Universidad Central “Marta Abreu” de Las Villa
Facultad de Matemática, Física y Computación

Trabajo de Diploma

Estudio de métodos de agrupamiento en el contexto del resumen de corpus textuales

Autores:
Yoisy Pérez Olmos
Juan M. Mederos Martínez

Tutores:
Lic. Leticia Arco García
Dr. Rafael E. Bello Pérez

Consultante:
Lic. Manuel Llanes Abeijón

2005

Hago constar que este trabajo ha sido realizado en la facultad de Matemática, Física y Computación de la Universidad Central “Marta Abreu” de las Villas como parte de la culminación de los estudios de licenciatura en Ciencias de la Computación, autorizando a que el mismo sea actualizado por la institución para los fines que estime conveniente, tanto de forma total como parcial y que además no podrá ser presentado en eventos ni publicado sin la previa autorización de la universidad.

_____ _____

Firma del Autor Firma del Autor

Los abajo firmantes, certificamos que el presente trabajo ha sido realizado según acuerdo de la dirección de nuestro centro, y que el mismo cumple con los requisitos que debe tener un trabajo de esta envergadura referido a la temática señalada.

_____ _____

Firma del Tutor Firma del Jefe del Seminario de Inteligencia Artificial

Dedicamos este trabajo a nuestros padres que han sabido colmar nuestras vidas de satisfacciones y esperanzas.

Agradecimientos

- No podríamos iniciar esta lista con otro nombre que no sea el de nuestra tutora Leticia Arco por ser tan especial, tan bondadosa, y tan dedicada a nuestro trabajo.
- A nuestro consultante Manuel Llanes, por brindarnos tanta ayuda y parte de su tiempo.
- Agradecemos a nuestros padres por su apoyo incondicional y confianza en nosotros.
- A todos nuestros profesores que a lo largo de estos años han influido en nuestra formación y madurez profesional.
- A nuestros compañeros de aula por brindarnos tantos momentos gratos.
- A todos nuestros amigos que de una forma u otra han contribuido a alcanzar esta meta, en especial a Zain, el Libero y Mirelys, por tantas consultas y noches de desvelo.
- A la "Charanga Camagüeyana", fieles amigos de siempre.
- A todos nuestros amigos del año anterior por tanta ayuda y paciencia, en especial a Amilkar, Yanibeisy, Yailén y La Tribu.

Resumen

En este trabajo se realizó el estudio de métodos que permiten el agrupamiento de documentos con el objetivo de obtener resúmenes extractos de un corpus textual. Para lograr estos propósitos nos insertamos en la herramienta CorpusMiner mediante el desarrollo de los módulos que permiten el agrupamiento y su evaluación, así como la obtención de resúmenes. Los algoritmos de agrupamiento implementados son: SKWIC (duro y determinista), Fuzzy SKWIC (borroso) y Extended Star (duro y con solapamiento). Permitimos, además, utilizar los resultados del agrupamiento del Extended Star para inicializar los dos antes mencionados y así obtener mejores resultados. Estos algoritmos parten de una representación VSM del corpus y devuelven una colección de clusters de documentos. Los métodos de agrupamiento implementados requieren calcular la similitud entre vectores, para ello incorporamos la similitud Coseno, distancias Jaccard binaria y pesada, y la distancia Euclidiana, aunque esta última no reporta buenos resultados en dominios textuales. En el módulo de evaluación implementamos las medidas Entropía, *F-Measure* (*precision* y *recall*) y *Overall Similarity* que permiten cuantificar la calidad del agrupamiento. El último módulo de CorpusMiner permite obtener un resumen extracto de cada uno de los clusters formados, extrayendo las oraciones que tienen la presencia de las palabras claves de cada cluster. La extracción de las oraciones por cada cluster puede realizarse solamente a partir del documento más representativo o de todos los documentos del cluster.

Abstract

This work is concerned with the classification of documents to produce abstracts of related-document clusters in a text corpus. To achieve this aim we developed a series of modules in the CorpusMiner system to cluster documents, to assess the quality of a clusterization and to produce extractive abstracts from document clusters. The clustering algorithms implemented in the clustering module include: SKWIC (deterministic hard clustering), Fuzzy SKWIC (fuzzy clustering) y Extended Star (hard and overlapping clustering). We also implemented the possibility of using the results of Extended Star clustering to initialize the other two clustering methods and thus achieve better results. All the algorithms implemented are based on a VSM representation of the corpus and produce a collection of document clusters. The implemented clustering methods are based on similarity measures such as the cosine, binary Jackard's distance and weighted Jackard's distance and Euclidian distance (the results of this last similarity measure are not the best for textual domains). In the assessment module we implemented the measurement based on entropy, the F-Measure (*precision* and *recall*) and *Overall Similarity* which allow us to quantify the quality of the clustering results. This last module within CorpusMiner allows us to produce an extractive abstract of each of the clusters obtained by extracting the sentences in the document cluster that contain the keywords of each. The keyword extraction can be performed from all texts in the cluster or from the most representative text in each cluster.

Índice

Introducción

En la Revolución de la Información que vivimos ha surgido una tarea difícil – los humanos no estamos diseñados para procesar cantidades masivas de información y encontrar asuntos de interés. La computadora primero encontró su uso en acelerar el funcionamiento de grandes cálculos numéricos eficientemente. Ahora es necesario que las computadoras resuelvan otra incompetencia humana, analizar grandes volúmenes de datos para encontrar elementos interesantes.

La proliferación de información disponible en el World Wide Web, intranets corporativas y bases de datos, cables de noticias electrónicas, y otros medios de comunicación es arrolladora. La creación y diseminación de información es soportada por un número creciente de herramientas, sin embargo, mientras que la cantidad de información disponible está continuamente creciendo, nuestra habilidad de procesarla y asimilarla permanece constante [DIX97] [LAN01].

El conocimiento es un recurso estratégico para el desarrollo económico, científico y social contemporáneo. Las tecnologías, métodos y herramientas asociadas con los procesos de adquisición, generación, gestión y transmisión del conocimiento se han desarrollado notablemente en los últimos años. Es imprescindible el empleo de técnicas y herramientas que le den sentido y utilidad a la información existente, ya que ésta es un elemento básico principal. Los motores de búsqueda pueden desempeñar un papel esencial en la viabilidad de los sistemas de información, pero es imprescindible que existan aplicaciones que puedan analizar y evaluar la relevancia de la información [FEB02].

El procesar automáticamente grandes cantidades de datos para encontrar conocimiento útil es el objetivo principal del área de Descubrimiento de Conocimiento en Bases de Datos o KDD (*Knowledge Discovery from Data base*) y podemos definirlo como un proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de datos, o como la extracción no trivial de información implícita, desconocida, y potencialmente útil de los datos [LEZ02].

La minería de datos (*Data Mining*) no es más que una fase del KDD, fase que integra los métodos de aprendizaje y estadísticas para obtener hipótesis de patrones y modelos. Las técnicas de minería de datos surgen como las mejores herramientas para realizar exploraciones más profundas y extraer información nueva, útil y no trivial que se encuentra oculta en grandes volúmenes de datos estructurados [LEZ02].

La minería de datos se involucra con la Estadística (correlación, regresión, agrupamiento numérico, y otros), la Inteligencia Artificial (Redes Neuronales Artificiales y Algoritmos Genéticos), técnicas de visualización, consultas SQL, OLAP (*Online Analytical Processing*), el uso de la inducción de árboles y reglas, así como con el desarrollo de nuevas herramientas de inducción de reglas de asociación e inducción de clasificadores bayesianos [LEZ02].

La limitante que existe es que las técnicas de minería de datos procesan información estructurada, y sin embargo, aproximadamente un 80% de la información está almacenada en forma textual no estructurada, de ahí que se desarrollen actualmente técnicas de minería de textos (*Text Mining*), que permiten encontrar patrones interesantes y útiles en un corpus de información textual no estructurada [DUR01].

Para lograr sus propósitos, la minería de textos necesita combinar varias técnicas, de ahí que sea un campo multidisciplinario que incluye la recuperación de información, el análisis de textos, la extracción de información, el agrupamiento, el resumen, la categorización, la clasificación, la visualización, la tecnología de bases de datos, el aprendizaje automático y la minería de datos [DIX97] [TAN99].

En estas áreas se ha trabajado intensamente, pero aún quedan muchos problemas por resolver. Los corpus de textos generalmente se representan siguiendo un modelo VSM. Una de las formas de extraer conocimiento de ellos es realizando resúmenes extractos. Pero estos corpus son generalmente heterogéneos, por tanto un problema a resolver es verificar su homogeneidad. Para ello, es necesario calcular los niveles de similitud entre textos, criterios estos que permiten agrupar la colección en clusters homogéneos de documentos.

Las preguntas de investigación que nos hemos formulado son:

- ¿Es factible verificar la homogeneidad de un corpus a partir del agrupamiento de los documentos similares que lo conforman?
- ¿Son los métodos de agrupamiento con solapamiento más efectivos que los métodos que logran particiones para identificar el grado de homogeneidad del corpus de textos?
- ¿Serán los resúmenes obtenidos verdaderos extractos del corpus original?

Para dar solución al problema planteado y respuesta a las preguntas de investigación nos hemos formulado como objetivo de investigación estudiar e implementar métodos de agrupamiento de documentos en su representación VSM (*Vector Space Model*) que permitan verificar la homogeneidad de un corpus de documentos en el contexto del resumen textual.

Para cumplimentar este objetivo hemos formulado algunos objetivos específicos que nos facilitarán el desarrollo del mismo:

- Analizar los principales algoritmos de agrupamiento en un dominio textual, así como estrategias que permitan obtener resúmenes extractos de documentos y determinar las variantes a implementar.
- Implementar en CorpusMiner métodos de agrupamiento que permitan verificar la homogeneidad de un corpus textual a partir de la similitud entre vectores documentos.
- Incorporar en CorpusMiner variantes que permitan obtener resúmenes extractos de clusters de documentos, teniendo en cuenta la selección del documento centro (más representativo) de cada grupo.
- Validar los métodos de agrupamiento implementados.

El capítulo 1 lo dedicaremos a describir las técnicas generales de la minería de textos, particularizando en los principales resultados publicados sobre el agrupamiento y resumen extracto de documentos. Abordaremos las características principales del resumen de un único texto, así como de múltiples documentos. Mencionaremos las diferentes técnicas de agrupamiento y métricas para comparar los objetos a agrupar. Ejemplificaremos con algoritmos de agrupamiento que utilizan técnicas duras y deterministas, duras y con solapamiento, y borrosas. El capítulo 2 estará dedicado a describir el diseño general de los módulos de la herramienta CorpusMiner que se dedican al agrupamiento de documentos y la obtención del resumen extracto del corpus ya sea seleccionando sólo las principales oraciones de los documentos más representativos de cada grupo o de todos los documentos que lo conforman, así como las medidas de similitud implementadas para comparar vectores documentos. Por último, en el capítulo 3 mostraremos la validación de los resultados del agrupamiento. Para evaluar la calidad del mismo utilizaremos las medidas *Overall Similarity*, *F-Measure* (*Precision* y *Recall*) y entropía. Además, describiremos como utilizar en CorpusMiner los módulos que permiten el agrupamiento y su evaluación, así como la generación del resumen extracto.

Capítulo 1 Acerca de las técnicas de la minería de textos para el agrupamiento y resumen extracto de documentos

Este capítulo lo dedicaremos a describir las técnicas generales de la minería de textos, particularizando en los principales resultados publicados sobre el agrupamiento y resumen extracto de documentos. Abordaremos las características principales del resumen de un único texto, así como de múltiples documentos. Mencionaremos las diferentes técnicas de agrupamiento, además de las medidas necesarias para calcular la distancia, similitud o disimilitud de los objetos a comparar en el análisis de clusters. Mencionaremos la aplicación del agrupamiento a la verificación de la homogeneidad de corpus textuales, particularizando en los algoritmos que utilizan técnicas duras y deterministas, duras y con solapamiento, y borrosas.

1.1 ¿Qué es la minería de textos?

La información histórica es útil para predecir información futura, ya que la mayoría de las decisiones de empresas, organizaciones e instituciones se basan en información de experiencias pasadas, extraídas de fuentes muy diversas.

El procesar automáticamente grandes cantidades de datos para encontrar conocimiento útil para un usuario y satisfacerle sus metas es el objetivo principal del área de descubrimiento de conocimiento en bases de datos o KDD. Este es el campo que está evolucionando para proporcionar soluciones al análisis automático, y en [LEZ02] aparece definido como un proceso no trivial de identificar patrones válidos, novedosos, potencialmente útiles y en última instancia comprensibles a partir de datos, es decir, no es más que la extracción no trivial de información implícita, desconocida, y potencialmente útil de los datos.

Es muy importante comprender el KDD, ya que la minería de datos no es más que una fase del mismo, fase que integra los métodos de aprendizaje y estadísticas para obtener hipótesis de patrones y modelos. Las técnicas de minería de datos permiten realizar exploraciones más profundas y extraer información nueva, útil y no trivial que se encuentra oculta en grandes volúmenes de datos [LEZ02].

Se puede decir que un sistema de minería de datos es una tecnología soporte para usuario final cuyo objetivo es extraer conocimiento útil y utilizable a partir de la información contenida en bases de datos; también se llama minería de datos al análisis de archivos y bitácoras de transacciones que sean útiles para la toma de decisiones. Las habilidades de la especie humana no nos permiten realizar, con la misma eficiencia, la tarea de analizar los trillones de datos almacenados

electrónicamente al monitorear las transacciones comerciales de una base de datos, independientemente de la capacidad humana de detectar patrones y descubrir tendencias.

La minería de datos se involucra con la Estadística (correlación, regresión, agrupamiento numérico, y otros), la Inteligencia Artificial (Redes Neuronales Artificiales y Algoritmos Genéticos), técnicas de visualización, consultas SQL, OLAP, el uso de la inducción de árboles y reglas, así como el desarrollo de nuevas herramientas de inducción de reglas de asociación e inducción de clasificadores bayesianos [LEZ02].

Hasta ahora hemos mencionado la minería de datos y sus aplicaciones y repercusión en la sociedad, pero vale destacar que la minería de datos trabaja sólo con datos estructurados, es decir, grandes bases de datos donde están muy bien organizados y definidos los datos a explorar. ¿Todos los datos que se presentan están en forma estructurada? No, todo lo contrario, aproximadamente un 80% de la información de las organizaciones está almacenada en forma textual no estructurada: informes, e-mail, actas de reuniones, artículos científicos, cables de noticias, etc [DUR01].

La lengua se ha convertido en un elemento clave de la llamada Sociedad de la Información (SI). Sin embargo, la tecnología asociada al procesamiento automático de textos, no se ha desarrollado al nivel suficiente para proporcionar la infraestructura que soporte a la SI. Además, si queremos crecer en las redes de comunicación, no solo es importante incrementar los contenidos, sino también es indispensable crear herramientas y recursos capaces de brindar servicios a partir de esos textos almacenados [TAL04].

Hasta ahora, las organizaciones están siendo incapaces de usar eficientemente grandes cantidades de datos textuales no estructurados (*free text*) que existen en varias fuentes, tales como la Web. La minería de textos es una tecnología poderosa que facilita la explotación efectiva de tales datos. Como un resultado, todos los textos no estructurados pueden ser automáticamente analizados y se pueden extraer los conceptos más importantes, etiquetar los documentos con términos claves, resumirlos y hacerlos disponibles en una forma normalizada y explotable [MAL04].

Se debe tener en consideración la aplicación de herramientas que automáticamente asistan a los usuarios en la organización y manipulación de ese exceso de información. Por ejemplo, esas herramientas podrían filtrar información relevante o interesante desde información no relevante acorde a intereses especificados por los usuarios. La información puede ser resumida y visualmente presentada. Además, la identificación de conocimiento oculto en datos no estructurados disponibles es un asunto crucial [LAN01].

La minería de textos, también conocida como minería de datos textuales o descubrimiento de conocimiento desde bases de datos textuales, pretende algo similar a la minería de datos: identificar relaciones y modelos en la información no cuantitativa. En pocas palabras, proveer de una visión selectiva y perfeccionada de la información contenida en documentos escritos y sacar consecuencias para la acción, así como detectar patrones no triviales e incluso información sobre el conocimiento almacenado en las mismas [OBE01].

La minería de textos es mucho más compleja que la minería de datos porque tiene que trabajar con datos textuales que son inherentemente no estructurados y borrosos. La minería de textos es un campo multidisciplinario que incluye recuperación de información, análisis de textos, extracción de información, agrupamiento, resumen, categorización, clasificación, visualización, tecnología de bases de datos, aprendizaje automático y minería de datos [TAN99] [DIX97].

A continuación describiremos cada una de las áreas que conforman la minería de textos:

Recuperación de información: Se encarga de recuperar documentos que puedan ser considerados relevantes para la tarea a realizar. Típicamente los usuarios del sistema pueden especificar conjuntos de documentos, pero el sistema debe ser capaz de filtrarlos y dejar fuera aquellos irrelevantes [DIX97] [FRA92].

Extracción de la información: Es el proceso de filtrar la información a partir de documentos ya seleccionados y de las especificaciones de los usuarios a través de preguntas complejas [DIX97] [FRA92] [FRA03].

Análisis de textos: Involucra todas las técnicas relacionadas con el análisis léxico, sintáctico y semántico de los textos [JAC02].

Resumen: Es el proceso de extraer conocimiento a partir de una fuente de información y presentar el contenido más importante al usuario en una forma condensada y sensitiva para las necesidades de la aplicación o del usuario [JAC02] [BER04].

Agrupamiento: Consiste en encontrar grupos de documentos que están relacionados por tópicos similares y extraer las palabras claves más importantes que son consideradas en esa clasificación [BER04].

Categorización: Es el proceso de clasificar los documentos por sus contenidos [JAC02] [BER04].

Clasificación: Se usa como un término más amplio que la categorización, para incluir cualquier asignación de documentos a clases, no necesariamente basados en el contenido [JAC02] [BER04].

1.2 Resumen extracto

Un problema de la minería de textos, en la actualidad, es encontrar documentos relevantes sobre la información que necesitamos, pero realmente la situación es incluso más compleja. Después de encontrar algunos documentos relevantes, el problema es encontrar el tiempo necesario para leerlos, por tanto, resumirlos puede contribuir a realizar una revisión en tiempo de los aspectos relevantes de ellos.

Resumir automáticamente textos consiste en tomar una fuente de información, extraer contenido de ésta, representando los conceptos más importantes de cada documento y presentar el contenido más importante al usuario en una forma compacta y sensible para satisfacer las necesidades de la aplicación o del usuario [NAN00]. Es un proceso que toma un documento como entrada, ofreciendo como salida uno documento más corto, llamado documento sustituto, que contiene el contenido más importante del inicial. La importancia puede ser considerada a partir de diferentes puntos de vista: fragmentos asociados a las palabras claves, requerimientos de usuarios y relevancia a partir de tópicos seleccionados, entre otros [JAC02].

Existen dos tipos principales de resúmenes: los resúmenes abstractos (*abstract*) y los resúmenes extractos (*summarization*). Sus aplicaciones y formas de obtención difieren sustancialmente. Un extracto es un resumen que es construido, sobre todo, escogiendo los fragmentos más relevantes del texto, quizás con algunas pequeñas revisiones. Un abstracto es un resumen que describe el contenido de un documento sin presentar, necesariamente, algunas de las partes del contenido del texto original explícitamente. En ambos casos, podemos pensar en el resumen como la compresión o la compactación de un documento. Un extracto realiza la compresión descartando el material menos relevante, mientras que un abstracto realiza la compresión de un modo más sofisticado, e.g. suprimiendo detalles y sustituyendo datos específicos con generalidades. Obviamente, se pudieran mezclar estos dos modos de compresión para obtener un resumen de mayor calidad [JAC02].

Otra forma de clasificar los resúmenes es en genéricos (*generis summaries*) y teniendo en cuenta las preguntas de los usuarios (*query-relevant summaries*). Los primeros se generan teniendo en cuenta todo el contenido del documento y los segundos resumen el contenido relevante a partir del entorno de la consulta realizada por el usuario.

La tarea de resumir textos puede ser dividida en dos fases: (a) construcción de una representación del texto; (b) generación del resumen, el cual puede incluir extracción de oraciones y/o construcción de nuevas oraciones. La construcción automática de nuevas oraciones es una tarea extremadamente

difícil, la mayoría de los sistemas generan un resumen extrayendo las oraciones más relevantes del documento original [LAR00]. En este trabajo pretendemos obtener un resumen extractivo.

Un resumen se puede realizar a partir de un único documento o de múltiples documentos relacionados o no. Este es otro aspecto que permite la clasificación de los resúmenes. A continuación describiremos en qué consiste realizar un resumen extracto de un único documento (*single summarization*) a partir de la generación de textos pequeños y concisos que representan las ideas generales presentadas en un texto, y un resumen extracto de múltiples documentos (*multiple summarization*) donde se extraen las ideas fundamentales a partir de múltiples documentos [LAR99].

1.2.1 Resumen extracto de un único texto

La forma más conocida para construir resúmenes extractos de documentos simples es teniendo un programa selector de fragmentos relevantes desde el documento y luego combinarlos en un extracto [JAC02].

En [MOE00], [JAC02] y [FUK03] se reportan varias técnicas de resumen extracto de un único documento. Algunas de estas técnicas son: resumen por selección de oraciones, resumen por selección de párrafos, resumen basado en discurso y resumen basado en co-referencia. A continuación describiremos cada una de estas estrategias y ejemplificaremos con sistemas reportados en la literatura.

1.2.1.1 Resumen por selección de oraciones

Un modo común de abordar un problema de investigación difícil es transformarlo en una tarea más sencilla que permita realizar la mayor parte del trabajo. Esta forma de enfrentar los problemas se puede utilizar al generar resúmenes extractos y teniendo en cuenta que la oración es frecuentemente (no siempre) seleccionada como la unidad para construir resúmenes. A partir de la selección de oraciones reducimos la generación del resumen de una tarea compleja a un problema de clasificación de las oraciones en relevantes o no para formar parte del extracto. Tener una clasificación permite incluir o excluir oraciones según la longitud deseada del resumen [JAC02].

La generación de resúmenes a partir de la selección de oraciones tiene sus ventajas y desventajas en dependencia del texto a resumir y de la longitud del resumen deseado. Es ventajoso utilizar esta estrategia cuando deseamos resumir noticias o textos no extremadamente largos. Un problema de esta estrategia es que los resúmenes resultantes son a menudo, desunidos y no se leen bien. Las

oraciones, además, tienen obviamente sus ventajas y desventajas respecto a usar unidades mayores (e.g. párrafos) o unidades pequeñas (e.g. frases).

1.2.1.2 Resumen por selección de párrafos

Como mencionamos anteriormente, problemas al considerar las oraciones como unidad básica al resumir son la desunión y la falta de coherencia, es decir, no se leen bien los resúmenes. La utilización de componentes básicos más grandes puede contribuir a la obtención de un resumen más coherente. Así, un enfoque puede ser asumir en un texto un número de párrafos que se pueden considerar mejores o relevantes, y que pueden describir el contenido del texto en su totalidad. Esto es particularmente efectivo para ciertos tipos de documentos, donde uno de los primeros párrafos, típicamente el primero, proporciona una descripción coherente de lo que sigue [JAC02].

La selección de párrafos ha sido bien estudiada, aunque no tiene tanta popularidad como la selección de oraciones. Ella es ventajosa si el resumen requerido es relativamente largo, o si el material es tal que el aspecto principal de un documento es probablemente contenido en un párrafo simple [JAC02].

1.2.1.3 Resumen basado en discurso

Como hemos descrito en las secciones 1.2.1.1 y 1.2.1.2, resumir documentos teniendo en cuenta las oraciones o párrafos como unidades tiene sus ventajas y desventajas. La estrategia de resumen basado en discurso propone determinar inicialmente la forma típica de discurso del documento a resumir, es decir, modelar inicialmente la estructura del documento que será resumido [MOE00]. Para ello, es necesario inicialmente dividir el documento en unidades de discurso coherentes. Los bloques de texto obtenidos deben reflejar los subtópicos contenidos en el texto. Para realizar la segmentación es necesario analizar léxicamente el texto, utilizar una medida de recuperación de la información para determinar la extensión de los bloques, e incorporar un diccionario y un algoritmo de desambiguación léxica estadística [YAR92].

Esta estrategia tiene gran utilidad porque los tipos diferentes de documentos tienen variaciones de su estructura, y por tanto, es muy útil identificar inicialmente la presencia o la ausencia de segmentos claves, para posteriormente realizar el extracto a partir de las unidades básicas detectadas. Un desventaja de esta estrategia es que generalmente funciona correctamente para un tipo particular de documentos y utilizándolo en un contexto específico.

1.2.1.4 Resumen basado en co-referencia

Co-referencia es un fenómeno lingüístico donde dos o más expresiones pueden representar o indicar la misma entidad. Así como otros fenómenos lingüísticos, la co-referencia puede admitir la ambigüedad. El concepto básico de co-referencia esta presente cuando dos expresiones lingüísticas, tales como 'Bill Gates' y 'The Chairman of Microsoft', se refieren ambas a la misma entidad.

Los métodos anteriormente mencionados están concebidos para obtener resúmenes genéricos, mientras que los métodos basados en co-referencia están más enfocados a generar el resumen teniendo en cuenta las preguntas de los usuarios y extrayendo el contenido relevante a partir del entorno de la consulta realizada por el usuario.

Las asociaciones que existen entre términos que tienen co-referencia pueden ser usadas para clasificar y seleccionar oraciones del documento al ser incorporadas en un resumen. Los métodos que siguen esta estrategia son capaces de generar resúmenes que son casi tan efectivos como el texto completo, ayudando al usuario a determinar la relevancia [BAL98]. La desventaja de esta estrategia es que se requiere realizar un preprocesamiento más costoso del documento (e.g. etiquetarlo, trabajar con tesauros).

1.2.1.5 Ejemplos

En [FUK03] se reporta un sistema que permite el resumen de un documento siguiendo la estrategia de selección de oraciones. Se extraen las oraciones con mayor peso según el cálculo TF/IDF, se organizan las oraciones en dependencia de su posición en los párrafos y se eliminan las partes innecesarias en las oraciones para resolver anáforas. En [FUK03] también se propone un sistema para generar resúmenes de documentos simples basado en la extracción de oraciones según TF/IDF. Las oraciones seleccionadas conforman el extracto.

Un ejemplo de la estrategia de resumen basado en el discurso es el sistema SALOMON, descrito en [JAC02], que antes de obtener el resumen extracto es capaz de identificar las partes principales del documento (e.g. título, introducción) y esto contribuye a la obtención de un resumen de mayor calidad.

Otro ejemplo del resumen basado en discurso se muestra en [MAR00]. La idea básica del método, nombrado *Rhetorical Structura Theory*, es que el texto puede ser descompuesto en dos tipos de unidades elementales: núcleo y satélite. El núcleo expresa algo esencial para el propósito del escritor, mientras que satélite expresa algo menos esencial. Los pasos que lo describen aparecen a continuación:

1. Identificar la estructura del discurso del texto, usando su '*rhetorical parsing algorithm*'.
2. Determinar un orden parcial de las unidades de la estructura del discurso.
3. Seleccionar las unidades en este orden, hasta cubrir un porciento especificado para el tamaño del resumen respecto al documento original.

Continuando con la estrategia basada en discurso, en [LAR00a] se presenta un algoritmo para obtener un resumen extractivo de un documento basado en la importancia de los tópicos contenidos en un documento. Primeramente se utiliza el algoritmo TextTiling para identificar los tópicos basado en la métrica TF-IDF, se calcula la relevancia de cada tópico en el documento y se extrae de cada tópico el número de oraciones proporcional a su relevancia.

En [IND01] se describe un programa que permite realizar el resumen de un documento combinando las estrategias basadas en discurso y co-referencia. Éste inicialmente describe las unidades de discurso del texto y elimina la información extraña, posteriormente realiza una detección robusta de co-referencia.

Otro ejemplo que combina varias estrategias es reportado en [LAR04]. Los autores proponen un sistema para resumir noticias y obtener una estructura argumentativa aproximada del texto original. Para alcanzar estos objetivos se usan varias técnicas y heurísticas, como descubrimiento de los conceptos principales en el texto, conectividad entre oraciones, presencia de nombres propios, anáforas, marcadores de discurso y una representación de árbol binario.

1.2.2 Resumen de múltiples textos

Si el resumen de un documento simple se dificulta, resumir múltiples documentos presenta aún más problemas. Sin embargo, el éxito en este empeño ofrecerá una utilidad real a muchos investigadores y "trabajadores de ciencia", por permitirles procesar colecciones enteras de documentos con menos esfuerzo que en la actualidad [JAC02].

En [STE00] se plantea que el resumen de documentos simples es sólo una de las tareas críticas necesarias para formar un resumen completo de múltiples documentos. Ejemplos de esto se mencionan a continuación:

- Identificar los temas de importancia en la colección de documentos.
- Seleccionar el resumen de un documento simple representativo por cada uno de los temas.

- Organizar los resúmenes representativos para formar el resumen final de múltiples documentos.

Resumir múltiples documentos es un subtópico que se aborda mucho en la actualidad dentro de las investigaciones sobre la obtención de resúmenes de documentos. Inicialmente, se pensó que se podían aplicar las técnicas usadas para resumir un único documento para obtener el resumen de múltiples documentos considerando la colección de todos los textos como un único documento. Aunque muchas de las técnicas usadas en el resumen de un único documento pueden ser usadas también en el resumen de múltiples documentos, existen al menos cuatro diferencias significativas [GOL99]:

1. El grado de redundancia en la información contenida en un grupo de documentos relacionados es mucho mayor que el grado de redundancia en un documento.
2. Un grupo de documentos puede contener una dimensión temporal, por ejemplo cuando nos referimos a reportes noticiosos.
3. El tamaño del resumen con respecto al tamaño del documento será típicamente mucho menor para colecciones de documentos que para un único documento.
4. El problema de la co-referencia en los resúmenes presenta retos mayores al resumir múltiples documentos que al resumir un único documento.

En [JAD00] se describen varias formas de crear resúmenes por extracto de múltiples documentos:

Resumir secciones comunes de los documentos: Encontrar las partes importantes y relevantes que la colección de documentos tiene en común (su intersección) y usarla como un resumen.

Resumir secciones comunes y secciones únicas de los documentos: Encontrar las partes importantes y relevantes que la colección de documentos tiene en común y las partes relevantes que son únicas y usarlas como un resumen.

Resumir el documento centro (más representativo): Crear el resumen de un único documento a partir del documento centro o más representativo de la colección.

Resumir el documento centro (más representativo) más el resto de los documentos de la colección: Crear el resumen de un único documento a partir del documento centro o más representativo de la colección y agregar alguna representación del resto de los documentos (pasajes o extracción de palabras claves) para proveer un cubrimiento total de la colección de documentos.

Resumir el documento más reciente más el resto de los documentos de la colección: Crear el resumen del documento que tiene información más reciente y adicionar alguna representación del resto de los documentos para proveer un cubrimiento de la colección de documentos.

Resumir secciones comunes y secciones únicas de documentos teniendo en cuenta el factor tiempo: Encontrar las partes relevantes e importantes que la colección de documentos tiene en común y las partes relevantes que son únicas. Tener en cuenta la secuencia de tiempo de la información extraída en la generación del resumen.

1.2.2.1 Ejemplos

A partir de la definición de las diferentes formas de resumir múltiples documentos, se reportan en la literatura varios ejemplos de sistemas que permiten realizar esta tarea.

En [FUK03] se muestra un método para resumir múltiples documentos donde el primer documento es resumido en la proporción de resumen requerida y los siguientes documentos son resumidos en la proporción de resumen requerida más un 10%. La unidad básica seleccionada es la oración. Las oraciones son colocadas según su orden original en la colección. Similar estrategia se sigue en [SCH02], donde se describe un método para resumir un gran corpus de documentos a partir de la aplicación de nuevas estrategias para seleccionar oraciones interesantes e informativas, incluyendo una medida de importancia derivada del análisis del corpus textual.

MEAD es un software reportado en [RAD00] que genera resúmenes de múltiples documentos usando los centros de los grupos obtenidos a partir de la detección de tópicos en la colección. A partir de la combinación de técnicas estadísticas y de aprendizaje automático se identifican en [MCK99] párrafos similares e intersección de frases similares en los párrafos para obtener un resumen conciso de múltiples documentos relacionados. En [LAR00b] se presenta un sistema para resumir noticias y obtener una estructura jerárquica del texto fuente.

En [WHI02] se presenta y evalúa un procedimiento aleatorio de búsqueda local de selección de oraciones para incluir en un resumen de múltiples documentos. Aquí, los autores persiguen la idea de saber favorecer la selección de los bloques de oraciones adyacentes en la construcción de un resumen de múltiples documentos. El reto es mejorar la inteligibilidad sin afectar excesivamente la información.

En [KEO99] se desarrolla un sistema de resumen de múltiples documentos para generar automáticamente un resumen conciso identificando y sintetizando semejanzas a través de un conjunto de documentos relacionados. Se presenta un enfoque único en la integración de técnicas

estadísticas y análisis sintáctico para identificar párrafos similares, intersección de frases similares dentro de los párrafos, y la generación de las expresiones que conformarán el resumen de la colección.

Se ha trabajado intensamente en la generación de resúmenes de noticias. Tal es el caso reportado en [JAM01], donde se definen los resúmenes temporales de actualidades dentro de un tópico de noticias, donde las historias son presentadas una por una y las oraciones de una historia deben ser clasificadas antes de que la siguiente historia pueda ser considerada.

Existen dos tipos de situaciones en las cuales los resúmenes de múltiples documentos pueden ser útiles: (1) cuando los documentos no están relacionados; (2) cuando los documentos están relacionados [GOL99]. Por tanto, es útil identificar inicialmente en una colección, si los documentos que la forman están relacionados o no, porque esto influye decisivamente en el resumen a obtener.

En la mayoría de los trabajos reportados en el área de la generación automática de múltiples documentos se tiene en cuenta un agrupamiento inicial del corpus, para después extraer las oraciones principales. De esta forma se verifica la homogeneidad del corpus antes de comenzar con la aplicación propiamente de las técnicas de generación de resúmenes de múltiples documentos. Resultados que con estas características se reportan en [WHI02], [GOL00], [LAR00a] [MAN02] y [FUK99].

Por tal motivo, una tendencia al resumir múltiples documentos es verificar la homogeneidad del corpus agrupando los textos en función de sus semejanzas semánticas.

En esta investigación obtenemos el resumen de un corpus de documentos (resumir múltiples documentos) verificando previamente la homogeneidad del corpus textual. Por tanto, previo a la extracción de las principales oraciones agruparemos los documentos similares de la colección para verificar su homogeneidad. Es por eso que a continuación definiremos qué es el agrupamiento de datos y describiremos las principales técnicas de agrupamiento, particularizando en el agrupamiento de documentos.

1.3 Técnicas de agrupamiento

El análisis de clusters o agrupamiento de datos es una actividad humana muy importante. Esta actividad usualmente forma las bases del aprendizaje y del conocimiento. El agrupamiento puede ser aplicado a disímiles campos. Por ejemplo, en los negocios para agrupar las necesidades comunes de consumidores, en la medicina para agrupar las enfermedades, y en la Biología, entre otros campos, para categorizar genes con funcionalidades similares.

La minería de textos no constituye una excepción respecto a la importancia de la aplicación de técnicas de análisis de clusters. El agrupamiento es una tarea importante en el correcto funcionamiento de muchos sistemas de minería de textos y de recuperación de información. El agrupamiento puede ser usado eficientemente para encontrar los vecinos más cercanos de un documento, para mejorar la *precision* y *recall* en sistemas de recuperación de información, en la organización de motores de búsqueda y últimamente en la personalización de los resultados de los motores de búsqueda, en la verificación de la homogeneidad de un corpus textual, en el resumen de colección de documentos y en la categorización de términos, entre otros.

En esta investigación abordaremos las principales técnicas de agrupamiento en función de su utilidad en la verificación de la homogeneidad de una colección textual, así como su influencia en la calidad del resumen de la colección a obtener.

El objetivo del agrupamiento de documentos es formar una colección de cluster (subconjuntos, grupos, clases) que cumplan las propiedades siguientes [HOP99]:

- La homogeneidad dentro de los clusters, i.e. los documentos que pertenecen al mismo cluster deben ser tan similares como se pueda.
- La heterogeneidad entre clusters, i.e. los documentos que pertenecen a clusters diferentes deben ser tan diferentes como se pueda.

Antes de mencionar los principales resultados en el agrupamiento de documentos, definiremos en qué consiste el agrupamiento de datos y describiremos las principales técnicas de agrupamiento que se reportan en la literatura.

1.3.1 ¿Qué es el agrupamiento de datos?

El agrupamiento es un proceso de división de un conjunto de datos u objetos en un conjunto de subclases significativas, llamadas clusters. Formalmente, dada una colección de *n* objetos cada uno de los cuales es descrito por un conjunto de p atributos, el objetivo del agrupamiento es derivar una división útil de los *n* objetos en un número de clusters. Un cluster es una colección de objetos que son similares a los otros del mismo cluster, basado en los valores de sus atributos, y puede ser tratado colectivamente como un grupo. El agrupamiento es útil para obtener una distribución interna del conjunto de datos [FUN02].

Un algoritmo de agrupamiento intenta encontrar grupos naturales de los datos basado en la similitud de los atributos. A continuación mencionaremos algunos requerimientos típicos del agrupamiento en la minería de datos [FUN02]:

Escalabilidad: Ser capaces de funcionar correctamente con grandes volúmenes de datos.

Alta dimensionalidad: Ser capaces de trabajar con espacios de gran dimensionalidad, descritos por un gran número de atributos.

Formas arbitrarias de los clusters: Ser capaces de asimilar diferentes formas de clusters en dependencia de las medidas utilizadas.

No ser sensitivos al orden de los datos de entrada: El orden de los objetos no puede influir en la calidad del agrupamiento a obtener.

Manipulación de datos ruidosos: Ser capaces de minimizar el impacto de los datos ruidosos.

Conocimiento previo del dominio: Ser capaces de reducir el número de parámetros iniciales que influyen en la calidad del agrupamiento y que deben ser suministrados por los usuarios, requiriéndose en conocimiento previo del dominio (e.g. especificar número de clusters a obtener).

1.3.2 Acerca de las principales técnicas de agrupamiento

Existen varias técnicas de agrupamiento, entre ellas, técnicas de agrupamiento incompleto o heurístico, técnicas de agrupamiento duro y determinista, técnicas de agrupamiento duro y con solapamiento, técnicas de agrupamiento probabilísticas, técnicas de agrupamiento borroso, técnicas de agrupamiento jerárquico, técnicas de agrupamiento basadas en funciones objetivos y técnicas de estimación de grupos [HOP99] [CHE98]. A continuación las describiremos brevemente:

Agrupamiento incompleto o heurístico: Aquí se utilizan métodos geométricos y técnicas de representación o proyección. Los datos multidimensionales son analizados por la reducción de la dimensión usando *principal component analysis* para obtener una representación gráfica en dos o tres dimensiones. Los clusters son determinados posteriormente (e.g. utilizando métodos heurísticos basados en la visualización de los datos).

Agrupamiento duro y determinista: Con estas técnicas cada dato puede ser asignado exactamente a un cluster de modo que la partición de clusters defina una partición ordinaria del conjunto de los datos.

Agrupamiento duro y con solapamiento: Cada dato será asignado a un cluster al menos, o puede ser simultáneamente asignado a varios clusters.

Agrupamiento probabilístico: Para cada dato, se determina una probabilidad de distribución sobre los clusters que especifica la probabilidad con la cual un dato es asignado a un cluster. Estas técnicas también son llamadas algoritmos de agrupamiento borrosos si las probabilidades son interpretadas como grados de pertenencia.

Agrupamiento borroso: Estas técnicas son algoritmos de agrupamiento borroso puro. Los grados de pertenencia indican en qué medida un dato pertenece a los clusters. La suma de las pertenencias de cada dato a todos los clusters es igual a uno.

Agrupamiento jerárquico: Estas técnicas agrupan los datos en un árbol de clusters. Un método es dividir los datos en varios pasos obteniendo clases con una granularidad cada vez más fina (*divisive*, construye el árbol *top-down*), y el otro, construye el árbol de una forma inversa, combinan pequeñas clases en otras con una granularidad más gruesa (*agglomerative*, construye el árbol *bottom-up*).

Agrupamiento basado en funciones objetivos: Estos métodos se basan en una función objetivo o de evaluación que asigna a cada posible partición un valor de calidad o error que tiene que ser optimizado. La solución ideal es la partición cluster que obtenga la mejor evaluación. Esta técnica nos conduce a un problema de optimización.

Agrupamiento de estimación de clusters: Estas técnicas adoptan una alternativa del algoritmo de optimización usado en la mayoría de métodos basados en función objetivo, pero usa ecuaciones heurísticas para construir particiones y estimar parámetros de cluster.

1.3.3 Diferentes medidas para comparar objetos

Al agrupar los objetos de un conjunto de datos, se requieren algunas medidas para cuantificar el grado de asociación entre ellos. Con este propósito, podemos utilizar una medida de distancia, o una medida de similitud o disimilitud. Algunos algoritmos de agrupamiento tienen un requerimiento teórico para el uso de una medida específica, pero lo más común es que el investigador seleccione qué medida utilizará con determinado método.

Una distancia ampliamente utilizada es la Euclidiana [GAB04]. Siguiendo esta distancia, podemos considerar que dos objetos son similares si la distancia entre ellos es cercana a cero y diferentes si la distancia entre ellos es cercana a uno. La distancia Euclidiana entre dos objetos O_i y O_j, descritos por k rasgos, es calculada a partir de la siguiente fórmula:

$$D(O_i, O_j) = \sqrt{\sum_{h=1}^{k} (O_{ih} - O_{jh})^2}$$

La métrica *Minkowski* es también ampliamente referenciada en la literatura [BER01]. Siguiendo esta métrica, se calcula la distancia entre dos objetos O_i y O_j, descritos por k rasgos y donde se cumple que $\gamma \geq 1$:

$$D(O_i, O_j) = \left(\sum_{h=1}^{k} |O_{ih} - O_{jh}|^{\gamma} \right)^{\frac{1}{\gamma}}$$

Cuando $\gamma = 1$, la forma de calcular la distancia entre los objetos se llama la métrica *Manhattan*. Si $\gamma = 2$, nos referimos a la distancia Euclidiana. Para los valores $\gamma \geq 2$, estamos en presencia de la métrica *Supermum* [BER01].

Otra forma de medir la distancia entre dos objetos O_i y O_j, descritos por k rasgos, es la métrica Canberra, calculada a partir de la siguiente fórmula [BER01]:

$$D(O_i, O_j) = \sum_{h=1}^{k} \frac{\left| \overrightarrow{O_{ih}} - \overrightarrow{O_{jh}} \right|}{\left(\left| \overrightarrow{O_{ih}} \right| + \left| \overrightarrow{O_{jh}} \right| \right)}$$

Otra forma muy utilizada de medir la distancia entre objetos es la distancia de *Hamming*. Tanto en su definición binaria a partir de la cantidad de atributos en que difieren dos objetos, como en sus variantes pesadas [DUC00]. En [KAR01] y en [DUC00] se presentan formas de medir la similitud entre objetos, considerando la medida como una relación asimétrica y teniendo en cuenta la probabilidad condicional de un objeto respecto al otro. Otro enfoque se muestra en [SAR00] [WIL97], donde obtienen la similitud entre dos objetos O_i y O_j calculando la correlación de *Pearson* entre ellos:

$$D(O_i, O_j) = \frac{\sum_{h=1}^{k} \left(O_{ih} - \overline{atributo_h} \right) \left(O_{jh} - \overline{atributo_h} \right)}{\sqrt{\sum_{h=1}^{k} \left(O_{ih} - \overline{atributo_h} \right)^2 \sum_{h=1}^{k} \left(O_{jh} - \overline{atributo_h} \right)^2}}$$

donde $\overline{atributo_h}$ es el valor promedio que toma el $atributo_h$ en el conjunto de datos.

Existe un gran número de medidas de similitud disponibles, sin embargo, existen pocos estudios comparativos de ellas y sus efectos en el agrupamiento. En la minería de textos, al comparar

documentos con el objetivo de agruparlos, la determinación de la similitud entre documentos depende de la representación del documento y de los pesos que se le asignen al caracterizarlo. A partir de estudios realizados, se ha demostrado que al agrupar documentos, los coeficientes Dice, Jaccard y Coseno, han reportado los mejores resultados [FRA92] [CRO02]. Consideramos que los dos objetos a comparar son los documentos D_i y D_j, con pesos asociados a los k términos que los describen y calcularemos la similitud S entre ellos.

El coeficiente Dice se define como sigue:

$$S(D_i, D_j) = \frac{2\sum_{h=1}^{k}(peso_{ih} \cdot peso_{jh})}{\sum_{h=1}^{k} peso_{ih}^2 + \sum_{h=1}^{k} peso_{jh}^2}$$

Si el peso de los términos es binario, el coeficiente Dice se reduce a

$$S(D_i, D_j) = \frac{2C}{A+B}$$

donde C es el número de términos que D_i y D_j tienen en común, y A y B son el número de términos de D_i y D_j respectivamente.

Siguiendo la notación anterior, el coeficiente Jaccard se define como sigue:

$$S(D_i, D_j) = \frac{\sum_{h=1}^{k}(peso_{ih} \cdot peso_{jh})}{\sum_{h=1}^{k} peso_{ih}^2 + \sum_{h=1}^{k} peso_{jh}^2 - \sum_{h=1}^{k}(peso_{ih} \cdot peso_{jh})}$$

En [GAS01] se muestra variantes binaria y pesada de la medida de Jaccard para determinar la similitud entre palabras, no obstante, esta medida puede ser extendida para comparar otro tipo de contexto sintáctico. A continuación describiremos las variantes de la medida para comparar documentos.

La medida de Jaccard binaria se puede utilizar para comparar dos documentos teniendo en cuenta los términos que ellos comparten:

$$S(D_i, D_j) = \frac{C}{A+B}$$

donde *A* es el número de palabras que describen a D_i, *B* es el número de palabras que describen a D_j y *C* es el número de palabras que aparecen tanto en D_i como en D_j.

Una variante de la medida de Jaccard pesada es considerar un peso local (*PL*) y otro global (*PG*) para cada término que describe los documentos.

El peso global tiene en cuenta como muchos documentos diferentes se asocian con un término dado, y se calcula con la siguiente fórmula:

$$PG(palabra_h) = 1 - \sum_{l=1}^{N} \frac{|p_{lh} \log p_{lh}|}{m_h}$$

donde *N* es el número total de documentos en la colección, m_k es el número de documentos en los cuales la $palabra_h$ ocurre y p_{lh} se calcula según la siguiente expresión:

$$p_{lh} = \frac{\text{frecuencia de la } palabra_h \text{ en } D_l}{\text{cantidad total de palabras en } D_l}$$

El peso local se calcula como:

$$PL(palabra_h, D_j) = \log(\text{frecuencia de la } palabra_h \text{ en } D_j)$$

El peso de un término es la multiplicación del peso global y el peso local:

$$peso(palabra_h, D_i) = PG(palabra_h) \cdot PL(palabra_h, D_i)$$

La similitud Jaccard entre dos atributos, siguiendo este cálculo de los pesos se plantea como:

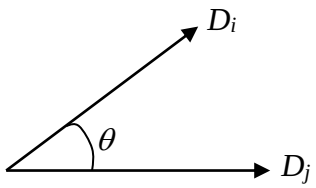
$$S(D_i, D_j) = \frac{\sum_{h=1}^{k} \min(peso(palabra_h, D_i), peso(palabra_h, D_i))}{\sum_{h=1}^{k} \max(peso(palabra_h, D_i), peso(palabra_h, D_i))}$$

El coeficiente Coseno se define en la siguiente expresión [FRA92]:

$$S(D_i, D_j) = \frac{\sum_{h=1}^{k} (peso_{ih} \cdot peso_{jh})}{\sqrt{\sum_{h=1}^{k} peso_{ih}^2 \cdot \sum_{h=1}^{k} peso_{jh}^2}}$$

Este coeficiente es ampliamente utilizado al determinar la similitud entre documentos y se basa en el coseno del ángulo que existe entre ellos. Si el coseno del ángulo es cercano a uno, se consideran

similares, y si es cercano a cero, los documentos son considerados diferentes. En la siguiente figura se muestra la gráfica de los vectores documentos, así como la expresión que describe el coseno del ángulo entre ellos.



$$\cos\theta = \frac{\overrightarrow{D_i} \cdot \overrightarrow{D_j}}{\|D_i\| \cdot \|D_j\|}$$

1.3.4 Principales algoritmos de agrupamiento en la minería de textos

A continuación haremos referencia a los principales algoritmos de agrupamiento que se han utilizado en la minería de textos, específicamente en el agrupamiento de documentos similares. Particularizaremos en aquellos que aplican técnicas duras y deterministas (cada documento es asignado a uno y solo un grupo, se forma una partición de la colección de documentos), las técnicas duras y con solapamiento (un documento puede ser asignado a más de un grupo), y las técnicas de agrupamiento borroso (cada documento es asignado a todos los grupos, pero con un grado de pertenencia asociado).

Generalmente en una colección de documentos, un documento no pertenece a un único grupo (e.g. agrupamiento de noticias, textos de divulgación científica, artículos científicos), por tanto el agrupamiento con solapamiento es muy útil, sin embargo, éste no da la posibilidad de conocer cuánto pertenece cada documento a cada grupo. En tal sentido, el agrupamiento borroso nos dará la posibilidad de agrupar los textos de un corpus de forma tal que conozcamos el grado de pertenencia de los textos a cada grupo. Una técnica de agrupamiento que tiene gran importancia en la minería de textos es el agrupamiento jerárquico, debido a que da la posibilidad de conocer cuán cercanos están los grupos obtenidos.

Un método muy utilizado es aplicar una red de Kohonen (*Self-Organizing Maps*) para agrupar documentos. En [NUR01] se utiliza esta red para agrupar documentos similares, donde no es necesario especificar el número de grupos que se desea obtener. Además se obtienen las palabras claves que describen cada grupo. Una deficiencia es que hay que definir el tamaño de la red manualmente.

Otro ejemplo de algoritmo de agrupamiento duro y determinista es Autoclass [LAR00]. Este, al igual que el anterior, no requiere especificar el número de grupos a obtener y además de obtener los

grupos de textos, obtiene las palabras claves de cada grupo. Este algoritmo ha aportado mejores resultados que la red Kohonen en el agrupamiento de documentos.

Un clásico en el análisis de clusters es el algoritmo *k-means* [QUE67] [JAN97] [BER04], éste tiene la desventaja que requiere que el número de clusters a obtener sea especificado a priori, por tanto requiere un cierto conocimiento del dominio, ya que es sensible a cómo se hizo inicialmente la partición. El algoritmo *k-means* agrupa los documentos teniendo en cuenta la similitud que existe entre ellos. A partir de este algoritmo se han derivado varios como el *Batch k-means* e Incremental *k-means*. El primero de estos tiene dos desventajas, una de ellas es que la calidad de la partición final depende de una buena selección de la partición inicial y además puede quedar atrapado en mínimos locales. El segundo, el Incremental *k-means*, resuelve las dos desventajas del *Batch k-means*, pero es más lento. Realizando mejoras a estos dos últimos algoritmos se obtuvo el algoritmo *Means* [BER04].

En [BAO00a] [BAO00b] se muestra un nuevo método de agrupamiento de documentos que utiliza una extensión de la teoría clásica de los conjuntos aproximados (*Rough Set Theory*), llamada *Tolerance Rough Set* que permite formar clases de tolerancia de las palabras y utilizarlas para realizar el agrupamiento de documentos.

Los algoritmos mencionados hasta el momento son duros y deterministas. Por la importancia en la minería de textos de los algoritmos jerárquicos, mencionaremos a uno reciente, el algoritmo *Principal Direction Divisive Partitioning* (PDDP) que no requiere que el número de clusters sea fijado inicialmente porque hace una subdivisión sucesiva de la colección inicial hasta detenerse cuando se cumpla cierto criterio de calidad [BOL98]. Este algoritmo no es basado en ninguna medida de distancia ni de similitud, y toma como ventaja lo dispersa que es la matriz de términos por documentos. Devuelve grupos de documentos, pero generalmente su salida es utilizada como entrada al algoritmo *Means* para mejorar su eficiencia y no tener que fijar manualmente el número inicial de clusters ni la partición inicial [BER04]. Otro algoritmo que usualmente se combina con el algoritmo *Means* es el algoritmo *Spherical Principal Directions Divisive Partitioning* (sPDDP), también jerárquico y que sigue la idea de [BOL98]. Este algoritmo no requiere que el número de clusters sea fijado inicialmente y mejora los resultados de PDDP. En [BAO00a] [BAO00b] también se muestra una variante del algoritmo duro y determinista que utiliza el modelo *Tolerance Rough Set*, para realizar un agrupamiento jerárquico.

El algoritmo *Ando*, también reportado en [BER04], es un algoritmo para el agrupamiento con objetivos específicos, ya que su idea general es identificar clusters pequeños en contextos limitados.

Este algoritmo tiene varias desventajas, una de las principales es que es poco escalable y funciona utilizando *Latent Semantic Indexing* y los vectores bases no son siempre ortogonales, aspecto esencial al buscar los valores y vectores propios.

Al agrupar documentos es muy útil obtener simultáneamente los grupos de documentos y las palabras claves de cada grupo. Esta práctica ayuda a dividir la colección de documentos en categorías más significativas y puede ser usada para generar automáticamente una descripción compacta de cada cluster en términos no solamente de los valores de los atributos, sino también de su relevancia.

Siguiendo la idea antes expuesta, el algoritmo *Simultaneous Clustering and Attribute Discrimination* (SCAD), es un algoritmo que utiliza una técnica dura y determinista logrando, simultáneamente, obtener los grupos de documentos similares y pesar los rasgos. SCAD tiene varias ventajas, una de ellas es pesar los rasgos continuos, lo cual provee una representación de la relevancia de los rasgos más rica que la selección de los rasgos binaria. Además, SCAD aprende una representación de la relevancia de los rasgos diferente para cada cluster. Este algoritmo utiliza la distancia Euclidiana para calcular la similitud entre documentos. Para el caso especial de documentos textuales es bien conocido que la distancia Euclidiana no es apropiada, de forma general, esta medida no es buena en problemas de alta dimensionalidad. En [BER04] se muestra una extensión de este algoritmo que permite simultáneamente agrupar documentos textuales y dinámicamente pesar el conjunto de palabras claves. Este nuevo enfoque es llamado *Simultaneous Keyword Identification and Clustering of text documents* (SKWIC), y es conceptualmente y computacionalmente simple. Este algoritmo es una extensión del *k-means* y funciona mejor que éste cuando no todos los rasgos son igualmente relevantes. SKWIC utiliza una medida de disimilitud basada en el coeficiente coseno para comparar los documentos, aunque puede ser adaptado a otras medidas de disimilitud. También la forma de pesar los términos puede ser modificada. Este algoritmo requiere que sea especificado el número inicial de clusters y devuelve la colección de clusters y la relevancia de las palabras en cada cluster obtenido [BER04].

Es muy importante al agrupar colecciones de textos utilizar algoritmos que no creen particiones, porque varios documentos de una colección pueden abordar más de un tema, por tanto pertenecer a varios clusters. Los algoritmos que aplican técnicas duras y con solapamiento pueden resolver este problema. Ejemplos de ellos son el algoritmo *Star* [ASL98] y el algoritmo *Extended Star* [GIL03]. Ambos algoritmos tiene la ventaja que no requieren que el número de clusters sea especificado inicialmente. Sin embargo, el primero de ellos tiene la desventaja que el algoritmo depende del

orden de los datos y esto puede provocar la construcción de clusters ilógicos. El algoritmo *Extended Star* supera esta desventaja, es decir, en la formación de los clusters no influye el orden de los datos.

Los algoritmos duros y con solapamiento ayudan a resolver el problema de la inclusión de un documento a más de un grupo, pero no se conoce en qué grado pertenecen los documentos a los grupos. Por tal motivo, es muy útil realizar el análisis de clusters aplicando algoritmos de agrupamiento borrosos. Un clásico de los algoritmos de agrupamiento siguiendo esta técnica es el *Fuzzy c-means* [BEZ73] [JAN97] [BER04], que reconoce nubes esféricas de puntos en un espacio p-dimensional, y asigna un grado de pertenencia de los documentos a los clusters. Siguiendo la idea del *Fuzzy c-means* se han desarrollado varios algoritmos, entre ellos el algoritmo *Relational Alternating Cluster Estimation* [RUN01] y el algoritmo *Simultaneous Soft Clustering and Term Weighting of Text Document* (Fuzzy SKWIC) [BER04]. Ambos algoritmos requieren que el número de clusters sea especificado inicialmente y además de devolver la colección de clusters, calculan simultáneamente la relevancia de las palabras en los grupos. Estos algoritmos tienen gran utilidad en el agrupamiento de documentos.

Capítulo 2 Diseño general de módulos que permiten el agrupamiento y resumen en CorpusMiner

En este capítulo mencionaremos inicialmente los principales conceptos de la programación orientada objetos y del Lenguaje Unificado de Modelación (*Unified Modeling Language* (UML)) que nos permitirán describir posteriormente el diseño general de CorpusMiner. Particularizaremos en los módulos del sistema que permiten el agrupamiento de documentos y la obtención del resumen extracto de cada grupo. Describiremos cada uno de los algoritmos, así como la forma en que fueron implementados en CorpusMiner.

2.1 Principales conceptos en el diseño e implementación orientados a objetos

Es importante tratar algunos conceptos fundamentales de la programación orientada a objetos (POO), el uso de interfaces, así como aspectos sobre la metodología de análisis y diseño orientada a objetos UML utilizada durante las etapas de desarrollo del sistema [RUM97a] [RUM97b].

2.1.1 Conceptos generales de la programación orientada a objetos

Los módulos que en este capítulo se describen son realizados utilizando el paradigma de programación orientada a objetos. Es por eso que es necesario precisar cuáles son las características particulares y propiedades que identifican los objetos que son utilizadas en este diseño. En [ROD99] se presentan las siguientes propiedades de los objetos:

Estado: Se define a partir de los valores que en un momento dado tienen los atributos del objeto. La estructura del objeto se define como el conjunto de todos los atributos o propiedades. Además un objeto puede conocer o contener a otros objetos, estas relaciones son también parte de su estado.

Comportamiento: Define cómo actúan los objetos frente a estímulos externos en términos de cambio de estados.

Identidad: Esta es la propiedad de un objeto que lo distingue del conjunto de todos los demás objetos del universo al que pertenece. Los modelos de POO son representaciones abstractas de este tipo.

El protocolo de objeto define la envoltura del comportamiento admitido por el objeto, representa todas las vistas estáticas y dinámicas del objeto. Para la mayoría de las abstracciones no triviales, es útil dividir los protocolos en grupos lógicos de comportamiento. Las colecciones que dividen el espacio del comportamiento de un objeto denotan los roles que un objeto puede jugar. Un role es una máscara con la cual se presenta y define un contrato entre la abstracción y sus clientes.

El marco de referencia conceptual en un sistema orientado a objeto es el modelo de objetos que incluye cuatro conceptos fundamentales que son: la abstracción, el encapsulamiento, la modularidad y la jerarquía. También existen otros conceptos secundarios dignos a tener en cuenta que son: los tipos, la concurrencia y la persistencia.

Abstracción: Denota las características esenciales de un objeto que lo distinguen de todos los demás tipos de objeto y proporciona así fronteras conceptuales nítidamente definidas respecto a la perspectiva del observador.

Encapsulamiento: Es uno de los principios más importantes de la POO, ha permitido la reusabilidad de objetos. Constituye el proceso de almacenar en un mismo compartimiento los elementos de una abstracción que constituyen su estructura y su comportamiento. El cliente se interesa por lo que hace el objeto y no cómo lo hace.

Modularidad: Es la propiedad que tiene un sistema que ha sido descompuesto en un conjunto de módulos cohesivos y débilmente acoplados.

Jerarquía: Es una clasificación u ordenamiento de abstracciones.

Concurrencia: Es la propiedad que distingue un objeto activo de uno que no está activo.

Persistencia: Es la propiedad de un objeto por la que su existencia trasciende el tiempo, el espacio, o ambos.

En [ROD99] se definen los conceptos de clase y tipo. La clase no es más que una representación abstracta que define la estructura y el comportamiento que le son comunes a un grupo de objetos. Mientras que el tipo es un protocolo usado en los mecanismos de comunicación e interacción entre objetos. Tiene identidad y generalmente está más relacionado a los mecanismos de comunicación que a la propia naturaleza de los objetos.

2.1.2 Interfaces

El uso de las interfaces en la programación orientada a objetos es una tendencia actual. Ellas esclarecen el diseño, lo hacen más cercano a la realidad y facilitan la implementación. En CorpusMiner ha sido muy efectivo en el diseño el uso de interfaces.

Cuando un objeto “implementa una interfaz” ese objeto implementa cada función miembro de la interfaz. Los objetos pueden, por supuesto, soportar simultáneamente múltiples interfaces. Las interfaces son inmutables, debido a que nunca pueden ser versionadas por lo que esto elimina los problemas de versionamiento. Una nueva versión de una interfaz, creada por la adición o

eliminación de funciones, o cambios semánticos, es enteramente una nueva interfaz a la cual se le asigna un nuevo identificador (ID) único. Por lo que la nueva interfaz no va a crear "conflictos" con la vieja.

La funcionalidad de encapsular los objetos accedidos a través de interfaces hace al sistema abierto y extensible. Es abierto en el sentido de que cualquiera puede proveer una implementación de una interfaz definida y cualquiera puede desarrollar una aplicación que use dichas interfaces y es extensible en el sentido de que interfaces nuevas, o extendidas puedan ser definidas sin cambiar las aplicaciones existentes y estas aplicaciones entienden las nuevas interfaces y las explotan mientras continúan interoperando con las viejas aplicaciones a través de las interfaces viejas.

La existencia de múltiples interfaces, nos lleva a una cuestión importante. Cuando un objeto cliente accede inicialmente a una interfaz, la pregunta es cómo ese cliente puede acceder a las demás interfaces del mismo objeto. Esta pregunta la realiza el cliente a través de la interfaz *IUnknown*, la cual posee una función miembro llamada *QueryInterface* y puede ser llamada en cualquier momento. *QueryInterface* está presente en *IUnknown* junto a dos métodos: *_AddReference* y *_Release* los cuales son encargados de controlar el tiempo de vida del objeto.

QueryInterface es la base para el proceso, llamado negociación de interfaces, que se realiza cada vez que un cliente le pide al objeto los servicios que este es capaz de proveer. La cuestión es pedir mediante un llamado al *QueryInterface* y pasarle a esa función el GUID de la interfaz solicitando a través de él los servicios de interés.

Esto trabaja de la siguiente manera: cuando un cliente gana acceso al objeto, este cliente recibirá como mínimo un puntero a la interfaz *IUnknown* (la interfaz más importante) mediante la cual él solo puede tener control del tiempo de vida del objeto y de ahí invocar al *QueryInterface*.

El objeto que implementa el *QueryInterface* tiene la opción de aceptar o desechar el pedido. Si se acepta el pedido, *QueryInterface* retorna un nuevo puntero a la interfaz requerida y así el cliente tiene acceso a los métodos de dicha interfaz. De desechar el pedido se retorna un puntero vacío o un error y el cliente no puede acceder a las funciones deseadas.

2.1.3 La metodología de análisis y diseño orientados a objetos

Las metodologías de análisis y diseño orientadas a objetos, como el resto de las metodologías de la ingeniería de software tratan de establecer pautas para el desarrollo de sistemas; indican los pasos a seguir durante las diferentes etapas por las que transita un software en su construcción.

Generalmente en las metodologías existen tres etapas bien definidas que son: la etapa de análisis, la etapa de diseño y la de implementación [KEN97].

Según Grady Booch [BOO99][BOO91]:

> "El análisis orientado a objetos es un método de análisis que examina los requisitos desde la perspectiva de las clases y objetos que se encuentran en el vocabulario del dominio del problema".
>
> "El diseño orientado a objetos es un método de diseño que abarca el proceso de descomposición orientado a objetos y una notación para describir los modelos lógico y físico, así como los modelos estático y dinámico del sistema que se diseña".
>
> "En el análisis de un problema se deben examinar los roles que un objeto puede jugar. Durante la etapa de diseño se refinan estos roles inventando las operaciones que llevan a cabo las responsabilidades de cada role".

El diseño de los módulos de CorpusMiner que en este trabajo se muestra, incluye los diagramas de clases así como las relaciones que existen entre las clases siguiendo las especificaciones de UML [BOO99].

Un diagrama de clases muestra la existencia de clases en el sistema y las relaciones entre las clases. Entre los elementos de modelación que se pueden encontrar en los diagramas de clases están:

- Clases con su estructura y comportamiento.
- Asociaciones, agregaciones, dependencias, y relaciones de herencia entre las clases.
- Multiplicidad e indicadores de navegación.

Al definir las clases es necesario tener en cuentas las operaciones que permiten representar el comportamiento de una clase y los atributos que representan su estructura.

No solo es necesario conocer como se definen correctamente los objetos, sino qué relaciones se pueden establecer entre ellos. En [BOO99] se definen tres tipos de relaciones entre objetos:

Asociación: Es una conexión bidireccional entre clases. Se muestra en el diagrama una línea conectando las clases relacionadas.

Agregación: Es una relación existente entre una pieza y sus partes. Una agregación se representa a través de una línea que conecta las clases relacionadas con un diamante en el extremo de la clase que representa a la clase agregada.

Dependencia: Es una forma débil de relación entre un cliente y un proveedor cuando el cliente no tiene un conocimiento semántico del proveedor. Una dependencia se representa como una línea discontinua desde el cliente hasta el suministrador.

Conceptos también muy utilizados son la multiplicidad y la navegación [BOO99]:

Multiplicidad: Es el número de instancias de una clase relacionadas con una instancia de otra clase y define cuantos objetos participan en una relación. Para cada asociación y agregación existen dos multiplicidades a definir, una por cada extremo de la relación.

Navegación: Las asociaciones y las agregaciones son en ambas direcciones por defecto, a menudo es deseable restringir la navegación en un solo sentido. Si la navegación es restringida es necesario añadir una flecha indicando el sentido de la misma**.**

La herencia la hemos usado en nuestro diseño y no es más que la relación entre una superclase y sus subclases. Existen dos formas de encontrar herencia: generalización y especialización. Así, los atributos comunes, operaciones y relaciones son mostrados en las clases de mayor nivel en la jerarquía.

Un elemento que facilita la organización del diseño en UML son los paquetes. Un paquete es un agrupamiento de elementos del modelo. Los paquetes en sí mismos pueden estar compuestos por otros paquetes. Un paquete puede contener paquetes subordinados y elementos ordinarios del modelo.

2.2 Generalidades del sistema CorpusMiner

La herramienta CorpusMiner permite obtener un resumen extracto de un corpus textual partiendo de la verificación de la homogeneidad del mismo utilizando métodos de agrupamiento. En este trabajo focalizaremos la atención en la realización del agrupamiento y la obtención del resumen extracto, no obstante, es necesario mencionar cada una de las etapas por las que transita el corpus de textos para comprender un procesamiento posterior. La entrada al módulo que permite agrupar los documentos es una representación VSM (*Vector Space Model*) del corpus.

Un elemento muy importante para obtener buenos resultados de agrupamiento es realizar una adecuada representación del corpus, que incluye la transformación del corpus, la extracción de términos, la reducción de la dimensionalidad (selección de rasgos y reparametrización), y la generación de vectores [LAN01]. Por tanto, es necesario en CorpusMiner transitar por varias etapas que permitan la representación del corpus, así como un procesamiento posterior para obtener su resumen extracto. En la figura 2.1 se muestran todas las etapas que conforman a CorpusMiner, desde

la especificación del corpus original hasta la obtención de su resumen extracto. Nótese que los módulos resaltados en la figura 2.1 son los que abordaremos en este trabajo.

En una primera etapa se realiza la transformación de un corpus de textos escrito en idioma Inglés. En CorpusMiner asumimos que la colección de documentos se encuentra almacenada en un fichero texto donde cada línea representa una oración y existen delimitadores de párrafos y textos. A partir de una colección con estas características se obtienen *tokens* a los cuales se les aplican transformaciones. Esta etapa incluye la transformación de todos los caracteres a mayúscula, la lematización (sustitución de cada forma lingüística por el lexema correspondiente), la sustitución de las contracciones por sus expansiones, de las abreviaturas por sus formas completas, la homogeneización de las convenciones ortográficas (según las normas británicas o las norteamericanas), así como la eliminación de números y símbolos [LAN01] [VAL05].

Luego, en una segunda etapa se obtiene una representación VSM del corpus textual a partir de los *tokens* indexados. Dicha matriz puede conformarse con términos indexados en sus filas y todos los documentos de la colección en sus columnas, donde las celdas representan la frecuencia de aparición de cada término en cada documento, o puede construirse considerando en las columnas cada una de las oraciones que conforman los documentos del corpus, por tanto cada celda corresponde a la frecuencia de aparición de los términos en la oración. Este módulo permite aplicar una variante de reducción de dimensionalidad por selección de rasgos, permitiendo o no la eliminación de las palabras gramaticales.

La representación VSM que se obtiene en [VAL05] para comenzar el proceso de agrupamiento puede tener todos los términos indexados con sus respectivas frecuencias absolutas en un contexto textual (e.g. oración, párrafo o documento), o puede haber sido sometida a una o varias de las tres acciones que describiremos a continuación. Una de ellas es calcular la calidad de términos que influye en una segunda acción que es reducir la dimensionalidad, y la última es normalizar y pesar la matriz. Estas acciones no tienen un orden predeterminado, pero en nuestra herramienta se sugiere primero normalizar y pesar la matriz y después calcular la calidad de los términos y aplicar técnicas de reducción de dimensionalidad, excepto para algunos criterios de calidad y reductores asociados a estos que requieren su aplicación con los valores de frecuencia absoluta de aparición de los términos. En el capítulo 3 de este trabajo mostraremos los resultados obtenidos por los métodos de agrupamiento a partir de cada una de estas variantes.

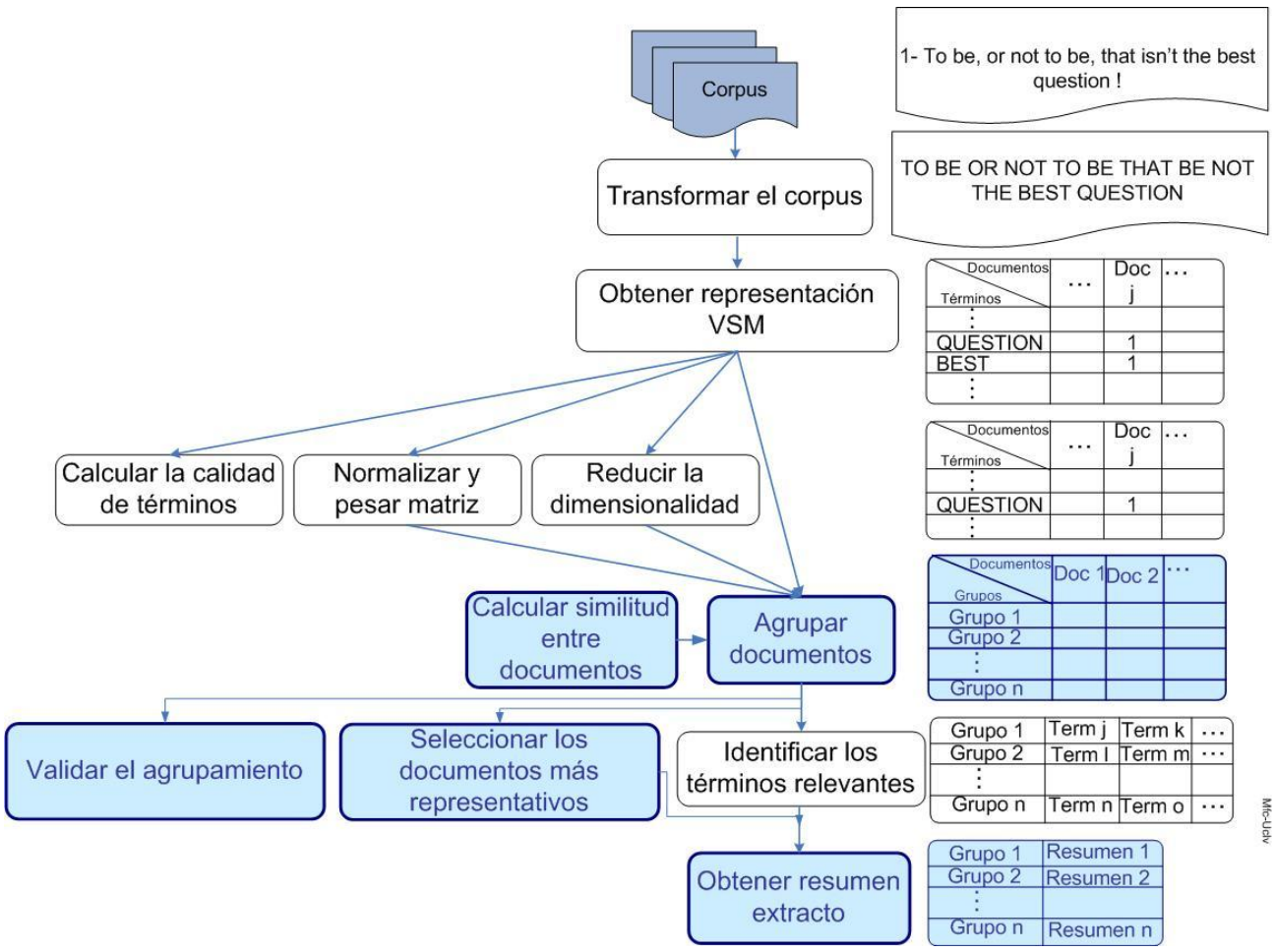


Figura 2.1 Etapas en el procesamiento de una colección de textos en CorpusMiner

En CorpusMiner la representación VSM puede ser normalizada dividiendo cada frecuencia absoluta de aparición por el número de tokens en cada documento [VAL05] y pesada a partir de la aplicación de dos formas de calcular *Term Frequency Inverse Document Frequency Weighting* (TF-IDF) [MAN00] [BER04].

Al procesar documentos es conveniente conocer la calidad de los términos, porque esto puede influir en la reducción de la dimensionalidad, que es una etapa vital en la representación de textos. En CorpusMiner hay varias medidas que permiten calcular la calidad de términos, ellas son: entropía [NUR01], skewness y kurtosis [FUK99], calidad de términos 1 y 2 [BER04]. Además, en el sistema se incluyen cuatro criterios que permiten clasificar los términos en tres categorías en función de su importancia y considerando el umbral de frecuencia de términos y documentos [VAL05]. A partir de las medidas de calidad de términos, los reductores de dimensionalidad logran seleccionar aquellos términos con mayor calidad, siguiendo alguno de estos criterios. Es necesario destacar que en

nuestra herramienta la reducción de dimensionalidad no es solo basada en medidas de calidad de términos.

Partiendo de una representación VSM de la colección de documentos, ya sea modificada o no por la aplicación de alguna técnica de normalización, pesado de la matriz, reducción de dimensionalidad o combinación de estas, es posible agrupar aquellos documentos que sean similares por su contenido. Los métodos de agrupamiento incluidos en CorpusMiner son *Simultaneous Keyword Identification and Clustering of Text Documents* (SKWIC) [BER04], *Simultaneous Keyword Identification and Fuzzy Clustering of Text Documents* (Fuzzy SKWIC) [BER04], y *Extended Star* [GIL03]. En cualquier caso se obtiene como resultado una colección de clusters de documentos. Estos algoritmos de agrupamiento requieren calcular la similitud que existe entre los documentos, es por eso que el módulo de agrupamiento está estrechamente relacionado con el de cálculo de similitud entre vectores. En los algoritmos SKWIC y Fuzzy SKWIC se calcula la distancia entre los vectores documentos a partir de una variante del cociente Coseno. En el algoritmo *Extended Star* es posible utilizar la similitud Coseno [FRA92] y las distancias de Jaccard binaria y pesada [GAS01] para comparar los vectores. La distancia Euclidiana [GAB04] también se encuentra implementada, pero en la literatura se reporta que no da buenos resultados en dominios textuales. La descripción, diseño e implementación de los algoritmos involucrados en ambos módulos aparece detallada en epígrafes posteriores.

El módulo evaluador permite reflejar la calidad de los resultados del agrupamiento a partir de la aplicación de las medidas de entropía, *Overall Similarity*, *precision*, *recall y Fmeasure* [SEB99] [STE00]. Los resultados con las distintas variantes de obtención y modificación de la representación VSM, así como de las técnicas de agrupamiento y medidas de similitud se exponen en el capítulo 3.

La reducción de dimensionalidad no es sólo una etapa de la representación textual, ella puede estar presente en otras fases del procesamiento de textos. En CorpusMiner, a partir de los resultados del agrupamiento, se requiere seleccionar aquellos términos que son relevantes y caracterizan cada cluster obtenido, para así, obtener un resumen extracto de cada tema que abordan los documentos de la colección. Existen tres formas en el sistema de seleccionar los términos relevantes de cada cluster: seleccionar los términos más relevantes a partir de los resultados del agrupamiento (esto solo es posible para los algoritmos SKWIC y Fuzzy SKWIC que devuelven la relevancia de los términos a cada cluster obtenido), seleccionar los términos que logran discernir entre clusters a partir de la aplicación del algoritmo ID3 [MIT97] [VAL05], o a partir de la intersección de ambos resultados [VAL05].

El último módulo de CorpusMiner es descrito también en este capítulo y permite obtener un resumen extracto de cada uno de los clusters formados. El resumen se logra a partir de la extracción de aquellas oraciones que tienen la presencia de las palabras claves de cada cluster. La extracción de oraciones por clusters puede considerar todos los documentos que lo forman o solo aquel más representativo.

A continuación describiremos como se concibió el diseño general de CorpusMiner a partir de las especificaciones mostradas en [VAL05].

2.3 Diseño del sistema CorpusMiner

El diseño del sistema CorpusMiner se dividió en tres capas fundamentales como se muestra en la figura 2.2. La primera capa o inferior es la capa del dominio, la segunda o intermedia es la capa controladora y la tercera o superior es la capa de interfaz de usuario [VAL05].

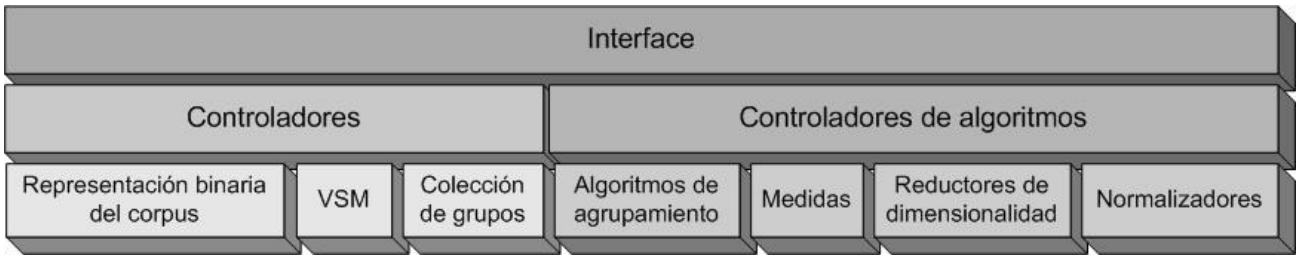


Figura 2.2 Diseño general del sistema CorpusMiner

En la capa inferior están las clases del dominio (i.e. las clases que representan elementos del dominio de aplicación). En esta capa, a su vez, se establecieron dos tipos de clases diferentes. En el primer tipo están aquellas clases que permiten la representación y manipulación de los datos (e.g., la representación VSM, la colección de cluster, las palabras claves de un cluster). Un segundo tipo incluye las clases correspondientes a los algoritmos que operan sobre estos datos (e.g., las medidas de calidad de términos, reductores de dimensionalidad, los algoritmos para normalizar y pesar la representación VSM). Por otra parte, la tercera capa es la encargada de la interfaz visual y posee todas las clases relacionadas con las formas visuales y la interacción con el usuario. La capa intermedia es la que posee todas las clases controladoras y es la encargada de establecer la comunicación entre las clases de las dos capas mencionadas; esta al igual que la primera capa, esta dividida en dos tipos de clases fundamentales las controladoras de las clases de datos y las controladoras de algoritmos.

Descrito en [VAL05], entre las principales clases del sistema se encuentran *TControllerView* de la cual heredan todas las clases controladoras de datos, *TAlgorithmController* y *TAlgorithmListController* de las que heredan todos los algoritmos implementados en CorpusMiner (Ver anexo 1).

2.4 Agrupamiento de documentos

A partir de la revisión bibliográfica realizada sobre los métodos de agrupamiento que son frecuentemente utilizados en la minería de textos, hemos decidido incorporar en CorpusMiner los algoritmos:

- *Simultaneous Keyword Identification and Clustering of Text Documents* (SKWIC) que realiza un análisis de clusters duro y determinista [BER04].
- *Simultaneous Keyword Identification and Fuzzy Clustering of Text Documents* (Fuzzy SKWIC) que utiliza una técnica de análisis de cluster borrosa [BER04].
- *Extended Star* que aplica una técnica de análisis de cluster dura y con solapamiento [GIL03].

Los tres algoritmos implementados tienen en común que parten de una representación *Vector Space Model* (VSM) del corpus textual y devuelven una colección de clusters. La estructura de los clusters varía en dependencia de los algoritmos de agrupamiento.

Los dos primeros nos brindan una colección de clusters, donde cada cluster tiene el vector centro de cluster, una lista de la relevancia de las palabras en el cluster y una lista de documentos con sus particularidades si es SKWIC o Fuzzy SKWIC. En el caso específico de SKWIC, al ser duro y determinista, cada cluster tiene la lista de los documentos que lo conforman y el algoritmo crea una partición. En el caso del algoritmo Fuzzy SKWIC, todos los documentos de la colección pertenecen a todos los clusters, pero con un grado de pertenencia asociado, por tanto la lista de documentos de cada cluster tiene por cada documento su grado de pertenencia al cluster. Existen varios algoritmos que se aplican al agrupamiento de documentos y que utilizan estas técnicas, sin embargo, hemos decidido implementar estos porque ambos logran calcular la relevancia de las palabras a los clusters simultáneamente al proceso de agrupamiento de documentos y porque utilizan una variante del coeficiente Coseno para calcular la disimilitud entre documentos, medida que tiene muy buenos resultados en el agrupamiento de textos. Los documentos, por sus características, pueden pertenecer a más de un grupo, de ahí la utilidad de implementar Fuzzy SKWIC, aunque SKWIC sea implementado. Ambos algoritmos tienen la desventaja que requieren que el número de clusters a obtener sea especificado a priori, por tanto, es necesario tener conocimiento del dominio para poder

definir ese valor y en la mayoría de las aplicaciones no contamos con el conocimiento suficiente. Otra desventaja de estos algoritmos es que en la primera iteración se seleccionan aleatoriamente los centros de clusters, por tanto, lograr que los centros queden estabilizados puede consumir varias iteraciones, además, debido a esa aleatoriedad, el algoritmo SKWIC puede devolver centros de clusters vacíos.

A partir de las desventajas mencionadas, decidimos implementar un tercer algoritmo de agrupamiento, el *Extended Star* [GIL03]. Este algoritmo también retorna una colección de clusters, donde cada cluster tiene señalado el documento que es centro del cluster, así como una lista con los documentos de la colección que pertenecen a él. Una desventaja de este algoritmo es que no calcula, simultáneamente al proceso de agrupamiento, la relevancia de las palabras a los clusters. Una gran ventaja, es que no requiere un conocimiento previo del dominio, porque no es necesario fijar el número de clusters inicialmente. El algoritmo *Extended Star* tiene otra ventaja, y es que permite calcular la similitud entre vectores utilizando diferentes medidas. Este algoritmo se puede utilizar para el agrupamiento de documentos y los mejores resultados se esperan con la similitud Coseno. Independientemente de la calidad del agrupamiento que se pueda obtener con *Extended Star*, la idea de su implementación en CorpusMiner es utilizarlo como un paso previo a los dos algoritmos anteriormente mencionados. Nuestra herramienta permite tomar la salida del método *Extended Star* para inicializar el número de clusters, así como sus centros, en los algoritmos SKWIC y Fuzzy SKWIC. Observe la figura 2.3.

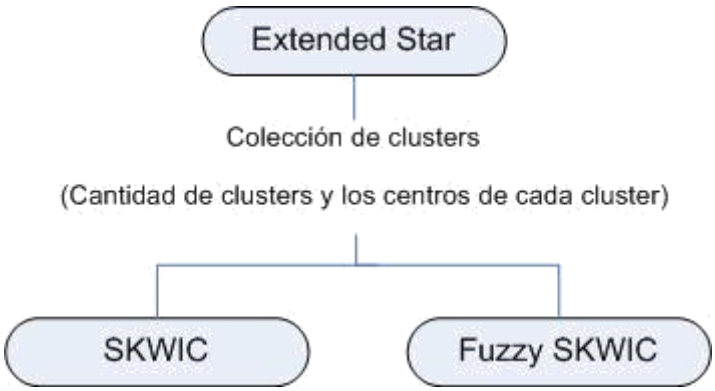


Figura 2.3 Combinación de los métodos de agrupamiento implementados en CorpusMiner.

A partir de esta combinación, ya no es necesario tener un conocimiento previo del dominio para definir el número de clusters a obtener, ni generar aleatoriamente los centros de clusters. Por tanto, se superan las deficiencias de los algoritmos SKWIC y Fuzzy SKWIC, donde se espera obtener mejores agrupamientos en un menor número de iteraciones (i.e. converger de una forma más

rápida). Además, al no generar los centros de clusters aleatoriamente en la primera iteración, esperamos que se reduzca la cantidad de clusters vacíos en SKWIC. A continuación describiremos estos tres algoritmos mencionados.

2.4.1 Algoritmo SKWIC

Una idea general del algoritmo *Simultaneous Keyword Identification and Clustering of Text Documents* (SKWIC) es [BER04]:

Entrada: Colección de documentos en su representación VSM.

Salida: Colección de clusters, donde cada cluster incluye el centro de cluster, la lista de documentos que pertenecen a él y la relevancia de términos en el cluster.

1. Fijar el número inicial de clusters.
2. Inicializar aleatoriamente los centros de clusters.
3. Inicializar las particiones y la relevancia de los términos.

REPETIR

a. Actualizar la relevancia de los términos por clusters (v_{ik}).

b. Calcular las distancias de los documentos a los centros de clusters ($D^k_{wc_{ij}}$).

c. Asignar cada documento a los cluster (en función de la menor distancia que exista entre un documento y todos los centros de clusters). Es decir, actualizar cada partición cluster χ_i.

d. Recalcular los centros de cluster (c_{ik}).

e. Recalcular los pesos δ_i por clusters.

HASTA (centros estabilizados)

En el algoritmo SKWIC se calcula la disimilitud entre cada documento x_j y los vectores centro de clusters c_i basándose en la medida de similitud Coseno, pero inicialmente se necesita calcular la distancia que hay entre dichos vectores para cada término k según la siguiente expresión:

$$D^k_{wc_{ij}} = \frac{1}{n} - (x_{jk} \cdot c_{ik})$$

La expresión anterior es utilizada para el cálculo de la distancia entre un documento x_j y un vector centro de cluster c_i:

$$\overline{D}^{k}_{wc_{ij}} = \sum_{k=1}^{n} v_{ik} D^{k}_{wc_{ij}}$$

Nótese que n es el número total de términos en la colección de N documentos, c_{ik} es el k-ésimo componente del vector centro del i-ésimo cluster, y $V=[v_{ik}]$ es la relevancia del término k en el cluster i.

El cálculo de la relevancia de los términos por clusters se realiza a partir de la siguiente fórmula:

$$v_{ik} = \frac{1}{n} + \frac{1}{2\delta i} \sum_{xj \in Xi} \left[\frac{1}{n} \sum D^{k}_{wcij} - D^{k}_{wcij} \right]$$

donde el peso δ_i se calcula mediante la expresión:

$$\delta_i^{(t)} = K_\delta \frac{\sum_{x_j \in \chi_i} \sum_{k=1}^{n} v_{ik}^{(t-1)} (D^{k^{(t-1)}}_{wc_{ij}})}{\sum_{k=1}^{n} (v_{ik}^{(t-1)})^2}$$

donde K_δ es una constante, y el subíndice (t-1) es usado sobre v_{ik} y c_{ik} para denotar sus valores en la iteración (t-1).

Al asignar un documento a un cluster, se toma en consideración la distancia que existe entre el documento a analizar y cada uno de los centros de clusters, asignándolo definitivamente al cluster más cercano:

$$X_i = \left\{ x_j \middle| \tilde{D}_{wcij} \leq \tilde{D}_{wckj} \forall k \neq i \right\}$$

Finalmente, es necesario recalcular los centros de clusters a partir de la siguiente expresión:

$$c_{ik} = \frac{\sum_{xj \in Xi} x_{jk}}{\sum_{xj \in Xi} 1}$$

Nótese que el algoritmo SKWIC crea una partición, como podemos ver en la figura 2.4.

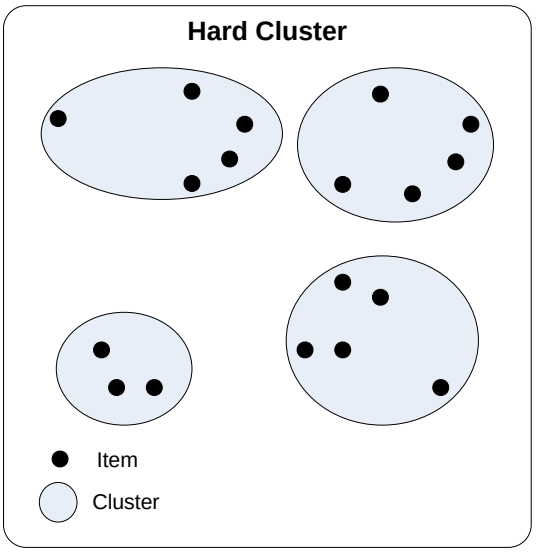


Figura 2.4 Partición creada por el algoritmo SKWIC.

Sin embargo, en los dominios textuales muchas veces un documento, por sus características, puede pertenecer a más de un grupo. Por tanto, en la sección 2.4.2 describiremos una versión borrosa de este método.

2.4.2 Algoritmo Fuzzy SKWIC

Una idea general del algoritmo *Simultaneous Keyword Identification and Fuzzy Clustering of Text Documents* (Fuzzy SKWIC) es [BER04]:

Entrada: Colección de documentos en su representación VSM.

Salida: Colección de clusters, donde cada cluster incluye el centro de cluster, la lista de todos los documentos y un valor de pertenencia de cada uno al cluster, además, la relevancia de términos en el cluster.

1. Fijar el número inicial de clusters
2. Inicializar aleatoriamente los centros de cluster
3. Inicializar los grados de pertenencia de los documentos a los clusters

REPETIR

   a. Calcular la relevancia de los términos (v_{ik}).

   b. Calcular la distancia de los documentos a los centros de cluster ($D^{k}_{wc_{ij}}$).

c. Actualizar los grados de pertenencia de los documentos a los clusters (en función de la menor distancia que exista entre un documento y todos los centros de cluster) (u_{ij}).

d. Recalcular los centros de cluster (c_{ik}).

e. Calcular los pesos de la relevancia de los términos (δ_i).

HASTA (centros estabilizados)

Esta variante, donde cada documento tiene un grado de pertenencia a cada cluster, muchas veces se ajusta más a las necesidades reales de problemas de minería de textos y por tanto es mucho más efectivo en el dominio textual, porque existen textos que por su contenido pertenecen a más de una categoría.

En el algoritmo Fuzzy SKWIC, al igual que en el SKWIC, se calcula la disimilitud entre cada documento x_j y los vectores centro de clusters c_i basándose en la medida de similitud Coseno, pero inicialmente se necesita calcular la distancia que hay entre dichos vectores para cada término k según la siguiente expresión:

$$D^{k}_{wc_{ij}} = \frac{1}{n} - (x_{jk} \cdot c_{ik})$$

La expresión anterior es utilizada para el cálculo de la distancia entre un documento x_j y un vector centro de cluster c_i:

$$\overline{D}^{k}_{wc_{ij}} = \sum_{k=1}^{n} v_{ik} D^{k}_{wc_{ij}}$$

Nótese que n es el número total de términos en la colección de N documentos, c_{ik} es el k-ésimo componente del vector centro del i-ésimo cluster, y $V=[v_{ik}]$ es la relevancia del término k en el cluster i.

Los grados de pertenencia de los documentos a los clusters se inicializan asignando el valor $1/C$, donde C es la cantidad de clusters.

El cálculo de la relevancia de los términos por clusters se realiza a partir de la siguiente fórmula:

$$v_{ik} = \frac{1}{n} + \frac{1}{2\delta_i} \sum_{j=1}^{N} \left(u_{ij}\right)^{m} \left[\frac{\tilde{D}_{wc_{ij}}}{n} - D^{k}_{wc_{ij}} \right]$$

donde el peso δ_i se calcula mediante la expresión:

$$\delta_i^{(t)} = K_\delta \frac{\sum_{j=1}^{N} \left(u_{ij}^{(t-1)}\right)^m \sum_{k=1}^{n} v_{ik}^{(t-1)} \left(D_{wc_{ij}}^{k^{(t-1)}}\right)}{\sum_{k=1}^{n} \left(v_{ik}^{(t-1)}\right)^2}$$

Para realizar el cálculo del grado de pertenencia de los documentos a los clusters se utiliza la fórmula siguiente:

$$u_{ij} = \frac{1}{\sum_{k=1}^{c} \left(\frac{\tilde{D}_{wc_{ij}}}{\tilde{D}_{wc_{ij}}} \right)^{\frac{1}{m-1}}}$$

Finalmente, es necesario recalcular los centros de clusters a partir de la siguiente expresión

$$c_{ik} = \begin{cases} 0, & \text{si } v_{ik} = 0 \\ \dfrac{\sum_{j=1}^{N} \left(u_{ij}\right)^m x_{jk}}{\sum_{j=1}^{N} \left(u_{ij}\right)^m}, & \text{si } v_{ik} > 0 \end{cases}$$

Nótese que en este algoritmo se realizan pasos similares al SKWIC pero en cada una de las expresiones se incorpora el grado de pertenencia de los documentos a los clusters.

2.4.3 Algoritmo Extended Star

El algoritmo *Extended Star* propuesto en [GIL03] es una mejora del *Star* reportado en [ASL00]. Por tanto, inicialmente comentaremos algunas ideas generales del algoritmo *Star*, así como sus desventajas para introducir posteriormente el *Extended Star*. A continuación definiremos algunos términos que son utilizados para describir los algoritmos.

Dos documentos se dicen β_0-semejantes si la semejanza entre ellos, dada por el valor de distancia entre sus vectores, es mayor o igual a β_0, donde β_0 es un parámetro definido por el usuario o calculado inicialmente a partir de la distancia entre los documentos.

Llamamos grafo β_0-semejante al grafo no dirigido cuyos vértices son los objetos a agrupar y existe una arista desde el vértice O_i al vértice O_j, si O_i es β_0-semejante a O_j.

El algoritmo *Star* consiste en formar un grafo β_0-semejante, donde los vértices son los documentos a agrupar. Este se basa en un cubrimiento *greedy* donde existen vértices satélites y vértices estrellas. Los vértices estrellas representan los documentos centros de clusters. Dos vértices estrellas nunca son adyacentes.

Alguna de las deficiencias que se mencionan del algoritmo *Star* es que los clusters obtenidos dependen del orden de los objetos [GIL03]. Obsérvese la figura 2.5.

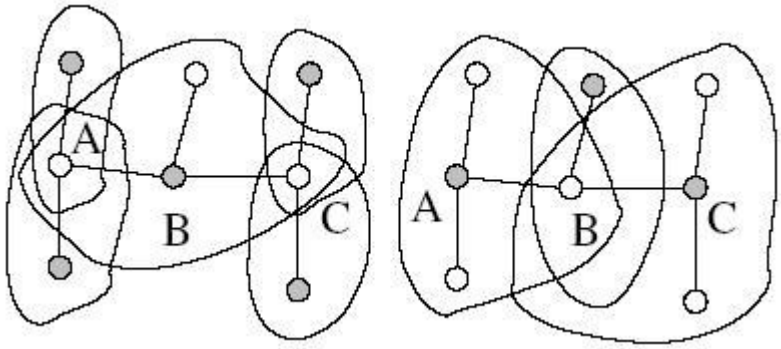


Figura 2.5 Agrupamiento *Star* donde influye el orden de los datos.

Otra desventaja del algoritmo *Star* es que se pueden formar clusters ilógicos, debido a que dos estrellas nunca pueden ser vecinas. Observe un ejemplo en la figura 2.6.

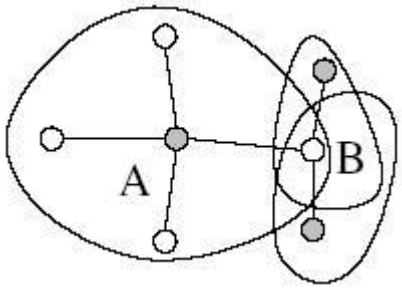


Figura 2.6 Clusters ilógicos según agrupamiento *Star*.

En [GIL03] se propone el algoritmo *Extended Star* que supera las deficiencias del *Star*. Los dos cambios fundamentales respecto al algoritmo *Star* son:

- Utilizar el grado complementario de un documento ($CD(O)$) definido como el grado del documento sin incluir aquellos vecinos ya agrupados en algún cluster

$$CD(O) = \left| \frac{N(O)}{Clu} \right|$$

donde $N(O)$ es el conjunto de documentos vecinos a O y Clu es el conjunto de documentos agrupados.

- Considerar un documento O como estrella si tiene al menos un vecino O**'** con menor o igual grado que O que satisface una de las condiciones siguientes:
  - O' no tiene vecinos estrellas.
  - El grado más alto de las estrellas que son los vecinos de O**'** no es mayor que el grado de O.

Una idea general del algoritmo *Extended Star* es [GIL03]:

Entrada: Colección de documentos en su representación VSM.

Salida: Colección de clusters, donde cada cluster incluye la estrella centro de cluster y la lista de todos los documentos que pertenecen a él.

1. Calcular todas las similitudes entre cada par de documentos para construir el grafo β_0-semejante.
2. Sea $N(O)$ los vecinos de cada documento O en el grafo β_0-semejante.
3. Para cada documento aislado O ($|N(O)| = 0$):
   a. Crear un cluster unitario $\{O\}$.
4. Sea L el conjunto de documentos no aislados.
5. Calcular el grado complementario de cada documento en L.
6. Mientras exista un documento no-agrupado:
   a. Sea M_0 el subconjunto de los documentos de L con máximo grado complementario.
   b. Sea M el subconjunto de los documentos de M_0 con máximo grado
   c. Para cada documento O en M
      i. Si O satisface la condición de ser estrella, entonces:

         Si $\{O\} \cup N(O)$ no existe:

            Crear un cluster $\{O\} \cup N(O)$

d. Eliminar los documentos procesados de *L* (*)

e. Actualizar el grado complementario de los objetos en *L*.

En el paso (*) se pueden eliminar de *L* los objetos en *M* o todos los objetos ya agrupados. Estas son versiones no restringidas y restringidas del algoritmo, respectivamente. En la versión restringida, solamente los objetos que no han sido agrupados, pueden ser estrella. Cada versión tiene ventajas y desventajas. La versión no restringida tiene más posibilidades de encontrar las mejores estrellas. La versión restringida es más rápida pero puede obtener clusters ilógicos [GIL03]. En la implementación realizada en CorpusMiner del algoritmo *Extended Star*, eliminamos los objetos de *L* considerando la versión no restringida.

La complejidad temporal del algoritmo *Extended Star* es de un O (n^2m), donde *n* es el número de objetos y *m* el número de rasgos [GIL03].

El algoritmo *Extended Star* [GIL03] crea clusters solapados y garantiza pares β_0-semejantes entre la estrella y sus vecinos. Los clusters se obtienen independientemente del orden de los datos. Si existen dos o más documentos con alta conectividad, este algoritmo selecciona como estrella a todos ellos. Además, la selección de estrellas usando el grado complementario permite reducir el solapamiento entre clusters.

2.4.3.1 Cálculo del umbral β_0

En la sección 2.4.3, al describir los conceptos principales relacionados con los algoritmos *Star* y *Extended Star*, hacíamos referencia a que el valor β_0 o umbral de semejanza, es un parámetro definido por el usuario o calculado inicialmente a partir de la distancia entre los documentos.

CorpusMiner tiene implementada tres variantes para el cálculo inicial del umbral β_0 y a continuación son descritas [GAR99]:

a) La primera variante calcula el umbral β_0 hallando la media de las distancias entre todos los pares de documentos posibles. Así se expresa en la siguiente fórmula:

$$\beta_0 = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} d(O_i, O_j)$$

b) La segunda forma implementada considera la próxima expresión :

$$\beta_0 = \frac{1}{n} \sum_{i=1}^{n} \max_{\substack{j=1..n \\ i \neq j}} \{d(O_i, O_j)\}$$

Esta segunda variante lo que hace es hallar la media de los valores máximos de las distancias entre cualquier par de documentos.

c) La tercera variante calcula el β_0 según muestra la siguiente expresión:

$$\beta_0 = \min_{\substack{i=1..n-1 \\ i \neq j}} \left\{ \min_{j=i+1..n} \left\{ d(O_i, O_j) \right\} \right\}$$

Aquí, lo que se hace es tomar la mínima de todas las distancias posibles en pares de documentos, sin tener en cuenta las distancias que sean cero.

La descripción de la notación utilizada es la siguiente: *n* es la cantidad de documentos de la colección y $d(O_i, O_j)$ es el valor de la distancia entre los vectores documento O_i y O_j.

2.4.4 Diseño e implementación de los algoritmos de agrupamiento

A continuación describiremos como fueron diseñados e implementados los algoritmos de agrupamiento, así como el cálculo de la similitud entre vectores ampliamente utilizado en este proceso.

2.4.4.1 Similitud entre vectores

La similitud entre documentos está estrechamente relacionada con los algoritmos de agrupamiento. La mayoría de estos métodos requieren el uso de medidas de similitud o disimilitud entre documentos. En los algoritmos de agrupamiento implementados en CorpusMiner, las medidas de similitud entre los documentos influyen considerablemente en la calidad de los grupos a obtener. Las medidas que implementamos en CorpusMiner son: la distancia Euclidiana [GAB04], las distancias de Jaccard binaria y pesada [GAS01], y la similitud Coseno [FRA92]. La distancia Euclidiana reporta malos resultados en aplicaciones de minería de textos, no obstante fue implementada para alguna verificación que deseara un usuario final del software. En [FRA92] se reportan las medidas Jaccard y Coseno como una de las mejores en el procesamiento textual.

Para la implementación de estas medidas de similitud entre documentos se definió la Interfaz *IDistance*, en la cual se declaran dos métodos con el uso de sobrecarga de funciones:

- *Distance(ItemVector1, ItemVector2: TVector):real;*
- *Distance(DispersedMatrix: TDispersedMatrix; Item1, Item2:integer):real*

Ambos prototipos representan la misma funcionalidad, solo los diferencian la forma de especificar los parámetros, es decir, los vectores documentos a comparar. En el primer caso, los dos vectores documentos son especificados directamente utilizando la clase *TVector*. En el segundo método se pasa la matriz de dispersión (términos - documentos) representada por la clase *TDispersedMatrix*, (véase anexo 2) y los índices de los vectores documentos que no están explícitamente en dicha matriz. En ambos casos se obtiene como resultado el valor real que expresa la similitud o disimilitud existente entre los vectores documentos que se comparan. Se presentan estas dos variantes, de tal forma que se utilice cada una de ellas en dependencia de los parámetros que se tengan.

A partir de este momento, se crean las clases correspondientes a cada una de las medidas de distancias que implementan la interfaz *IDistance*. Ellas son: *TDistanceEuclidian, TDistanceCosine* y *TDistanceBinaryJaccard, TDistanceWeightJaccard*. Todas estas clases son clases algoritmos (según diseño mostrado en epígrafe 2.3 y en [VAL05]) y solamente contienen los métodos de la interfaz *IDistance*, excepto en el caso específico de *TDistanceWeightJaccard* que cuenta con funciones propias: la función *GlobalWeigh* para calcular el peso global de un término y la función *LocalWeight* que se utiliza en el cálculo del peso local de un término (Véase anexo 3).

Las medidas para calcular distancias se insertan fácilmente en el diseño general de CorpusMiner. Se crea la clase *TDistanceController* que tiene como atributo una variable *_Distance* de tipo *IDistance*, y entre sus métodos se encuentra *Execute* para ejecutar el método *Distance* de cálculo de distancia. Esta clase hereda de *TAlgorithmController* que tiene como atributo un *_Name* de tipo *string* que no es más que el nombre que identifica la distancia en todo sus usos. También se define la clase *TDistanceListController* que hereda de *TAlgorithmListController* la cual tiene como atributo una lista de *TAlgorithmController*, además de métodos para la manipulación de la misma. Esta lista es llenada en el *Create* de *TDistanceListController* con objetos que instancian a *TDistanceController* por cada una de las distancias creadas como atributos. También se implementa un método *GetDistance* que dado el nombre (*_Name)* devuelve la referencia de tipo *IDistance* para su uso según el contexto en que se defina (ver anexo 4).

2.4.4.2 Implementación de los algoritmos de agrupamiento

En CorpusMiner se implementaron tres algoritmos de agrupamiento, pero el sistema es flexible en su diseño y permite asimilar tantos métodos para agrupar documentos como se deseen implementar.

La concepción del diseño parte de la creación de estructuras comunes a todos los métodos de agrupamiento implementados y a su vez adecuadas para almacenar los resultados de interés al ser aplicados. Tenemos clases como:

- *TCluterItem*: Identifica a un documento.
- *TFuzzyItem*: Hereda de *TCluterItem* y almacena un valor que representa el nivel de pertenencia (*_Membership*) de un documento a cada uno de los clusters en caso de ser borroso.
- *TCluster*: Permite almacenar la lista de documentos que se agrupan ya sea de manera dura o borrosa (véase anexo 5).
- *TClusterCollection*: Almacena distintos grupos de documentos y nos permite conocer:
    - Cantidad de cluster vacíos.
    - Cantidad de cluster con un solo elemento.
    - Cluster en que se encuentra un determinado elemento.

Los algoritmos de agrupamiento que hemos implementado en esta primera versión de CorpusMiner responden a las técnicas de agrupamiento: duro y determinista, duro con solapamiento y agrupamiento borroso. Por tal motivo, se implementaron clases que heredan de *TCluster* y de *TClusterCollection* en dependencia del tipo de agrupamiento seleccionado y sus necesidades. Se crea la clase *THardCluster* que hereda de *TCluster*, y la clase *TFuzzyCluster* que igualmente hereda de *TCluster* y además devuelve el grado de pertenencia de un documento al cluster. De esta clase *TCluster* también hereda la clase *TOverlappedCluster* que representa los grupos formados por el algoritmo *Extended Star*, por lo que tiene la funcionalidad de devolver el elemento estrella del cluster. De igual manera se crearon clases que representan colecciones de cluster duras y borrosas definidas como *THardClusterCollection* y *TFuzzyClusterCollection* respectivamente. Ambas heredan de *TClusterCollection* y representan las salidas de los algoritmos de agrupamiento descritos.

Uno de los resultados parciales de nuestra herramienta es la obtención de las palabras claves correspondientes a cada cluster. Para ello se necesita la relevancia de los términos obtenidas a partir de los algoritmos de agrupamiento SKWIC y Fuzzy SKWIC. Por tal motivo, fue necesario crear la interfaz *IRelevance* con el objetivo de asociar los clusters obtenidos a través de dichos algoritmos al ofrecer la relevancia de los términos por cada cluster formado, independientemente

que el resultado sea duro o borroso. A partir de aquí se crea la clase *THardClusterRelevance* que hereda de *THardCluster* e implementa la interfaz *IRelevance* para el caso del algoritmo SKWIC. Análogamente se definió la clase *TFuzzyClusterRelevance* que hereda de *TFuzzyCluster* y también implementa la interfaz *IRelevance* para el método Fuzzy SKWIC. Tanto la clase *THardCluster* como *TFuzzyCluster* heredan de la clase *TCluster* mencionada anteriormente (ver anexo 6).

Diseño e implementación de SKWIC y Fuzzy SKWIC

Con el objetivo de hacer extensible la implementación de los algoritmos de agrupamiento se definió una interfaz para cada técnica implementada, dejando abierta la posibilidad de insertar nuevos algoritmos de agrupamiento, y a su vez, nuevas técnicas de agrupamiento en CorpusMiner. Para las técnicas duras se creó la interfaz *IHardClusterizer*, que permite el agrupamiento a partir de la representación VSM de los corpus de documentos, y además, ofrece una colección dura de los grupos de documentos formados. Debido a que las técnicas duras y con solapamiento también devuelven una colección de clusters dura se decidió incluirlas en esta interfaz. Análogamente, se creó la interfaz *IFuzzyClusterizer* para las técnicas de agrupamiento borroso, mediante la cual se obtiene una colección borrosa de los clusters de documentos formados.

En la implementación de estos métodos de agrupamiento se definieron las clases *THardClusterizerSKWIC* (que implementa la interfaz *IHardClusterizer*) y *TFuzzyClusterizerSKWIC* (que implementa la interfaz *IFuzzyClusterizer*) para soportar los algoritmos SKWIC y Fuzzy SKWIC respectivamente. Este diseño se muestra en los anexos 7.a) y 7.b). Siguiendo los pasos descritos por ambos algoritmos los principales métodos implementados en dichas clases permiten:

- Inicializar los centros de clusters y la relevancia de los términos. La inicialización de los centros de clusters puede ser aleatoria o mediante una lista de documentos centros de clusters que se especifica al crear el objeto (esta lista se obtiene como resultado de otro proceso de agrupamiento).
- Agrupar cada uno de los documentos teniendo en cuenta la relevancia de los términos y la similitud con los centros de cada cluster.

Diseño e implementación de *Extended Star*

Para la implementación del algoritmo *Extended Star* se creó la clase *TOverlappedClusterizer* que también implementa la interfaz *IHardClusterize* (Ver anexo 7.c)). Esta clase necesita una medida de cálculo de distancia *IDistance* y un umbral β_0 para crear el grafo β_0-semejante. Esta clase, además de agrupar mediante el algoritmo *Extended Star*, ofrece las siguientes funcionalidades:

- Devolver los documentos seleccionados como estrellas (centros de clusters), los cuales servirán para inicializar alguno de los métodos descritos anteriormente.
- Verificar cuando un nodo se corresponde con un documento estrella y en este caso crear el cluster correspondiente.
- Obtener los valores de grado complementario de los nodos.

Es importante señalar que para almacenar el grafo β_0-semejante que se forma al agrupar mediante *Extended Star*, reutilizamos la estructura definida para almacenar la representación VSM del corpus (i.e. la matriz de dispersión).

Como habíamos señalado anteriormente, para el cálculo del umbral β_0 se ofrecen varias opciones. Para ello, definimos la interfaz *IBoInterface*, en la cual se declaró el método *B_0Compute* que usa una medida de cálculo de distancia *IDistance* y la matriz en su representación VSM (de tipo *TDispersedMatrix*) para el retorno del valor a calcular. A partir de aquí, se crean las clases *TMinDistanceComputeBo*, *TMeanDistanceComputeBo* y *TMaxDistanceComputeBo* que implementan la interfaz *IBoInterface*, y de esta forma se hace extensible nuestro sistema para cualquier otro método de cálculo de β_0 (Ver anexo 8).

Inserción de los algoritmos de agrupamiento en el sistema CorpusMiner

Para la manipulación de los algoritmos de agrupamiento en CorpusMiner, se creó una clase controladora por cada una de las técnicas implementadas: *THardClusterizerController* y *TFuzzyClusterizerController*. Luego, se definió una clase controladora para cada uno de los algoritmos implementados: *THardClusterizerSKController* y *TFuzzyClusterizerSKController*, que heredan de *THardClusterizerController* y *TFuzzyClusterizerController* respectivamente. Para el caso específico del algoritmo *Extended Star* se definió la clase *TOverlappedClusterizerController* que también hereda de *THardClusterizerController*, pero que además, implementa la interfaz *IResultClusterizerToInitilize* que devuelve un *TVector* con los elementos estrellas (centros de clusters) obtenidos a partir del algoritmo *Extender Star*. Estos centros de cluster son usados en la inicialización de los demás algoritmos de agrupamiento implementados en CorpusMiner que han sido referenciados anteriormente. De igual forma se crea la clase *TClusterizerInitializeController* con un atributo de tipo *IResultClusterizerToInitilize*, con el objetivo de agrupar las clases correspondientes a los algoritmos que devuelven resultados útiles en la inicialización de otros métodos. Se crea entonces la clase *TClusterizerListUsedToInitialize* que manipula la lista de clases controladoras con métodos de inicialización (ver anexo 9).

También se definieron las clases *THardClusterizerListController* que contiene la lista de los controladores de algoritmos duros y *TFuzzyClusterizerListController* que contiene la lista de los algoritmos borrosos.

2.5 Obtención del resumen extracto de cada grupo de documentos

En el capítulo anterior habíamos señalado que los resúmenes extractos se pueden realizar a partir de un único documento, o de múltiples documentos relacionados o no. En CorpusMiner se realizó un resumen extractivo de múltiples documentos. Para ello, partimos de una colección de clusters que son formados a través de los distintos algoritmos de agrupamiento implementados en nuestra herramienta. Agrupar los documentos similares antes de obtener un resumen extracto permite verificar la homogeneidad del corpus, por tanto, los extractos se generarán a partir de grupos homogéneos de documentos.

En el subepígrafe 1.2.2 hicimos referencia a diferentes formas de crear resúmenes por extracto de múltiples documentos. En CorpusMiner hemos permitido dos de ellas: crear el resumen de un único documento a partir del documento centro o más representativo de la colección, y crear el resumen de un único documento a partir del documento centro o más representativo de la colección y agregar alguna representación del resto de los documentos del cluster para proveer un cubrimiento total de cada cluster de documentos. Nótese que en el algoritmo Fuzzy SKWIC todos los documentos pertenecen a todos los clusters, por tanto para obtener un resumen del documento más representativo y el resto de los documentos, se seleccionan aquellos documentos que tengan un valor de pertenencia mayor que un umbral definido como parámetro por el usuario.

En CorpusMiner generamos resúmenes por cada cluster de la colección mediante la extracción de las oraciones que contienen las palabras claves, ya sea del documento más representativo del cluster, o del documento más representativo del cluster y del resto de los documentos del cluster. Por tanto, es importante conocer cuál es el documento más representativo de un cluster. Este proceso será descrito en la próxima sección.

2.5.1 Cálculo del documento más representativo de cada cluster

En CorpusMiner se posibilitan diferentes formas para seleccionar el documento más representativo de cada uno de los grupos formados como resultado de aplicar alguno de los algoritmos de agrupamiento implementados.

Los grupos formados por el método SKWICK tienen un vector centro de cluster que no necesariamente coincide con un vector documento, sino que es un documento ideal que se obtuvo a partir de las iteraciones realizadas por el algoritmo para determinada colección. Por tal motivo, para seleccionar el documento más representativo del grupo se utilizaron las medidas de cálculo de distancias entre vectores documentos (descritas en el subepígrafe 1.3.3). De esta forma, al comparar cada documento del grupo con el vector centro de cluster, se selecciona como documento más representativo de dicho grupo aquel documento que menor distancia tenga con respecto al centro de cluster.

Para lograr una implementación adecuada, se le adicionó a la clase *THardClusterSKRelevance* un método *GetItemMoreRepresentative,* al cual se le pasa como parámetro una variable de tipo *IDistance* y una bandera de tipo *Boolean* que es la encargada de identificar si se desea obtener el documento más representativo del grupo o todos los documentos.

Al aplicar el algoritmo de agrupamiento de documentos Fuzzy SKWICK a un corpus, se obtiene como resultado la lista de todos los documentos y un valor de pertenencia de cada uno de ellos al cluster, por lo que se selecciona como documento más representativo de cada grupo aquel documento que mayor valor de pertenencia tenga a dicho grupo.

La implementación se realizó de manera similar a la identificación del documento más representativo cuando los grupos se obtienen a partir del algoritmo SKWICK. La única diferencia está en los parámetros del método *GetItemMoreRepresentative,* en este caso serán el valor de umbral y la bandera de tipo *Boolean* (este último parámetro mantiene la misma funcionalidad que en el algoritmo anterior). Nótese que aquí el cálculo de la distancia entre vectores no es necesario.

Por último, identificar el documento más representativo cuando los grupos son formados por el algoritmo *Extender Star* es mucho más sencillo que en los casos anteriores. Este algoritmo devuelve por cada cluster un elemento estrella que no es más que el vector centro del cluster y coincide con un vector real de la colección, por tanto, éste es el documento más representativo del cluster. En este caso el método *GetItemMoreRepresentative* sólo necesita como parámetro la bandera de tipo *Boolean*.

2.5.2 Obtención de las palabras claves de cada grupo de documentos

En CorpusMiner se han implementado varias opciones para obtener las palabras claves de los grupos formados por los diferentes métodos de agrupamiento [VAL05]. Una de ellas es considerar como palabras claves aquellas que sobrepasan un umbral de relevancia en el cluster o tomar un

número definido por el usuario de las mejores palabras según su relevancia en el cluster. Nótese que esta variante sólo es posible aplicarla cuando hemos agrupado los documentos con los algoritmos SKWIC o Fuzzy SKWIC, que son los algoritmos implementados que simultáneamente agrupan y calculan la relevancia de las palabras en cada cluster. Otra variante que es posible aplicar es utilizando el algoritmo ID3 [MIT97] para resultados de agrupamiento a partir de los algoritmos *Extended Star* o SKWIC. Este método logra extraer aquellas palabras que logran discernir entre clusters, aunque no necesariamente tiene una frecuencia de aparición alta. Es por esto que para el algoritmo SKWIC es aconsejable interceptar las palabras relevantes con aquellas obtenidas por el ID3. La tercera variante es considerar los reductores de dimensionalidad descritos en [VAL05] e implementados en CorpusMiner, pero partiendo de una representación VSM donde solamente se incluyen los documentos que pertenecen al cluster. Este último criterio es aplicado a cualquiera de los tres algoritmos de agrupamiento implementados.

La clase *TKeywords* fue creada para almacenar el conjunto de las palabras claves de un cluster obtenidas a través de alguna de estas opciones. Esta clase almacena los índices de las palabras en un *TVector* y además guarda el identificador del cluster al que pertenecen. También se creó la clase *TKeyworlist* que contiene una lista de *TKeywords*. Esta clase es utilizada cuando se desea obtener las palabras claves de la colección completa (ver anexo 10).

En CorpusMiner brindamos la posibilidad de extraer las palabras claves de determinado cluster o las palabras claves de todos los clusters de la colección. Ya que la colección de clusters se puede expandir, por lo que en lugar de una colección tenemos cada cluster por separado. Sin embargo, el ID3 no se puede aplicar a un grupo en particular, ya que requiere para su funcionamiento el resultado completo del agrupamiento.

2.5.3 Obtención del resumen extracto de cada grupo

Después de haber precisado como obtener las palabras claves y el documento más representativo de cada cluster, podemos mencionar las principales clases utilizadas para formar el extracto. Inicialmente creamos la clase de datos *TClusterExtract* que almacena los índices de las oraciones seleccionadas para formar el extracto de un cluster determinado. También creamos la clase *TClusterCollectionExtract* para la formación del extracto de todos los grupos de la colección. La clase algoritmo *TSummarizerCluster* nos permite obtener un extracto del cluster a partir de sus palabras claves, el documento más representativo o el documento más representativo y el resto de los documentos del grupo (según de desee el usuario). Un objeto de tipo *TWordController* fue creado para guardar la referencia de la indexación previa de todas las palabras, las oraciones, los

párrafos y los documentos del corpus. Además, creamos la clase *TSummarizerCCluster* que utiliza objetos de los tipos *TKeywordlist* y *TSummarizerCluster*, y devuelve un *TClusterCollectionExtract*. Para todas estas clases creadas se definieron distintas clases controladoras ya sea formando el extracto a partir de un solo cluster o a partir de la colección (ver anexo 11).

Capítulo 3 Evaluación de los resultados e interfaz de usuarios

En este capítulo mostraremos la evaluación de los resultados obtenidos a partir de la aplicación de los diversos métodos de agrupamiento implementados en CorpusMiner. Las medidas utilizadas en la evaluación son entropía, *F-Measure* (*Precision* y *Recall*) y *Overall Similarity*. Mostraremos algunos fragmentos de resúmenes obtenidos, así como las palabras relevantes que permitieron su extracción. Además, describiremos como utilizar las opciones de agrupamiento y su evaluación, así como la obtención de extractos en CorpusMiner para orientar al usuario en el momento de hacer uso de nuestra herramienta.

3.1 Evaluación del agrupamiento de documentos

En la literatura se reportan diferentes medidas para validar la calidad del agrupamiento de documentos. En CorpusMiner utilizaremos algunas de ellas para evaluar los grupos generados por cada uno de los métodos implementados y sus combinaciones. Entre ellas se encuentran, por ejemplo: entropía, *F-Measure* y *Overall Similarity* [STE00] [FOR03] [SEB99].

3.1.1 Diferentes medidas para validar el agrupamiento

Para el agrupamiento, dos medidas de calidad son ampliamente usadas [STE00]. Un tipo de medida permite comparar diferentes conjuntos de grupos sin referenciar el conocimiento externo y es llamada medida de calidad interna. En tal sentido, utilizaremos la medida *overall similarity* basada en calcular la similitud de pares de documentos en un cluster. Existe otro tipo de medidas que nos permite evaluar cuán bueno es el agrupamiento mediante la comparación de los grupos producidos por la técnica de agrupamiento con las clases que se identifican en la colección. Este tipo de medida es llamada medida de calidad externa. Ejemplos de medidas externas son la entropía y *F-Measure*.

Hay muchas medidas de calidad diferentes y los resultados obtenidos al aplicarlas a diferentes algoritmos de agrupamiento, puede variar sustancialmente dependiendo de cual medida fue usada. Sin embargo si un algoritmo de agrupamiento funciona mejor que otros algoritmos de agrupamiento para la mayoría de las medidas, entonces podemos tener la confidencia que ciertamente ese algoritmo es el mejor para la situación que fue evaluada. A continuación describiremos cada una de estas medidas mencionadas.

3.1.1.1 Entropía

En [STE00] usan la entropía como medida de calidad de los clusters (con la advertencia que la mejor entropía es obtenida cuando cada cluster contiene exactamente un documento). Sea *CS* un resultado de agrupamiento. Para cada cluster es calculada primero la distribución de las clases (i.e. para un cluster *j* se calcula p_{ij}, la probabilidad que un miembro del clusters *j* pertenezca a la clase *i*)

$$p_{ij} = \frac{n_j^i}{n_j}$$

donde n_j^i es el número de documentos de la clase *i* que están asignados al cluster *j*. Entonces, usando esa distribución de la clase, la entropía de cada cluster *j* es calculada usando la fórmula estándar

$$E_j = -\sum_i p_{ij} \log\left(p_{ij}\right)$$

donde la sumatoria es aplicada sobre todas las clases. La entropía total para el conjunto de clusters *i* es calculada como la suma de las entropías de cada cluster y han sido pesadas por el tamaño de cada cluster

$$E_{CS} = \sum_{j=1}^{m} \frac{n_j * E_j}{n}$$

donde n_j es el tamaño del cluster *j*, *m* es el número de clusters, y *n* es el números total de documentos.

3.1.1.2 F-Measure

La segunda medida de calidad externa es *F-Measure* [STE00], e suna medida que combina las ideas de *precision* y *recall* de la recuperación de información [FRA92]. En [STE00] se trata cada cluster como si este fuera el resultado de una consulta y cada clase como si esta fuera el conjunto de documentos deseados para una consulta. Así, se calcula *precision* y *recall* de los clusters para cada clase dada. Más específicamente, para el cluster *j* y la clase *i*

$$recall(i, j) = n_{ij} \Big/ n_i \qquad precision(i, j) = n_{ij} \Big/ n_j$$

donde n_{ij} es el número de miembros de la clase *i* en el cluster *j*, n_j es el número de miembros del cluster *j* y n_j es el número de miembros de la clase *i*.

Entonces, según [STE00] *F-Measure* del cluster *j* y la clase *i* es entonces dada por

$$F(i, j) = \frac{2 \cdot recall(i, j) \cdot precision(i, j)}{recall(i, j) + precision(i, j)}$$

Un valor de *F-Measure* es calculado para toda la colección calculando un promedio pesado de todos los valores de las medidas *F-Measure*, según la siguiente expresión

$$F = \sum_i \frac{n_i}{n} \max\{F - Measure(i, j)\}$$

Es bien conocido de la práctica diaria de la recuperación de información que los niveles más altos de *precision* generalmente se obtienen con valores bajos de *recall*. La siguiente expresión permite calcular el valor F-Measure, proporcionando un parámetro α ($0 \leq \alpha \leq 1$) que permite ponderar *precision* y *recall*.

$$F_\alpha(i, j) = \frac{1}{\alpha \frac{1}{precision(i, j)} + (1 - \alpha) \frac{1}{recall(i, j)}}$$

En esta fórmula α puede verse como el grado relativo de importancia atribuida para *precision* y *recall*. Si α=1 entonces $F_\alpha(i,j)$ coincide con *precision*, si α=0 entonces $F_\alpha(i,j)$ coincide con *recall*. El valor α=0.5 es usado usualmente, así se considera igual importancia para *precision* y *recall*. En CorpusMiner utilizamos esta última variante del cálculo de $F_\alpha(i,j)$.

3.1.1.3 Overall Similarity

En la ausencia de información externa, tales como las etiquetas de clases, la cohesión de los clusters puede ser usada como una medida de similitud de clusters [STE00]. Un método para calcular la cohesión de un cluster es usar la similitud pesada de la similitud interna del cluster,

$$OverallSimilarity = \frac{1}{|S|^2} \sum_{\substack{d \in S \\ d' \in S}} \text{distance}(d', d)$$

donde $|S|$ es el número de documentos que pertenecen al cluster a evaluar. En [STE00] utilizan el cociente Coseno para calcular la distancia entre los vectores, sin embargo, en CorpusMiner nosotros permitimos calcular la distancia con cualquier medida de similitud implementada.

3.1.2 Evaluación de los algoritmos de agrupamiento implementados en CorpusMiner

A continuación mostraremos los resultados de evaluar el agrupamiento obtenido por cada método o combinación de ellos utilizando una colección de documentos extraída de un corpus textual de la agencia Reuters de noticias[1]. El corpus publicado por David D. Lewis tiene noticias referentes a 135 categorías. Todas ellas se encuentran etiquetadas y en muchos casos multiclasificadas.

A partir de esta colección de la agencia Reuters, conformamos un corpus que contiene 72 noticias generalmente de tamaño pequeño, que abordan 7 tópicos. La distribución de los documentos por tópicos la podemos apreciar en la tabla 3.1.

Tópicos	**Número de documentos**
acq	17
cocoa	7
earn	8
crude	18
money-supply	6
trade	9
coffee	9

Tabla 3.1 Cantidad de documentos en cada tópico.

Como habíamos mencionado anteriormente las medidas utilizadas para evaluar el agrupamiento son: entropía, *F-Measure* (*Precision* y *Recall*) y *Overall Similarity*. La evaluación se centró fundamentalmente en verificar el funcionamiento de los algoritmos de agrupamiento teniendo en cuenta las formas de pesar la matriz y de calcular la similitud entre los documentos; aunque esta última es válida solamente para el método *Extended Star* ya que los demás sólo trabajan con una variante la similitud coseno.

[1] Extraído de una colección de noticias de la agencia Reuters, con un total de 135 tópicos. Publicado por David D. Lewis en diciembre de 1996.

A continuación mostraremos los resultados de evaluar el agrupamiento obtenido a partir de de cada uno de los métodos implementados y sus combinaciones, en función de dichas medidas. Nótese que para la aplicación de todas las medidas, excepto *Overall Similarity* es necesario considerar documentos previamente etiquetados.

Cada fila en las tablas 3.2 y 3.3 corresponde a un algoritmo de agrupamiento o combinación de éstos. El orden considerado es el siguiente:

I. Algoritmo de agrupamiento SKWIC.

II. Algoritmo de agrupamiento Fuzzy SKWIC

III. Combinación de los algoritmos de agrupamiento SKWIC-*Extended Star*.

IV. Combinación de los algoritmos de agrupamiento Fuzzy SKWIC-*Extended Star*.

V. Algoritmo de agrupamiento *Extended Star*.

En la tabla 3.2 mostraremos los resultados de evaluar el agrupamiento teniendo en cuenta diferentes medidas de similitud entre documentos:

1. Similitud Coseno
2. Distancia Jaccard binaria
3. Distancia Jaccard pesada

Téngase en cuenta que para los algoritmos SKWIC y Fuzzy SKWIC no es posible evaluar los resultados para las medidas Jaccard binaria y pesada, ya que ellos sólo permiten calcular la similitud entre documentos utilizando una variante de la similitud coseno.

La representación VSM fue pesada con TF-IDF I para realizar los agrupamientos utilizando estas medidas.

	Entropía			**F-Measure**			**Precision**			**Recall**		
	1	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**	**1**	**2**	**3**
I	0.22	-	-	0.7	-	-	0.87	-	-	0.76	-	-
II	0.21	-	-	0.86	-	-	0.97	-	-	0.93	-	-
III	0.08	0.42	0.32	0.75	0.56	0.66	0.96	0.75	0.71	0.68	0.73	0.81
IV	0.37	2.41	3.21	0.75	0.29	0.29	0.86	0.17	0.17	0.95	1.02	1.02
V	0	0.07	0	0.42	0.26	0.28	1.02	1.02	1.02	0.29	0.15	0.16

Tabla 3.2 Resultado de la evaluación del agrupamiento teniendo en cuenta las diferentes medidas de similitud.

Nótese en la tabla 3.2 que los mejores resultados los reporta la similitud Coseno. Las distancias Jaccard binaria y pesada se comportan de manera similar y no reportan buenos resultados. Esta evaluación corrobora los criterios reportados en la literatura acerca de la similitud Coseno, donde se ha publicado reiteradamente que ofrece los mejores resultados en dominios textuales.

En la tabla 3.3 mostramos los resultados de evaluar el agrupamiento partiendo de una matriz previamente pesada a través de TF-IDF I y TF-IDF II. La medida de similitud entre documentos que se aplicó fue Coseno. El umbral β_0 utilizado en el algoritmo Extended Star para delimitar cuando dos vectores son β_0-semejantes fue 0.38 para un pesado con TF-IDF I y 0.30 para un pesado con TF-IDF II. En cada caso, β_0 fue calculado por la media de los máximos.

	Entropía		**F-Measure**		**Precision**		**Recall**	
	TF-IDF I	TF-IDF II	TF-IDF I	TF-IDF II	TF-IDF I	TF-IDF II	TF-IDF I	TF-IDF II
I	0.22	0.25	0.7	0.75	0.87	0.86	0.76	0.81
II	0.21	0.2	0.86	0.73	0.97	0.87	0.93	0.79
III	0.08	0.12	0.75	0.69	0.96	0.93	0.68	0.62
IV	0.37	0.17	0.75	0.69	0.86	0.83	0.95	0.69
V	0	0	0.42	0.36	1.02	1.02	0.29	0.23

Tabla 3.3 Resultado de la evaluación del agrupamiento partiendo de una matriz previamente pesada con TF-IDF I o TF-IDF II.

Nótese en la tabla 3.3 que los resultados obtenidos al pesar con TF-IDF I y TF-IDF II son similares, no obstante se aprecian resultados ligeramente mejores con TF-IDF I respecto a TF-IDF II en las medidas *F-Measure*, *Precision* y *Recall*. Los buenos resultados de *F-Measure* expresan que los clusters al ser compactos agrupan en él una gran cantidad de elementos de la clase asociada al mismo. Observe en el siguiente ejemplo un fragmento de los resultados del agrupamiento:

```
…
Cluster 7
     Document 3
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 5
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 15
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 19
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 32
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 33
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 41
```

```
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 48
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 50
          First Sentence: <TOPICS><D>trade</D></TOPICS>
     Document 68
          First Sentence: <TOPICS><D>trade</D></TOPICS>

Cluster 13
     Document 2
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 6
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 8
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 10
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 17
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 23
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 26
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 27
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 28
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 29
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 39
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 46
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 52
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 65
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 70
          First Sentence: <TOPICS><D>crude</D></TOPICS>
     Document 71
          First Sentence: <TOPICS><D>earn</D></TOPICS>
…
```

El algoritmo que reporta los mejores valores de entropía es el *Extended Star*, esto se debe a que el mismo forma grupos muy compactos debido a que utiliza la similitud Coseno para comparar los vectores documento. Las combinaciones usando TF-IDF II de los algoritmos *Extended Star* –

SKWIC y *Extended Star* – Fuzzy SKWIC reportan mejores valores de entropía que SKWIC y Fuzzy SKWIC respectivamente. Un elemento que influye en estos resultados es considerar la salida del Extended Star para inicializar SKWIC y Fuzzy SKWIC, y de esta forma los centros no se inicializan aleatoriamente.

El algoritmo *Extended Star* no reporta buenos valores de *recall*, esto se debe a que crea muchos clusters aislados, además, crea clusters muy pequeños y varios de ellos de un mismo tópico. Sin embargo, sus valores de *precision* son buenos porque los clusters que crea son muy homogéneos.

Nótese que los resultados de SKWIC y Fuzzy SKWIC se comportan similares porque el corpus conformado para la evaluación no tiene noticias con múltiples etiquetas.

La figura 3.4 refleja los resultados del agrupamiento utilizando *Overall similarity*. Nótese que esta forma de evaluar se realiza por cluster y no requiere que los documentos estén previamente etiquetados.

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**	**Cluster 6**	**Cluster 7**	**Cluster 8**	**Cluster 9**	**Cluster 10**	**Cluster 11**	**Cluster 12**	**Cluster 13**	**Cluster 14**	**Promedio**
I	-	0.21	0.07	0.2	-	0.3	0.07	0.07	0.25	-	0.24	0.14	-	0.17	0.17
II	0.2	0.23	-	-	0.08	-	0.09	-	-	0.3	0.13	0.2	0.11	0.09	0.16
III	0.3	-	0.14	0.07	0.2	0.23	-	0.15	0.3	0.6	0.28	0.2	0.14	0.2	0.22
IV	0.36	0.25	-	0.14	-	-	-	0.07	-	-	-	-	0.14	0.23	0.19
V	0.45	0.45	0.45	0.6	0.46	0.38	0.49	0.74	0.4	0.64	0.58	1	0.43	0.53	0.46

Tabla 3.4 Resultados de la evaluación del agrupamiento con *Overall Similarity*.

Las celdas que aparecen con un guión (-), en la tabla 3.4, reflejan clusters vacíos, donde no es posible calcular *Overall Similarity*. En los algoritmos Fuzzy SKWIC y *Extended Star* – Fuzzy SKWIC ningún cluster es vacío, pero representamos con un guión (-) aquellos que coinciden con otros ya evaluados.

Un criterio que permite tener una noción de cuán bueno son los resultados que refleja *Overall Similarity* es la comparación de estos valores con los umbrales β_0 calculados como la media de los máximos (β_0=0.38) y la media de las distancias de todos los documentos (β_0=0.06). Tuvimos en cuenta para el cálculo de los β_0 la similitud Coseno. Observando la tabla 3.4 apreciamos que todos

los resultados de *Overall Similarity* por cluster son superiores al valor de β_0 obtenido a partir de la media de las distancias de todos los documentos.

Por las características comentadas anteriormente sobre el algoritmo *Extended Star* es que éste logra los mejores resultados. *Extended Star* logra valores de *Overall Similarity* superiores a β_0=0.38 calculado como la media de los máximos. Este criterio de evaluación considera solamente los clusters que tienen más de un documento, es por esto que la generación de clusters aislados no se puede reflejar en este criterio.

Nótese, a partir del valor promedio de *Overall Similarity* según cada algoritmo, que el agrupamiento con las combinaciones de *Extended Star* – SKWIC y *Extended Star* – Fuzzy SKWIC los resultados son mejores que cuando los grupos son formados con SKWIC y Fuzzy SKWIC respectivamente.

3.2 ¿Cómo utilizar módulos de agrupamiento, resumen y evaluación en CorpusMiner?

Este epígrafe lo dedicaremos a describir cómo utilizar en CorpusMiner las opciones referidas al agrupamiento, resumen y evaluación. Asumimos que el usuario domina el resto de las opciones de CorpusMiner que le permiten representar el corpus adecuadamente, así como obtener las palabras claves por cada grupo obtenido.

3.3.1 Operaciones sobre la representación VSM

El módulo de agrupamiento funciona a partir de una representación VSM del corpus textual. Sin embargo, realizar el agrupamiento no es la única opción disponible en CorpusMiner después de obtener una representación VSM de la colección de documentos. Sobre la representación VSM existen cuatro operaciones posibles (ver figura 3.1).

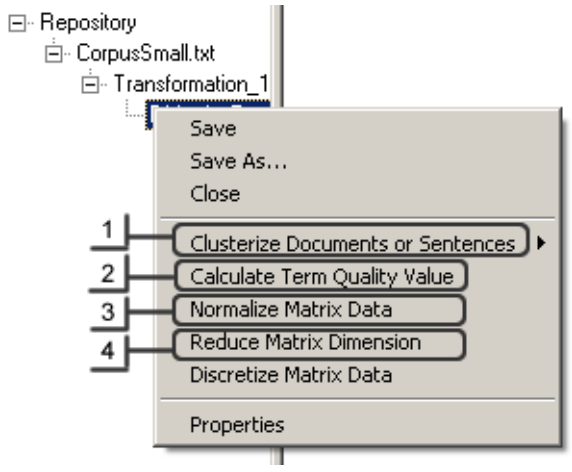


Figura 3.1 Opciones posibles a partir de una representación VSM del corpus de textos.

2. Calcular la calidad de los términos (*Calculate Term Quality Value*) [VAL05].

3. Pesar la matriz (*Normalize Matriz Data*) [VAL05].

4. Reducir la dimensionalidad de la matriz (*Reduce Matrix Dimension*) [VAL05].

Las opciones 2, 3 y 4 permiten transformar la representación VSM y se sugieren aplicar antes de realizar un proceso de agrupamiento, ya que contribuyen a mejorar la calidad del mismo.

1. Agrupar los documentos o sentencias (*Clusterize Documents or Sentence*): Permite seleccionar una de las técnicas de agrupamiento implementadas en CorpusMiner.

Nuestro sistema da la posibilidad de agrupar objetos a través de algoritmos que utilizan técnicas duras o borrosas. Es por eso, que si decidimos realizar un proceso de agrupamiento se despliega un submenú que permite decidir si la técnica a aplicar es dura o borrosa, opciones 1 o 2 respectivamente (ver figura 3.2).

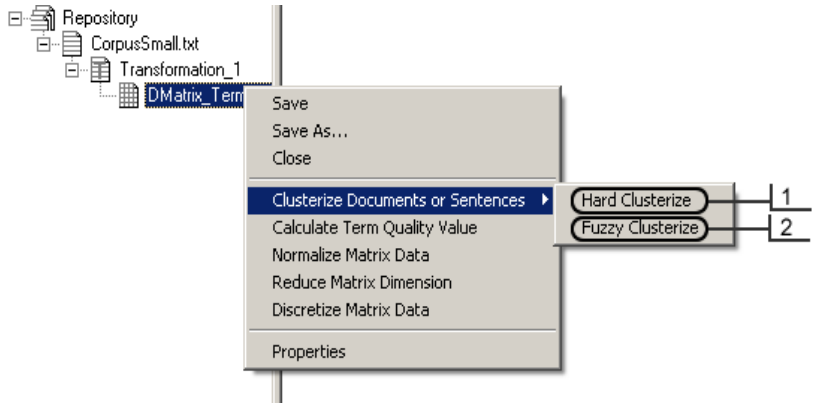


Figura 3.2 Selección de agrupamiento duro o borroso.

Después de seleccionar el tipo de agrupamiento a realizar se elige el algoritmo de agrupamiento mediante el cual se desea agrupar, en dependencia de la técnica seleccionada. Si decidió agrupar utilizando una técnica dura, entonces se presenta en CorpusMiner un diálogo como el de la figura 3.3. En dicha figura la zona seleccionada con un 1 representa los dos algoritmos duros que ofrece CorpusMiner: SKWIC (*Hard SK*) o *Extended Star* (*Overlapped Star*). Por el contrario, si la decisión fue métodos borrosos, sólo se muestra el algoritmo Fuzzy SKWIC (*Fuzzy SK*).

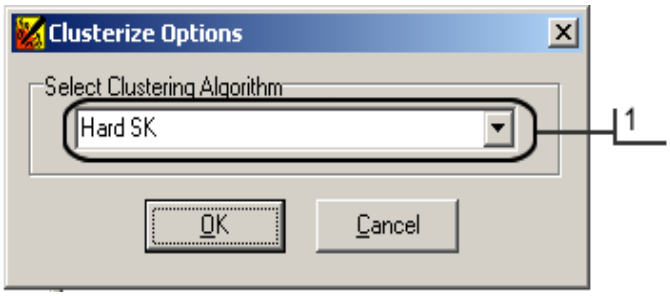


Figura 3.3 Algoritmos de agrupamiento duros.

A continuación presentaremos la interfaz visual de estos algoritmos e iremos explicando cada uno de sus parámetros.

Extended Star

Este algoritmo requiere el cálculo de la distancia entre los vectores documentos para realizar el agrupamiento. Además, requiere que se especifique o calcule un umbral β_0 que es el que permite decidir si dos documentos son β_0-semejantes a partir de la comparación de la distancia y el umbral. En la figura 3.4 se detallan cada una de las opciones en *Extended Star*.

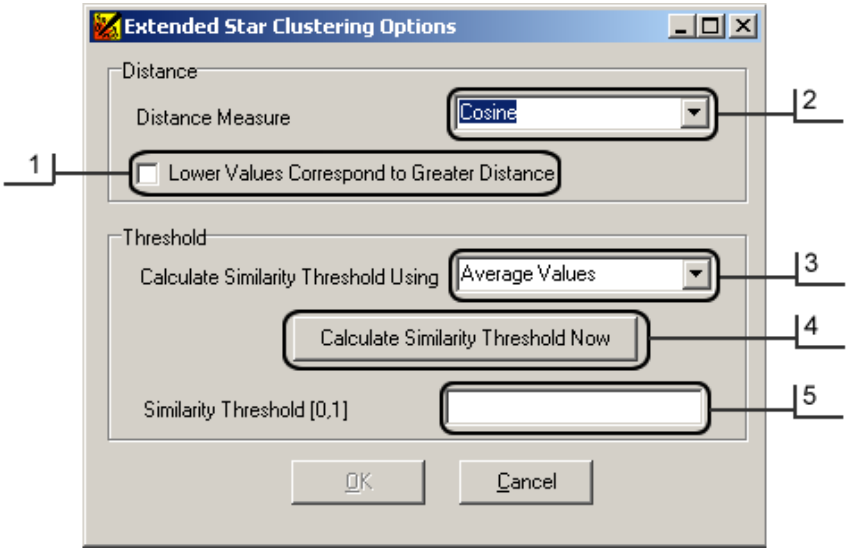


Figura 3.4 Opciones para aplicar el algoritmo Extended Star.

1. Se utiliza para especificar si la medida es de similitud o disimilitud. En función de eso, son mejores lo mayores o menores valores.
2. Permite seleccionar la medida para calcular la similitud entre documentos: Coseno, Jaccard binaria y pesada. También se permite la selección de la distancia Euclidiana, aunque ésta no es adecuada para comparar documentos.

3. Posibilita seleccionar la forma de cálculo del umbral β_0: mínimo, media y máximo. La forma de calcular β_0 influye en que se forman más o menos grupos, y que sean más o menos compactos.
4. Es el botón que ejecuta la acción de calcular el umbral β_0.
5. Esta opción le permite al usuario especificar directamente un valor β_0. Nótese que el cálculo del β_0 es costoso computacionalmente, por lo que en muchas ocasiones y teniendo conocimiento del dominio, el usuario puede desear definir el umbral.

SKWIC

Como mencionamos en el capítulo 2, este algoritmo requiere que sea especificado el número de clusters a obtener e inicializa aleatoriamente los centros de clusters. Sin embargo, una propuesta de este trabajo es utilizar los resultados de algoritmo *Extended Star* para inicializar el algoritmo SKWIC. Por tal motivo, los usuarios tienen dos opciones:

- Aplicar el algoritmo SKWIC (*Hard SK*) directamente definiendo la cantidad de clusters a obtener y esto inicializándose aleatoriamente.
- Aplicar el algoritmo *SKWIC* a partir de los resultados obtenidos por el algoritmo *Extended Star*.

El diálogo de la figura 3.5 permite decidir si se desea aplicar o no un algoritmo previo a SKWIC.

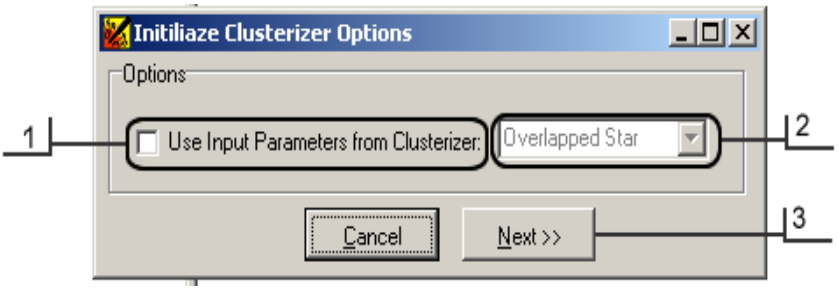


Figura 3.5 Selección de la aplicación de un algoritmo previo a SKWIC.

1. Define si se inician o no los parámetros a partir de otro algoritmo de agrupamiento.
2. Selecciona el algoritmo que devuelve los datos de inicialización, en este caso sólo es posible el algoritmo *Extended Star*.
3. Botón que ejecuta la acción de seleccionar el algoritmo inicial.

Aplicar el algoritmo SKWIC definiendo directamente la cantidad de clusters a obtener

Las opciones del algoritmo SKWIC se muestran en el siguiente diálogo (ver figura 3.6).

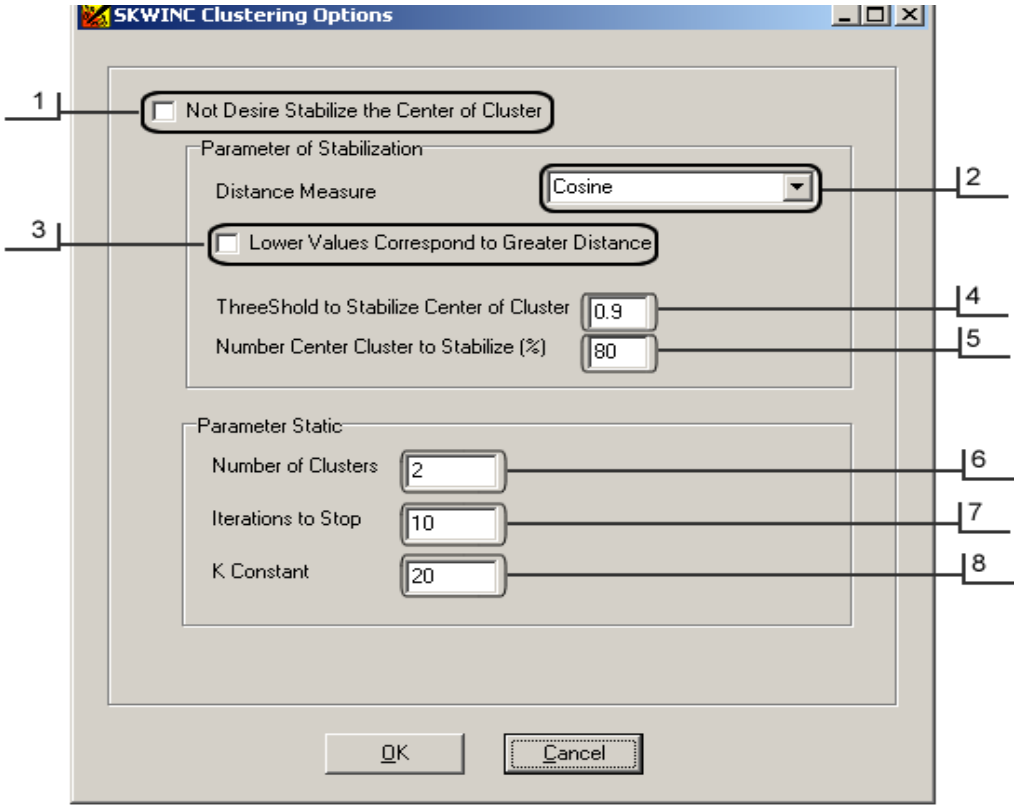


Figura 3.6 Definición de parámetros para aplicar el algoritmo SKWIC.

1. Posibilita seleccionar si se desea incluir o no la estabilización de los clusters como condición de parada del algoritmo.
2. Permite seleccionar la medida para calcular la similitud entre los centros de clusters necesaria para la estabilización.
3. Permite especificar si la medida utilizada para comparar es de similitud o disimilitud.
4. Permite especificar el umbral que decidir que un centro de cluster está estabilizado.
5. Posibilita especificar el porciento de clusters que debe estar estabilizado para considerar una estabilización total.
6. Especificar el número de clusters a obtener.
7. Definir la cantidad de iteraciones que se realizarán en la ejecución. Si se especifica el número de iteraciones a realizar y también los parámetros de estabilización, el algoritmo se detiene si logra

converger antes de cumplirse el número de iteraciones, o cuando se cumpla el número de iteraciones aunque no haya convergido.

8. Define el valor de la constante *K* utilizada en el algoritmo para el cálculo de los pesos de la relevancia de las palabras.

Aplicar el algoritmo SKWIC a partir de los resultados obtenidos por el método *Extended Star*

Cuando previamente utilizamos la salida del algoritmo *Extended Star* para inicializar SKWIC, es necesario especificar los parámetros que se muestran en el diálogo de la figura 3.7.

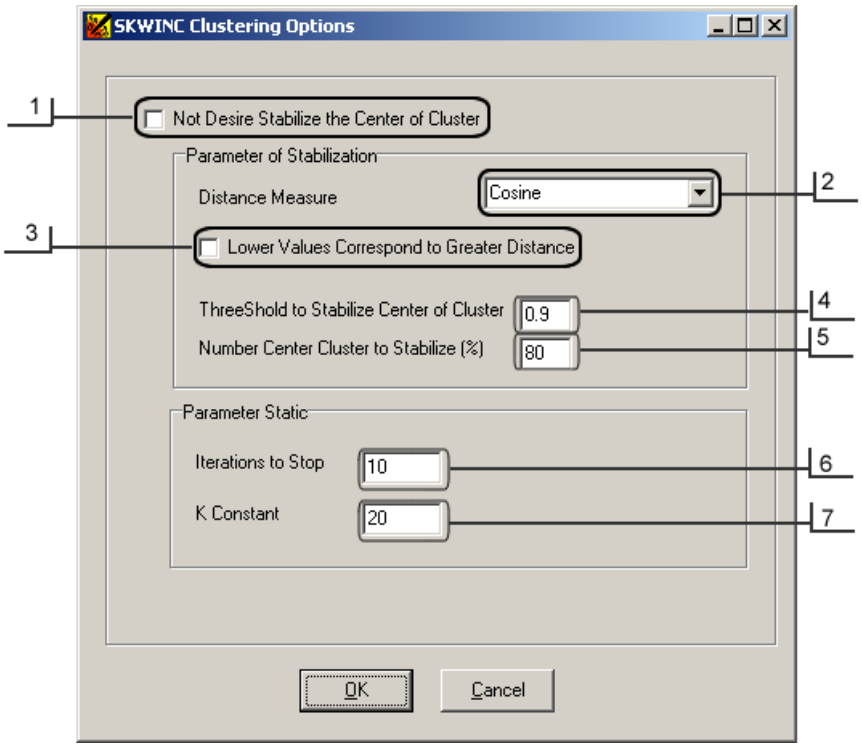


Figura 3.7 Opciones del algoritmo SKWIC cuando previamente se aplicó *Extended Star*.

Las opciones de la 1 a la 5 coinciden exactamente con las explicadas en el diálogo anterior porque se refieren a la estabilización de los centros de clusters. Las opciones 6 y 7 coinciden con las opciones 7 y 8 del diálogo de la figura 3.6, respectivamente. Nótese que todos los parámetros se mantienen al combinar ambos métodos, excepto la especificación del número inicial de clusters. Aunque la repercusión de la combinación es aún mayor porque no es necesario generar aleatoriamente los centros de clusters, sino inicializarlos con los resultados del Extended Star y se logrará la estabilización mucho más rápido.

Fuzzy SKWIC

Al igual que como sucede en SKWIC, al aplicar este algoritmo el usuario tiene dos posibilidades:

- Aplicar el algoritmo Fuzzy SKWIC (*Fuzzy SK*) directamente definiendo la cantidad de clusters a obtener.
- Aplicar el algoritmo Fuzzy SKWIC (*Fuzzy SK*) a partir de los resultados obtenidos por el algoritmo *Extended Star*.

A continuación describiremos cada una de estas variantes.

Aplicar el algoritmo Fuzzy SKWIC definiendo directamente la cantidad de clusters a obtener

Al aplicar directamente el algoritmo Fuzzy SKWIC es necesario especificar los parámetros que se muestran en el diálogo de la figura 3.8.

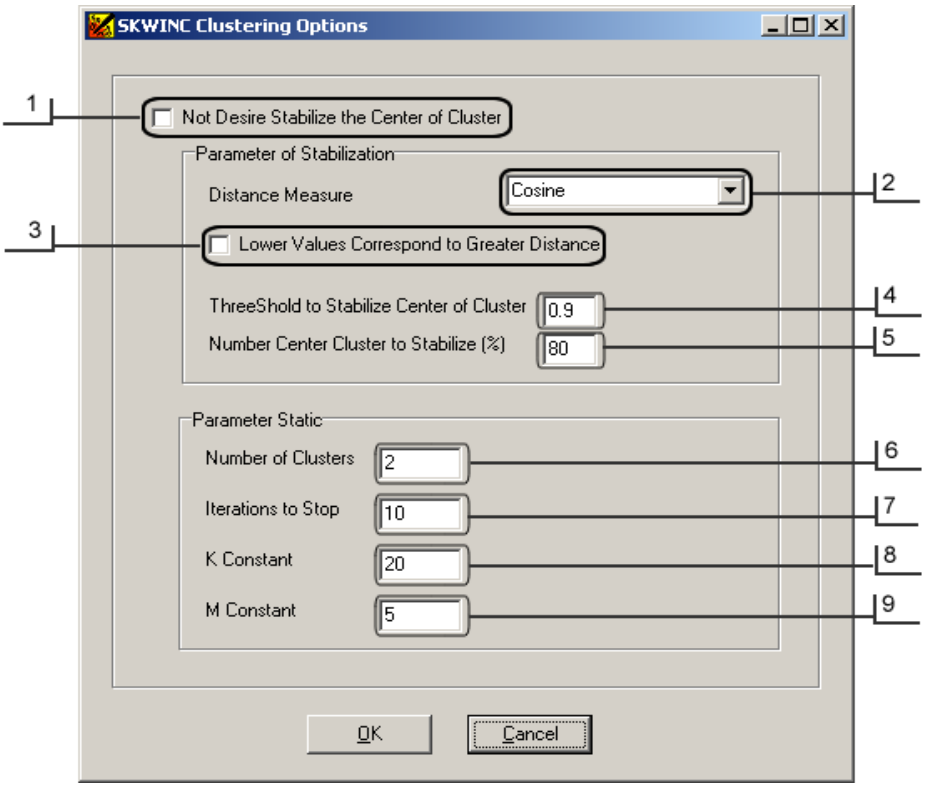


Figura 3.8 Definición de parámetros para aplicar el algoritmo Fuzzy SKWIC.

Las opciones de la 1 a la 8 coinciden exactamente con las opciones para definir los parámetros del algoritmo SKWIC, solo en este caso es necesario especificar la opción 9 (constante M) que se utiliza en el algoritmo para darle menor o mayor importancia al grado de pertenencia de los documentos a los clusters.

Aplicar el algoritmo Fuzzy SKWIC a partir de los resultados obtenidos por el algoritmo *Extended Star*

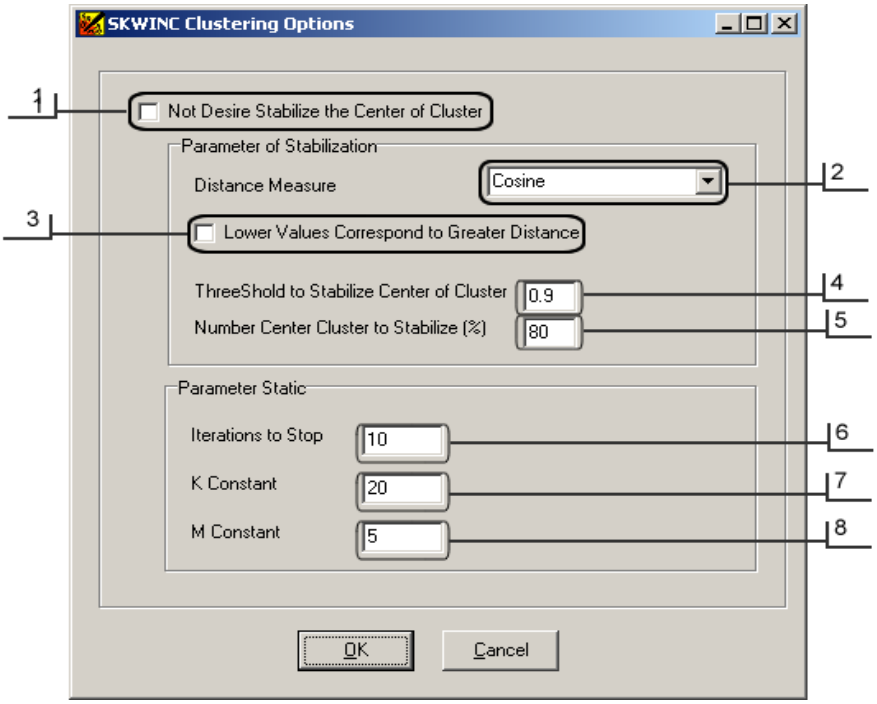


Figura 3.9 Opciones del algoritmo Fuzzy SKWIC cuando previamente se aplicó *Extended Star*.

Todas las opciones para la aplicación de Fuzzy SKWIC después de aplicar *Extended Star* (ver figura 3.9) coinciden exactamente con las especificadas en la figura 3.8, excepto que en este caso no es necesario que el usuario defina el número de clusters a obtener, ya que este dato se obtiene del resultado de aplicar *Extended Star*.

3.3.2 Acciones a realizar sobre una colección de clusters

Después de obtener una colección de clusters, cinco operaciones son posibles en CorpusMiner:

1. **ID3**: Obtener las reglas que caracterizan cada cluster en función de los términos que los describen y logran discernir entre ellos [VAL05].
2. **Expandir** (*Expand*): Expandir la colección de cluster, es decir, crear un nodo en el árbol por cada cluster de la colección.
3. **Palabras claves** (*Keywords*): Calcular las palabras claves de cada cluster [VAL05].
4. **Resumir** (*Sumarize*): Permite seleccionar el tipo de resumen a realizar.

5. **Evaluar** (*Evaluate*): Permite evaluar la calidad de los métodos de agrupamiento implementados.

A continuación describiremos como resumir y evaluar el agrupamiento en CorpusMiner.

Resumir

Como mencionamos en el capítulo 2, en CorpusMiner es posible resumir seleccionando las oraciones principales del documento más representativo de cada cluster, o incluir además las oraciones principales de todos los documentos que conforman el cluster. Estas opciones pueden ser seleccionadas como se muestra en la figura 3.10.

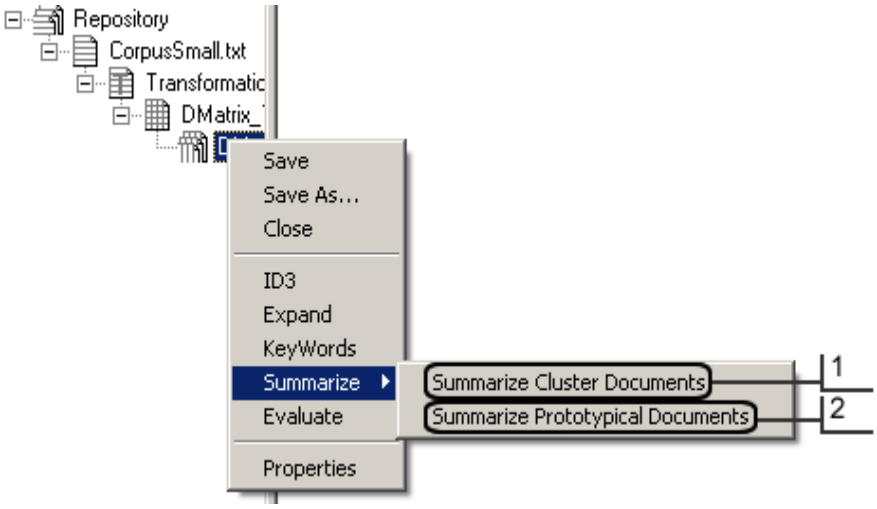


Figura 3.10 Selección de las formas de obtener un resumen extracto.

1. Realizar el resumen a partir de todos los documentos de un cluster.
2. Realizar el resumen a partir del documento más representativo del cluster.

Después de haber decidido qué resumen obtener es necesario especificar cómo se calcularán las palabras claves. Estas palabras claves son imprescindibles para extraer las principales oraciones que conformarán el extracto. Las posibilidades de cálculo de palabras claves están en dependencia del algoritmo de agrupamiento aplicado.

Formas de calcular las palabras claves en CorpusMiner:

1. Palabras relevantes a cada cluster (sólo válido cuando agrupamos con SKWIC y Fuzzy SKWIC, porque realizan simultáneamente el agrupamiento y el cálculo de la relevancia de los documentos a los clusters).
2. Aplicando el algoritmo ID3 (sólo válido cuando agrupamos con SKWIC y *Extended Star*, porque son algoritmos que utilizan una técnica dura).

3. Interceptar las palabras relevantes y las obtenidas por el algoritmo ID3 (sólo válido para SKWIC).
4. Considerar los reductores de dimensionalidad descritos en [VAL05] e implementados en CorpusMiner, pero partiendo de una representación VSM donde solamente se incluyen los documentos que pertenecen al cluster (es aplicado a cualquiera de los tres algoritmos de agrupamiento implementados).

A continuación describiremos varias formas de extraer el resumen extracto teniendo en cuenta si deseamos resumir todo el cluster o solo su documento más representativo, el tipo de agrupamiento utilizado, y en función de él, la forma de calcular las palabras claves.

El diálogo de la figura 3.11 está disponible en CorpusMiner cuando deseamos obtener un extracto de todos los documentos por clusters y la técnica de agrupamiento es dura (SKWIC o Extended Star). En la opción 1 aparecen las formas de calcular las palabras claves. Si agrupamos con SKWIC son posibles todas las variantes y si lo hicimos con Extended Star solo son posibles las variantes 2 y 4.

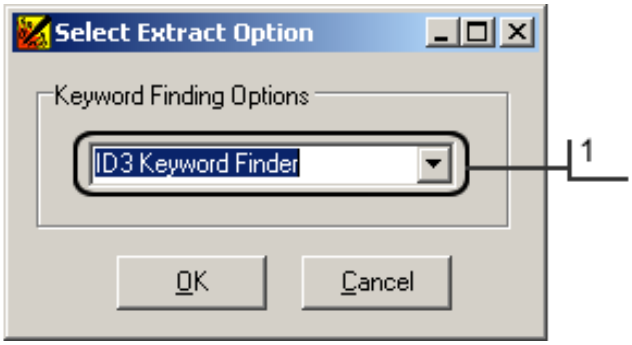


Figura 3.11 Resumen extracto de cada cluster cuando se agrupó con Extended Star o SKWIC.

El diálogo de la figura 3.12 está se presenta en CorpusMiner cuando deseamos obtener un extracto de todos los documentos por clusters y la técnica de agrupamiento es borrosa (Fuzzy SKWIC). Nótese que es necesario especificar qué documentos consideraremos en la colección (opción 2), ya que con esta técnica todos los documentos pertenecen a todos los clusters pero con determinado grado de pertenencia. Así, solo consideraremos aquellos documentos que sobrepasen el umbral especificado. Aquí las variantes de cálculo de palabras claves se muestran en la opción 1, y son posibles la 1 y la 4.

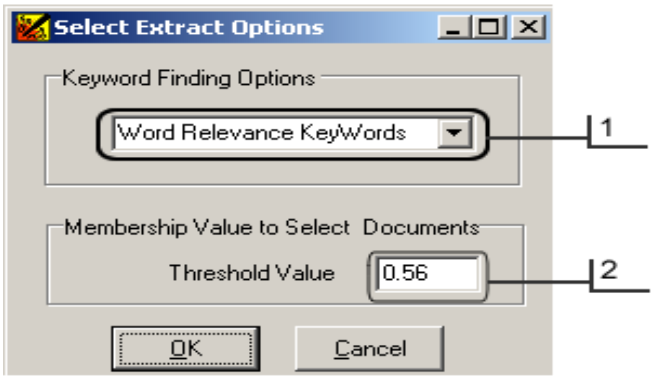


Figura 3.12 Resumen extracto de cada cluster cuando se agrupó con Fuzzy SKWIC.

Los diálogos anteriormente descritos están en función de obtener un resumen extracto que incluya a todos los documentos de cada cluster, donde el primero de ellos es válido cuando se realizó un agrupamiento duro y el segundo cuando se realizó un agrupamiento borroso. Sin embargo, cuando se desea obtener el resumen extracto de sólo el documento más representativo de cada cluster, es necesario tener en cuenta, más que la técnica de agrupamiento en sí, el propio algoritmo de agrupamiento utilizado.

Cuando el algoritmo de agrupamiento es SKWIC, se busca el documento del cluster más cercano a su centro y éste se selecciona como el documento más representativo del cluster (recuerde que en SKWIC el centro es un documento ideal, no coincide con uno de la colección y por eso no se puede tomar como más representativo). Observe la figura 3.13.

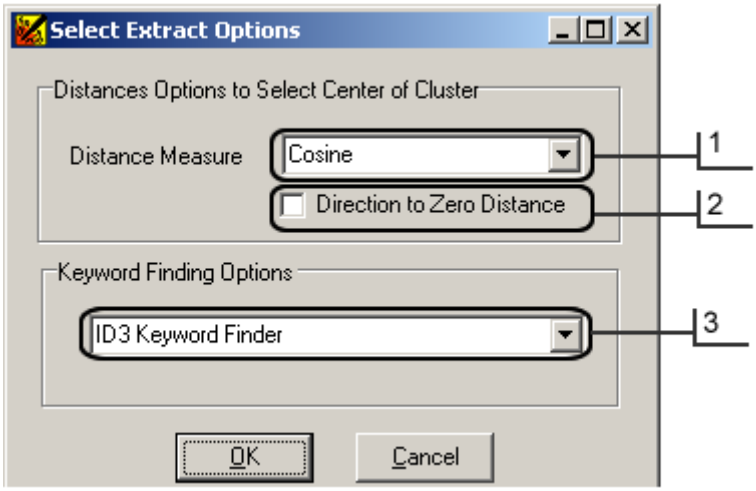


Figura 3.13 Resumen extracto del documento más representativo de cada cluster cuando se agrupó con SKWIC.

1. Selecciona la medida de cálculo de distancia entre los documentos y el centro de cluster.
2. Permite especificar si la medida es de similitud o disimilitud.
3. Selecciona la forma de calcular las palabras claves.

Cuando se realiza un agrupamiento borroso con Fuzzy SKWIC, o uno duro con *Extended Star*, y deseamos resumir solamente el documento más representativo de cada cluster, el diálogo que ofrece CorpusMiner coincide exactamente con el de la figura 3.11, sólo modificando las opciones de cálculo de palabras claves en función de cada uno de estos métodos. Esto se debe a que la identificación del documento más representativo cuando se agrupó con Fuzzy SKWIC no requiere especificar ningún parámetro adicional, ya que se toma aquel documento que tiene mayor grado de pertenencia al cluster. En el caso de *Extended Star*, el centro de cluster coincide con un documento real de la colección, por tanto, el centro de cada cluster es su documento más representativo y tampoco requiere especificar parámetros adicionales.

Evaluar

Existen tres formas de evaluar el agrupamiento realizado: *Overall Similarity*, Entropía y *F-Measure*. Este último criterio requiere la especificación de un parámetro α que se sugiere sea 0.5 para calcular *F-Measure* y pesar de igual forma los resultados de *precision* y *recall* que intervienen en su expresión de cálculo. Si se desea calcular solamente *precision*, entonces α=1. Si se desea calcular solamente *recall*, entonces α=0. Observe en la figura 3.14 el diálogo que ofrecemos en CorpusMiner para evaluar los resultados del agrupamiento.

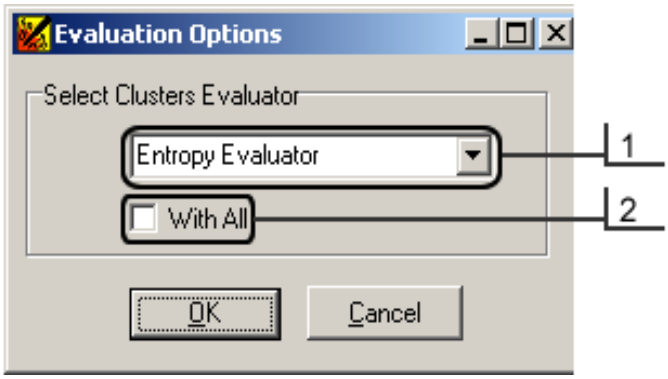


Figura 3.14 Selección de criterios para evaluar los resultados del agrupamiento.

1. Selecciona el criterio para realizar la evaluación.
2. Significa incluir las tres medidas implementadas para evaluar el agrupamiento.

Si el agrupamiento a evaluar utilizó una técnica borrosa, en este diálogo se incluye el umbral de pertenencia de los documentos a los clusters, de forma tal que se pueda definir qué documentos pertenecen a cada cluster.

Si en el diálogo de la figura 3.14 se presiona el botón OK, entonces es necesario especificar las opciones de la ventana de la figura 3.15. Nótese que todos los criterios de evaluación, excepto *Overall Similarity* requieren que los documentos estén previamente clasificados por tópicos. Por tanto, en el siguiente diálogo es necesario especificar en qué formato aparecerán los tópicos a los que pertenecen cada documento y un fichero donde se muestren todos los temas tratados en el corpus. Si no contamos con documentos previamente etiquetados, sólo podemos aplicar el criterio *Overall Similiarity*.

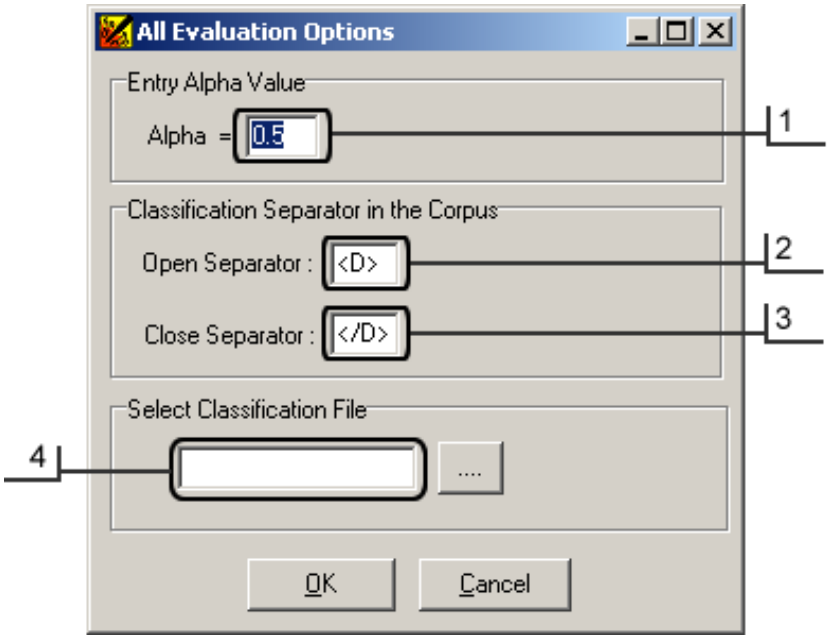


Figura 3.15 Especificación de la clasificación de cada documento y su formato.

1. Parámetro α que se utiliza en la medida *F-Measure*, su valor por defecto es α=0.5.
2. Delimitador de inicio utilizado para especificar en la primera línea de cada documento los tópicos abordados.
3. Delimitador de fin utilizado para especificar en la primera línea de cada documento los tópicos abordados.
4. Especifica el camino del fichero donde aparecen los tópicos correspondientes a la colección de documentos previamente agrupados.

Conclusiones

Con la realización de este trabajo se incorporaron a CorpusMiner los principales métodos de agrupamiento que facilitan la obtención de un resumen extracto, lográndose de esta forma dar cumplimiento al objetivo planteado ya que:

- Como resultado del análisis bibliográfico se seleccionaron los métodos de agrupamiento Extended Star, SKWIC y Fuzzy SKWIC. Estos dos últimos permiten simultáneamente formar grupos y obtener la relevancia de las palabras por cada cluster obtenido. Además se seleccionaron las estrategias que permiten obtener resúmenes extrayendo las principales oraciones del documento centro (más representativo) de cada cluster o de este más el resto de los documentos.
- El diseño concebido para la implementación de los métodos de agrupamiento facilita tanto aplicar los algoritmos por separado como la combinación de ellos. Además permite la incorporación de nuevos métodos de agrupamiento. Para ello, se incorporó el manejo de la similitud entre vectores documentos en función de lograr la homogeneidad de los clusters formados. En el diseño se incorporaron cuatro variantes de cálculo de similitud y es extensible a la incorporación de nuevas medidas con tales propósitos.
- Se programaron en CorpusMiner dos variantes que permiten obtener resúmenes extractos. La primera de ellas, al extraer los resúmenes del documento más representativote cada cluster, permitirá obtener resúmenes más sintéticos. La segunda variante, al extraer las oraciones de todos los documentos de cada cluster, generó resúmenes más explícitos, aunque puedan tener presencia de redundancia y anáforas.
- Se validaron los métodos de agrupamiento partiendo de un corpus textual de la agencia de noticias Reuters con 72 documentos previamente etiquetados en 7 tópicos. Teniendo en cuenta las medidas de similitud entre documentos, los mejores resultados fueron reportados cuando se utilizó la similitud Coseno para la comparación. Los mejores valores de entropía fueron obtenidos con el algoritmo *Extended Star*, mientras que éste reportó los peores valores de *F-Measure*. Las combinaciones de *Extended Star* con SKWIC y Fuzzy SKWIC logran mejores valores de entropía y de *Overall Similarity*, respecto a SKWIC y Fuzzy SKWIC, lo que garantizó la construcción de clusters homogéneos útiles en el proceso de construcción de resúmenes extractos. El algoritmo que más clusters aislados produce es el *Extended Star*.

Recomendaciones

A partir de los resultados obtenidos en este trabajo recomendamos:

- Estudiar algoritmos de agrupamiento jerárquico e incorporarlos a CorpusMiner para identificar la cercanía que existe entre los tópicos que se abordan en un corpus.
- Incluir en CorpusMiner nuevas medidas que permitan comparar la similitud entre documentos, diseñando por cada una de ellas una clase que implemente la interfaz *IDistance*. Por ejemplo, incorporar la medida *Dice* que según la literatura reporta buenos resultados en dominios textuales.
- Reducir redundancias y eliminar anáforas en los resúmenes extractos obtenidos.

Referencias bibliográficas

[ASL00] Aslam, J. Pelekhov, K. Rus, D. Scalable Information Organization. Proceedings of RIAO. 2000.

[ASL98] Aslam, J. Pelekhov, K. Rus, D. Static and Dynamic Information Organization with Star Clusters. Proceedings of the Conference of Information Knowledge Management, Baltimore, MD. 1998.

[BAL98] Baldwin, B. Morton, T. Co-reference-Based Summarization. T. Firm Hand & B. Sundheim (Eds), TIPSTER-SUMMAC Summarization Evaluation. Proceedings of the TIPSTER Text Phase III Workshop. 1998.

[BER01] Bergo, A. Text Categorization and Prototypes. 2001. www.illc.uva.nl/Publications/ ResearchReports/MoL-2001-08.text.pdf

[BER04] Berry, M. Survey of Text Mining. Clustering, Classification, and Retrieval. Springer-Verlag. ISBN 0-387-95563-1. 2004.

[BEZ73] Bezdek, J.C. Fuzzy Mathematics in Pattern Classification. Ph. D. Thesis, Applied Math. Center, Cornell University, Ithaca. 1973.

[BOL98] Boley, D.L. Principal direction divisive partitioning. Data Mining and Knowledge Discovery. 2(4). pp. 325-344. 1998.

[BOO91] Booch, G. Object Oriented design: with applications. The Benjamin Cummings Publishing's Company Inc. U.S.A. 1991.

[BOO99] Booch, G. Rumbaugh, J., Jacobson, I. The Unified Modeling Language User Guide. Addison-Wesley Longman. 1999.

[CHE98] Cherkassky, V. Mulier, F. Learning from Data. Concepts, Theory, and Methods. A Wiley-Interscience Publication. John Wiley & Sons, Inc. 1998.

[CRO02] Cross, V. Sudkamp, T. Similarity and Compatibility in Fuzzy Set Theory. Assessment and Applications. Phisica-Verlag. Springer-Verlag Company. ISSN 1434-9922. 2002.

[DIX97] Dixon, M. An Overview of Document Mining Technology. 1997.

www.geocities.com/ResearchTriangle/Thinktank/1997/mark/writings/dixm97_dm.ps

[DUC00] Duch, W. Similarity-based methods: a general framework for classification, approximation and association. Control and Cybernetics. Vol. 29. No. 4. 2000.

[DUR01] Dürsteler, J.C. Minería de Textos. Inf@Vis! La revista digital de InfoVis.net. Mensaje No. 27. 2001.

[FEB02] Febles, J.P., González, A. Aplicación de la minería de datos en la Bioinformática. Centro Nacional de Bioinformática. 2002.

[FOR03] Forman, G. An Extensive Empirical Study of Feature Selection Metrics for Text Classification. Journal of Machine Learning Research 3. pp. 1289-1305. 2003.

[FRA03] Franke, J., Nakhaeizadeh, G., Renz, I. Advances in Soft Computing. Text Mining.

Theoretical Aspects and Applications. Physica-Verlag. ISBN 3-7908-0041-4. 2003.

[FRA92] Frankes, W. B., Baeza-Yates, R. Information Retrieval. Data Structures & Algorithm. Prentice Hall PTR. ISBN 0-13-463837-9. 1992

[FUK03] Fukumoto, J. Text summarization based on itemized sentences and similar parts detection between documents. Proceedings of the Third NTCIR Workshop. 2003.

[FUK03] Fukumoto, J. Text summarization based on itemized sentences and similar parts detection between documents. 2003.

http://research.nii.ac.jp/ntcir/workshop/OnlineProceedings3/NTCIR3-TSC-FukumotoJ.pdf

[FUK99] Fukuhara, T. Takeda, H. Nishida, T. Multiple-text Summarization for Collective Knowledge Formation. Proceedings of Workshop of Social Aspects of Knowledge and Memory. IEEE Systems, Man and Cybernetics Conference. 1999.

[FUN02] Fung, B. Hierarchical Document Clustering using Frequent Itemsets. Master of Science in the School of Computing Science. Simon Fraser University. 2002.

[GAB04] Gabrielsson, S. MOVSOM-II - analysis and visualization of movieplot clusters. 2004. http://www.pcpinball.com/movsom.

[GAR99] García, M.M. Monografía de reconocimiento de patrones. Universidad Central "Marta Abreu" de Las Villas. 1999.

[GAS01] Gasperin. C, Gamallo. P. López, G. Lima, V. Using Syntactic Contexts for Measuring Word Similarity. 2001. www.cl.cam.ac.uk/users/cvg20/esslli01.pdf

[GIL03] Gil-García, R. Badía-Contelles, J.M. Pons-Porrata, A. Extended Star Clustering Algorithm. Proceedings of CIARP. 2003.

[GOL00] Goldstein, J. Mittal, V. Carbonell, J. Callan, J. Creating and Evaluating Multi-document Sentences Extract Summaries. CIKM 2000. pp. 165-172. 2000.

[GOL99] Goldstein, J. Automatic Text Summarization of Multiple Documents. Thesis Proposal. 1999.

[HOP99] Höppner, F. Klawonn, F. Rudolf, K. Runkler, T. Fuzzy Cluster Analysis. Methods for Classification, Data Analysis and Image Recognition. John Wiley & Sons Ltd. 1999.

[IND01] Inderjeet, M. Gates, B. Bloedorn, E. Improving Summaries by Revising Them. 2001. acl.ldc.upenn.edu/P/P99/P99-1072.pdf

[JAC02] Jackson, P. Moulinier, I. Natural Language Processing for Online Applications. Text Retrieval, Extraction and Categorization. John Benjamins Publishing Company. ISBN 902724988. 2002.

[JAC02] Jackson, P. Moulinier, I. Natural Language Processing for Online Applications. 2002. http://www.benjamins.nl/jbp/series/IPJDD/12-1/art/0015a.pdf

[JAM01] James, A. Gupta, R. Center for Intelligent Information Retrieval Temporal Summaries of News Topics. 2001. www-ciir.cs.umass.edu/~allan/Papers/2001-sigir.pdf

[JAN97] Jang, J.S. NeuroFuzzy and Soft Computing. Prentice Hall. ISBN: 0-13-261066. 1997.

[KAR01] Karypis, G. Evaluation of Item-Based Top-N Recommendation Algorithms. 2001. http://www.cs.umn.edu/~karypis/suggest

[KEN97] Kendall, K.E., Kendall, J.E. Análisis y diseño de sistemas. Tercera Edición. Pretice Hall. 1997.

[KEO99] McKeown, K.R. Klavans, J.L. Towards Multidocument Summarization by Reformulation: Progress and Prospects. 1999. http://newsblaster.cs.columbia.edu/papers/stim2.pdf

[LAN01] Lanquillon, C. Enhancing Text Classification to Improve Information Filtering. Dissertation Ph.D. Fakultat fur Informatik der Otto-von-Guericke Universitat Magdeburg. 2001.

[LAR00a] Larocca, J. Santos, A. Kaestner, C. Freitas, A. Generating Text Summaries Through the Relative Importance of Topics. Int. Joint Conf. IBERAMIA 2000 (7th Ibero-American Conf. on Artif. Intel.) & SBIA-2000 (15th Brazilian Symp. On Artif. Intel.). Brazil. pp 300-309. 2000.

[LAR00b] Larocca, J. Santos, A. A trainable algorithm for summarizing news stories. 4th European Conference on Principles and Practice of Knowledge Discovery in Databases (PKDD'2000). 2000.

[LAR04] Larocca, J. Santos, A. Celso, A. A trainable algorithm for summarizing news stories. 2004. http://www.chirouble.univ-lyon2.fr/~pkdd2000/Download/WS4_04.pdf

[LAR99] Larocca, J. Santos, A. DocClustering: um sistema para agrupamento de documentos. International Seminar on Document Management. Curitiba, Brazil. 1999.

[LEZ02] Lezcano, R.D. Minería de Datos. 2002.

http://www.google.com.cu/depar/areas/informatica/SistemasOperativos/MineriaDatosLezcano.pdf

[MAL04] Maloney, J. Text Mining Solutions. Services & Solutions. 2004.

[MAN00] Manning, C. Shütze, H. Foundations of Statistical Natural Language Processing. MIT Press, Cambridge, MA. 2000.

[MAN02] Mani, I. House, D. Klein, G. Hirschman, L. The TIPSTER SUMMAC Text Summarization Evaluation. The MITRE Corporation. 2002. http://acl.ldc.upenn.edu/E/E99/E99-1011.pdf

[MAR00] Marcu, D. The Theory and Practice of Discourse Parsing and summarization. Cambridge, MA: MIT Press. 2000.

[MCK99] McKeown, K. Klavans, J. Hatzivassiloglou, V. Towards Multidocument Summarization by Reformulation: Progress and Prospects. American Association for Artificial Intelligence. 1999.

[MIT97] Mitchell, T. Machine Learning. McGraw-Hill Science, Engineering, Math. ISBN 0070428077. 1997.

[MOE00] Moens, M.F. Automatic Indexing and Abstracting of Document Texts, Chapter 7. Norwell, MA: Kluwer Academic. 2000.

[NAN00] Nanba, H. Okumura, M. Producing More Readable Extracts by Revising Them.

Proceedings of the 18th International Conference on Computational Linguistics (COLING-2000), pp. 1071-1075. 2000.

[NER01] Neri, F. A new way to Explore Patents Databases. Text Mining Solutions – Lexical Systems. 2001. http://www.synthema.it/textmining

[NUR01] Nürnberger, A. Klose, A. Kruse, R. Clustering of Document Collection to Support Interactive Text Exploration. Studies in Classification, Data Analysis and Knowledge Organization. Exploratory Data Analysis in Empirical Research. Proceedings of the 25th Annuals Conference of the Gesellschaft für Klassification. pp 291-299. 2001.

[OBE01] Obeso, C. Apuntes de éxito. 2001. www.infonomia.com

[PRO03] Proyecto México-Cuba. Redes Neuronales para la Minería de Datos y Textos: Aplicación al Análisis Exploratorio y Descubrimiento de Conocimiento en Grandes Bases de Datos de Información Biomédica. 2003.

[QUE67] McQueen, J. Some methods for classification and analysis of multivariate observations. Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability. pp. 182-297. 1967.

[RAD00] Radev, D. Hongyan, J. Malgorzata, B. Centroid-based summarization of multiple documents: sentence extraction,utility-based evaluation, and user studies. 2000. http://acl.ldc.upenn.edu/W/W00/W00-0403.pdf

[ROD99] Rodríguez, M. El concepto de tipo y la teoría de programación actual. Revista GIGA. No. 4. 1999.

[RUM97a] Rumbaugh, J. Booch, G. Jacobson, I. UML Notation Version 1.1. 1997.

[RUM97b] Rumbaugh, J. Booch, G. Jacobson, I. UML Semantics Version 1.1. 1997.

[RUN01] Runkler, T.A. Bezdek, J.C. Relational Clustering for the Analysis of Internet Newsgroups. Proceedings of the 25th Annuals Conference of the Gesellschaft für Klassifikation. University of Munich. pp. 291-299. 2001.

[SAL71] Salton, G. The SMART Retrieval System. Prentice-Hall, Englewood Cliffs, NJ. 1971.

[SAR00] Sarwar, B. Karypis, G. Konstan, J. Riedl, J. Analysis of Recommendation Algorithms for E-Commerce. 2000. http://www.grouplens.org/papers/padf/ec00.pdf

[SCH02] Schiffman, B. Nenkova, A. McKeown, K. Experiments in Multidocument Summarization. Proceedings of HLT 2002 Human Language Technology Conference. San Diego, CA. 2002.

[SEB99] Sebastiani, F. A Tutorial on Automated text Categorisation. 1999. http://citeseer.ist.psu.edu/cache/papers/cs/12228/http:zSzzSzfaure.iei.pi.cnr.itzSz~fabriziozSzPublicationszSzASAI99zSzASAI99.pdf/sebastiani99tutorial.pdf

[SHA48] Shannon, C. A mathematical theory of communication, Bell System Technical Journal. 1948.

[STE00a] Stein, G. Bagga, A. Wise, G.B. Multi-document summarization: Methodologies and evaluations. Proccedings of TALN-2000. 2000.

[STE00a] Steinbach, M. Karypis, M. A Comparison of Document Clustering Techniques. 2000. http://www.l.cs.cmw.edu/~dunja/KDDpapers/steinbach-IR.pdf

[TAL04] Taller de Herramientas y Recursos Lingüísticos para el desarrollo del Español y el Portugués. IX Congreso Iberoamericano de Inteligencia Artificial (IBERAMIA). 2004.

[TAN99] Tan, A. Text Mining: The state of the art and the challenges. Proceedings of PAKDD'99. pp 65-70. 1999.

[VAL05] Valdés, L. Representación de textos y su reducción de dimensionalidad. Trabajo de diploma. Universidad Central "Marta Abreu" de Las Villas. 2005.

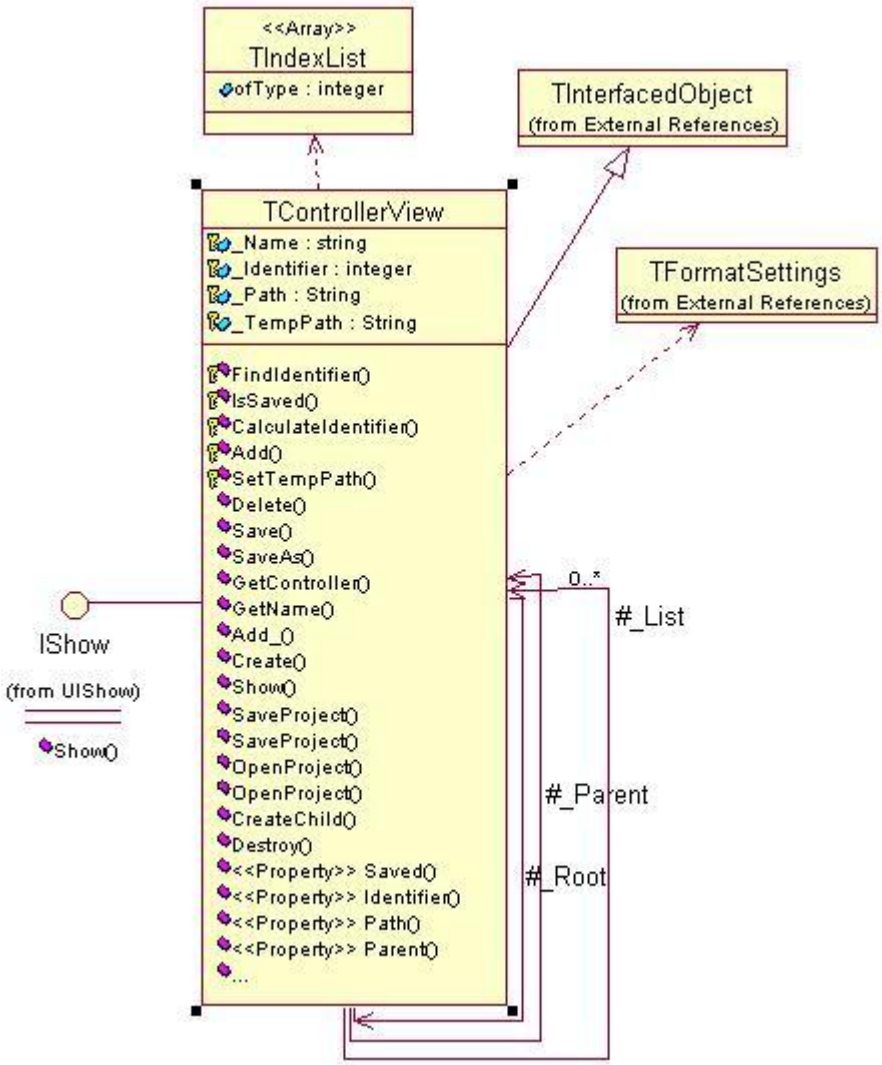
[WHI02] White, M. Selecting Sentences for Multidocument Summaries using. Randomized Local Search.CoGenTex, Inc. 840 Hanshaw Road. Ithaca, NY 14850, USA 2002. http://www.cogentex.com/papers/acl-02-summ-wkshop.pdf

[WIL97] Wilson, D.R. Martinez, T.R. Improved Heterogeneous Distance Functions. Journal of Artificial Intelligence Research. No. 6. pp. 1-34. 1997.

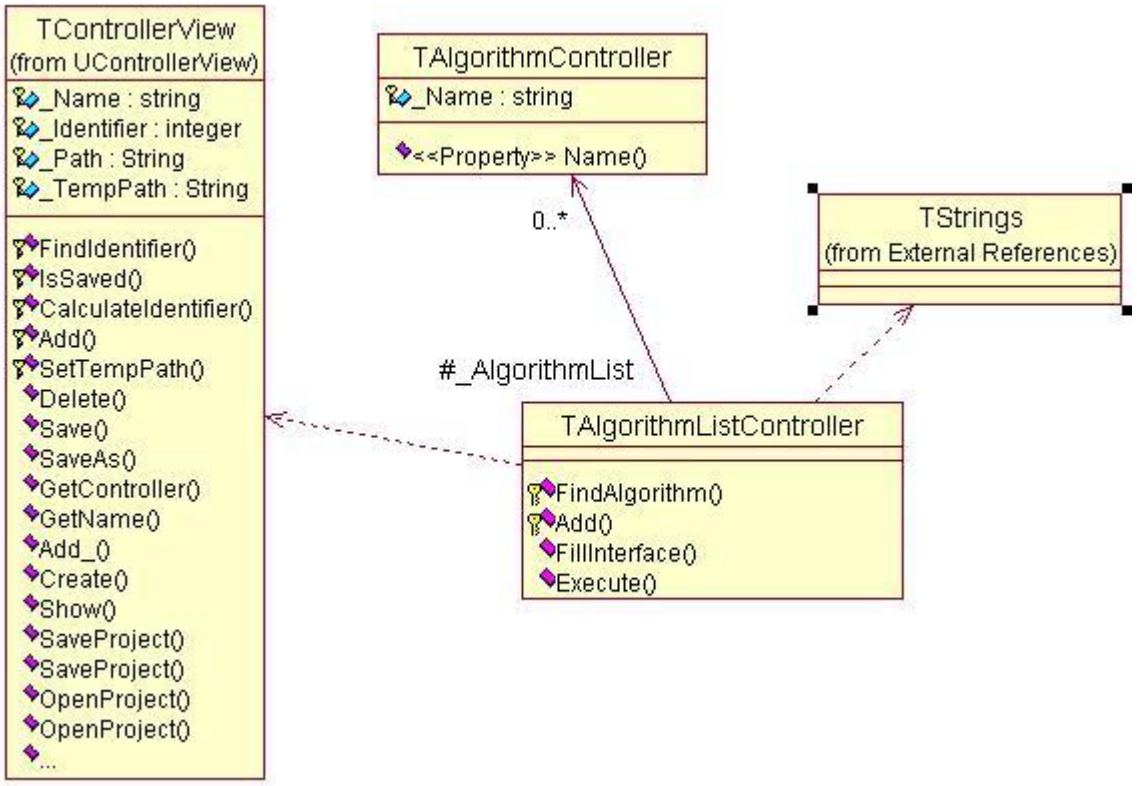
[YAR92] Yarowsky, D. Word sense disambiguation using statistical models of roget's categories trained on large corpora. In Proceedings of the Fourteenth International Conference on Computational Linguistics. pp. 454-460, Nantes, France. 1992.

Anexos

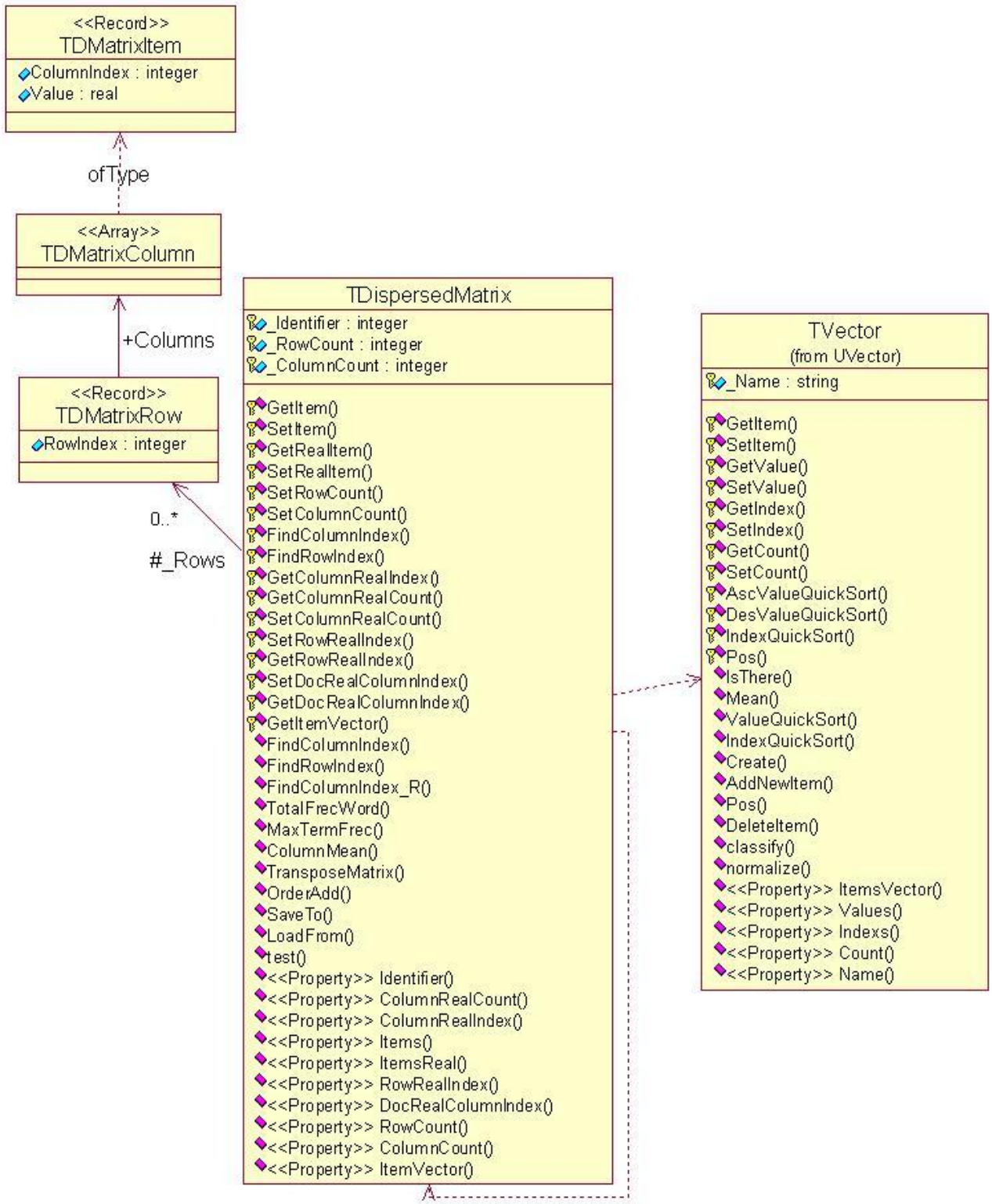
***Anexo 1*:** Diseño UML de *TControllerView* , *TAlgorithmController* y *TAlgorithmListController*



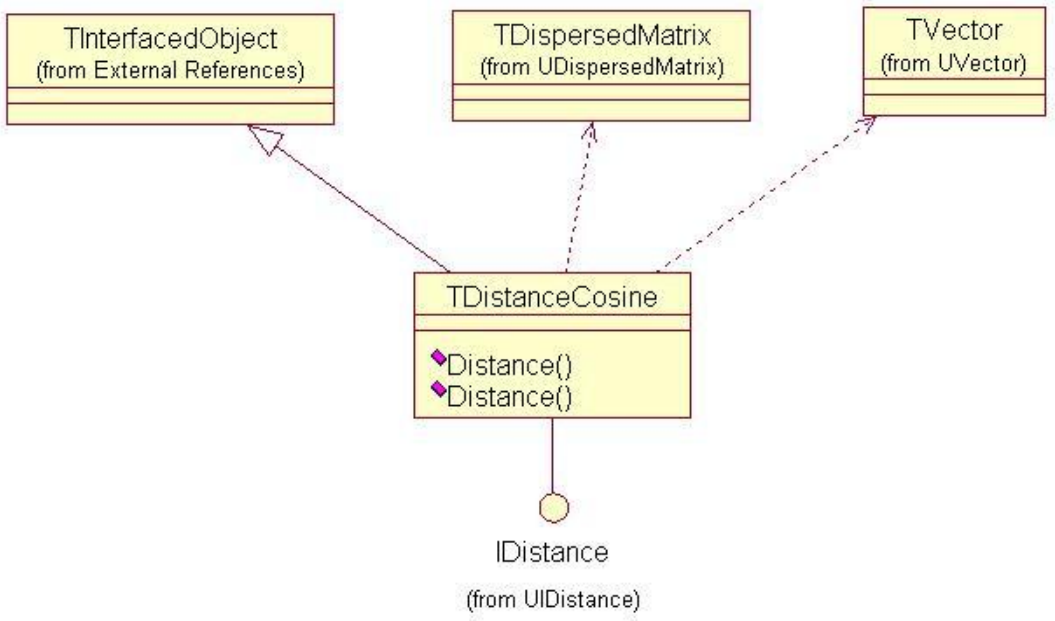
***Anexo 1*: (continuación)**



Anexo 2: Diseño de clases de *TVector* y *TDispersedMatrix*

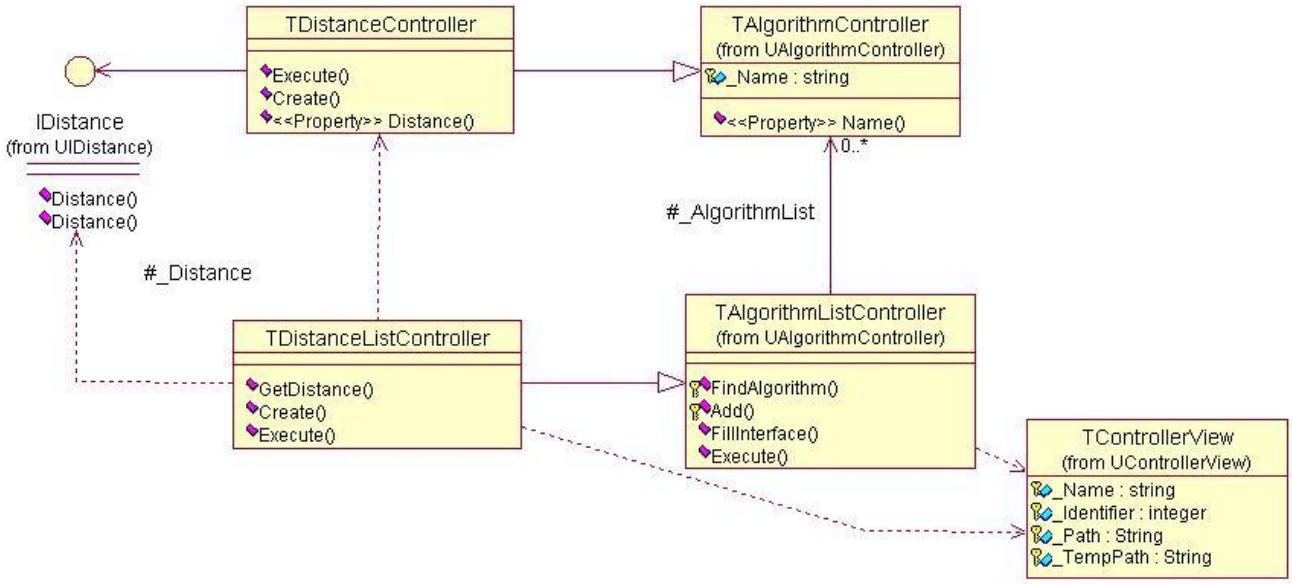


Anexo 3: Diseño de clases para el cálculo de las distancias (*TDistanceCosine*)

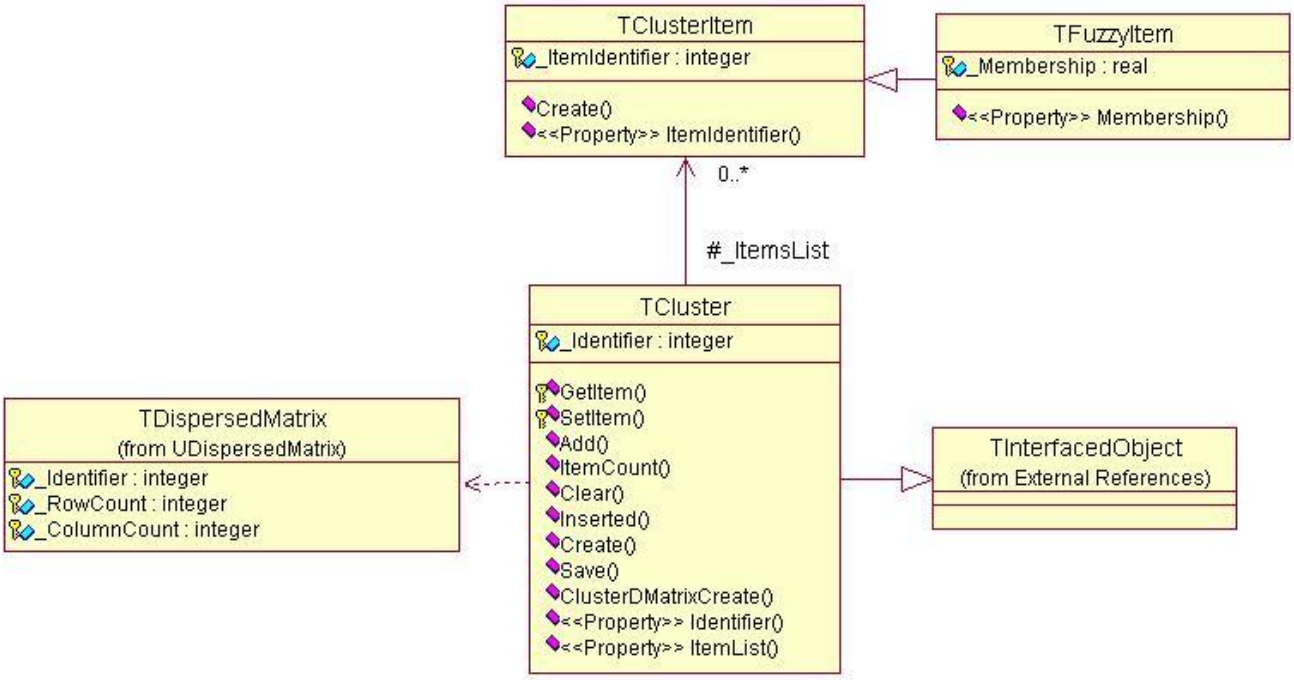


De forma similar se definen las clases: *TDistanceEuclidian*, *TDistanceBinaryJaccard* y *TDistanceWeightJaccard.*

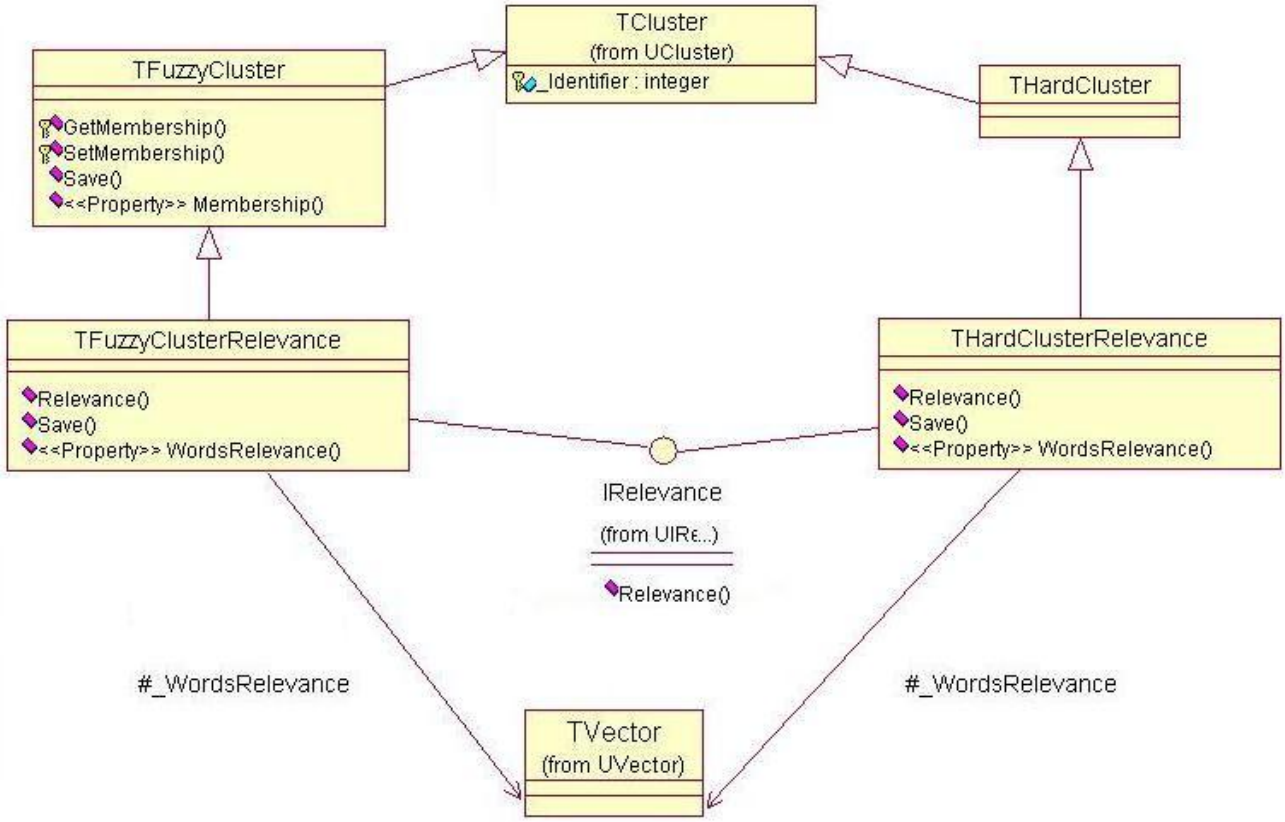
Anexo 4: Diseño de las clases *TDistanceController* y *TDistanceListController*



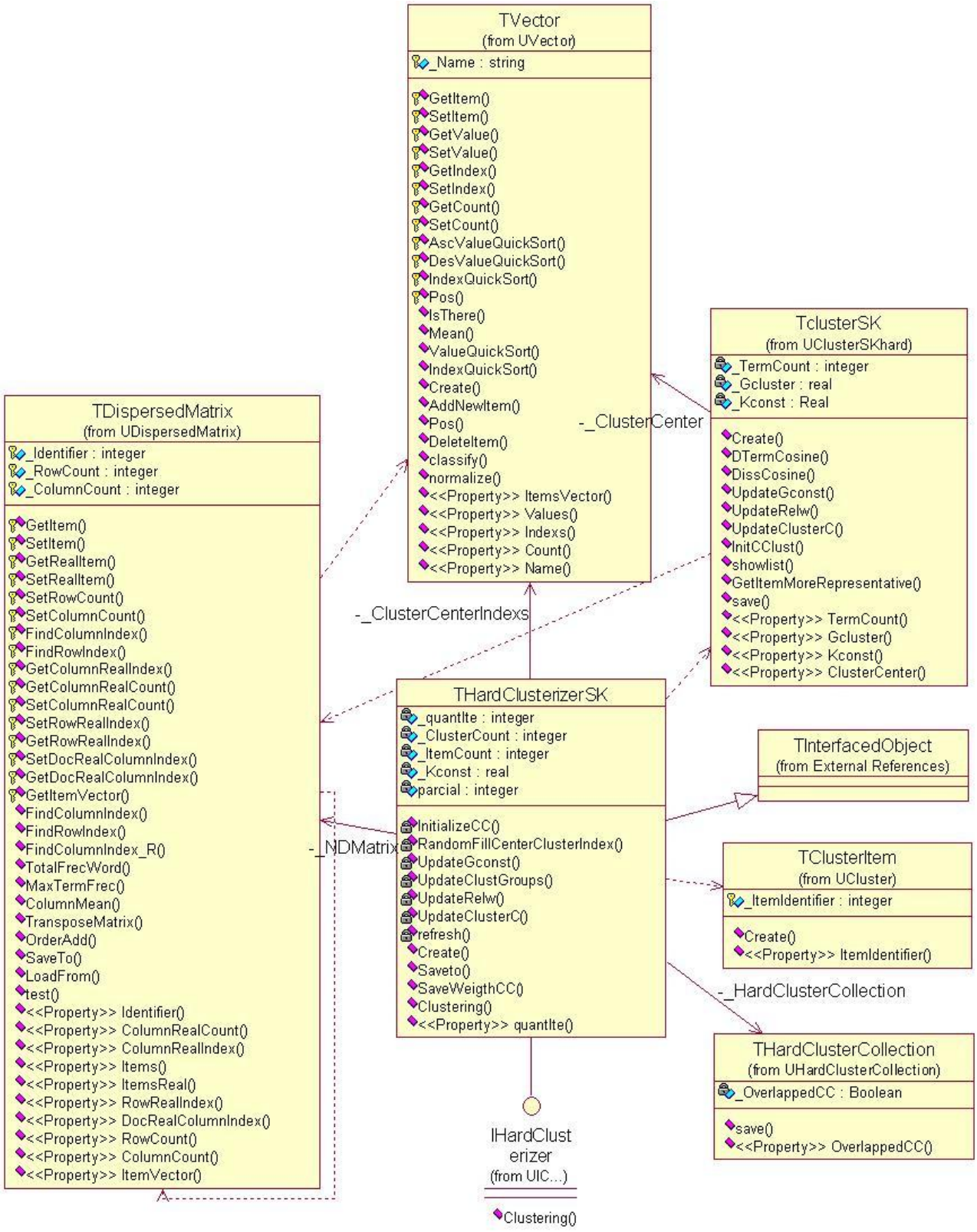
Anexo 5: Diseño de las clases *TCluster*, *TClusterItem* y *TFuzzyItem*



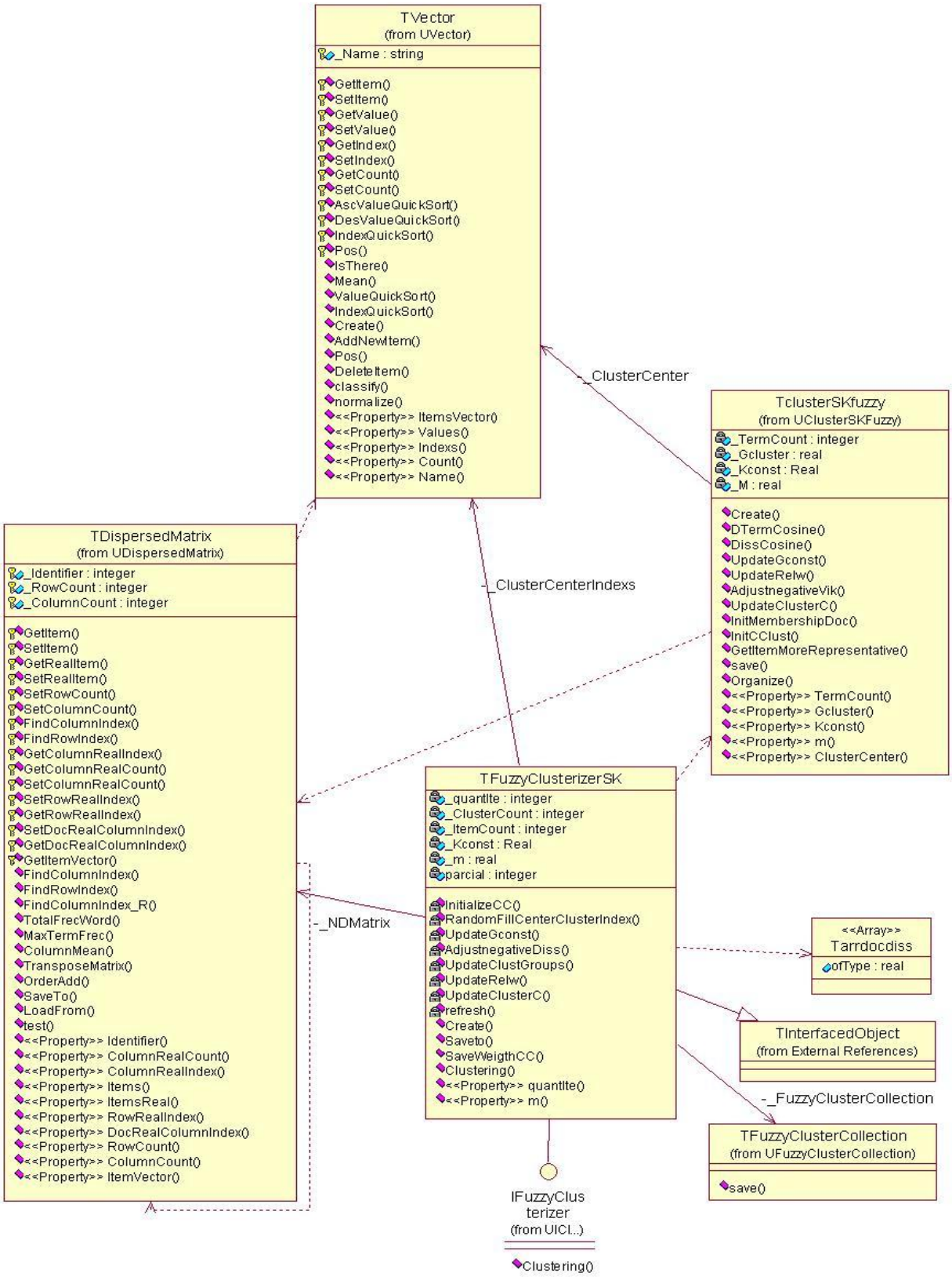
Anexo 6: Diseño de las clases *THardClusterRelevance* y *TFuzzyClusterRelevance*



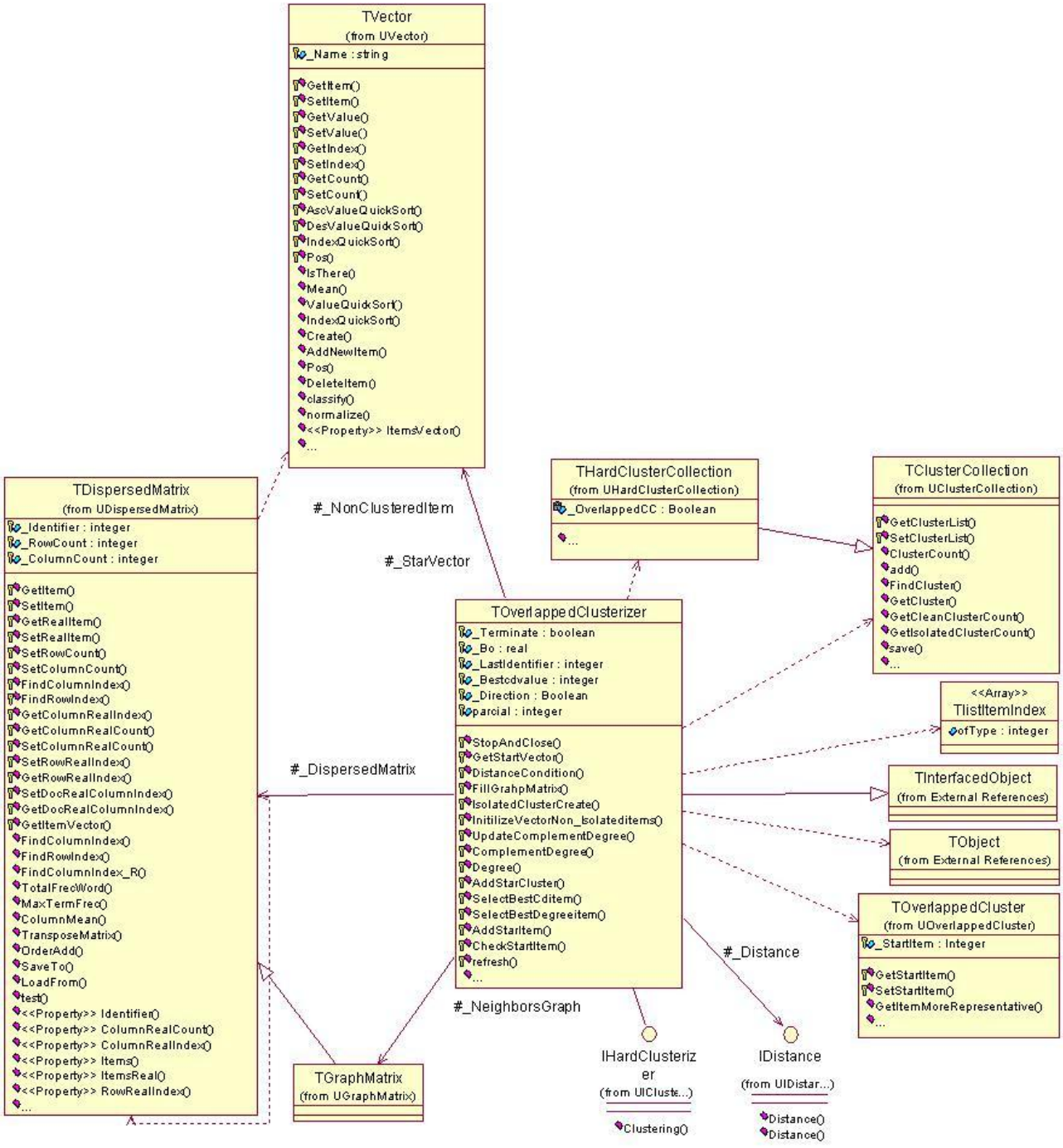
Anexo 7: a) Diseño de la clase *THardClusterizer*



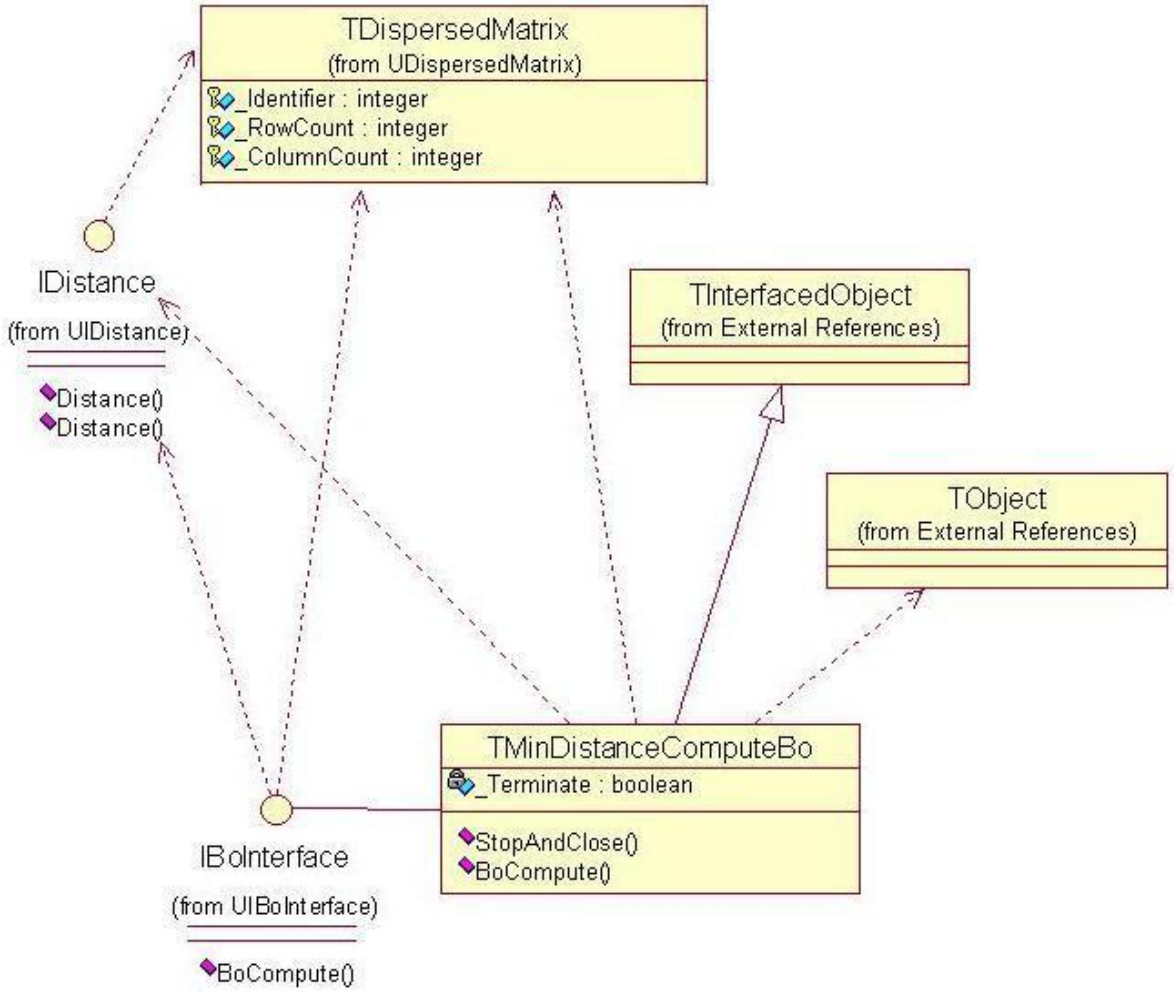
Anexo 7: b) Diseño de la clase *TFuzzyClusterizer*



Anexo 7: c) Diseño de la clase *TOverlappedClusterizer*

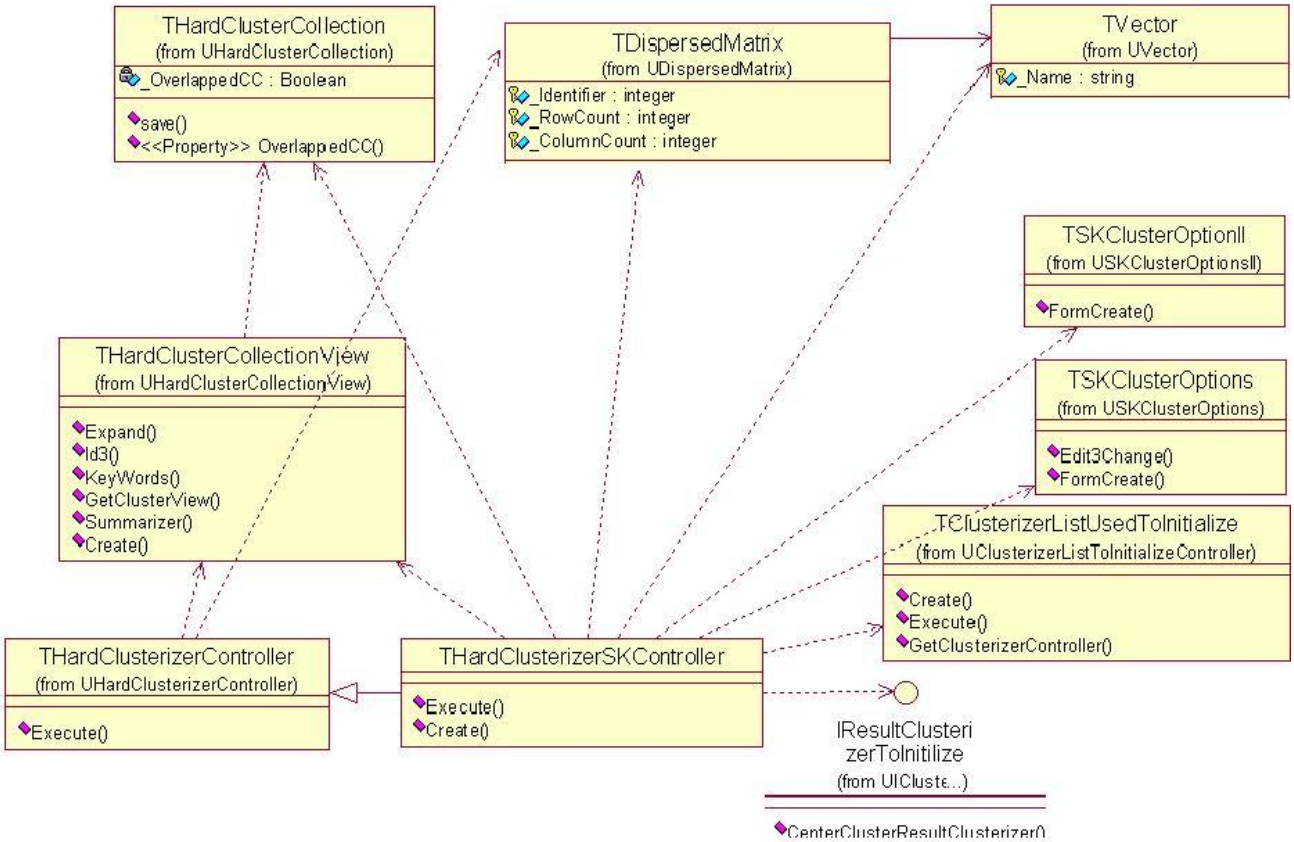


Anexo 8: Diseño de clases para el cálculo del parámetro β_0: *TMinDistanceComputeBo*

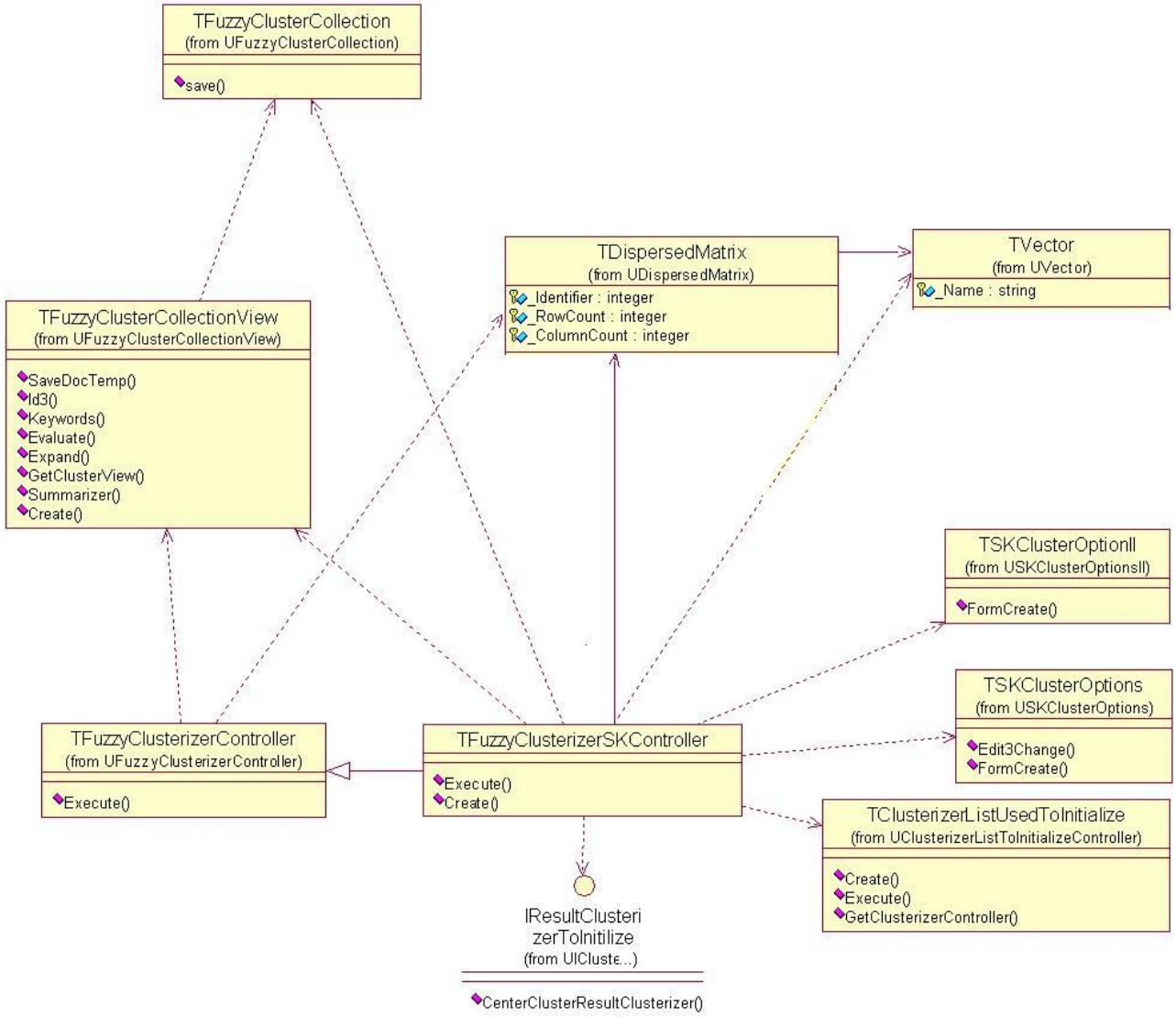


De forma similar se definen las clases: *TMeanDistanceComputeBo* y *TMaxDistanceComputeBo*.

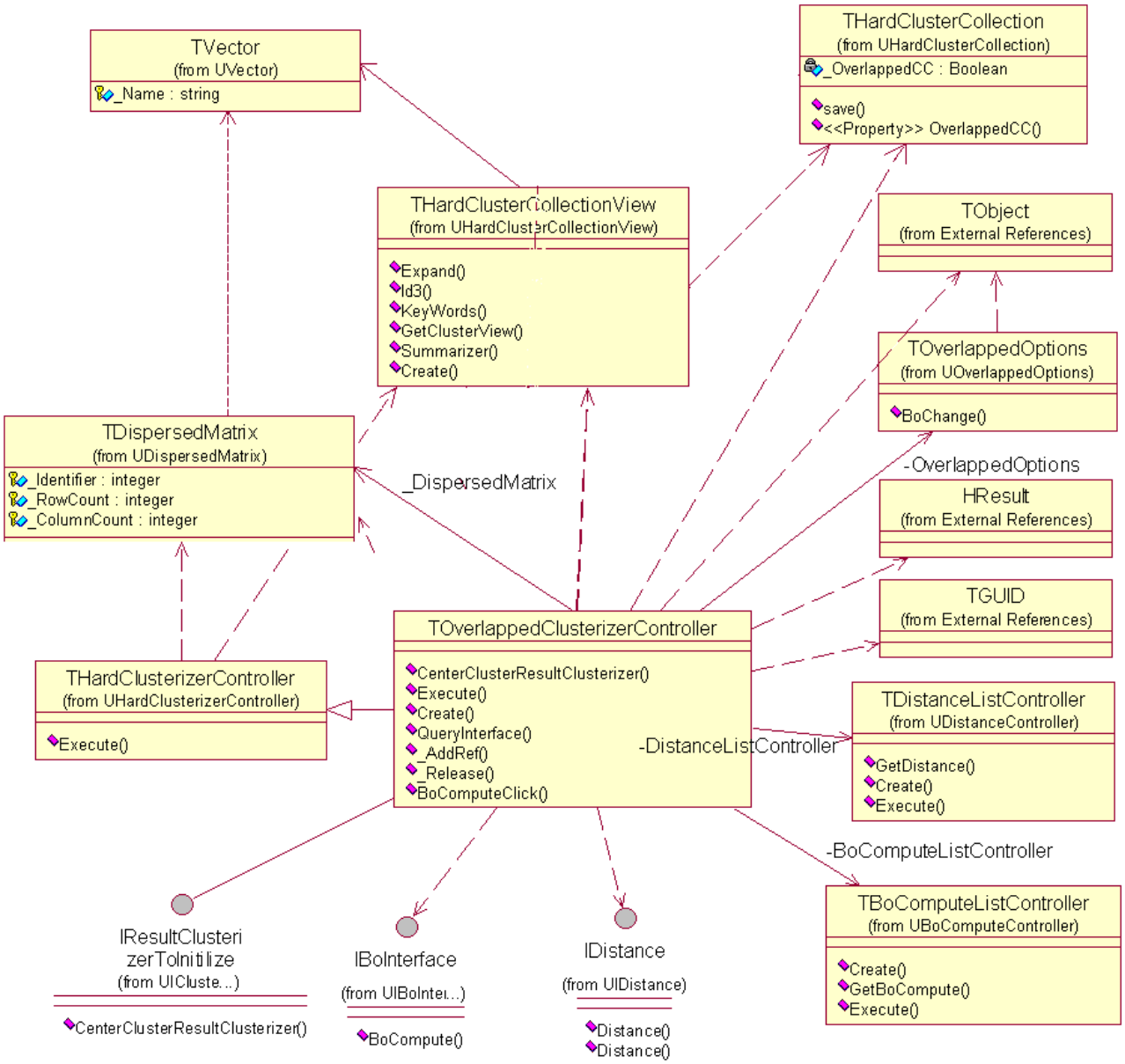
Anexo 9: a) Diseño de la clase controladora: *THardClusterizerSKController*



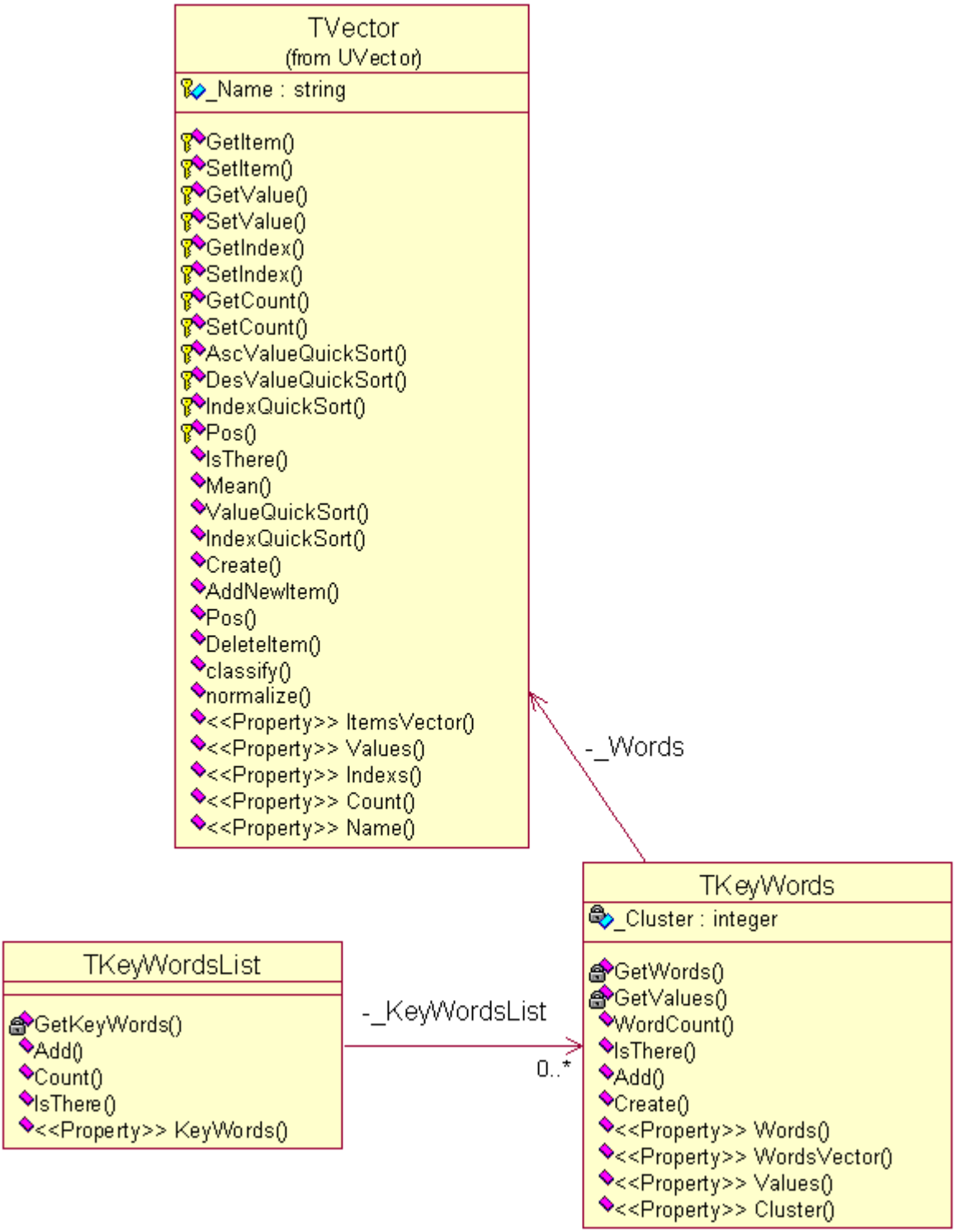
Anexo 9: b) Diseño de la clase controladora: *TFuzzyClusterizerSKController*



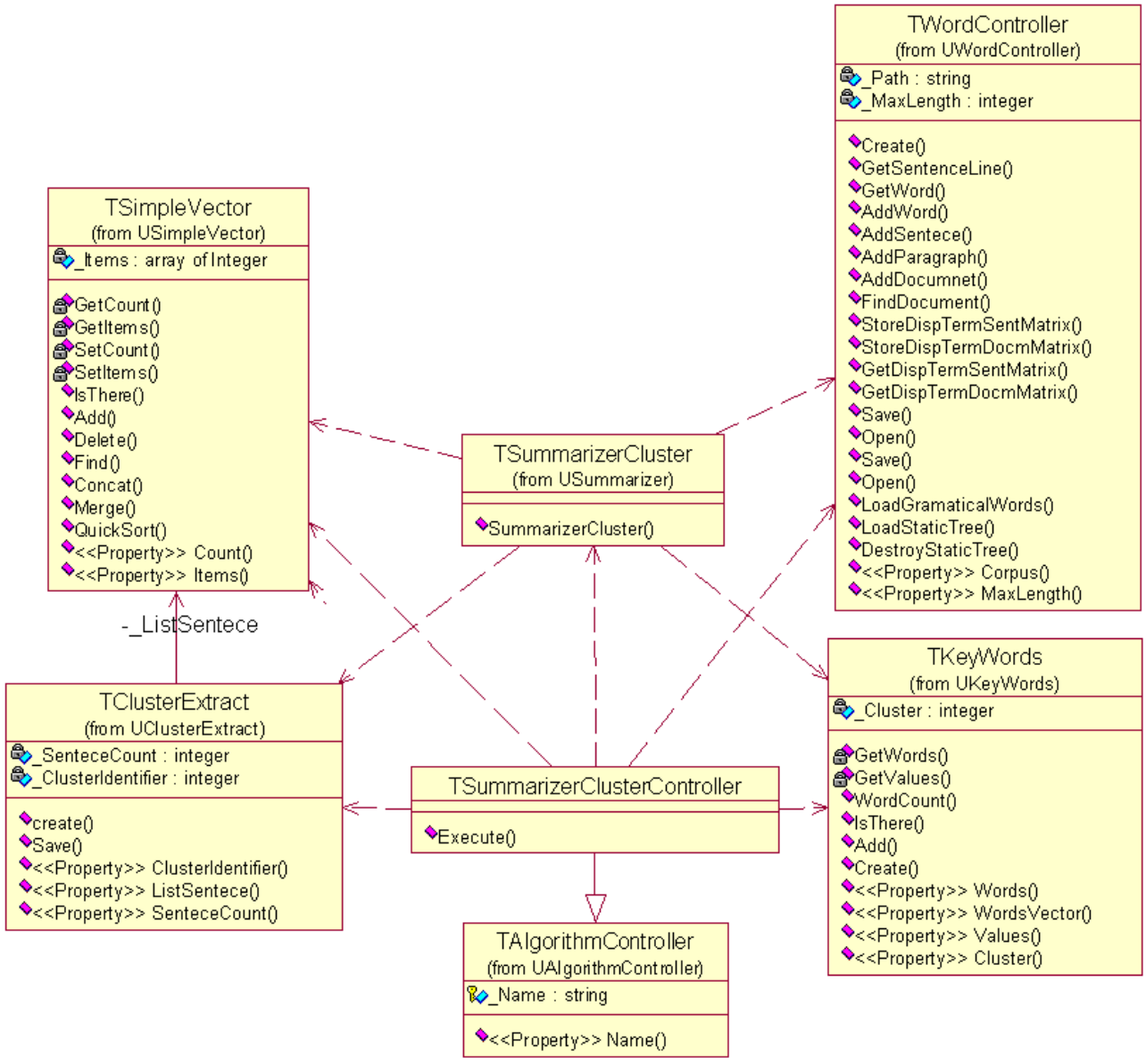
Anexo 9: c) Diseño de la clase controladora: *TOverlappedClusterizerController*



Anexo 10: Diseño de la clase *TKeywords* para la obtención de palabras claves



Anexo 11: Diseño de la clase *TClusterExtract* para la selección del extracto



De forma similar se define la clase: *TClusterCollectionExtract.*