documentos con etiquetas de referencia (Porter).

Con el *stemmer* de Lancaster:

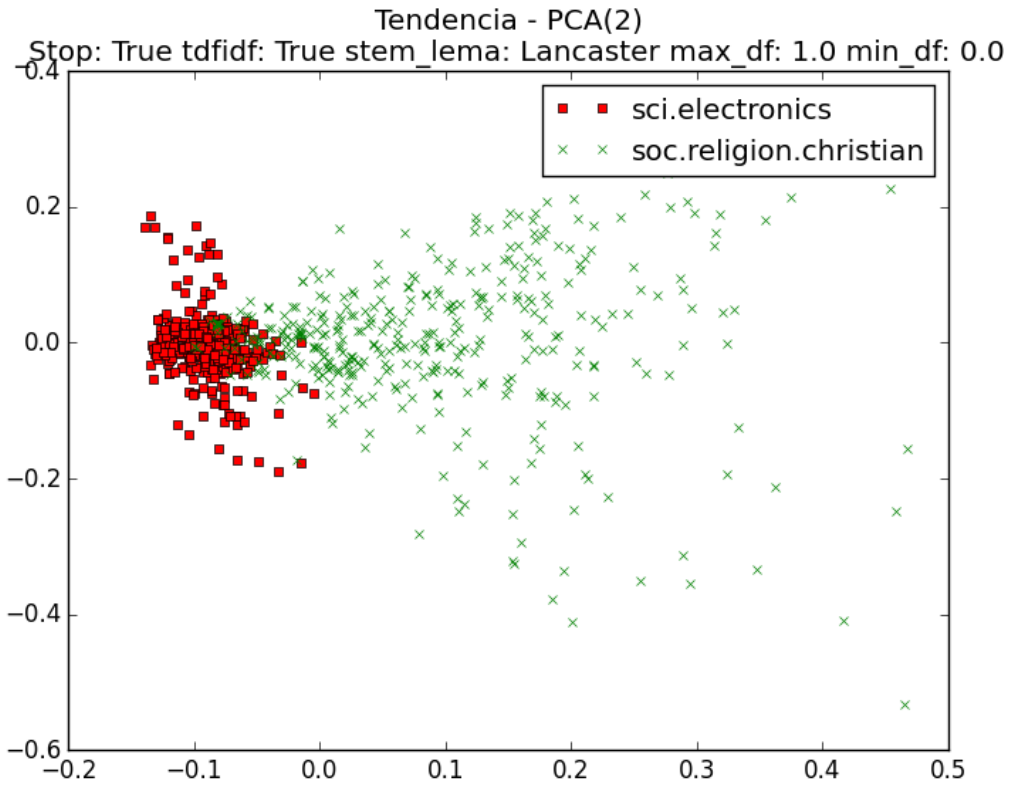

Ilustración 6. 3.Documentos con etiquetas de referencia (Lancaster).

Con lematización usando Wordnet:

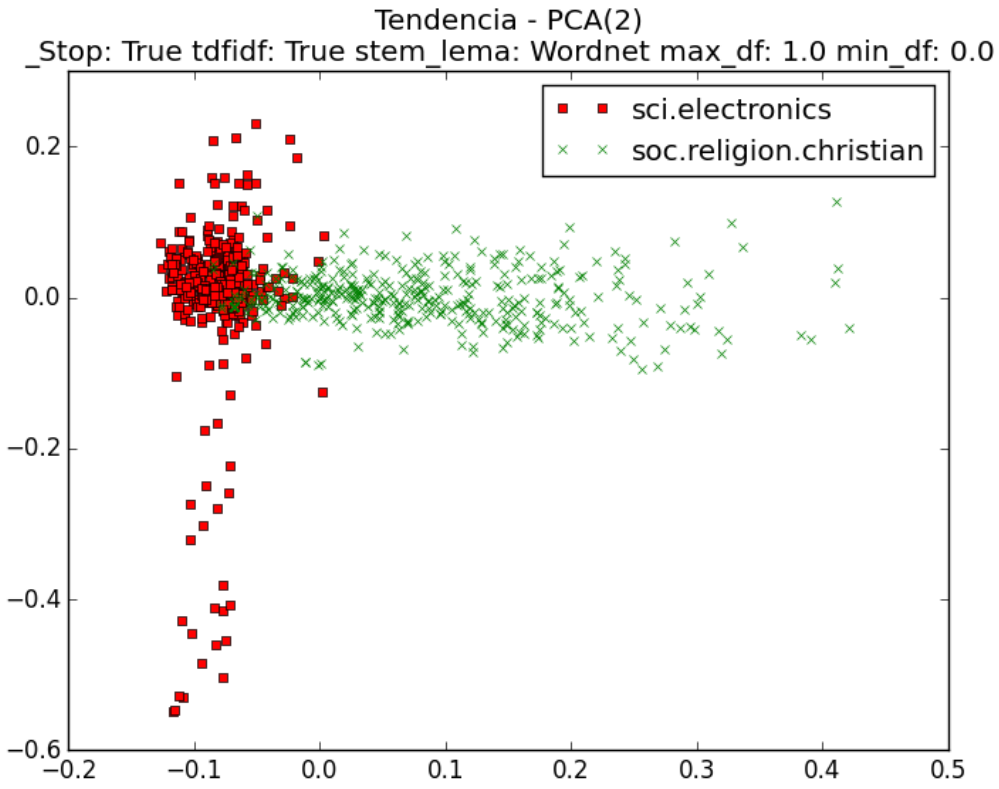

Ilustración 6. 4.Documentos con etiquetas de referencia (Wordnet).

6.1.3. Índices de evaluación

Externos:

- Completitud.
- Homogeneidad.
- Medida-V.
- Índice de Rand ajustado.
- Índice de información mutua ajustado.

Interno:

- Coeficiente de silueta.

Todos estos índices se explican en la sección 4.2.

6. 2. Resultados obtenidos

Las tablas 6.1, 6.4 y 6.7 muestran los resultados para los algoritmos k-means, Ward y agrupamiento espectral, respectivamente. Empezaremos analizando los resultados de cada algoritmo.

6.2.1. k-means

En la siguiente tabla se recogen los resultados del algoritmo k-means. Se ejecutó con 20 reinicios. Para la validación externa, los mejores resultados se obtienen para la combinación de Wordnet y no aplicar ninguna reducción. Para el coeficiente de silueta todas las opciones con Lancaster y Porter son igual de buenas, y además no hay mucha diferencia respecto a las otras combinaciones.

k-means Preproc.	Reducción	Homogeneidad	Completitud	Medida-V	ARI	AMI	Coeficiente Silueta
Ninguna	*Sin*	0.391	0.444	0.416	0.351	0.390	0.011
	PCA(10)	0.327	0.394	0.357	0.267	0.326	0.011
	PCA(2)	0.327	0.394	0.357	0.267	0.326	0.011
Porter	*Sin*	0.416	0.464	0.439	0.385	0.416	**0.013**
	PCA(10)	0.393	0.446	0.418	0.354	0.393	**0.013**
	PCA(2)	0.360	0.420	0.388	0.310	0.360	**0.013**
Lancaster	*Sin*	0.423	0.470	0.445	0.394	0.423	**0.013**
	PCA(10)	0.375	0.432	0.402	0.330	0.375	**0.013**
	PCA(2)	0.373	0.430	0.4	0.327	0.373	**0.013**
Wordnet	*Sin*	**0.458**	**0.497**	**0.477**	**0.440**	**0.457**	0.012
	PCA(10)	0.391	0.444	0.416	0.351	0.390	0.012
	PCA(2)	0.369	0.427	0.396	0.322	0.368	0.012

Tabla 6. 1. Resultados para el algoritmo k-means. Los mejores resultados están en negrita.

Por lo que se refiere a los índices externos, la reducción de la dimensionalidad afecta negativamente a los resultados del algoritmo. En todas las opciones de preprocesamiento es mejor no reducir la dimensionalidad que reducirla a diez componentes, y esto a su vez es mejor que reducirla a dos (ver tabla de detalle 6.3).

La imagen 6.5. muestra el agrupamiento de referencia (arriba) y el mejor que se ha obtenido con este algoritmo (abajo). En la parte central, que es la complicada, el algoritmo no hace un buen trabajo y sobre representa la categoría de electrónica. En las zonas en las que no hay mezcla los resultados son mejores.

Hay que resaltar algo válido para todos los gráficos de agrupamiento. Ya se comentó que no hay que tener en cuenta el valor absoluto de las etiquetas (colores) resultantes de un algoritmo de agrupamiento. De cualquier manera y para facilitar la comparación rápida los colores se han escogido teniendo en cuenta las etiquetas de referencia.

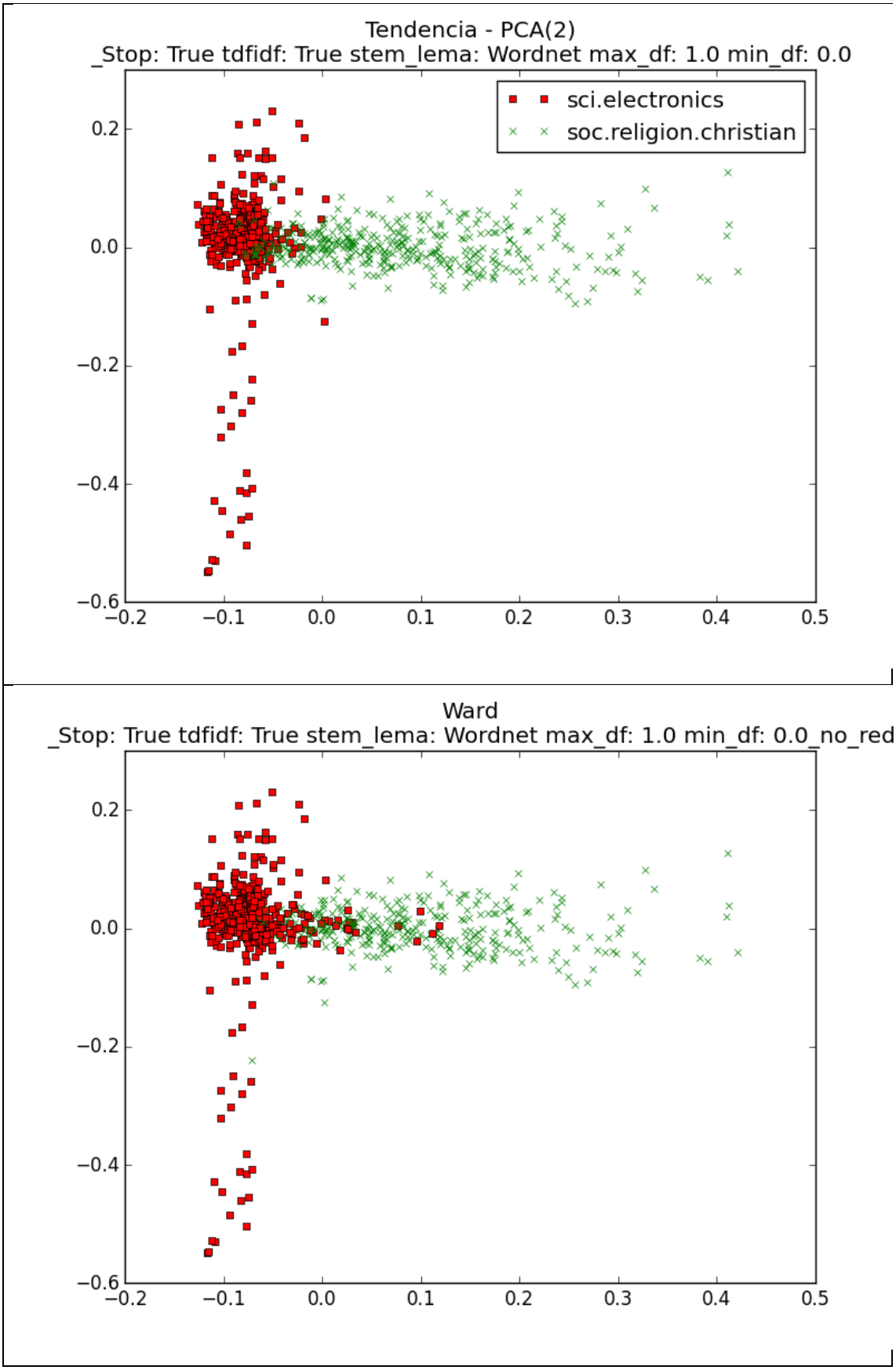

Ilustración 6. 5. Mejor agrupamiento generado por k-means (abajo) y clasificación de referencia (arriba).

La siguiente tabla es un detalle de la anterior, muestra los mejores resultados obtenidos según la reducción aplicada. Entre paréntesis está la opción de preprocesamiento con que se obtuvo. Es interesante que cambia en los tres casos.

	Medida-V	ARI	AMI
Sin reducción (Wordnet)	**0.477**	**0.440**	**0.457**
PCA 10 (Porter)	0.418	0.354	0.393
PCA 2 (Lancaster)	0.4	0.327	0.373

Tabla 6. 2. Mejor resultado para cada reducción (k-means).

La siguiente tabla también es un detalle de la 6.1., muestra los mejores resultados obtenidos según la opción de preprocesamiento aplicada. Entre paréntesis está la opción de reducción de la dimensionalidad con que se obtuvo.

	Medida-V	ARI	AMI
Sin prepro. (sin reducción)	0.416	0.351	0.390
Porter (sin reducción)	0.439	0.385	0.416
Lancaster (sin reducción)	0.445	0.394	0.423
Wordnet (sin reducción)	**0.477**	**0.440**	**0.457**

Tabla 6. 3. Mejor resultado para cada opción de preprocesamiento (k-means).

6.2.2. Ward

En la siguiente tabla se recogen los resultados del algoritmo de Ward. Para las métricas externas los mejores resultados se obtienen con el *stemmer* de Porter y la reducción a diez componentes. Esta opción tiene un coeficiente de silueta muy bajo. Por otro lado, el mejor coeficiente de silueta se obtiene con *stemmer* de Porter y la reducción a dos componentes, que tiene unos valores muy bajos en los índices externos.

Ward Preproc.	Reducción	Homogeneidad	Completitud	Medida-V	ARI	AMI	Coeficiente Silueta
Ninguna	*Sin*	0.546	0.548	0.547	0.64	0.545	0.008
	PCA(10)	0.634	0.645	0.639	0.685	0.633	0.201
	PCA(2)	0.483	0.484	0.483	0.584	0.482	0.394
Porter	*Sin*	0.563	0.571	0.567	0.640	0.562	0.011
	PCA(10)	**0.703**	**0.705**	**0.704**	**0.790**	**0.703**	0.177
	PCA(2)	0.265	0.347	0.300	0.190	0.264	**0.575**
Lancaster	*Sin*	0.65	0.657	0.653	0.715	0.650	0.012
	PCA(10)	0.628	0.639	0.634	0.685	0.628	0.199
	PCA(2)	0.380	0.435	0.406	0.336	0.373	0.535
Wordnet	*Sin*	0.549	0.559	0.554	0.616	0.548	0.0102
	PCA(10)	0.676	0.682	0.679	0.750	0.676	0.183
	PCA(2)	0.194	0.295	0.234	0.112	0.194	0.539

Tabla 6. 4.Resultados para el algoritmo de Ward. Los mejores resultados están en negrita.

Igual que con k-means, la reducción a dos dimensiones es la peor opción en todos los casos. La diferencia está en que, salvo para el *stemmer* de Lancaster, la reducción a diez dimensiones es mejor que no hacer ninguna reducción, con una mejora en torno al 10% en los índices. En el caso de Lancaster la opción de no reducción es solo ligeramente mejor.

La siguiente imagen muestra el agrupamiento de referencia (arriba) y el mejor que se ha obtenido con este algoritmo (abajo). También es el mejor de los tres algoritmos. A simple vista se puede observar que obtiene un alto grado de coincidencia, incluso en la zona en que las categorías se mezclan.

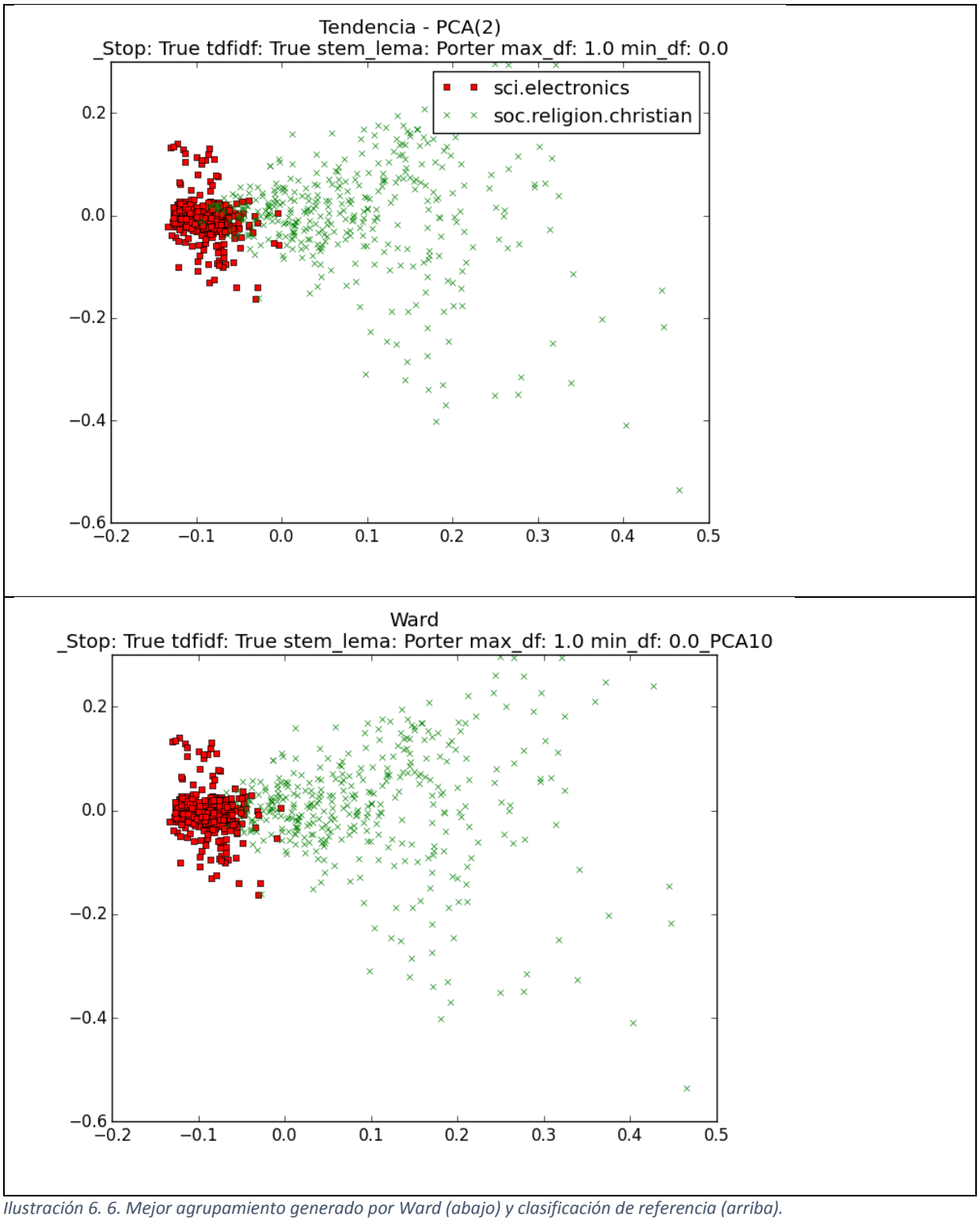

Ilustración 6. 6. Mejor agrupamiento generado por Ward (abajo) y clasificación de referencia (arriba).

La siguiente tabla es un detalle de la anterior, muestra los mejores resultados obtenidos según la reducción aplicada. Entre paréntesis está la opción de preprocesamiento con que se obtuvo. Si se usa la reducción a dos componentes, el mejor resultado se obtiene sin utilizar opciones de preprocesamiento, en lugar de con el *stemmer* de Porter como en los

otros casos. En el caso de la reducción a dos componentes los resultados son notablemente peores.

	Medida-V	ARI	AMI
Sin reducción (Porter)	0.653	0.715	0.650
PCA 10 (Porter)	**0.704**	**0.790**	**0.703**
PCA 2 (sin preproc.)	0.483	0.584	0.482

Tabla 6. 5. Mejor resultado para cada reducción (Ward).

La siguiente tabla también es un detalle de la 6.4, muestra los mejores resultados obtenidos según la opción de preprocesado aplicada. Entre paréntesis está la opción de reducción de la dimensionalidad con que se obtuvo.

	Medida-V	ARI	AMI
Sin prepro. (PCA 10)	0.639	0.685	0.633
Porter (PCA 10)	**0.704**	**0.790**	**0.703**
Lancaster (sin proc.)	0.653	0.715	0.650
Wordnet (PCA 10)	0.679	0.750	0.676

Tabla 6. 6.Mejor resultado para cada opción de preprocesamiento (Ward).

6.2.3. Agrupamiento espectral

En la siguiente tabla se recogen los resultados del agrupamiento espectral. Los mejores resultados en los índices externos se obtienen con el *stemmer* de Porter y la reducción a diez componentes. La reducción a diez componentes es la mejor opción en todos los casos. El coeficiente de silueta apenas cambia en todos los experimentos.

Spectral	Reducción	Homogeneidad	Completitud	Medida-V	ARI	AMI	Coeficiente silueta
Ninguna	*Sin*	0.319	0.388	0.350	0.256	0.318	0.011
	PCA(10)	0.350	0.412	0.378	0.296	0.349	0.011
	PCA(2)	0.327	0.3934	0.357	0.267	0.326	0.011
Porter	*Sin*	0.335	0.400	0.365	0.277	0.334	0.012
	PCA(10)	**0.412**	**0.460**	**0.435**	**0.378**	**0.411**	**0.013**
	PCA(2)	0.367	0.425	0.394	0.319	0.366	**0.013**
Lancaster	*Sin*	0.333	0.399	0.363	0.275	0.332	**0.013**
	PCA(10)	0.391	0.444	0.416	0.351	0.390	**0.013**

	PCA(2)	0.380	0.435	0.406	0.336	0.379	**0.013**
Wordnet	*Sin*	0.345	0.408	0.374	0.291	0.345	0.012
	PCA(10)	0.384	0.439	0.410	0.342	0.384	0.012
	PCA(2)	0.378	0.433	0.404	0.333	0.377	0.012

Tabla 6. 7. Resultados para el agrupamiento espectral. Los mejores resultados están en negrita.

La siguiente tabla es un detalle de la anterior, muestra los mejores resultados obtenidos según la reducción aplicada. Entre paréntesis está la opción de preprocesamiento con que se obtuvo. Es interesante que cambia en los tres casos.

	Medida-V	ARI	AMI
Sin reducción (Wordnet)	0.374	0.291	0.345
PCA 10 (Lancaster)	**0.435**	**0.378**	**0.411**
PCA 2 (sin preproc.)	0.406	0.336	0.379

Tabla 6. 8. Mejor resultado para cada reducción (espectral).

La siguiente imagen muestra el agrupamiento de referencia (arriba) y el mejor que se ha obtenido con este algoritmo (abajo). Los resultados son peores que en los otros dos casos. La diferencia con el algoritmo de Ward salta a la vista. Con k-means es menos evidente pero también se puede observar.

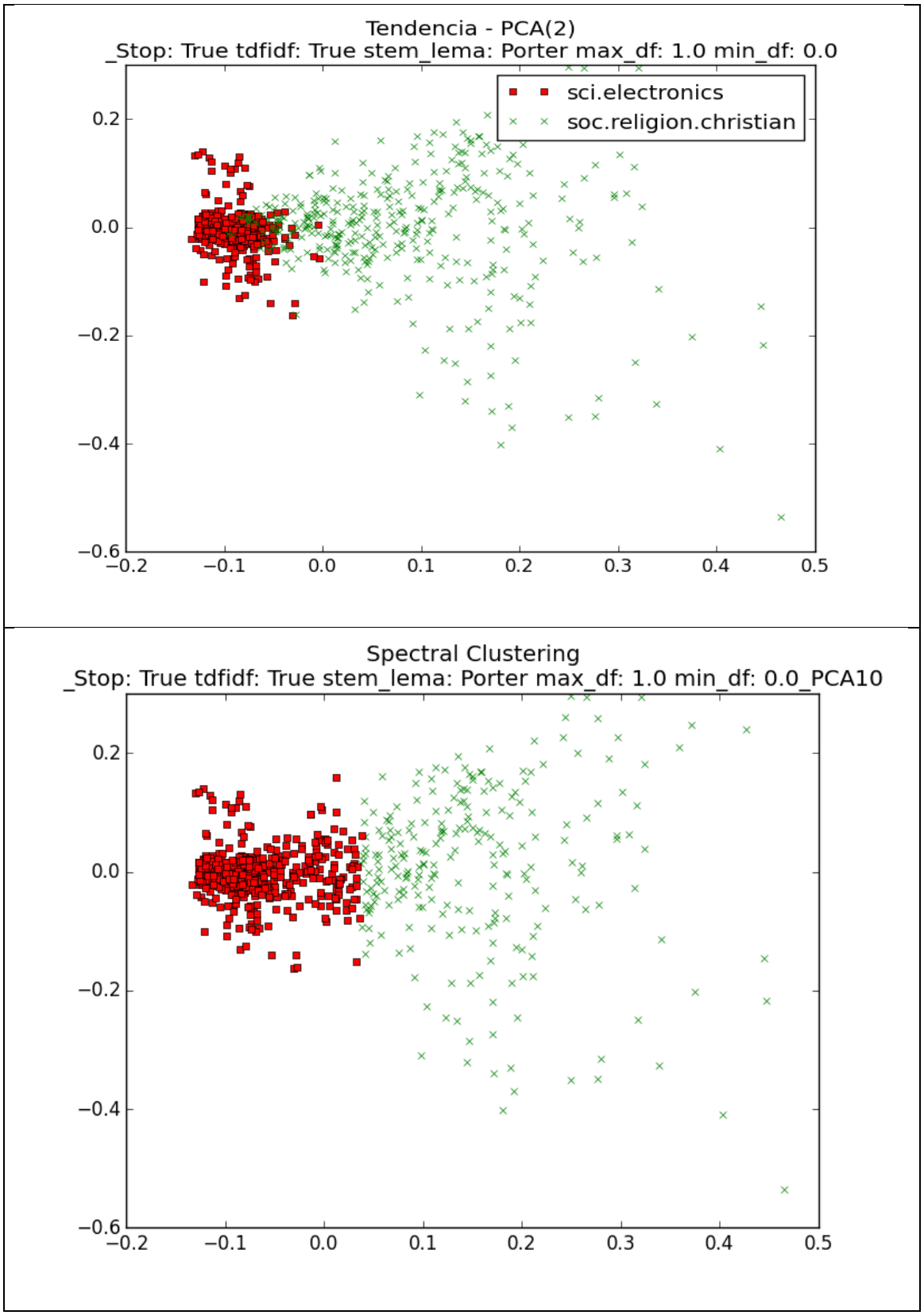

Ilustración 6. 7. Mejor agrupamiento generado por agrupamiento espectral (abajo) y clasificación de referencia (arriba).

La siguiente tabla es un detalle de la 6.7, muestra los mejores resultados obtenidos según la opción de preprocesado aplicada. Entre paréntesis está la opción de reducción de la dimensionalidad con que se obtuvo. Los resultados de ambos *stemmers* y lematización son muy similares.

	Medida-V	ARI	AMI
Sin prepro. (PCA 10)	0.378	0.296	0.349
Porter (PCA 10)	**0.435**	**0.378**	**0.411**
Lancaster (sin proc.)	0.416	0.351	0.390
Wordnet (PCA 10)	0.410	0.342	0.384

Tabla 6. 9.Mejor resultado para cada opción de preprocesamiento (espectral).

6.2.4. Conclusiones

- Los mejores resultados se obtienen con el algoritmo de Ward. La diferencia con los otros algoritmos es importante. La siguiente tabla muestra los mejores resultados para cada algoritmo.

	Medida-V	ARI	AMI
k-means	**0.477**	**0.440**	**0.457**
Ward	**0.704**	**0.790**	**0.703**
Espectral	**0.435**	**0.378**	**0.411**

Tabla 6. 10.Mejor resultado para cada algoritmo.

- El *stemmer* de Porter y la reducción a diez componentes principales es la mejor combinación para el algoritmo de Ward y el agrupamiento espectral.
- La reducción a dos componentes sacrifica demasiada información. Con diez componentes se obtienen los mejores resultados en los algoritmos de Ward y agrupamiento espectral. Para k-means, se obtienen mejores resultados sin reducción; queda por determinar si algún punto intermedio mejora los resultados como en los otros algoritmos.
- El preprocesamiento mejora generalmente los resultados. En ningún algoritmo se ha alcanzado la mejor solución sin utilizar *stemming* o lematización.
- El *stemmer* de Porter es la mejor opción de preprocesamiento para estos datos, aunque no es un ganador claro. Obtiene los mejores resultados en dos de los algoritmos. De cualquier manera el de Lancaster en algunas combinaciones es mejor.

- La lematización con Wordnet es la mejor opción para k-means y en general mejora los resultados frente a no a hacer nada. También es la mejor opción en algunas combinaciones de los otros algoritmos.
- El coeficiente de silueta es bajo en k-means y el agrupamiento espectral. Además, con esos algoritmos el valor apenas cambia en las diferentes combinaciones. Solo alcanza valores algo mejores en el algoritmo de Ward.

7. Conclusiones y trabajos futuros

El trabajo realizado tiene tres áreas básicas: agrupamiento, minado de texto y la aplicación realizada. Tanto el agrupamiento como el minado de texto son campos muy amplios. El trabajo presentado solo se puede considerar una primera aproximación a los mismos.

Una de las principales conclusiones del trabajo es que diferentes tipos de grupos requieren diferentes algoritmos. Por ejemplo, los datos del experimento del capítulo 6 son muy poco apropiados para un algoritmo basado en densidades (al menos, con las formas escogidas para representar los documentos). Por tanto es necesario conocer una variedad amplia de algoritmos y conocer para qué tipos de datos son más o menos apropiados. Es especialmente importante en el caso de grandes conjuntos, donde hay que usar algoritmos eficientes.

Otro aspecto importante es la reducción de la dimensionalidad. Se ha podido comprobar en el experimento que la eficiencia no es el único motivo para usarla, sino que además puede mejorar los resultados al liberar a los algoritmos de información poco relevante.

En el campo del minado de texto también hay muchas posibilidades para seguir investigando. Una primera ampliación sería cambiar el modelo de representación de los documentos por uno más sofisticado. Algunas opciones son:

- N-gramas. En el trabajo se obtenía una representación de los documentos basada en la frecuencia de palabras individuales. También es posible analizar, por ejemplo, la frecuencia de aparición de secuencias de dos palabras.
- Word2vec. Esta herramienta parte de una colección de documentos y proporciona una representación vectorial de cada palabra. Estos vectores tienen la capacidad de capturar información lingüística, así que se pueden usar posteriormente para aplicaciones de procesamiento de lenguaje natural.

También se pueden incorporar mejoras en el preprocesamiento lingüístico, como las que se comentaron en el capítulo 2:

- Etiquetado gramatical. La función gramatical de una palabra se puede usar como característica de la misma y además es un paso previo para otra técnicas, como las dos que siguen.
- Desambiguación. Por ejemplo, usando Wordnet [Amine] para sustituir cada palabra su concepto asociado.

- Troceado de texto (*text chunking*). Como ya comentamos puede ser una aproximación útil para caracterizar documentos.

La aplicación realizada sirve para realizar algunos experimentos cómodamente, pero tiene también muchas posibilidades de ampliación y mejora:

- Incrementar el número de algoritmos disponibles y permitir manejar todos los parámetros disponibles en cada algoritmo.
- Automatizar la realización de los experimentos. Aplicar varios algoritmos al mismo conjunto de datos a la manera de un script, generando un documento resumen.
- Mejorar la salida de los datos: ficheros resumen y en formato CSV/XML.
- Añadir una opción para realizar agrupamiento por consenso.
- Opción para aplicar un algoritmo con un rango de valores en sus parámetros para buscar los valores óptimos.
- Nuevos conjuntos de datos y la opción de usar una colección de documentos proporcionada por el usuario.

Por otro lado, la aplicación desarrollada en el proyecto utiliza un conjunto de datos de prueba con el objetivo de comparar algunos algoritmos habituales. El siguiente paso lógico sería intentar utilizar las técnicas de agrupamiento para resolver un problema real. A continuación se citan algunas aplicaciones interesantes, obviamente desde un punto de vista personal:

- Sistemas de recomendación. Son un ejemplo de uso habitual de los algoritmos de agrupamiento, ya que se basan en buscar elementos parecidos a los que el usuario ya ha visitado. Tienen un gran interés comercial y se utilizan en diversos contextos. En un periódico muestran noticias relacionadas con la que se está leyendo; en una tienda, artículos relacionados con las últimas búsquedas del usuario.
- Agrupamiento descriptivo. Al problema del agrupamiento se le añade el de generar una descripción en lenguaje natural para cada grupo que se encuentre [Weiss].
- Recopilación de noticias. Se trata de una aplicación habitual en los buscadores web. Se toman noticias de prensa de fuentes diferentes y se muestran agrupadas

según al hecho al que se refieran. Fue una aplicación que se valoró para el proyecto.

- Detección de plagio. La idea es detectar trozos de documentos atípicos en relación al resto del documento en el que se encuentren [Akiva]. Estos trozos, identificados mediante troceado de texto, podrían no haber sido escritos por el (supuesto) autor del documento.

8. Referencias

[20Newsgroups] http://qwone.com/~jason/20Newsgroups/

[Akiva] Navot Akiva. 'Using Clustering to Identify Outlier Chunks of Text'. http://ceur-ws.org/Vol-1177/CLEF2011wn-PAN-Akiva2011.pdf

[Almeida] Almeida, Gómez, Yamakami.
http://www.dt.fee.unicamp.br/~tiago/smsspamcollection/doceng11.pdf

[Amine] Amine, Elberrichi, Simonets. 'Evaluation of Text Clustering Methods Using WordNet'. The International Arab Journal of Information Technology, Vol. 7, No. 4, October 2010.

[Ankerst] Mihael Ankerst, Markus M. Breunig, Hans-Peter Kriegel, Jörg Sander (1999). *OPTICS: Ordering Points To Identify the Clustering Structure*. ACM SIGMOD international conference on Management of data. ACM Press. pp. 49–60.

[Arthur] Arthur, Vassilvitskii. k-means++: The Advantages of Careful Seeding
http://ilpubs.stanford.edu:8090/778/1/2006-13.pdf

[Bezdek] J. C. Bezdek and R. J. Hathaway, "VAT: A Tool for Vis-ual Assessment of (Cluster) Tendency," Proceedings of the 2002 International Joint Conference on Neural Net-works, Honolulu, 12-17 May 2002, pp. 2225-2230.

[Cárdenas] Cárdenas-Montes. Índice de Davies-Bouldin.
http://wwwae.ciemat.es/~cardenas/docs/lessons/Davies-Bouldin.pdf

[Dean] Dean, J. http://jalt.org/test/bro_30.htm

[Ester]Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu (1996). Evangelos Simoudis, Jiawei Han, Usama M. Fayyad, eds. *A density-based algorithm for discovering clusters in large spatial databases with noise*.

[Frey] http://www.psi.toronto.edu/affinitypropagation/FreyDueckScience07.pdf

[Grygo] Grygorash, Zhou, Jorgensen. Minimum Spanning Tree Based Clustering Algorithms. http://www4.ncsu.edu/~zjorgen/ictai06.pdf

[Inquirer] www.wjh.harvard.edu/~inquirer/

[Kai] Kainulainen, Jukka. Clustering Algorithms: Basics and Visualization.
http://www.niksula.hut.fi/~jkainula/pdfs/clustering.pdf

[Kaufman] Kaufman, L. and Rousseeuw, P.J. (1987), Clustering by means of Medoids.

[Kwale] Francis M. Kwale. An Efficient Text Clustering Framework. International Journal of Computer Applications. Volume 79 – No8, October 2013.

[Luxburg] Luxburg. A Tutorial on Spectral Clustering. Statistics and Computing, 17 (4), 2007.

[Mirkes] Mirkes, E. M. http://www.math.le.ac.uk/people/ag153/homepage/KmeansKmedoids/Kmeans_Kmedoids.html

[NLTK] http://www.nltk.org/

[OPTICS] https://en.wikipedia.org/wiki/OPTICS_algorithm

[Pana] Panayirci E., Dubes R.C. 'A test for multidimensional clustering tendency', Pattern Recognition,Vol. 16(4), pp. 433–444, 1983.

[Porter] Porter, M.F. An algorithm for suffix stripping. http://tartarus.org/martin/PorterStemmer/def.txt

[Potts] Potts, Christopher. Sentiment Symposium Tutorial: Stemming. http://sentiment.christopherpotts.net/stemming.html

[Python] https://www.python.org/

[Qt] http://www.qt.io

[Rendón] Eréndira Rendón, Itzel Abundez, Alejandra Arizmendi and Elvia M. Quiroz. 'Internal versus External cluster validation indexes'. INTERNATIONAL JOURNAL OF COMPUTERS AND COMMUNICATIONS. Issue 1, Volume 5, 2011

[Scikit] http://scikit-learn.org/stable/

[Strehl] http://strehl.com/diss/node1.html

[Sumathy] Sumathy, Chidambaram. 'Text Mining: Concepts, Applications, Tools and Issues – An Overview'.

[Theo]Pattern Recognition, 4ª edición. S. Theodoridis, K. Koutroumbas.

[Vallinaku] Vallinaku, Meyyappan. Survey of Text Mining. International Conference on Technology and Bussiness Management, pp. 508–514, 2013.

[Vinh]N. X. Vinh, J. Epps, J. Bailey. Information Theoretic Measures for Clusterings Comparison: Variants, Properties, Normalization and Correction for Chance. Journal of Machine Learning Research 11 (2010) 2837-2854.
http://www.jmlr.org/papers/volume11/vinh10a/vinh10a.pdf

[Voronoi] https://en.wikipedia.org/wiki/Voronoi_diagram

[Wang] Wang, Nguyen, Bezdek, Leckie, Ramamohanarao. iVAT and aVAT: Enhanced Visual Analysis for Cluster Tendency Assesment. Advances in Knowledge Discovery and Data Mining. 14th Pacific-Asia Conference, PAKDD 2010. Proceedings, Part 1.

[Ward] Ward, J. H., Jr. (1963), "Hierarchical Grouping to Optimize an Objective Function", *Journal of the American Statistical Association*, 58, 236–24.
http://www.psi.toronto.edu/index.php?q=affinity%20propagation

[Weiss] 'Descriptive Clustering as a Method for Exploring Text Collections'.
http://www.cs.put.poznan.pl/dweiss/site/publications/download/dweiss-phd-thesis.pdf

[Word2Vec] https://code.google.com/p/word2vec/

[Wordnet] Princeton University 'About WordNet.' WordNet. Princeton University. 2010. http://wordnet.princeton.edu