UNIVERSIDAD AUTÓNOMA METROPOLITANA
Unidad Azcapotzalco

DESARROLLO DE UNA METODOLOGÍA E IMPLEMENTACIÓN PARA LA AGRUPACIÓN DE DOCUMENTOS XML

TESIS

Que para obtener el grado de
MAESTRA EN CIENCIAS DE LA COMPUTACIÓN

presenta

Ing. Cristal Karina Galindo Durán

Dr. Héctor Javier Vázquez
Director de Tesis

Dra. Mihaela Juganaru de Mathieu
Co-Directora de Tesis

México, Distrito Federal
Julio 2012

"Haz lo necesario para lograr tu más ardiente deseo, y acabarás lográndolo."

Ludwig Van Beethoven

A Dios

Por llenarme de bendiciones día a día, por todo el amor con el que me rodeas y por que me tienes en tus manos.

A mi madre

Por haberme dado la vida, por cuidar de mi, por apoyarme a culminar un sueño más.

A mi esposo

Por creer en mí en todo momento, por todo el apoyo que me has dado para continuar y seguir con mi camino. Gracias por estar conmigo, por tu amor, comprensión y respeto, por ser mi compañero de toda la vida, te amo.

A mi hija Lila

Gracias por alegrarme cada día con tu sonrisa, por darle un sentido diferente a mi vida, por enseñarme tanto siendo aún tan pequeña, te amo.

A mi hermana Laura y su esposo

Gracias por estar conmigo y apoyarme siempre.

A Magu

Por continuar siendo mi compañera fiel.

A la Dra. Mihaela Juganaru de Mathieu

Le agradezco por sus ideas invaluables, por su apoyo y conducción para lograr este proyecto posible.

Al Dr. Héctor Javier Vázquez

Le agradezco por confiar en mí, por tenerme la paciencia necesaria, por apoyarme en momentos difíciles, por darme siempre ánimos para continuar. Por ser una buena persona.

A todos mis profesores

Quisiera nombrar a cada uno de ustedes; pero son muchos, sin embargo eso no quiere decir que no los tenga presentes a todos. Les agradezco por regalarme sus conocimientos y compartirme cada uno sus experiencias, lo que ha transformado mi vida.

A mis compañeros

Por compartir este tiempo dentro de las aulas.

A la Universidad Autónoma Metropolitana

Por darme la oportunidad de formar parte de ella, gracias.

Resumen

El Lenguaje eXtensible de Marcas (XML, por sus siglas en inglés) es un lenguaje de descripción de datos para la representación, intercambio y almacenamiento de la información de manera estructurada y/o semi-estructurada, sin necesidad de especificar el cómo debe de ser vista la información. Esto significa que el contenido y la presentación son independientes.

XML también ofrece los medios para compartir información de manera fiable y fácil; permitiendo la compatibilidad entre diferentes sistemas y plataformas, así como en diferentes aplicaciones. Estas razones hacen que XML sea el lenguaje preferido para codificar datos textuales y no textuales. Coadyuvando en la construcción de diversas aplicaciones como : servidores web, aplicaciones en línea y para la creación de colecciones de documentos.

La mayoría de los datos textuales no se almacenan en bases de datos. Para esto XML permite almacenar colecciones de texto, estructurado o no, con enlaces a otros documentos, datos e imágenes (dentro y fuera de la colección).

El objetivo de este trabajo es proponer un marco para el diseño de una aplicación modular y configurable para la minería XML, en particular, para la agrupación de documentos en grandes colecciones, como INEX (Iniciativa para la Evaluación y Recuperación XML), integrada por información textual semiestructurada. Este marco está destinado a conducir el desarrollo de una aplicación para el minado de colecciones de documentos XML, de manera flexible y configurable que permita al usuario elegir los siguientes parámetros : la colección a minar, trabajar con todos los documentos XML o sólo una muestra, el tipo de tokens a minar, el tipo de normalización, el tipo de algoritmo a aplicar y dónde guardar los resultados de agrupamiento; así como los índices de Dunn y Davies-Bouldin, los cuales evaluan la calidad de los grupos obtenidos. La tesis culmina con resultados concretos obtenidos mediante el minado aplicado a una muestra representativa de la colección INEX.

Contenido

Lista de Figuras

Lista de Tablas

Capítulo 1

Introducción

Actualmente se almacena una gran cantidad de datos numéricos y/o textuales en formato XML lo que implica, un nuevo e importante desafío, el cual trae consigo : la extracción de conocimiento que no se encuentra explícito en las colecciones XML almacenadas.

XML fue concebido inicialmente como un formato de representación de información para Internet, pero su poder de representación y su uso actual sobrepasa mucho más. A finales de 1990 se presentó la necesidad de poder realizar páginas web vistosas y de diversos usos, por lo que se propuso separar el contenido de la presentación, con el fin de facilitar el manejo del contenido y evitar el rediseño de la interfaz. Es así como surge XML como lenguaje de almacenamiento e intercambio de información.

XML, por sus siglas en inglés de eXtensible Markup Language (lenguaje de marcas extensible), es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium[1], el cual ha surgido como un estándar prometedor para el almacenamiento, intercambio y recuperación, que lleva a la integración de la información [Harold04]. Desde su creación en 1998 ha sido muy útil para almacenar datos estructurados o semi-estructurados [Abiteboul01] en una forma independiente de cualquier presentación final, esto facilita el desarrollo de aplicaciones en línea e inclusive las de comercio electrónico y la creación de colecciones de documentos XML, sin limitar su uso a Internet. Sin embargo, para analizar, organizar y tratar la gran cantidad de información almacenada. En el formato XML se

[1] *http://www.w3c.org/XML/*

requiere de nuevas metodologías y herramientas computacionales y estadísticas. De aquí resulta la minería XML.

La minería XML [Candillier07] se define como el proceso que permite el descubrimiento de nuevos conocimientos que no están explícitos en la información que se analiza, pero que surge al relacionar el contenido de varios documentos estructurados. La minería XML se encarga de descubrir información a partir de la estructura y/o contenido de documentos de varias fuentes.

La investigación en minería XML es reciente; aunque es importante aclarar que el proceso de minería comparte en gran medida etapas propuestas por las técnicas de minería de datos [Ramírez04] y de minería de texto [Srivastava09]. La forma multidimensional y el tratamiento de la información almacenada en XML ha generado la necesidad de incluir en el proceso de minería la estructura.

Hoy en día existe la Iniciativa para la Evaluación de Recuperación de XML llamada INEX[2]. En ésta se proponen herramientas y pruebas para evaluar los métodos en forma cualitativa y cuantitativa, así como herramientas para la búsqueda de información, rutinas de consulta, manejo de colecciones, uso de eBook, etc. Todas ellas propuestas por distintos grupos de investigación.

Desde el 2007, el objetivo de dicha conferencia es poner a disposición de los investigadores de todo el mundo grandes volúmenes de documentos XML como colecciones de prueba. Hasta el 2010 un tema principal para la conferencia INEX era el agrupamiento (clustering) y el presente trabajo aportó los resultados mostrados en el apéndice H, en las secciones H.1, H.2, H.3 y H.4.

1.1. XML

XML es un metalenguaje extensible de etiquetas que permite definir la gramática de lenguajes específicos (de la misma manera que HTML es a su vez un lenguaje definido por SGML). Por lo tanto, XML no es realmente un lenguaje de programación, sino un lenguaje de descripción de la información y también una manera de definir lenguajes para

[2] *https://inex.mmci.uni-saarland.de/*

diferentes necesidades. Algunos de estos lenguajes que usan XML para su definición son XHTML, SVG, MathML, tal como se muestra en la figura 1.1.

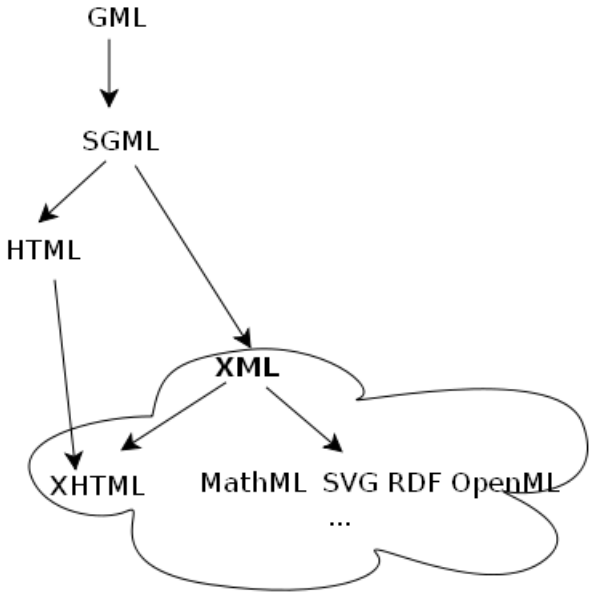


Figura 1.1: Evolución del XML

XML es una tecnología sencilla que tiene a su alrededor tecnologías adyacentes que la complementan (XSLT, CSS, etc.), la hacen mucho más poderosa, con posibilidades mucho mayores en términos de programación, de presentación o de diversas transformaciones. XML juega un papel muy importante en la actualidad ya que permite la compatibilidad entre sistemas para compartir la información de una manera segura, fiable y fácil.

A continuación se presenta un ejemplo de un documento XML:

```
<?xml version="1.0" encoding= "UTF-8"?>
<lista>
 <postit>
  <fecha>2010-06-12</fecha>
  <quien>Pablo</quien>
  <quien>Paola</quien>
  <texto>Escribir
   <bf>una carta</bf> para los abuelos
  </texto>
 </postit>
 <postit>
  <fecha>2010-06-10</fecha>
  <quien>Pablo</quien>
  <quien>Ana</quien>
  <texto>
  Ir al cine y ver
   <bf>La era de hielo</bf>
  </texto>
 </postit>
 <postit>
  <fecha>2010-06-09</fecha>
  <quien>Ana</quien>
  <texto>
  Llamada
   <bf>55 555 666 777</bf>
  </texto>
 </postit>
</lista>
```

Dicho ejemplo lo tomaremos como base para ejemplificar principios del proceso de minería.

Un documento XML se compone de un encabezado y un contenido formado por etiquetas que arman la estructura, partes textuales o numéricas e hipervínculos. En el encabezado del documento XML se especifica el idioma, referencias a descripciones de estructuras, vínculos hacia otros documentos, etc.

El cuerpo de un documento XML se puede representar mediante un árbol de parseo, debido a que existe sólo un elemento que forma la raíz del documento.

En la figura 1.2 , se puede observar el correspondiente árbol de parseo, el cual es una representación equivalente del documento XML, cuya raíz es llamada <`lista`>.

Las ventajas de uso de XML son [Harold04], [XML10]:

- En un documento XML existen etiquetas y contenidos de elementos los cuales son significativos.
- Fácilmente procesable tanto por humanos como por las aplicaciones.

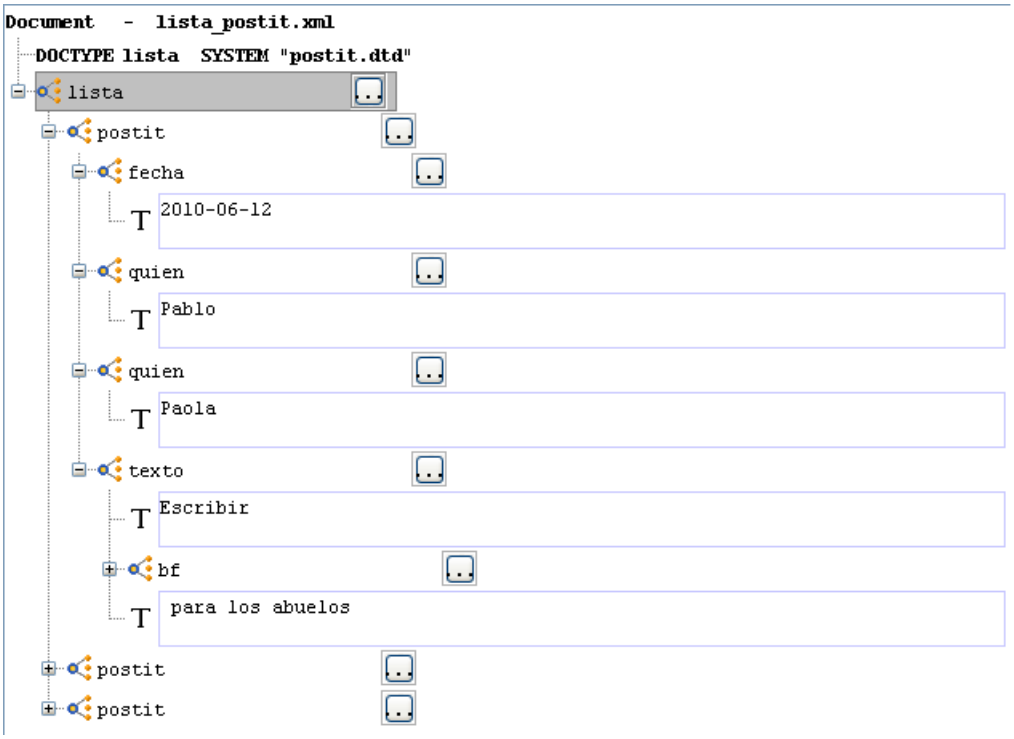


Figura 1.2: Árbol de parseo XPath del documento XML, mostrado previamente

- Separa radicalmente la información o el contenido de su presentación o formato.
- Diseñado para ser utilizado en cualquier lenguaje o alfabeto.
- Su análisis sintáctico es fácil debido a las reglas estrictas que rigen la composición de un documento.
- Capacidad de utilizar espacios de nombres y posibilidad de utilizar descriptores y estructuras definidas antes.
- Estructura jerárquica
- El número de marcas es ilimitado
- Relaciones explícitas entre los recursos o partes de los recursos, a través de enlaces XLink.

Existen diversas herramientas para la consulta del contenido de documentos XML de tipo XPath y XQuery que permiten la búsqueda de información precisa. También se pueden transformar documentos XML y visualizar el contenido. Por ejemplo, XSLT permite seleccionar parte del contenido dentro de un archivo XML, transformarlo a otra estructura como HTML o bien transformarlo a texto plano. Sin embargo no existe una aplicación accesible para procesar y transformar documentos XML a conocimiento sintético de tipo patrones repetitivos u otro tipo de conocimiento extraído por técnicas de minería, como agrupamiento.

1.2. Colecciones XML

En la actualidad existen diferentes proyectos de investigación que ponen a disposición varias colecciones de documentos XML entre las que destacan : Shakespeare en XML, Wratislavia XML Corpus, INEX, etc.

Shakespeare en XML (http://www.ibiblio.org/xml/examples/shakespeare/)

Es un directorio que contiene la colección de las obras de Shakespeare en formato XML, realizadas por Jon Bosak. La colección cuenta con 37 obras, lo que representa la

colección completa de Shakespeare, dicha colección data de 1999. Elliotte Rusty Harold posee los derechos reservados sobre la colección completa.

Wratislavia XML Corpus (http://pskibinski.pl/research/Wratislavia/)

Wratislavia XML fue desarrollado específicamente para probar y comprobar nuevos algoritmos de compresión de XML. La colección contiene 7 archivos que van desde un tamaño de 2 MB hasta 151 MB, en total 471 MB. Los archivos se encuentran seleccionados estratégicamente en base a su capacidad de proporcionar resultados representativos de rendimiento en los algoritmos de compresión.

Colecciones INEX (https://inex.mmci.uni-saarland.de/)

XML es un formato ideal de almacenamiento para diferentes propósitos, colecciones de información de diferentes tipos: textuales, de datos, con vínculos (ligas) sobre otros documentos de la misma colección o fuera, con referencias a imágenes y otros archivos multimedia.

Las colecciones de documentos XML de INEX son versiones de artículos de Wikipedia[3]. Estas colecciones también contienen bolsas de palabras con términos frecuentes que se encuentran en los documentos, frecuencias de varias estructuras para el agrupamiento XML en forma de etiquetas, árboles, ligas y nombres de entidades. La colección de INEX se centra básicamente en documentos XML, dicha colección para el año 2007 poseía dos colecciones divididas una de gran tamaño (2.7 millones de documentos) y una colección de menor tamaño (50,000 documentos). Para la versión del año 2008 posee de igual forma dos colecciones, una de menor tamaño (659,410 documentos) y la de gran tamaño (5 millones de documentos).

Cabe señalar que los documentos XML en la colección INEX se encuentran organizados en carpetas y para este proyecto inicialmente se propuso trabajar con sólo una carpeta de la colección del año 2008. Sin embargo para fines de experimentación el tamaño de la colección es poco relevante, pues la experimentación se puede realizar bajo principios estadísticos [Devore01],[Sheaffer87], por técnicas de muestreo estadístico dichas técnicas permiten trabajar con una muestra que tome documentos de todas las carpetas de la colección.

[3] *http://www.wikipedia.org/*

Finalmente en el mundo de las empresas existen numerosas colecciones de documentos XML, las cuales son generadas por diferentes aplicaciones con el objetivo de almacenar la información; así como para el intercambio entre sistemas de información.

1.3. Aplicaciones e interés de las técnicas de minería

Las técnicas de minería se utilizan para el análisis y el descubrimiento de nueva información en documentos textuales ya sea planos o jerárquicos (XML). Actualmente, la minería se está utilizando cada vez más en soluciones para la inteligencia de negocios y el apoyo a la toma de decisiones. Las aplicaciones comunes de minería incluyen [Ramírez04] :

- Finanzas
  - Análisis de riesgos : Dado un conjunto de clientes actuales, sus finanzas y su historial de seguro, se puede construir un modelo predictivo el cual se puede utilizar para clasificar un nuevo cliente en una categoría de riesgo.
  - Mercado dirigido : Dado un conjunto de clientes actuales y su historial de compras, predecir sus respuestas a las promociones.
  - Retención de clientes : Dado un conjunto de clientes antiguos, su comportamiento antes de salir, predecir quién tiene más probabilidades de salir y tomar medidas proactivas.
  - Detección de fraudes : Al detectar divergencia del patrón de conducta histórico para determinar actividades fraudulentas de forma proactiva.
- Medicina
  - Genética : En el descubrimiento de patrones asociados a enfermedades permite establecer pronósticos en personas sanas. Lo que abre la posibilidad de mejorar los sistemas de prevención de enfermedades.
  - Farmacología : Con el objeto de descubrir propiedades de los fármacos, con base en su estructura. Esto evitaría gastos importantes de investigación para su desarrollo.

– Diagnóstico : El estudio de los expedientes de un hospital permitiría descubrir asociaciones entre las características del paciente, su calidad de vida y diferentes grupos de enfermedades. Esto ayudaría a establecer el diagnósticos sin necesidad de efectuar pruebas de laboratorio o estudios de diagnósticos costosos.

Muchos otros nuevos dominios de aplicación están emergiendo a medida que se exploran nuevas áreas.

Con el fin de conocer más a fondo la metodología aplicada Minería XML, se procedió a realizar un clasifiador de artículos, el cual se publicó en la revista de Administración y Organizaciones en diciembre 2010, (veáse anexo H, página 155).

1.4. Justificación de nuestro trabajo

La investigación encaminada al estudio de minería XML por su naturaleza semiestructurada de los datos y por su tipo de contenido más textual que numérico, es incompatible con las técnicas de almacenamiento y recuperación de la información en las bases de datos (relacionales). El tratamiento y el almacenamiento de la información XML requiere entonces nuevas formas o soportes. Por otra parte, hay una gran dificultad en el uso directo de una colección XML; las tecnologías XML son poderosas pero no son tan simples para el uso de un usuario final.

Es por tal motivo que se ve la necesidad de implementar métodos de minería XML para la agrupación de los documentos, los cuales ayuden a la organización y sistematización de la información de los documentos XML, lo que coadyuvará en la búsqueda y recuperación de información de documentos XML, a usuarios y empresas con la aplicación resultado de este trabajo.

Capítulo 2

Estado del arte

2.1. Minería XML

La minería XML es el área que se encarga de aplicar técnicas para la extracción de conocimiento[1] de documentos XML. Esta área se ve influenciada por diferentes dominios como lo es la minería de datos, recuperación de información, la estadística, reconocimiento de patrones, inteligencia artificial.

La investigación en el campo de la Minería XML ha desarrollado diversas técnicas y herramientas que tratan del minado de frecuencia de datos con patrones nativos de XML, además de minar las frecuencias de patrones dentro de los árboles de parseo. Hoy en día, se hace más necesario el estudio y desarrollo de nuevas técnicas de minería XML ya que XML presenta una ventaja por su estructura, haciendo posible el minado en diferentes colecciones [Tran08b] .

El proceso de extracción de información de XML se puede sintetizar de la siguiente manera:

1. Selección del conjunto de documentos XML, comunmente llamado colección.

2. Preprocesamiento de los documentos XML [Jung-Wong04]:

[1] *Se considera que conocimiento es una forma sintética y a veces no explícita de varias informaciones almacenadas, el conocimiento es frecuentemente usado para el aprendizaje o toma de decisiones [Usama Fayyad96].*

- Descubrimiento de la estructura.
- Identificación de elementos similares.
- Extracción de características comunes.

3. Seleccionar y aplicar la técnica de minería de XML, como la clasificación y el agrupamiento.
4. Extracción de información.
5. Interpretación y evaluación.

2.2. Agrupamiento

Una técnica fundamental de la minería es el agrupamiento de los documentos, el cual se aplica a un conjunto de objetos definidos por un número de características, con el fin de generar subgrupos sin usar ninguna información previa.

El proceso de búsqueda de grupos se realiza sin supervisión, pues en teoría, el usuario no interviene directamente en la generación de grupos (figura 2.1).

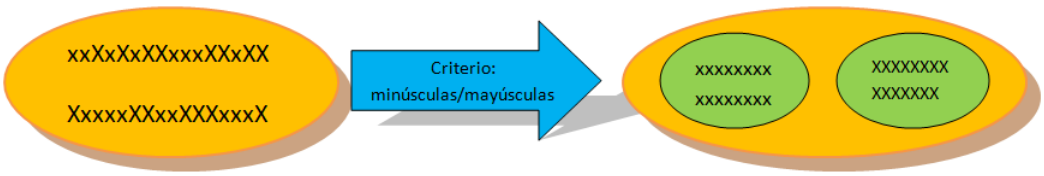


Figura 2.1: Noción de agrupamiento

Cada subgrupo (cluster) comparte propiedades comunes que resultan de la aplicación de funciones de distancias, de similitud y de disimilaridad, esto permite evaluar qué tan similares son los objetos. Existen diversas métricas de distancias cuando las características son numéricas, la más utilizada es la distancia euclidiana; sin embargo cuando las propiedades son nominales se establecen otras funciones, que cuentan la presencia o ausencia de atributos. El detalle de estas métricas se presenta en el Marco teórico y tecnológico en la página 31.

Las distancias entre documentos XML se pueden realizar de tres maneras:

- Distancia entre los términos textuales, esta distancia se realiza a través de una distribución de frecuencia de términos entre los documentos.
- Distancia derivada de la estructura, en esta distancia se consideran unicamente las etiquetas o la distancia entre árboles y los caminos formados para transformar un documento a otro.
- Distancia basada en los vínculos de los documentos, en esta distancia se hace una matriz de ausencia y presencia de los vínculos existentes.

En resumen, un agrupamiento permite obtener un conjunto de subgrupos que se definen entre los elementos o categorías con propiedades similares establecidas después de la aplicación de uno o varios algoritmos, bajo ciertos criterios.

Para poder emplear los métodos de agrupamiento en la información XML se requiere de un tratamiento especial para convertir los elementos de un documento XML (etiquetas, contenido e hipervínculos) a información numérica.

2.3. Trabajos actuales sobre agrupamiento XML

Las investigaciones sobre el agrupamiento de documentos XML son numerosas; sin embargo en la siguiente lista indicamos algunos trabajos que son relevantes :

- **Clustering XML by Structure** [Dalamagas04a]. Presenta un sistema de agrupamiento para documentos XML basado en la estructura, explotando distancias que estiman la similitud entre las estructuras de árboles en términos de la relación jerárquica de los nodos.
- **Clustering XML Documents using Structural Summaries** [Dalamagas04b]. Estudia métodos de agrupamiento para documentos XML por su estructura, explotando distancias en su estructura y resumen estructural. Definen métricas de distancia sobre la estructura para estimar la similitud entre dos documentos XML.

- **A Methodology for Clustering XML Documents by Structure** [Dalamagas06]. Presenta una metodología para la agrupación de documentos XML por su estructura. El agrupamiento estructural se refiere a la tarea de agrupar en conjunto los datos de estructura similar. A diferencia del artículo Clustering XML by Structure, proponen los pasos para realizar el minado de documentos XML.

- **Combining Structure and Content Similarities for XML Document Clustering** [Tran08a]. Introduce una mejora al agrupamiento basado en 2 métricas distintas, las cuales exploran las similaridades en la estructura y el contenido de los documentos XML.

- **XEdge: Clustering Homogeneous and Heterogeneous XML Documents Using Edge Summaries** [Antonellis08]. Propone un algoritmo unificado para la agrupación de documentos heterogéneos y homogéneos XML. Dependiendo del tipo de documento XML, se modifica la métrica de distancia para adaptarla a las características estructurales de los documentos XML homogéneos o heterogéneos.

Por supuesto que existen muchas vías de investigación; que van desde la transformación de texto en forma numérica, pasando por la búsqueda de asociaciones textuales con referencias internas o externas hasta el desarrollo de sistemas de inferencia basados en técnicas de inteligencia artificial.

2.3.1. Diferencias importantes entre el trabajo actual y otros trabajos realizados

En general los trabajos clásicos sólo realizan minería XML analizando etiquetas (las cuales describen la estructura del documento), con el contenido de las etiquetas (CDATA) o bien hipervínculos, sin embargo hasta el momento no se ha realizado ningún trabajo que mine las tres partes escenciales de un documento XML : estructura, hipervínculos y contenido al mismo tiempo.

En el presente trabajo se pretende realizar minería XML analizando las tres partes esenciales de un documento XML : estructura, contenido y presencia/ausencia (sin valores)

de hipervínculos. En lo que respecta al manejo de información se presentan diferentes opciones, por ejemplo generar árboles e índices para localizar el contenido dentro del documento, o extraer la información y representarla en forma de atributos. Se propone representar la información en tablas. En las referencias estudiadas se almacenan las tablas como tales en forma de archivos, en nuestro caso se propone almacenar y administrar las tablas a través de un Sistema de Gestión de Base de Datos. Esta solución de almacenamiento permite aplicar distintos algoritmos de minería de datos a las tablas mencionadas sin necesidad de reprocesar los documentos originales.

En el prototipo, se pueden aplicar varios algoritmos con diferentes parámetros sobre los mismos atributos extraidos de la base de datos. Para una mejor comprensión y más opciones en los algoritmos de agrupamiento se agregó el algoritmo K-Means [Jain10]. En los trabajos realizados en otras investigaciones, en algunos casos se almacenan parte de los documentos o los árboles de parseo completos. Esto además de representar problemas de manejo de información, requiere de recursos computacionales adicionales.

Otro punto a mencionar es la existencia de una interfaz gráfica de usuario, la cual permite manejar y configurar la aplicación facilmente.

Otra novedad del presente trabajo es también la integración de dos algoritmos de construcción de grupos, uno apoyado en distintas métricas y estrategias (algoritmo Jerárquico) y otro apoyado en mezclas de funciones de probabilidad en particular del modelo Gaussiano (Máxima expectación). Además, el programa de esta aplicación permitirá la integración de otros algoritmos de agrupamiento y distintas métricas para continuar nuestra investigación en el futuro.

Capítulo 3

Objetivos

3.1. Definición del problema

Actualmente los documentos XML han llegado a ser utilizados ampliamente como soporte en disciplinas diferentes , debido a la riqueza de su formato y a la posibilidad de explotarlo con diferentes tecnologías. Esta flexibilidad dio lugar a un incremento significativo en las colecciones XML como fuentes de información, tal es el caso de INEX la cual pone a disposición colecciones de documentos XML. Es por tal motivo que se plantean métodos de minería XML para la obtención de agrupaciones de documentos que permitan la recuperación de manera eficiente de la información.

3.2. Delimitación del problema

Se realizará una aplicación configurable (esta característica se define en la sección 3.2.2) que implemente el proceso de extracción de información (estructura, contenido e hipervínculos) sobre la colección de documentos XML INEX, que permita pasar de información textual a información numérica. Una vez que se haya obtenido la información numérica en forma de tablas de frecuencia o de presencia/ausencia de atributos, se procederá a la aplicación de métodos o algoritmos de agrupamiento. Las métricas serán función de las características de los datos y de las pruebas de validación.

Los dos métodos elegidos para realizar las agrupaciones de documentos son :

Jerárquico y Máxima Expectación (EM).

Una cualidad importante de esta aplicación es la flexibilidad para poder configurar en base a : elegir colecciones, considerar diferentes atributos de los documentos, posibilidad de tomar toda la colección o una muestra de ésta, aplicar diferentes tipos de normalización, diferentes métricas para la distancia entre los individuos, diferentes algoritmos de agrupamiento, así como poder guardar el estado de la aplicación. En la figura 3.1 se puede apreciar los puntos donde la aplicación es configurable.

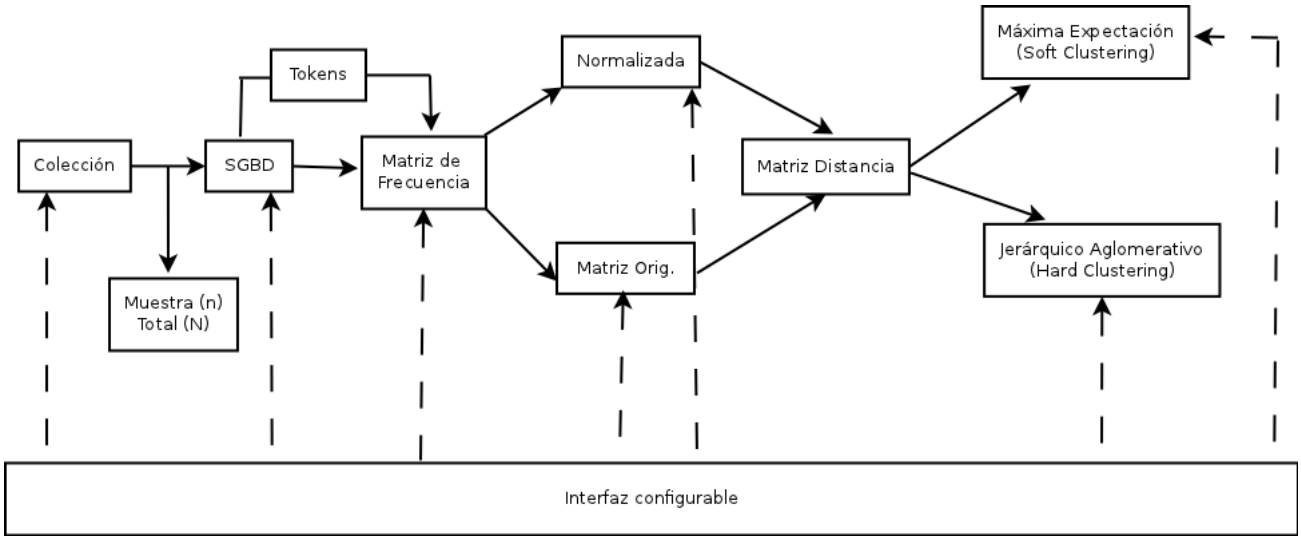


Figura 3.1: Configurabilidad de la aplicación

Por configurable se entiende, la posibilidad de elegir diferentes opciones de manejo y extracción de datos a lo largo del proceso, por ejemplo : elegir una colección diferente, elegir una muestra de documentos, elegir ciertos atributos, aplicar un método específico de normalización, elegir entre diversas métricas o algoritmos de agrupamiento.

De una colección de documentos XML pueden obtenerse valores numéricos en un arreglo para luego aplicar tratamiento o cálculos específicos.

Se pretende que esta aplicación funcione en lotes en forma “asíncrona”, es decir, se realiza en varias etapas :

- Obtención de la colección de documentos XML y almacenamiento local.

- Procesamiento de la totalidad de la colección o de una muestra representativa de la colección.
- Almacenamiento de atributos en el SGBD.
- Generación de una o varias tablas de frecuencias, dando al usuario la libertad de elegir en forma exclusiva o mezclada atributos de contenido, etiquetas o hipervínculos.
- Aplicación de diferentes formas de normalización y distintos algoritmos de agrupamiento con diferentes métricas.

Por supuesto que este proceso no se realiza en tiempo real y los documentos no son procesados en forma continua "on the fly", es decir, en sincronía con la llegada de nuevos documentos a la colección.

3.3. Objetivos

Objetivo General

Diseñar e implementar una aplicación configurable para la extracción de información de colecciones de documentos en formato XML. Esto con el fin de establecer agrupamientos basados en los documentos.

Objetivos Específicos

- Diseñar una aplicación para transformar información descrita en documentos XML que pueda ser procesada por al menos dos algoritmos de agrupamiento.
- Se proponen los siguientes módulos:
  - Módulo de preprocesamiento y análisis de documentos XML. Ver la figura A.1, la cual muestra el caso de uso.
  - Módulo de generación de matrices de frecuencias y normalización. En la generación de frecuencias se tomarán en cuenta el contenido, la estructura, así como también los vínculos dentro de los documentos y las referencias a otros contenidos múltimedia, a partir de un SGBD. Ver la figura A.2, la cual muestra el caso de uso.

  - Módulo de cálculo de distancias. Ver la figura A.3, la cual muestra el caso de uso.
  - Definición del módulo para el uso de los algoritmos de agrupamiento. Ver figura A.4, la cual muestra el caso de uso.
  - Módulo de Interfaz Gráfica de Usuario (GUI) para manejar y configurar la aplicación.

- Búsqueda de criterios de eficacia [Vázquez07]. Ver la figura A.5, en donde se muestra el diagrama UML.

- En base a los resultados de evaluación se propongan mejoras para el proceso.

Capítulo 4

Marco teórico y tecnológico

En este capítulo se presentan detalles de los conceptos y tecnologías involucradas en esta tesis.

4.1. Minería XML

Con el fin de entender las distintas fases del proceso de minería XML y revelar los fundamentos teóricos del proceso, parece importante desarrollar un diagrama conceptual. Esto no quiere decir que en la fase de desarrollo de la aplicación sea necesario seguir exactamente el proceso, ya que por razones prácticas o por falta de algunos conceptos, el diagrama conceptual quizá deba ser modificado en la parte de la implementación.

En la figura 4.1 se pueden apreciar las distintas etapas del proceso de Minería XML.

El pre-proceso de los documentos XML se realiza para identificar la estructura y el contenido, así como realizar una limpieza en el documento. Por ejemplo, la estructura se revela a través del parseo del documento. Esta operación genera un árbol que corresponde a la disposición de elementos de la estructura del documento.

En este caso, la minería XML consiste en obtener matrices de frecuencias tokens (contenido, estructura e hipervínculos), aplicar diferentes funciones de distancia, generar matrices de similitud y aplicar diversos algoritmos de agrupamiento.

La evaluación de resultados consiste en evaluar la calidad de grupos. Por ejemplo

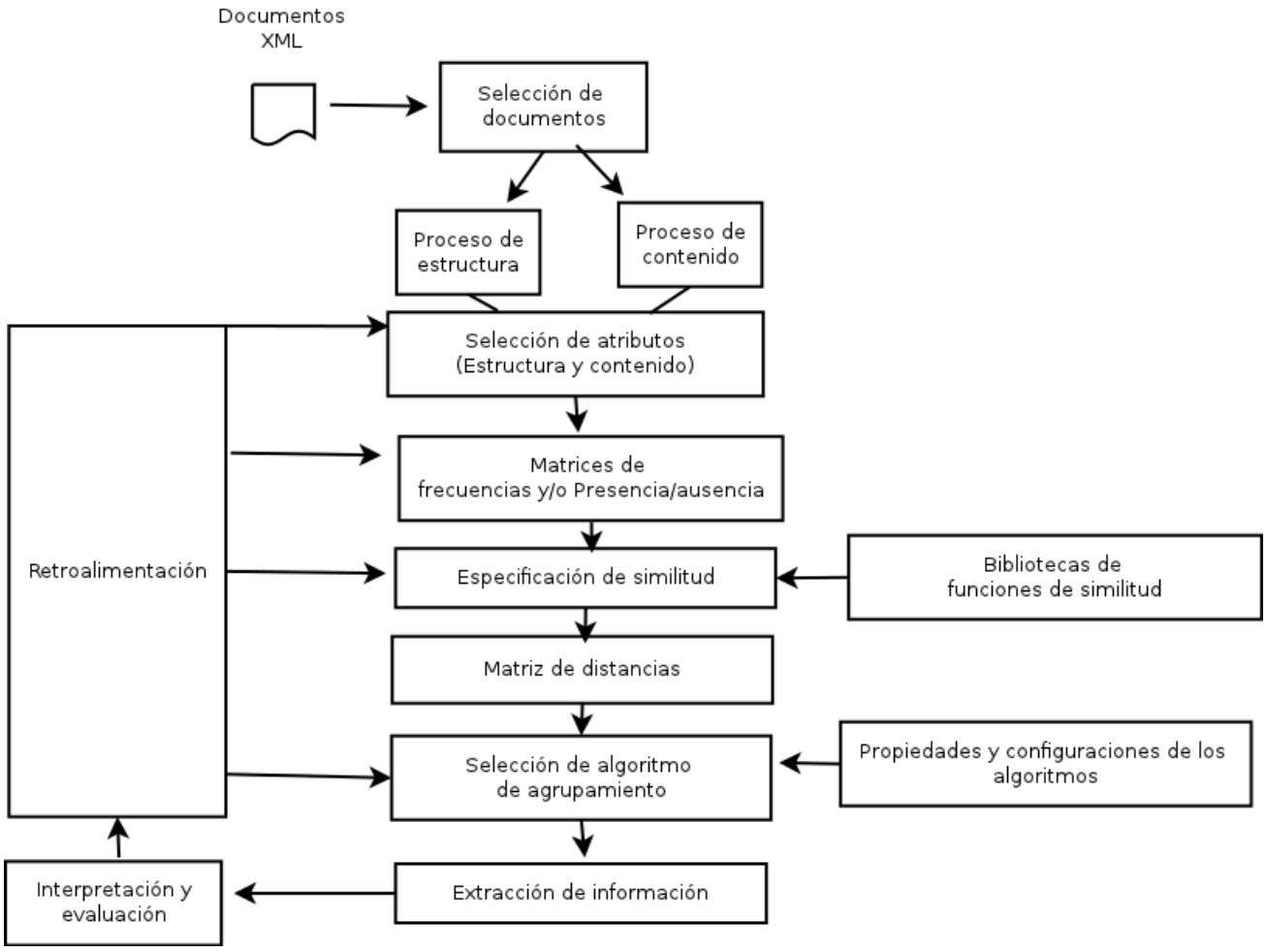


Figura 4.1: Diagrama conceptual de la Minería XML

distancias intragrupos pequeñas, buena definición de grupos y una distancia intragrupos.

La retroalimentación se presenta en cada fase de la minería XML, pues en cada momento se verifican procesos y se realizan los ajustes necesarios.

4.2. Parseo

Para poder acceder a la información almacenada por archivos XML, el World Wide Web Consortium (W3C) ha especificado varios mecanismos para acceder a documentos XML y trabajar con ellos. Se trata de normas que indican a los desarrolladores la manera de acceder a los documentos. Estas normas incluyen una jerarquía de objetos que tienen métodos y atributos con los que tendremos que trabajar y que nos simplificarán las tareas relativas al recorrido y acceso a las partes del documento.

Los mecanismos con mayor importancia se encuentran implementados en SAX y DOM. SAX se utiliza para hacer un recorrido secuencial de los elementos del documento XLM y DOM por su parte genera un árbol en memoria que contiene todo el documento XML, y al que se hace cualquier tipo de recorrido y acciones con los elementos que se deseen.

La API Simple para XML (SAX, por sus siglas en inglés) y el Modelo de Objetos de Documentos (DOM, por sus siglas en inglés), son especificaciones de interfaz, existen varias implementaciones de DOM y SAX como libxml [1], la cual es un conjunto de bibliotecas OpenSource para la manipulación de XML. La Interfaz de Programación de Aplicaciones (API, por sus siglas en inglés) se escribe regularmente en lenguajes de alto nivel como Java, C, Perl, etc.

Los parseadores se utilizan para obtener los tokens : etiquetas y contenido de tipo CDATA.

4.2.1. SAX

SAX [SAX10] procesa la información por eventos, también procesa la información en XML conforme ésta sea presentada (evento por evento), manipulando cada elemento a un determinado tiempo, sin incurrir en uso excesivo de memoria. Las características más

[1] *http://xmlsoft.org/*

notables de SAX son :

- Es un "parser" ideal para manipular archivos de gran tamaño, ya que no crea un un árbol en memoria principal, como lo realiza DOM.
- La especificación más reciente de SAX es 2.0, y al igual que DOM 2.0, ésta se incluye en casi todos los "Parsers" disponibles en el mercado.

4.2.2. DOM

DOM [DOM10] es una interfaz de programación de aplicaciones propuesta por la World Wide Web que proporciona un conjunto estándar de objetos para representar documentos HTML y XML, es un modelo sobre cómo pueden combinarse dichos objetos y una interfaz estándar para acceder a ellos y manipularlos. A través del DOM, los programas pueden accederse y modificarse, es decir agregar o eliminar el contenido y/o estructura; así como también pueden modificar la estructura de los documentos HTML y XML.

DOM define la manera en que objetos y elementos se relacionan entre sí en el navegador y en el documento. Cualquier lenguaje de programación adecuado para el diseño web puede utilizar los métodos y objetos de DOM, cada objeto tiene un nombre, el cual es exclusivo y único. Cuando existe más de un objeto del mismo tipo en un documento web, estos se organizan en un vector.

Sin embargo, el parseo no es suficiente para el análisis de los documentos XML, ya que sólo realizan la extracción de partes textuales que se encuentran como cadenas de carácteres entre dos etiquetas y no a nivel de palabra. Para esto se requiere realizar un proceso adicional con el software de TreeTagger, el cual se usa para obtener los tokens de tipo palabra en una forma invariable y no ambigua.

4.3. TreeTagger

TreeTagger es una herramienta de análisis léxico que determina la categoría gramatical de las palabras así como la forma invariante de dichas palabras. TreeTagger fue desarrollado por Helmut Schmid en el proyecto TC Poject [2] en el Instituto para la Compu-

[2] *http://www.ims.uni-stuttgart.de/projekte/tc/*

tación Lingüística en la Universidad de Stuttgart [3]. TreeTagger ha sido utilizado con éxito en idiomas como : alemán, inglés, francés, italiano, español, búlgaro, ruso, griego, chino, entre otros.

A continuación se presenta un ejemplo de TreeTagger :

"The children play in the park."

Palabra Tipo Forma Inv.
The DT the
children NNS child
play VBZ play
in IN in
the DT the
park NN park

Para la descomposición anterior se tiene que la primera palabra es la cadena de texto que lee del enunciado, la segunda palabra es la abreviatura del tipo gramatical al que pertenece y la tercera palabra índica la forma invariante.
Por ejemplo :

children, es la palabra que lee.
NNS, es la abreviatura del tipo gramatical *"Noun, plural"* o sustantivo, plural.
child, es la forma invariante de la palabra children.
De esta manera dicha herramienta permite obtener de manera rápida la forma invariable de los tokens de contenido.
Cabe destacar que el uso de TreeTagger como paquete (TT4J) en Java se ve limitado, ya que los paquetes escritos poseen ciertos errores, los cuales no permiten invocar los métodos necesarios para que esté inmerso dentro del mismo programa.
A la combinación del proceso de parsear y del análisis léxico surge el proceso llamado tokenización.

[3] *http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/*

4.4. Tokenización

Si en el dominio de minería de datos se considera como elemento básico de análisis a los datos numéricos, en la minería de texto es la palabra, en el campo de XML, el elemento básico es el token.

- Etiquetas abiertas (las etiquetas cerradas forman parejas).
- Nodos de tipo especial (hipervínculos, vínculos o referencia a imágenes).
- Palabras que componen un nodo de tipo CDATA.

A partir de la información de contenido, etiquetas e hipervínculos se pueden construir matrices de frecuencia/ausencia, frecuencias absolutas por nivel o por camino, normalizadas. El tamaño de esas matrices : si hay M documentos y N tipos distintos $M \times N$ o $M \times K_1 \times N$, $M \times K_2 \times N$ con K_1 y K_2, dos factores - K_1 es la profundidad del árbol de un documento y K_2 es el número de caminos posibles.
Antes de elegir la construcción de los diferentes tipos de matrices, es necesario conocer la información original para observar como se comportan los datos.
Para ejemplificar lo anterior se presenta la figura 1.2, donde se representan documentos tipo block de mensajes ("*postit*"). En ellos se puede identificar la etiqueta raíz (lista), la etiqueta post it, los nodos, y la profundidad de cada etiqueta. Una vez que se conoce la estructura se estudia el contenido, identificando o removiendo objetos con características similares. A lo largo del proceso todos o un subconjunto de atributos es seleccionado y se cuentan para obtener la frecuencia o sólo establecer su presencia o ausencia. El ejemplo *postit* :

```
<?xml version="1.0" encoding="UTF-8"?>
<lista>
 <postit>
  <fecha>2010-06-12</fecha>
  <quien>Pablo</quien>
  <quien>Paola</quien>
  <texto>Escribir
   <bf>una carta</bf> para los abuelos
  </texto>
 </postit>
 <postit>
  <fecha>2010-06-10</fecha>
  <quien>Pablo</quien>
  <quien>Ana</quien>
  <texto>
```

```
  Ir al cine y ver
   <bf>La era de hielo</bf>
  </texto>
 </postit>
 <postit>
  <fecha>2010-06-09</fecha>
  <quien>Ana</quien>
  <texto>
  Llamada
   <bf>55 555 666 777</bf>
  </texto>
 </postit>
</lista>
```

El contenido, presentado sobre el postit tiene los siguientes elementos: "...","2010-06-12", "Pablo", "Paola", "escribir y para los abuelos". Para el propósito de minería el último elemento se descompone en los tokens siguientes: "escribir","y","para", "los" y "abuelos". Estos datos pueden integrarse en una o varias matrices. Por ejemplo Tien [Tran08a] propone una matriz para atributos sobre la estructura (tokens, marcadores, camino donde se encuentra el token, profundidad u otro atributo relacionado con la estructura) y otra matriz para registrar cuentas de tokens relacionados con el contenido (compuesto principalmente de texto). Si se sigue esta propuesta se obtendrían las siguientes matrices:

- Matriz de frecuencia de la estructura se presenta en la tabla 4.1, en la primera columna se encuentra la Ruta, la cual representa el camino de las etiquetas contenidas en los documentos. A partir de la segunda hasta la cuarta columna representan el acumulado de la etiqueta en cuestión del documento " *lista_postit.xml*" hasta "*lista_postit_n.xml*".

Tabla 4.1: Matriz de frecuencia de la estructura

Ruta	lista_postit.xml	lista_postit_1.xml		lista_postit_n.xml
lista/post it/fecha	1	1	...	1
lista/post it/quien	1	1	...	0
lista/post it/quien	1	1	...	0
lista/post it/texto	1	1	...	1
lista/post it/texto/bf	1	0	...	0

- Matriz de frecuencia del contenido que se muestra en la tabla 4.2.

Tabla 4.2: Matriz de frecuencia de palabras

término	lista_postit.xml	lista_postit_1.xml		lista_postit_n.xml
2010-06-12	1	1	...	1
pablo	1	1	...	1
paola	1	1	...	0
Escribir	1	2	...	1
a los abuelos	1	1	...	0

Las matrices presentadas en las tablas 4.1 y 4.2, también se pueden integrar en una sóla matriz.

4.5. Normalización

La normalización es el proceso de ajustar los valores de los atributos en el interior de un intervalo de forma reducida hacia un modelo establecido. La normalización permite mejorar la eficiencia de los algoritmos de minería. Existen diversas normalizaciones; sin embargo para este proyecto se aplican las siguientes :

- **Normalización Z**: Al proceso de transformación de un polígono de frecuencias a una curva normal [Larose07] [Jajuga00], se llama normalización Z y para ello se hace un cambio de escala mediante la normalización o tipificación de las puntuaciones, es decir, los valores del campo (X) se transforman en valores Z mediante la siguiente ecuación de transformación :

$$X^* = \frac{X - \overline{X}}{S_X} \tag{4.1}$$

- **Normalización mínima-máxima**: La normalización mínima-máxima funciona identificando el valor mínimo de un campo; así como el valor máximo. La normalización se produce por la diferencia del valor actual del campo menos el valor mínimo entre la

diferencia del valor máximo menos el valor mínimo [Larose07] [Jajuga00]. La fórmula se define como :

$$X^* = \frac{X - min(X)}{max(X) - min(X)} \tag{4.2}$$

De la normalización mínima-máxima se obtiene que los valores mínimos X^* de la variable tienen un valor en cero.

- **Normalización máxima**: Funciona identificando el valor máximo de un campo (atributo), para poder realizar el cociente del valor actual entre el valor máximo [Jajuga00]. La fórmula se define como :

$$X^* = \frac{X}{max(X)} \tag{4.3}$$

Aúnque el proyecto presenta una serie de normalizaciones para los datos de la matriz de frecuencias, se trabajó con las frecuencias originales.

4.5.1. Medidas de distancia, similitud o disimilaridad

Antes del agrupamiento, es necesario aplicar ciertas medidas o métricas de similitud, las cuales reflejen el grado de cercanía entre dos objetos dentro de un conjunto de objetos. Estas medidas se aplican de acuerdo a las características evaluadas del problema, no existen métricas universales que se puedan utilizar para todos los problemas, es por ello que es pertinente elegir una medida o métrica de similitud de acuerdo al algoritmo de agrupamiento que se vaya a aplicar; así como al tipo de atributos que se analiza (incluso al origen y sentido de la información minada) [Roux85].

- **Manhattan :** La distancia Manhattan también conocida como la distancia de la ciudad de bloque o como la longitud de Manhattan, se define como la distancia entre dos puntos medidos a lo largo de los ejes del ángulo recto. La distancia Manhattan se define como :

$$dManhattan(\vec{x}, \vec{y}) = \sum_{i=1}^{n} |x_i - y_i| \tag{4.4}$$

donde se puede suponer que x y y son dos vectores correspondientes a cada documento. $x = (x_1, x_2, x_3, ..., x_n)$ y $y = (y_1, y_2, y_3, ..., y_n)$

- **Euclidiana :** La distancia Euclideana es una métrica estándar para problemas geométricos. Esta es una distancia ordinaria entre puntos y puede fácilmente medirse en tres o más dimensiones del espacio. La distancia Euclideana es ampliamente utilizada en problemas de agrupamiento de texto. Es por defecto la medida de distancia en el algoritmo K-Medias. La distancia Euclidiana entre dos documentos se define como :

$$dEuclidiana(\vec{x}, \vec{y}) = \sqrt[2]{\sum_{i=1}^{n} (x_i - y_j)^2} \tag{4.5}$$

- **Minkowski :** Es una métrica en el espacio euclideano que se puede considerar como una generalización tanto de la distancia Euclideana y la distancia de Manhattan. La distancia de Minkowski se define como :

$$dMinkowski(\vec{x}, \vec{y}) = (\sum_{i=1}^{n} |x_i - y_i|^P)^{\frac{1}{P}} \tag{4.6}$$

donde P es el número de propiedades a evaluar.

Las distancias Euclideana, Manhattan y Minkowski se aplican por lo general a variables de tipo continuo, intervalo o de razón.

- **Binaria :** Si la distancia entre dos documentos es mayor al umbral, entonces son diferentes, si la distancia es menor que el umbral, se consideran iguales. Por lo que la métrica binaria se puede definir como :

$$\delta(\vec{x}, \vec{y}) = \begin{cases} 0 & \text{si} \quad d(\vec{x}, \vec{y}) \leq h \\ 1 & \text{si} \quad d(\vec{x}, \vec{y}) > h \end{cases} \tag{4.7}$$

donde h es el umbral.

- **Máxima :** Esta medida de similitud tiende a separar a los individuos en mayor medida que la indicada por sus disimilaridades iniciales.

- **Coseno :** Esta medida de similitud es utilizada cuando los documentos son representados como vectores, la similitud de los dos documentos corresponde a la correlación entre los vectores. Esto se cuantifica como el coseno del ángulo de dichos vectores. La medida de similitud coseno es una de las más utilizadas en los documentos de texto, como ejemplo existen aplicaciones de recuperación de información y agrupamiento [Huang08]. Su expresión es la siguiente, siendo n el número de propiedades; y representando x_i y y_i el valor del atributo i para el caso X y para el caso Y :

$$dCoseno(\vec{x}, \vec{y}) = \frac{\sum_{i=1}^{F} (x_i \cdot y_i)}{\sqrt[2]{\sum_{i=1}^{F} x_i^2 \sum_{i=1}^{n} y_i^2}} \tag{4.8}$$

4.5.2. Índices de similitud

Los índices de similitud se aplican para elementos con atributos binarios. A continuación se mencionan algunos de los índices más importantes :

- **Coeficiente de Jaccard :** También conocido como el coeficiente de Tanimoto. Para documentos de texto, se compara el peso total de los términos comunes en los documentos p y q, los términos propios en el documento p y finalmente los términos propios en el documento q. La definición formal es [Huang08]:

$$CJaccard(\vec{x}, \vec{y}) = \frac{c}{p + q - c} \tag{4.9}$$

  El coeficiente de similitud de Jaccard es una medida de similitud cuyos rangos oscilan entre 0 y 1, donde 1 significa que dos objetos son iguales y 0 que dos objetos son completamente diferentes [Rodríguez01].

En el caso de presencia o ausencia de atributos, se puede dar mayor importancia a las presencias o a las ausencias o bien la misma importancia. Cuando las presencias o ausencias tienen diferente importancia es común utilizar diferentes índices; sin embargo, cuando es más importante la presencia se utilizan los índices que se muestran en la tabla 4.3:

Tabla 4.3: Índices de similitud dónde la presencia de atributos es más importante

Jaccard	$\frac{c}{(p+q-c)}$
Dice-Sorensen	$\frac{2c}{(p+q)}$
Sokal & Sneath-2	$\frac{c}{2(p+q)-3c}$
Kulczinski-1	$\frac{c}{(p+q-2c)}$
Kulczinski-2	$\frac{(\frac{c}{p}+\frac{c}{q})}{2}$
Ochiai	$\frac{c}{\sqrt{p*q}}$
Simpson	$\frac{c}{Min(p,q)}$
Kochen & Wong	$\frac{nc}{pq}$

Cabe mencionar que para los índices de similitud mencionados anteriormente, se considera como atributos únicamente los atributos significativos, es decir aquellos que presenten 1.
Si la presencia o ausencia presentan la misma importancia, se consideran las medidas presentadas en la tabla 4.4 :

Notación

p = número de atributos del primer individuo;

q = número de atributos del segundo individuo;

c = número de atributos compartidos entre los dos individuos;

d = número de atributos ausentes en ambos individuos.

n = número total de atributos.

Los índices de disimilaridad y similitud descritos se aplican para datos de tipo binario, presencia ausencia y de frecuencias. Estos generalmente se relacionan con información de tipo nominal u ordinal.

4.6. Algoritmos de agrupamiento

Los algoritmos de agrupamiento en general permiten clasificar un conjunto de elementos en un determinado número de grupos basándose en las semejanzas y diferencias existentes

Tabla 4.4: Índices de similitud dónde la presencia y ausencia de atributos son igual de importantes

Sokal & Michener	$\frac{(c+d)}{n}$
Sokal & Sneath-1	$\frac{2(c+d)}{n+c+d}$
Rogers & Tanimoto	$\frac{(c+d)}{2n-(c+d)}$
Sokal & Sneath-3	$\frac{(c+d)}{(p+q-2c)}$
Sokal & Sneath-4	$\left\{\begin{array}{c} S_1 = \frac{c}{p} + \frac{c}{q} \\ S_1 = \frac{d}{(n-p)} + \frac{d}{(n-q)} \\ S = \frac{S_1+S_2}{4} \end{array}\right\}$
Sokal & Sneath-4	$\frac{cd}{\sqrt{(pq(n-p)(n-q))}}$
Roux-1	$\left\{\begin{array}{c} D_1 = Min(p,q) \\ D_2 = Min(n-p, n-q) \\ s = \frac{(c+d)}{D_1+D_2} \end{array}\right\}$
Roux-2	$\frac{ncd}{(pq(n-p)*(n-q))}$
Hamann	$((c+d)-(p-c)-(q-c))$
Yule	$\left\{\begin{array}{c} N = cd-(p-c)(q-c) \\ D = \sqrt{pq(n-p)(n-q)} \\ s = \frac{N}{D} \end{array}\right\}$
Phi de Pearson	$\left\{\begin{array}{c} N = cd-(p-c)(q-c) \\ D = \sqrt{pq(n-p)(n-q)} \\ s = \frac{N}{D} \end{array}\right\}$

entre los componentes de la muestra [Jain10].
Una gran variedad de algoritmos de agrupamiento han surgido en los últimos años los cuales se pueden clasificar en [Garre07]:

- Métodos de Particionado y Recolocación [Kutty08], en la figura 4.2 se muestra la noción de dichos métodos.
  - Algoritmos basados en la densidad
    - Agrupamiento de conectividad basada en densidad (Density-Based Connectivity Clustering)
    - Agrupamiento basado en Funciones de Densidad
  - Métodos de los k-vecinos
  - Métodos de las k-medias [Jain10]
  - Agrupamiento probabilístico [Garre07]
- Métodos Jerárquicos [Tran08b], en la figura 4.3 se muestra la noción de dichos métodos.
  - Algoritmos aglomerativos
  - Algoritmos divisivos
- Métodos basados en rejillas
- Métodos basados en la co-ocurrencia de datos categóricos
- Agrupamiento basado en restricciones
- Agrupamiento subespacial
- Técnicas de co-agrupamiento

Desde el punto de vista del resultado del agrupamiento; existen algoritmos de tipo soft clustering (agrupamiento suave) o hard clustering (agrupamiento duro). En el agrupamiento suave un elemento puede pertenecer a uno o varios grupos según una probabilidad de pertenencia, mientras que en el hard clustering un elemento pertenece a uno y sólo un grupo.

4.6.1. Método de las K-Medias (K-Means)

El método K-Means consiste en dividir una base de datos dada en K grupos (K es fijo) [4] [Jain10].

La idea principal es definir K centroides de manera aleatoria (uno para cada grupo), generalmente se extraen K elementos desde el conjunto inicial y luego tomar cada punto de la base de datos y situarlo en la clase de su centroide más cercano. El próximo paso es recalcular el centroide de cada grupo y volver a distribuir todos los objetos según el centroide más cercano. El proceso se repite hasta ya no realizar cambios en los grupos de un paso al siguiente [Wu09].

El problema del empleo de estos esquemas es que fallan cuando los puntos de un grupo están muy cerca del centroide de otro grupo, también cuando los grupos poseen diferentes tamaños y formas.

En la figura 4.2, se muestra la noción del algoritmo K-Medias.

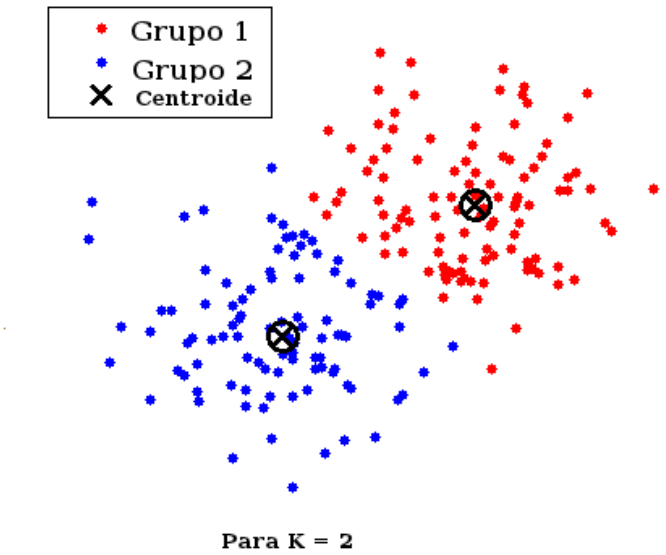


Figura 4.2: Agrupamiento por particiones

[4] *El valor de K es elegido por el experto del dominio o bien se toman varios valores lo bastante pequeños según los datos que se estén trabajando.*

4.6.2. Método jerárquico

Los métodos de agrupamiento jerárquico (divisivo y aglomerativo), se consideran de tipo hard, ya que un documento puede pertenecer a uno y sólo un grupo es decir : Hard Clustering : Agrupamiento con métodos jerárquicos con funciones de distancia para el cálculo de la matriz de distancias entre los documentos. El algoritmo es el siguiente:

1. Existen N elementos a agrupar.

2. Se buscan los 2 elementos más próximos, los cuales serán agrupados para generar un nuevo subgrupo.

3. Se calculan las distancias entre el nuevo elemento p subgrupos y los elementos restantes.

4. Se buscan de nuevo los 2 elementos más próximos, generando un nuevo subgrupo, se calculan las nuevas distancias entre el nuevo grupo y los otros, se itera el algoritmo hasta que se quede un número pequeño de subgrupos.

En los algoritmos jerárquicos los grupos se visualizan en una estructura de árbol llamada dendograma, donde los grupos son creados de manera recursiva (métodos divisivos) o combinacionales (aglomerativos) [Larose07]. El algoritmo anterior es aglomerativo.
En la figura 4.3, se presenta el funcionamiento de dicho algoritmo.
El criterio básico para cualquier agrupación es la distancia. Existen diversas estrategias de selección de éstas como son:

- Simple : Está basada en la distancia mínima entre los elementos del grupo A y el grupo B. En otras palabras; la similitud del grupo está basada en la menor distancia entre los elementos de cada grupo [Larose07].

- Completa : Está basada en la distancia máxima entre los elementos del grupo A y grupo B. De otra manera la semejanza de los miembros está basado en los miembros más distintos de cada grupo [Larose07].

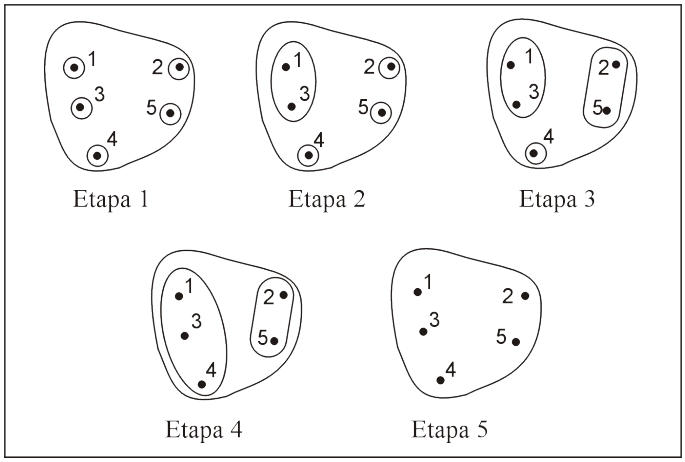


Figura 4.3: Agrupamiento jerárquico aglomerativo

- Promedio : Esta distancia reduce la dependencia de los valores extremos de distancias, el criterio de distancia promedio de todos los elementos del grupo A y del grupo B [Larose07].

- Centroide : Está basada en la distancia centroide entre grupos. Este criterio de agrupamiento presenta el inconveniente de dejarse influir excesivamente por los grupos de mayor tamaño [Larose07].

- Ward : Ward [Lebart84] propuso que la pérdida de información que se produce al integrar los distintos individuos en grupos puede medirse a través de la suma total de los cuadrados de las desviaciones entre cada punto (individuo) y la media del cluster en el que se integra. Para que el proceso de agrupamiento resulte óptimo, en el sentido de que los grupos formados no distorsionen los datos originales, proponía la siguiente estrategia: En cada paso del análisis, considerar la posibilidad de la unión de cada par de grupos y optar por la fusión de aquellos dos grupos que menos incrementen la suma de los cuadrados de las desviaciones al unirse.

  La estrategia de fusión Ward es una de las más utilizadas en la práctica; posee casi todas las ventajas del método de la media y suele ser más discriminativo en la determinación de los niveles de agrupación. Una investigación llevada a cabo por Kuiper

y Fisher [Gutiérrez94] probó que esta estrategia era capaz de acertar mejor con la clasificación óptima que otros métodos (mínimo, máximo, media y centroide).

Cuando todas las propiedades son binarias, se le puede dar una interpretación no geométrica, tomando toda la sumatoria del denominador como el número de atributos comunes entre las dos distancias y el denominador como la media geométrica del número de atributos que poseen ambas instancias.

En la figura 4.4, se pueden observar la noción de los distintos criterios de distancia mencionados anteriormente.

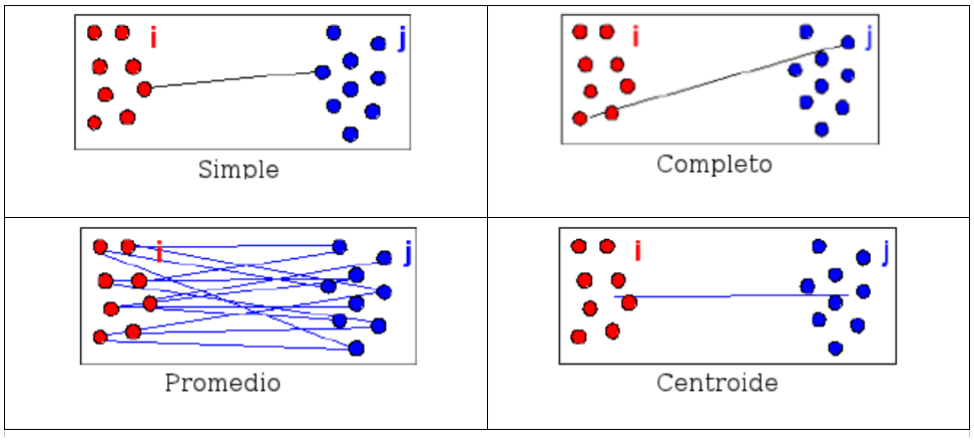


Figura 4.4: Criterios de agrupamiento.

Para un conjunto de datos, los grupos que se construyen dependen de la especificación de ciertos parámetros:

- El método de agrupamiento define las reglas para la formación de grupos. Es decir, se puede usar la distancia mínima, promedio o máxima entre objetos.
- La medida define la fórmula para el cálculo de distancia.
- La estandarización permite igualar el efecto de las variables medidas sobre diferentes escalas.

Para el ejemplo de *postit* se considera cada elemento <*postit*> como un documento y aplicamos el método jerárquico sobre la coleccion formada por 5 documentos. En la tabla 4.5, se muestra la matriz inicial.

Tabla 4.5: Matriz de distancias inicial

	post_it_1	post_it_2	post_it_3	post_it_4	post_it_5	post_it_6
post-it_1	0	4	36	81	196	225
post_it_2	4	0	16	49	144	169
post_it_3	36	16	0	9	64	81
post_it_4	81	49	9	0	25	36
post_it_5	196	144	64	25	0	1
post_it_6	225	169	81	36	1	0

Partiendo de la matriz anterior se agruparían el post_it_4 y post_it_5 en la tabla 4.6 se presenta la forma de agruparlos.

Tabla 4.6: Matriz resultante de agrupar el post_it_4 y post_it_5

	post_it_1	post_it_2	post_it_3	post_it_4	post_it_5 y post_it_6
post-it_1	0	4	36	81	196
post_it_2	4	0	16	49	144
post_it_3	36	16	0	9	64
post_it_4	81	49	9	0	25
post_it_5 y post_it_6	196	144	64	25	0

Partiendo de la matriz anterior se agruparían el post_it_1 y post_it_2 en la tabla 4.7 se presenta la forma de agruparlos.

Como paso final se procede a agrupar el post_it_3 y post_it_4, en la tabla 4.8 se presenta la matriz final. El algoritmo termina ya que cuenta con sólo dos grupos, que a su vez puede decirse que forman parte de un único grupo.

Tabla 4.7: Matriz resultante de agrupar el post_it_1 y post_it_2

	post_it_1 y post_it_2	post_it_3	post_it_4	post_it_5 y post_it_6
post_it_1 y post_it_2	0	16	49	144
post_it_2	16	0	9	64
post_it_3	49	9	0	25
post_it_5 y post_it_6	144	64	25	0

Tabla 4.8: Matriz resultante de agrupar el post_it_3 y post_it_4

	post_it_1 y post_it_2	post_it_3 y post_it_4	post_it_5 y post_it_6
post_it_1 y post_it_2_1	0	16	144
post_it_3 y post_it_4	16	0	25
post_it_5 y post_it_6	144	25	0

Una vez elegida la métrica se obtienen las matrices de similitud o disimilaridad. Como por ejemplo la tabla 4.9 es calculada con la métrica de distancia Euclidiana :

Tabla 4.9: Matriz de similitud

	post_it_1	post_it_2	post_it_3	post_it_4
post-it_1	0	64	81	100
post_it_2	64	0	1	4
post_it_3	81	1	0	1
post_it_4	100	4	1	0

A partir de esta matriz se pueden generar los agrupamientos o conglomerados mediante la aplicación de algoritmos de agrupamiento.

4.6.3. Método EM-Based

El algoritmo de Máxima Expectación (EM) tiene un enfoque amplio para el cálculo iterativo de funciones de Máxima Verosimilitud (MV) [Dempster77], es utilizado en una amplia variedad de problemas de estimación de parámetros con datos incompletos.

Descripción del algoritmo

Se desea obtener un número K de conglomerados $C_k,\ k = 1, ..., K$ a partir de un conjunto de N observaciones $X = x_1, ..., x_N$ caracterizadas, cada una, por d atributos. Para esto se puede establecer un modelo de la forma $P(X/\theta)$ que permita calcular la probalidad de que la observación x_i se encuentre en un conglomerado C_k. El modelo $P(X/\theta)$ se puede representar como una mezcla de modelos, representado de la siguiente manera :

$$P(X/\theta) = \sum_{k=1}^{K} P(C_k) * P(X_i)/C_k, \theta_K) \tag{4.10}$$

donde

$X = \{x_1, ..., x_N\}$ son las N observaciones (en el caso de este trabajo los N documentos).
$C_k,\ k = 1, ..., K$ son los conglomerados o el número de componentes de la mezcla.
$P(C_k) = p_k$ son las proporciones con las que participa cada conglomerado k en la mezcla, tal que $\sum_{k=1}^{K} p_k = 1$.

θ= $[p_1, ...,\ p_k,\ \theta_1, ..., \theta_k]$ son los parámetros de cada componente de la mezcla. El objetivo es estimar la proporción de cada componente en la mezcla y los parámetros θ, en particular $\theta_k = \mu_k, \sum_k$.

donde

μ_k es el valor esperado.
$\sum_k$ la matriz de covarianza del componente k.

Si se supone que las N observaciones son identicas e independientemente distribuidas, se

puede establecer una función de verosimilitud $L(X/\theta)$ para estimar los parámetros θ, de la forma :

$$L(X/\theta) = \sum_{i=1}^{N} ln\left(\sum_{k=1}^{K} p_k * P(x_i/C_k, \theta_k)\right) \tag{4.11}$$

La maximización de $L(X/\theta)$ permite obtener un estimador de máxima verosimilitud

$$\hat{\theta} = \arg_\theta \, max L(X/\theta) \tag{4.12}$$

Sin embargo el procedimiento analítico clásico para máximizar ésta función basada en las derivadas de $L(X/\theta)$, no siempre permite obtener la solución analítica [Bilmes98]. El algoritmo EM simplifica considerablemente el problema de estimación de parámetros en modelos de mezclas finitas.

El algoritmo se realiza en dos fases. Antes de iniciar la fase E, se proponen el número de componentes del modelo $P(X/\theta)$ (ecuación 4.11) y se proponen valores iniciales para los parámetros θ^0; es decir, los coeficientes de mezclado $p_k = P(C_k)$, los promedios μ_k y las matrices de covarianza $\sum_k$, para cada componente. A partir de esto se calculan las probabilidades condicionales *a priori* $P(x_i/C_k, \theta_k)$.

La fase E consiste en calcular las probabilidades $t_k(x_i)$, $1 \leq k \leq K$ y $1 \leq i \leq N$, condicionales *a posteriori* de la forma $P(C_k/x_i, \theta^i)$, obtenidas mediante los valores de los parámetros de la etapa anterior.

Estas probabilidades representan las probabilidades de pertenencia de x_i al conglomerado C_k, es decir, $P(x_i \in C_k)$. Esto se logra mediante el teorema de Bayes dado por la siguiente fórmula [Devore01] :

$$P(C_k/x_i, \theta^{i-1}) = \frac{P(C_k) * P(x_i/C_k, \theta^{(i-1)})}{P(x_i)} \tag{4.13}$$

donde

$P(C_k)$ son las probabilidades *a priori* de cada componente.
$P(x_i/C_k, \theta^{i-1})$ es la probabilidad *a priori* de x_i dado el conglomerado C_k.
$P(C_k/x_i, \theta^{i-1})$ son las probabilidades *a posteriori*.

La fase M consiste en estimar (restimar) los parámetros $\theta^i = \{p_1, ..., p_k,\ \mu_1, ..., \mu_K, \sum_1, ..., \sum_K\}$ mediante las siguientes fórmulas :

$$p_k = \frac{\sum_{i=1}^{N} t_k(x_i)}{\sum_{k=1}^{K}\sum_{i=1}^{N} t_k(x_i)} \tag{4.14}$$

$$\mu_k = \frac{\sum_{i=1}^{N} t_k(x_i) * x_i}{\sum_{i=1}^{N} t_k(x_i)} \tag{4.15}$$

y

$$\sum_k = \frac{\sum_{i=1}^{N} t_k(x_i) * (x_i - \mu_k) * (x_i - \mu_k)^T}{\sum_{i=1}^{N} t_k(x_i)} \tag{4.16}$$

para $k = 1, ..., K$.

Esta fase conduce a un nuevo vector de parámetros θ^i de aquí regresar a la fase E para obtener nuevos valores de $t_k(x_i)$ y reestimar el valor de parámetros a θ^{i+1}. El algoritmo itera hasta obtener una convergencia.
Fraley [Fraley06] propone representar a cada componente $P(x_i/C_k, \theta_k)$ del modelo 4.11 por un modelo gaussiano de la forma :

$$N_k(x_i/\mu_k, \sum_k) = \frac{e^{-\frac{1}{2}(x_i-\mu_k)^T * \sum_k^{-1}(x_i-\mu_k)}}{\sqrt{|2\pi \sum_k|}} \tag{4.17}$$

Con promedios μ_k y varianzas $\sum_k$.
La propuesta anterior intenta dar una visión del proceso para estimar parámetros de una mezcla de la forma $P(x_i/C_k, \theta_k)$, en particular de modelos gaussianos, mediante el algoritmo EM. Existen otros enfoques para presentar el algoritmo EM. Por ejemplo [Bilmes98] propone

obtener una mejor estimación del vector de parámetros mediante la siguiente fórmula :

$$\theta_t \leftarrow argmax Q(\theta, \theta_{t-1}) \tag{4.18}$$

donde $Q(\theta, \theta_{t-1})$ es una función equivalente a la diferencia entre la verosimilitud de la iteración actual y la anterior : $L(\theta) - L(\theta - 1)$. Una descripción más detallada del algoritmo EM, también se encuentra en el libro editado por Wu[Wu09].

En lo que respecta a la programación de EM, en la literatura se proponen diferentes implementaciones, programas y bibliotecas, algunos relajando ciertas suposiciones, otros imponiendo restricciones [Gan09]. El proyecto de R [Team11] propone diferentes bibliotecas bien probadas y validadas. Por ejemplo Fraley propone descomponer la matriz de covarianzas $\sum_k$ en :

$$\sum_k = \lambda_k S_k \Lambda_k S_k' \tag{4.19}$$

donde :

$\lambda_k = \left| \sum_k \right|^{\frac{1}{d}}$, es una matriz ortogonal constituida a partir de $\sum_k$.

Λ_k, es una matriz diagonal, definida y positiva, con determinante 1.

S_k, es la matriz ortogonal de vectores propios de $\sum_k$ y sirve para determinar la orientación de los elipsoides de equidensidad de μ_k.

En términos de μ_k, λ_k, S_k y Λ_k, se pueden construir hasta 27 familias de modelos de mezclas que surgen de la combinación de las variantes del :

- Volumen (λ_k) : Diferente para cada k.
- Forma (Λ_k) : Λ_k, diferente para cada k
- Orientación (S_k) : S_k diferente para cada k

Así el modelo EVI denota un modelo en el cual el volumen de todos los conglomerados es igual (E, de equal en inglés), la forma de los conglomerados puede variar (V, de varying) y la orientación es la identidad (I, de identity).

Para decidir entre uno u otro modelo se usa el criterio BIC, el cual se encuentra definido por :

$$BIC = -lnp(\hat{\theta}; x, M) + (v/2)ln(n) \tag{4.20}$$

donde

$\hat{\theta}$ son los parámetros.
x son las observaciones.
n es el número de observaviones.
$M_j(j = 1, ..., m)$ los modelos asociados.
v es el número de parámetros.

El criterio BIC es un criterio consistente y asimptótico para evaluar los modelos M_j y decidir por el modelo con mayor probabilidad $P(M/x)$.
Un ejemplo del algoritmo EM para una mezcla de dos modelos gaussianos univariados se presenta en el apéndice C (ver página 119).

4.7. Criterios de evaluación de agrupamiento

Los algoritmos de agrupamiento suelen evaluarse o utilizan en su funcionamiento distintas medidas internas (u objetivas) como el índice de Davies-Bouldin o el índice de Dunn, que intentan reflejar propiedades estructurales del resultado del agrupamiento. Sin embargo, la presencia de estas propiedades estructurales no garantiza el uso de los resultados para el usuario, una propiedad subjetiva reflejada por medidas externas como la medida F y que determinan hasta que punto los grupos obtenidos se asemejan a los que se hubieran logrado con una categorización manual real [Srivastava09].
A continuación se detallan los índices a utilizar en este proyecto.

- **Índice de Davies-Bouldin :** El índice de Davies-Bouldin puede ser calculado por

la siguiente fórmula [Halkidi01] :

$$DB = \frac{1}{k}\sum_{i=1}^{k} \max_{i \neq j}\left(\frac{\sigma_i + \sigma_j}{d(c_i, c_j)}\right) \tag{4.21}$$

dónde K es el número de grupos, c_x es el centro de gravedad del grupo x, σ_x es la distancia media de todos los elementos del grupo x de centroide c_x y $d(c_i, c_j)$ es la distancia entre los centroides c_i y c_j. Ya que los algoritmos que producen los grupos con baja de racimo dentro de la distancia (dentro del clúster de alta similitud) y alta inter-cluster distancias (bajo inter-cluster similitud) tendrá un bajo índice de Davies-Bouldin, el algoritmo de clustering que produce un conjunto de grupos con el más pequeño Davies-Bouldin índice es considerado el mejor algoritmo basado en este criterio.

Debido a la forma en que se define, es una función de relación de dispersión dentro de un grupo, es decir, el valor más pequeño representa la agrupación mejor.

- **Índice de Dunn :** El índice de Dunn tiene como objetivo identificar racimos densos y bien separadas. Se define como el cociente entre la distancia mínima entre los cluster de máxima intra-cluster distancia. Para cada partición de clúster, el índice de Dunn se puede calcular mediante la siguiente fórmula [Halkidi01] :

$$D = min_{i=1,\ldots,K}\left\{min_{j=1,K,j\neq K}\left(\frac{d(C_i, Cj)}{max_{K=1,\ldots,K} diam(C_K)}\right)\right\} \tag{4.22}$$

dónde K representa el número de grupos que vienen de la misma división, $dist(C_i, C_j)$ es la distancia entre los grupos C_i y C_j y $diam(C_K)$ es el diámetro calculado para cada grupo.

El índice de Dunn depende de K, es decir, del número de grupos en el conjunto. Si el número de grupos no es conocido *a priori*, entonces se elige la configuración con el índice de Dunn más grande que se tenga.

4.8. Búsqueda de criterios de eficiencia y eficacia

Para evaluar la aplicación es importante presentar lo que se entiende por eficiencia y eficacia. Para esto tomaremos como base el trabajo publicado por Vázquez y Colaboradores [Vázquez07].

El control debe involucrar hacer bien las cosas [eficiencia] y hacer las cosas correctas [eficacia]. Es importante notar que hacer las cosas bien "doing things right" no significa que se hagan las cosas correctas "doing the right things" [Ackoff03]. En numerosas situaciones se hacen las cosas bien aunque éstas no sean las correctas. Desafortunadamente, mientras más hacemos bien las cosas, pero éstas son cosas incorrectas, nos equivocamos más. Esto significa que el aumento en eficiencia puede hacer decrecer la eficacia. Existen muchas maneras de incrementar el control de tal manera que se logre que un sistema sea más eficiente; sin embargo, ellas no pueden cambiar al sistema para que deje de hacer las cosas equivocadas y hacer las correctas. Esas opciones sólo pueden incrementar la eficiencia con que el sistema lleva a cabo sus funciones, pero no incrementa su eficacia.

De aquí podemos decir que la eficiencia está relacionada con el hecho de hacer bien las cosas. Si integramos lo anterior con la definición de eficiencia y eficacia presentada en otras disciplinas. La eficiencia, dependiendo de la disciplina tiene distintas definiciones (http://es.wikipedia.org/wiki/Eficiencia) : en Ingeniería de procesos (físicos, biológicos, químicos, etc.) la eficiencia de un proceso es la relación entre la energía útil y la energía invertida. En economía, la eficiencia es la cantidad mínima de entradas (horas-hombre, capital invertido, materias primas, etc.). En agricultura, la eficiencia (por ejemplo en sistemas de producción basados en el riego) es el porcentaje del volumen de agua derivada en un sistema de riego con relación al volumen de agua efectivamente utilizado por las plantas. En estadística, la eficiencia de un estimador es una media de su varianza. La efectividad se define como la capacidad de lograr un efecto deseado. Esto nos orienta a la necesidad de definir con claridad los recursos invertidos, los resultados obtenidos y su relación con lo que se considera es correcto.

En el caso del presente proyecto, la aplicación puede o no resultar eficiente y eficaz desde uno o varios puntos de vista. Por ejemplo podemos considerar como criterios de eficiancia:

- Agrupar un gran volumen de documentos y un gran número de atributos con un mínimo de recursos de comunicación por internet y de almacenamiento.
- Obtener un número mínimo de grupos de documentos a partir de grandes volúmenes de documentos con varianza mínima.
- Obtener un número mínimo de grupos de documentos a partir de grandes volúmenes de documentos en un tiempo mínimo.
- Realizar un número máximo de pruebas en un tiempo mínimo.

Y como criterios de eficacia :

- Lograr una agrupación adecuada con el fin de discriminar en grandes volúmenes de documentos.
- Tiempos mínimos de procesamiento.
- Procesamiento de documentos en forma continua.

Es importante considerar que no necesariamente estos criterios son consistentes. Por ejemplo para obtener un número de grupos de documentos con varianza mínima puede requerir tiempos importantes de procesamiento o bien si se requiere que los documentos sean procesados en forma continua, se pueden requerir recursos importantes y fiables en comunicación.
En nuestro caso, de acuerdo a lo planteado en los objetivos se desea una aplicación lo suficientemente flexible o configurable para agrupar documentos XML manejando diferentes opciones (métodos de normalización, métricas, algoritmos, muestras, secuencias, etc), utilizando una cantidad reducida en recursos de almacenamiento, sin importar que esto se realice o no en tiempo real. Además es importante considerar que no todas las agrupaciones son útiles, pues agrupar un gran número de documentos en un solo grupo o en un número muy grande de grupos puede ser poco útil a pesar de que los índices anteriormente expuestos señalen lo contrario.
Tampoco se considera necesario, para la realización de pruebas a todos los documentos ya que la teoría estadística del muestreo nos aporta resultados adecuados con un volúmen

reducido de documentos. Esto puede ser considerado un criterio de eficacia. Es decir realizar un máximo número de pruebas en un tiempo mínimo, con métodos que den resultados fiables.

Capítulo 5

Construcción de la aplicación

En este capítulo se presenta la construcción de la aplicación para el minado de documentos XML. Poniendo particular atención al inicio del desarrollo, para modelar tanto el funcionamiento de la aplicación; así para modelar los datos en el modelo relacional. Por esta razón se empieza con una sección dedicada al modelado y posteriormente se aborda la implementación del modelo relacional a la base de datos.

5.1. Modelado UML

El modelado es una parte central de todas las actividades que conducen a la producción de un buen software. Se construyen modelos para visualizar y controlar la arquitectura de un sistema; en general para comprender mejor el sistema que se está desarrollando.
El Lenguaje Unificado de Modelado (UML por sus siglas en inglés), es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad; está respaldado por el OMG (Object Management Group). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema.
UML se puede usar para modelar distintos tipos de sistemas; así como su funcionamiento y las diversas interacciones que el usuario pueda tener con el sistema y para ellos ofrece nueve diagramas [Rumbaugh99] de los cuales se utilizaron los siguientes:

- Diagramas de casos de uso, para modelar el comportamiento que tendrá el usuario

con diversas partes del sistema (ver anexo A).

- Diagramas de clases, para modelar la estructura estática de las clases de el sistema.

Los diagramas de clases son utilizados durante el proceso de análisis y diseño de los sistemas, donde se crea el diseño conceptual de la información que se manejará en el sistema, y los componentes que se encargaran del funcionamiento y la relación entre uno y otro. Por tal motivo el diagrama de clases se convierte en una pieza fundamental para la creación de nuestro sistema.

Con el fin de modelar el diseño de la aplicación tanto a nivel procesos y base de datos se propone usar una metodología que nos permita dar seguimiento, para esto se utilizará la herramienta CASE Umbrello UML Modeller [1].

En la figura 5.1, se puede apreciar el diagrama de clases de la aplicación. Las clases son las plantillas de los objetos en los que se pueden ver representados sus atributos o características y su comportamiento o métodos; así como las diversas relaciones que existen entre ellas. En el diagrama de clases, las primeras clases que se definieron fueron las clases Coleccion y la clase Documento. Estas clases son directamente asociadas con una multiplicidad [1,*], dado que una colección posee varios documentos. Una clase importante es también Dist_Frecuencia, la cual se encuentra asociada con varios documentos y varios tokens. Los tokens son objetos identificados en el documento y puede tener varias categorías (por ejemplo estructural o gramatical). Las clases restantes están relacionadas con el proceso de minería, los cuales aplican algoritmos de agrupamiento, basados en una métrica dada una matriz de frecuencia (revelada por las clases : Token y Dist_Frecuencia). Las clases propuestas son : Coleccion, Documento, Dist_Frecuencia, Token, Criterio, Eleccion, Metrica, Aplicar, Cluster y Algoritmo.

En la figura 5.2, se muestra el diagrama de clases actualizado de la aplicación para el minado de documentos XML, el cual presenta la introducción de nuevas clases que permiten tener una mejor conceptualización del sistema.

La tabla 5.1 muestra en la primera columna el nombre de la clase, en la segunda columna muestra los atributos que posee así como la visibilidad del atributo, en la tercera columna

[1] *http://uml.sourceforge.net/*

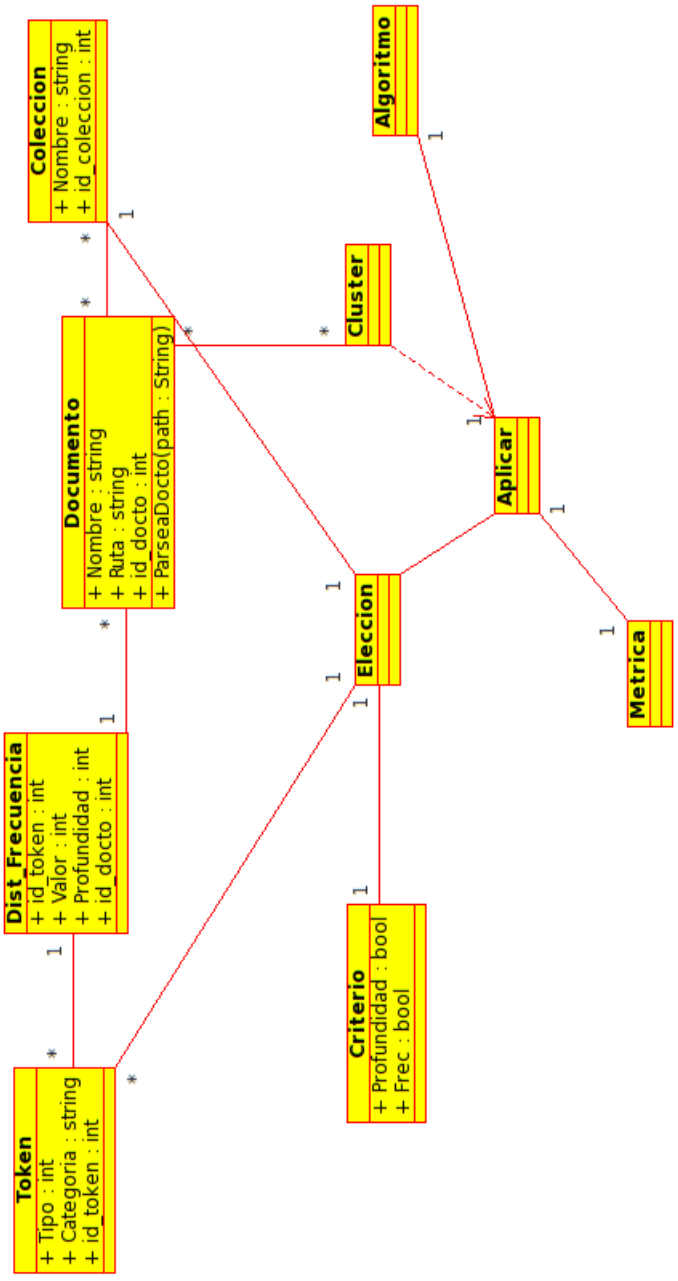


Figura 5.1: Diagrama de clases de la aplicación

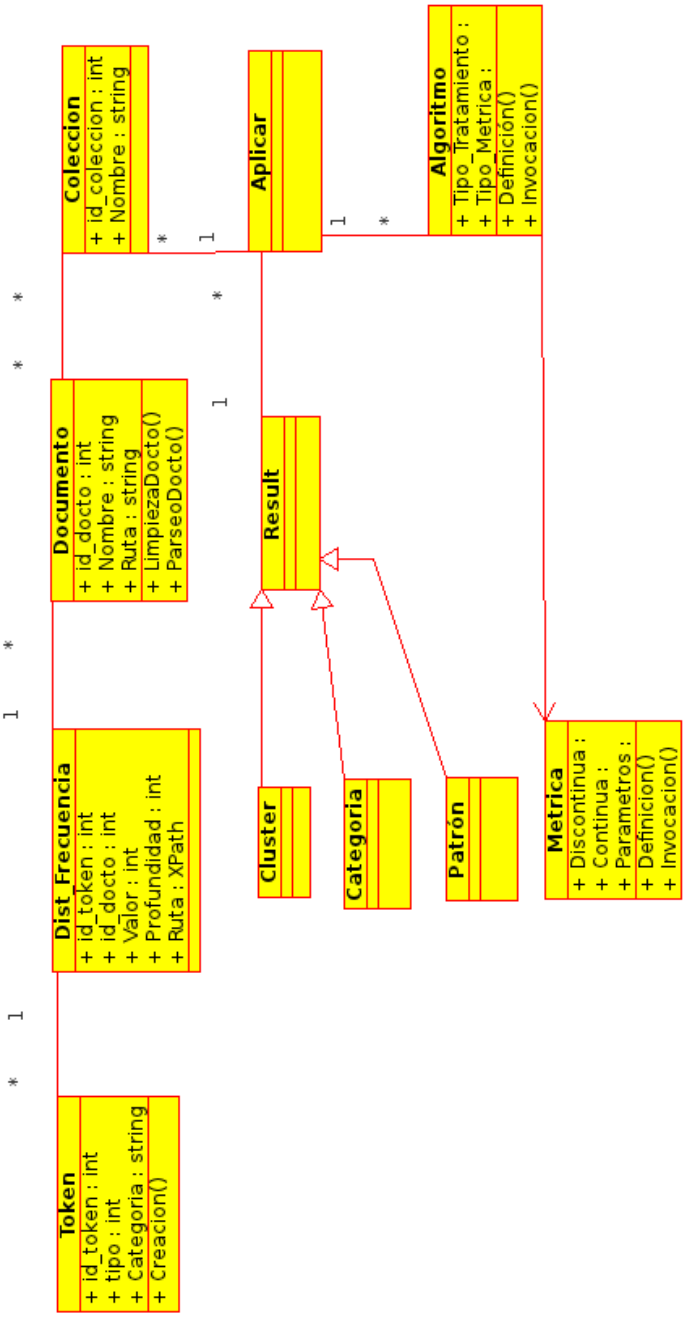


Figura 5.2: Diagrama de clases de la aplicación actualizado

muestra sus métodos, en la cuarta columna muestra las clases con las que posee una relación; así como el tipo de relación. La quinta columna muestra la multiplicidad [2] que posee dicha clase con las clases con las que posee relación.
Del estudio anterior se estableció la especificación para el diseño de la aplicación de minería de documentos XML. Este trabajo fue presentado en el congreso IMECS 2011 en Hong Kong en Marzo 16-18, 2011 y publicado [Galindo-Durán11](veáse anexo H, página 169).

5.2. Modelo relacional

Los diagramas UML desarrollados muestran la estructura y algunos detalles de la implementación de la aplicación. A partir del diagrama de clase se obtiene el modelo relacional de la base de datos.

5.2.1. Diseño del modelo relacional

El modelo relacional representa a los atributos de una colección de datos, relacionados en formas simples. En la figura 5.3, se puede apreciar el modelo relacional de la aplicación, el cual se obtuvo a partir del diagrama de clases de la aplicación.
A continuación se presenta una breve descripción de cada una de las tablas del modelo.

- **COLECCION**

  La tabla COLECCION es la que posee la información de la ubicación de los documentos de las colecciones; en la tabla E.2, se presenta la descripción de dicha tabla.

  Ejemplos de los datos contenidos en la tabla COLECCION se muestran en la tabla 5.3, que se encuentra en la página 58.

  La tabla COLECCION posee una relación 1 a muchos $(1 \ldots *)$ con la tabla DOC_COLECC, ya que una colección puede contener muchos documentos XML, en la tabla 5.4, se presenta la descripción de la tabla DOC_COLECCION.

[2] *Existen diversos tipos de multiplicidad como son: uno y solo uno (1), cero a uno (0...1), Desde N hasta M (N...M), cero a varios (0...*), uno o varios (1...*).*

Tabla 5.1: Descripción del diagrama de clases.

CLASE	ATRIBUTOS	MÉTODOS	RELACIÓN	MULTIPLICIDAD
Coleccion	id_coleccion : Público Nombre :Público		Documento : Asociación Eleccion : Asociación	[*...*] [*...1]
Documento	id_documento : Público Nombre : Público Ruta : Público	LimpiezaDocto() ParseaDocto()	Dist_Frecuencia : Asociación Cluster : Asociación	[*...1] [*...*]
Dist_Frecuencia	id_token : Público id_docto : Público Valor : Público Profundidad : Público CaminoToken : Público		Token : Asociación Documento : Asociación	[1...*] [1...*]
Token	id_categoria : Público Tipo : Público Categoria : Público		Dist_Frecuencia : Asociación Eleccion : Asociación	[*..1] [1...1]
Criterio	Profundidad : Público Frec : Público		Eleccion : Asociación	[1...1]
Eleccion			Token : Asociación Criterio : Asociación Coleccion : Asociación Aplicar : Asociación	[1...1] [1...1] [1...*] [1...1]
Metrica	id_Metrica : Público		Aplicar : Asociación	[1...1]
Aplicar			Eleccion : Asociación Metrica : Asociación Cluster : Dependencia Algoritmo : Asociación	[1...1] [1...1] [1...*] [1...1]
Cluster			Aplicar : Dependencia Documento : Dependencia	[*...1] [*...*]
Algoritmo	id_alg : Público		Aplicar : Asociación	[1...1]

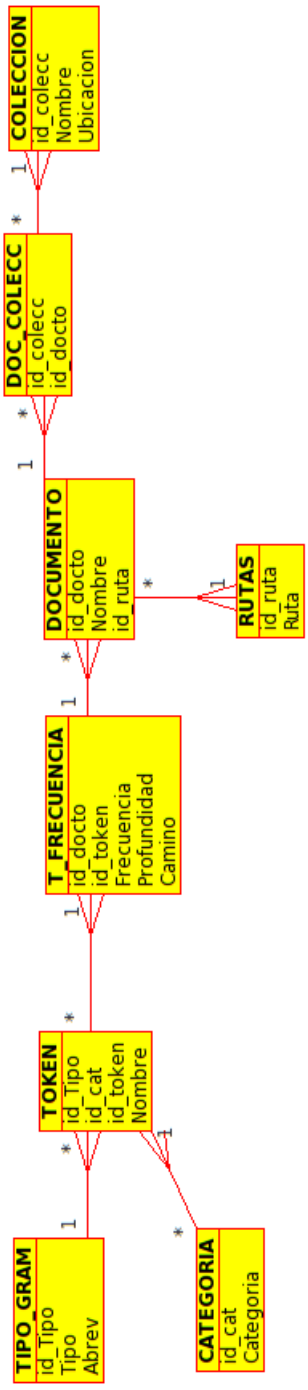


Figura 5.3: Modelo relacional de la aplicación

Tabla 5.2: Estructura de la tabla COLECCION

Columnas	**Tipo de datos**	**Descripción**
Id_colecc	Número	Representa el número de colección, es un número secuencial
Nombre	Cadena de caracteres	Es el nombre que posee la colección
Ubicación	Cadena de caracteres	Es la ruta donde se encuentra ubicada la colección con documentos XML a analizar

Tabla 5.3: Contenido de la tabla COLECCION

id_coleccion	**nombre**	**path**
1	INEX	/mnt/windows/ColeccINEX/ColeccDesc

- **DOC_COLECCION**

Tabla 5.4: Estructura de la tabla DOC_COLECCION

Columnas	**Tipo de datos**	**Descripción**
Id_colecc	Número	Representa el número de colección, es un número secuencial
Id_docto	Número	Representa el número del documento

La tabla DOC_COLECCION posee dos relaciones, la primer relación es muchos a 1 ($*\ldots 1$) con la tabla COLECCIÓN, lo cual significa que muchos documentos pueden estar contenidos en una colección. La segunda relación es una relación muchos a 1 ($*\ldots 1$) con la tabla DOCUMENTO, lo cual significa que un documento puede estar presente en múltiples colecciones, por ejemplo se puede considerar una colección reducida a partir de una colección de gran tamaño.

- **DOCUMENTO**

La tabla DOCUMENTO contiene todos los documentos tratados, en la tabla 5.5, muestra la descripción de dicha tabla.

Tabla 5.5: Estructura de la tabla DOCUMENTO

Columnas	**Tipo de datos**	**Descripción**
Id_docto	Número	Representa el número del documento
Nombre	Cadena de caracteres	Es el nombre de cada uno de los documentos
Id_ruta	Número	Representa el identificador de la ruta donde se encuentran los documentos

La tabla DOCUMENTO posee 3 relaciones, la primera relación es $(1 \ldots *)$ con la tabla DOC_COLECC. La segunda es una relación uno a muchos $(1 \ldots *)$ con la tabla RUTAS y la tercera relación es muchos a uno $(* \ldots 1)$ con la tabla T_FRECUENCIA.

En la tabla 5.6, se muestran algunos ejemplos de la información contenida en la tabla DOCUMENTO.

Tabla 5.6: Parte del contenido de la tabla DOCUMENTO

id_docto	**nombre**	**id_ruta**
1	36174.xml	1
2	45222.xml	1
3	16975.xml	1

- **RUTAS**

  La tabla 5.7, presenta la estructura de dicha tabla. En dicha La tabla se guardar la localización de los documentos XML.

  La tabla RUTAS presenta una relación $(* \ldots 1)$ con la tabla DOCUMENTO.

- **T_FRECUENCIA**

Tabla 5.7: Estructura de la tabla RUTAS

Columnas	**Tipo de datos**	**Descripción**
Id_ruta	Número	Es el identificador de las rutas de los documentos XML
Ruta	Cadena de caracteres	Es el camino completo y absoluto para encontrar cada documento

La tabla T_FRECUENCIA es la tabla que contiene las frecuencias de los tokens de etiquetas, contenido e hipervínculos encontrados en los documentos XML, en la tabla 5.8, se muestra su descripción.

Tabla 5.8: Estructura de la tabla T_FRECUENCIA

Columnas	**Tipo de datos**	**Descripción**
Id_docto	Número	Representa el número de documento
Id_token	Cadena de caracteres	Representa el identificador de cada token
Frecuencia	Número	Número de veces que se repite cada token por nivel
Profundidad	Número	Profundidad en la que se encuentra el token
Camino	Cadena de caracteres	Camino del token dentro del árbol de parseo del documento XML

La tabla T_FRECUENCIA posee dos relaciones, la primer relación uno a muchos $(1 \ldots *)$ con la tabla DOCUMENTO, la segunda relación muchos a uno $(* \ldots 1)$ con la tabla TOKEN.

- **TOKEN**

  La tabla TOKEN contiene los tokens encontrados en los documentos XML, en la tabla 5.9, se muestra la descripción de dicha tabla.

  La tabla TOKEN posee tres relaciones, la primera relación uno a muchos $(1 \ldots *)$

Tabla 5.9: Estructura de la tabla TOKEN

Columnas	**Tipo de datos**	**Descripción**
Id_tipo	Número	Representa el identificador del tipo
Id_cat	Número	Representa el identificador del tipo de token (etiqueta,contenido,hipervínculo)
Id_token	Número	Representa el identificador de cada token

con la tabla T_FRECUENCIA, la segunda relación uno a uno (1...1) con la tabla TIPO_GRAM, lo cual indica que un token solo le corresponde un sólo tipo gramatical y la tercera relación uno a uno (1...1) con la tabla CATEGORIA, lo cual representa que un token sólo puede provenir de una etiqueta, contenido o hipervínculo.

- **TIPO_GRAM**

  La tabla TIPO_GRAM posee los diversos tipos de palabras que existen en el idioma inglés, en la tabla 5.10, se describe su estructura. Esta tabla es invariable.

Tabla 5.10: Estructura de la tabla TIPO_GRAM

Columnas	**Tipo de datos**	**Descripción**
Id_tipo	Número	Representa el identificador del tipo gramatical
Id_token	Número	Representa el identificador de cada token
Abrev	Cadena de caracteres	Abreviatura del tipo gramatical

La tabla TIPO_GRAM posee una relación uno a uno (1...1) con la tabla TOKEN.

Ejemplos de los datos contenidos en la tabla TIPO_GRAM son los que se muestran en la tabla 5.11.

- **CATEGORIA**

  La tabla CATEGORIA contiene que tipo de token existe, en la tabla 5.12, se muestra la descripción de su estructura. Esta tabla también es invariable con respecto a cualquier

Tabla 5.11: Contenido de la tabla TIPO_GRAM

id_tipo	**tipo**	**abrev**
1	Coordinating conjunction	CC
2	Cardinal number	CD
3	Determiner	DT

colección.

Tabla 5.12: Descripción de la tabla CATEGORIA

Columnas	**Tipo de datos**	**Descripción**
Id_cat	Número	Representa el identificador del tipo
Categoria	Cadena de caracteres	Representa el tipo de categoría de cada token (etiqueta,contenido,hipervínculo

La tabla CATEGORIA tiene una relación uno a uno (1...1) con la tabla TOKEN.

Ejemplos de los datos contenidos en la tabla CATEGORIA son los que se muestran en la tabla 5.13

Tabla 5.13: Tabla que muestra el contenido de la tabla categoria

id_cat	**categoría**
1	contenido
2	etiqueta
3	hipervínculo
4	otros

Conforme al avance de la implementación de este proyecto se realizaron cambios en el modelo relacional, lo que lleva a una segunda aproximación de los módulos conceptuales y que se pueden apreciar en la figura 5.4.

En la segunda aproximación se pueden apreciar modificaciones a las tablas : COLECCION,

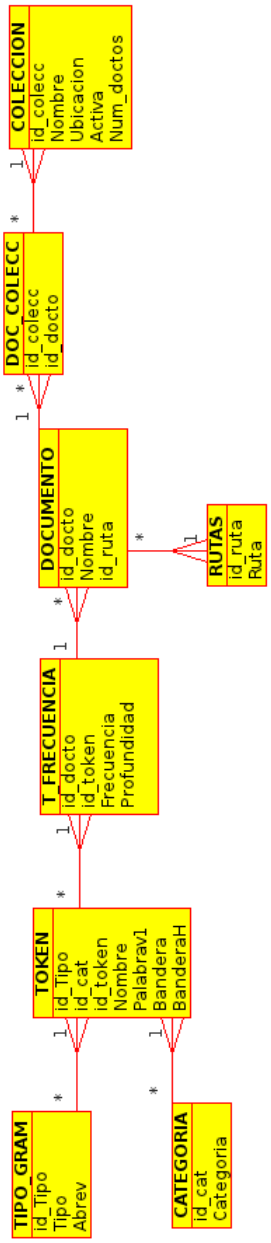


Figura 5.4: Modelo relacional de la aplicación en una segunda aproximación

T_FRECUENCIA, DOCUMENTO y TOKEN, a continuación se describen estas modificaciones. En la tabla COLECCION se le agregaron 2 atributos o columnas adicionales, las cuales son : Activa y Num_doctos. En la tabla 5.14 , se muestra la estructura general de dicha tabla.

Tabla 5.14: Estructura de la tabla COLECCION en la 2da aproximación

Columnas	**Tipo de datos**	**Descripción**
Id_colecc	Número	Representa el identificador de cada colección
Nombre	Cadena de caracteres	Representa el nombre de cada colección
Ubicación	Cadena de caracteres	Representa la ruta de cada colección
Activa	Cadena de caracteres	Representa si una colección está activa o no
Num_doctos	Número	Representa el número total de documentos que posee la colección

En la tabla T_FRECUENCIA en la segunda aproximación se omitió el atributo o columna camino, la tabla 5.15, muestra la estructura general de dicha tabla.

Tabla 5.15: Estructura de la tabla T_FRECUENCIA en la 2da aproximación

Columnas	**Tipo de datos**	**Descripción**
Id_docto	Número	Representa el número de documento
Id_token	Cadena de caracteres	Representa el identificador de cada token
Frecuencia	Número	Número de veces que se repite cada token por nivel
Profundidad	Número	Profundidad en la que se encuentra el token

En la tabla TOKEN se le agregó un atributo o columna adicional, la cual es : BanderaH, el cual servirá para identificar los hipervínculos internos y externos, también se omitió el

atributo Palabrav2. En la tabla 5.14, se muestra la estructura general de dicha tabla.

Tabla 5.16: Estructura de la tabla TOKEN en la 2da aproximación

Columnas	**Tipo de datos**	**Descripción**
Id_tipo	Número	Representa el identificador del tipo
Id_cat	Número	Representa el identificador del tipo de token (etiqueta,contenido,hipervínculo)
Id_token	Número	Representa el identificador de cada token
Nombre	Cadena de caracteres	Representa el texto de la palabra o la etiqueta del token
Palabrav1	Cadena de caracteres	Representa si el token es una palabra es vacia
Bandera	Número	Representa si las etiquetas son hard o soft
BanderaH	Número	Representa si un hipervínculo es interno o externo

5.3. Detalles de implementación

La construcción de la aplicación de agrupamiento está basada desde su análisis UML, programación, almacenamiento de los datos y tokenización (análisis léxico) del contenido textual de los documentos XML, en herramientas y lenguajes de software libre u OpenSource.

5.3.1. Condiciones de implementación

La aplicación fue desarrollada en Java y el modelo relacional se implementó en el Sistema Gestor de Base de Datos (SGBD) PostgreSQL Server versión 9.0.2 en i686-pc-linux de 32 bits, el acceso al SGBD se realizó con el paquete (API) JDBC para PostgreSQL. Fue probado en Linux bajo la distribución de Linux ARCH, entorno de escritorio Gnome 2.32 y 3.0.

Toda la aplicación está estructura alrededor de la base de datos, ya que aquí se encuentra

la mayoría de la información manejada en el código. Esta estrategia es justificada debido a la necesidad de evitar los mismos cálculos cada vez que se necesita la aplicación de un algoritmo de agrupamiento o con parámetros diferentes.

La parte que consume más recursos computacionales y de tiempo es la extracción de los tokens; así como la obtención de las frecuencias y los indicadores de presencia que constituyen los valores de los vectores que se manejan durante la fase de los algoritmos de agrupamiento.

De manera global la aplicación posee tres etapas : la primera el pre-tratamiento se ejecuta una sóla vez por cada documento tratado hay varios accesos a la base de datos (desde la inserción en la tabla DOCUMENTO hasta el fin de la tokenización) se realiza en modo transaccional, atómica, coherente y consolidado. Posteriormente se ejecuta la fase de aplicación de algoritmos de agrupamiento y finalmente la evaluación de la solución calculada. Las dos últimas etapas pueden realizarse varias veces y el experto en el dominio decidirá si la solución es satisfactoria.

En la figura 5.5, se presenta un diagrama de flujo donde se puede apreciar las interacciones que tiene la aplicación con la base de datos.

A continuación se presenta el pseudocódigo de las interacciones que se realizan a la base de datos en la fase de pre-tratamiento.

```
Insert into DOCUMENTO tabla;
Get id_docto de la tabla DOCUMENTO;
Ruta="";
Profundidad=0;
Lee el documento XML(empezando desde la raíz)
For each nodo n del árbol XML(obtenido por DOM)
Ruta=Ruta(concatenar)"/"(concatenar)n.getNodoName;
Profundidad=Cuenta(/)-1;
  if n es outsidelink o collectionlink o weblink o wikipedialink
  n es un hipervínculo
    if n es collectionlink o wikipedialink
    n es un hipervínculo interno;
    else
    n es un hipervínculo externo;
  else
  n es una etiqueta
    if n es p o section o emph2 o emph3 article o document o item o name
      n es una hard tag;
    else
      n es una soft tag;
  end if;
  Consult TOKEN tabla
    if token no existe
      Insert into TOKEN tabla;
  Cosult tabla T_FRECUENCiA tabla con (id_docto, id_token,Profundidad)
    if Consult no retorno resultado
```

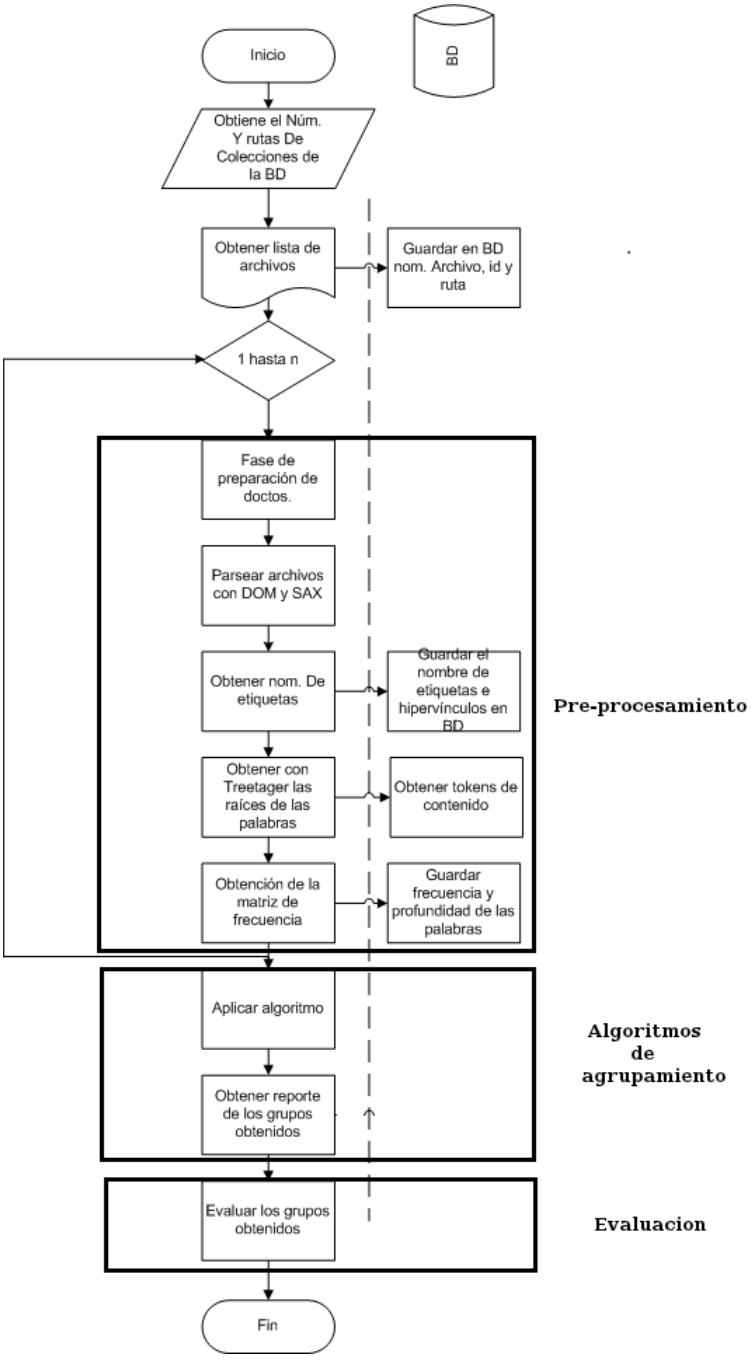


Figura 5.5: Interacciones de la aplicación con la BD

```
      Inser into T_FRECUENCIA tabla;
    else
      Get Frecuencia;
      Frecuencia++;
      Update T_FRECUENCIA set frecuencia=Frecuencia;
  Get CDATA de cada nodo n;
  Set archivo con el CDATA;
  Save CDATA en archivo txt;
  Call a TreeTagger;
  While Lee archivo
    Tokeniza cada línea de archivo;
    Get forma invariable del Token;
    Get abreviatura del tipo del Token;
    Consult TIPO_GRAM tabla;
    Get id_tipo del token;
    Consult TOKEN tabla
    if no existe
    Insert into TOKEN tabla;
  end While
    Consult TOKEN tabla
    if token no existe
      Insert into TOKEN tabla;
  Cosult tabla T_FRECUENCiA tabla con (id_docto, id_token,Profundidad)
    if Consult no retorno resultado
      Inser into T_FRECUENCIA tabla;
    else
      Get Frecuencia;
      Frecuencia++;
      Update T_FRECUENCIA set frecuencia=Frecuencia;
end For;
```

En la arquitectura de la aplicación presentada en el Marco teórico figura 4.1, se vió la necesidad de modificar el proceso e introducir nuevas fases quedando de manera más detallada y explícita como se muestra en la figura 5.6.

Cabe señalar que en la fase de pre-tratamiento, de los datos contenidos en la base de datos, se pueden utilizar en varios procesos de minería, como lo son : la clasificación o búsqueda de patrones y durante un eventual proceso de búsqueda de información.

5.3.2. Interfaz Gráfica de Usuario (GUI)

Para la flexibilidad de configuración se diseñó y se creo una Interfaz Gráfica de Usuario (GUI), que permita configurar el archivo o colección a trabajar, considerando diferentes atributos de los documentos y la posibilidad de tomar toda o una muestra de ésta, o seleccionar diferentes tipos de normalización (Z, máxima-mínima y máxima), o diferentes métricas (Euclideana, Manhattan, Minkowski, Máxima y Binaria) con diferentes estrategias de agrupamiento (Ward, sencilla, completa, promedio, Centroide y media), así como

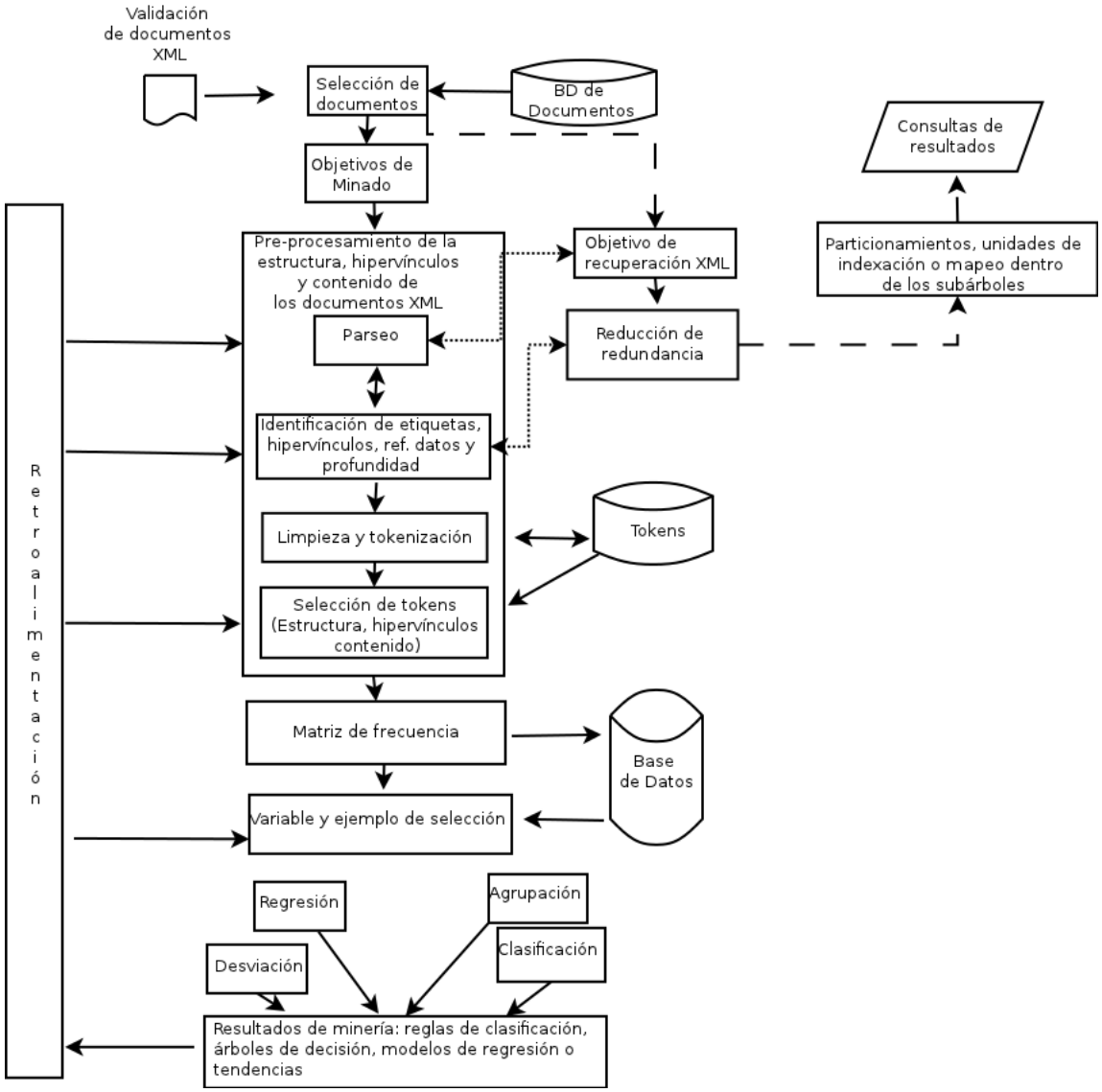


Figura 5.6: Diagrama conceptual modificado de la Minería XML

los algoritmos de agrupamiento.

Se presentan las diferentes pantallas que conforman las GUI's de esta aplicación en las páginas 70,71, 71, 72, 72, 73 y 73, la explicación detallada de las pantallas que conforman la interfaz se presenta en el anexo E. Cabe mencionar que dichas pantallas se contruyeron a partir de los diagramas de casos de uso.

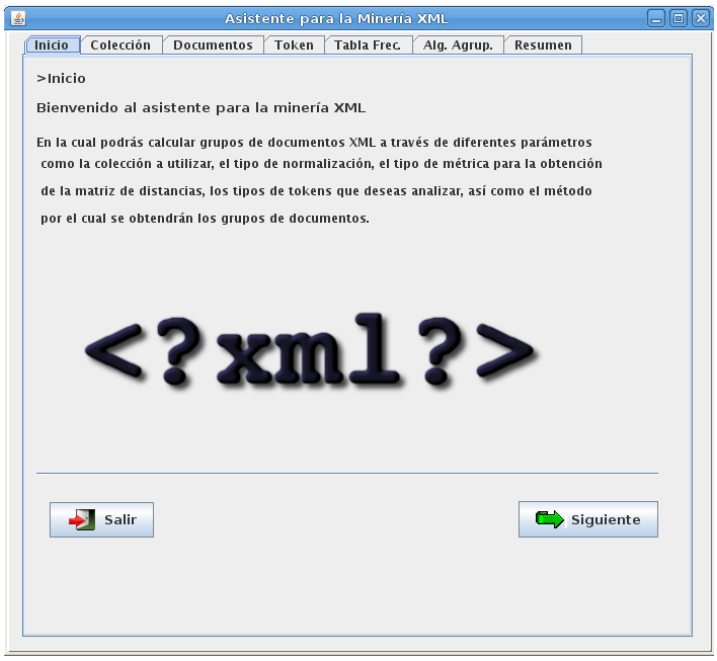


Figura 5.7: Pantalla de inicio del asistente para la minería XML.

5.3.3. Herramientas CASE

Para coadyuvar con la realización de este proyecto fue necesario la utilización de herramientas CASE para facilitar el desarrollo de diversos diagramas, como el de clases y el modelo relacional, específicamente de la herramienta CASE Umbrello UML Modeller, mencionada en el marco teórico. Umbrello no sólo facilita la elaboración de diagramas, sino que en diagramas de clase y de entidad relación permite la generación de código, muy útil que prácticamente es compatible con cualquier manejador de bases de datos. A continuación se muestra el código producido del diagrama entidad relación por Umbrello :

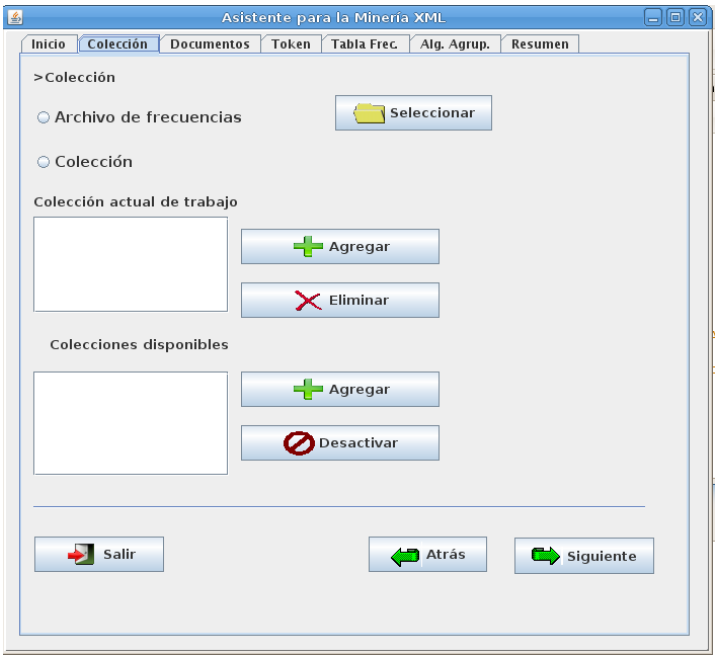


Figura 5.8: Pantalla que muestra las opciones para comenzar a trabajar.

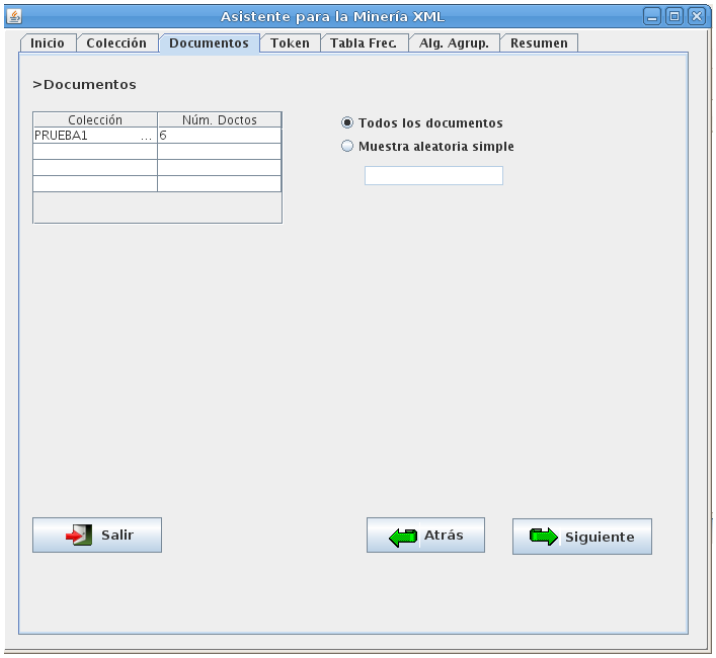


Figura 5.9: Pantalla que muestra el número de documentos de la colección.

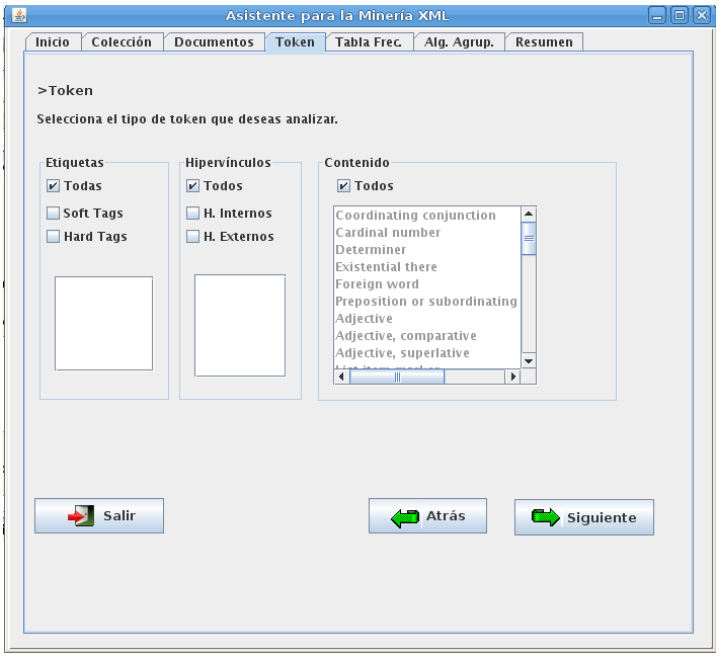


Figura 5.10: Pantalla que muestra las elecciones posibles de los tokens.

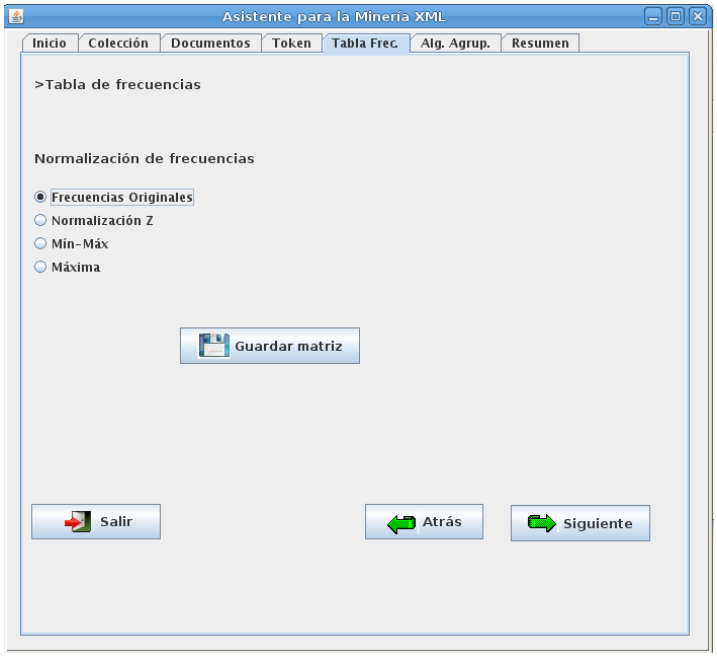


Figura 5.11: Pantalla que muestra los métodos para normalizar las frecuencias.

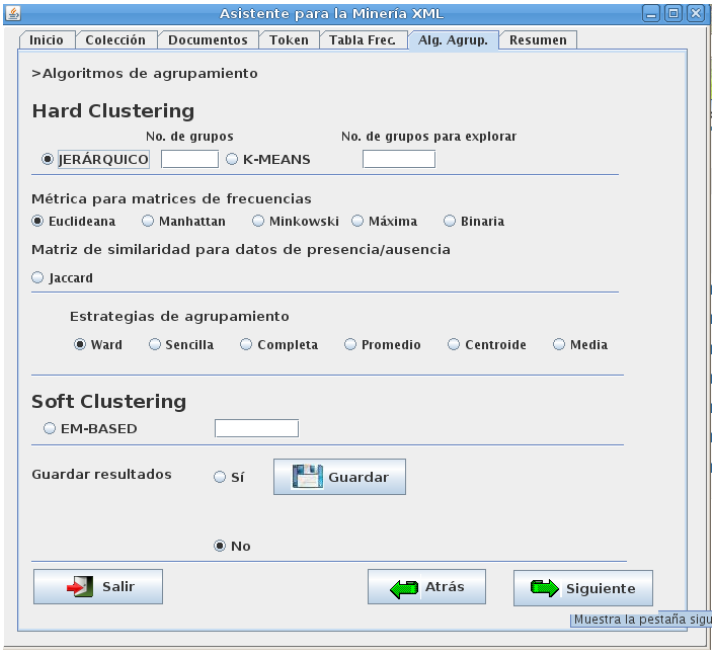


Figura 5.12: Pantalla que muestra los métodos de agrupamiento disponibles.

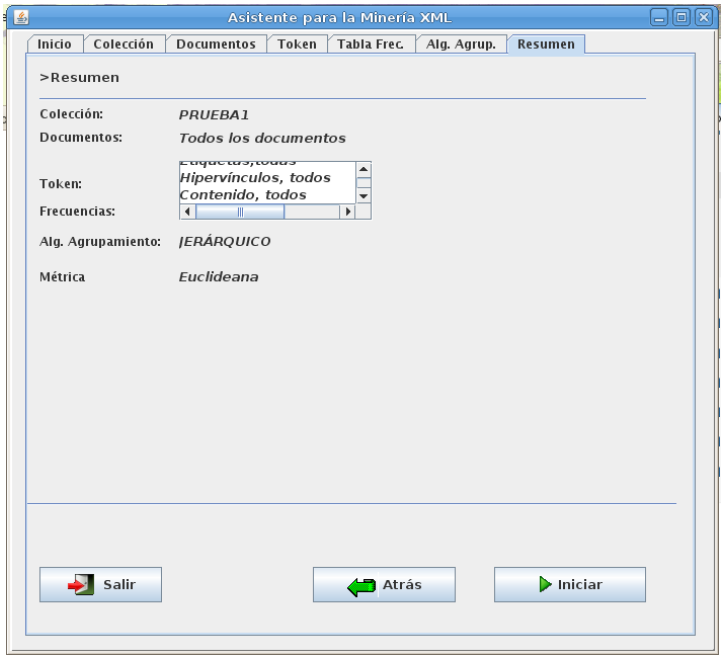


Figura 5.13: Pantalla que muestra el resumen de los parámetros de configuración del asistente.

```
CREATE TABLE COLECCION (
  id_colecc int2 NOT NULL ,
  Nombre char(100) NOT NULL ,
  Path char(250) NOT NULL,
  PRIMARY KEY (id_colecc));

CREATE TABLE DOC_COLECC (
  id_colecc int2 NOT NULL ,
  id_docto int8 NOT NULL ,
  FOREIGN KEY (id_colecc) REFERENCES COLECCION (id_colecc),
  FOREIGN KEY (id_docto) REFERENCES DOCUMENTO (id_docto));

CREATE TABLE DOCUMENTO (
  id_docto int8 NOT NULL ,
  Nombre char(100) NOT NULL ,
  id_ruta int8 NOT NULL,
  Cluster_Jerarquico int2,
  Cluster_EM int2,
  PRIMARY KEY (id_docto),
  FOREIGN KEY (id_ruta) references RUTAS(id_ruta));

CREATE TABLE RUTAS (
  id_ruta int8 NOT NULL ,
  Ruta char(200) NOT NULL,
  PRIMARY KEY(id_ruta));


CREATE TABLE T_FRECUENCIA (
  id_docto int8 NOT NULL ,
  id_token int8 NOT NULL ,
  Frecuencia int2 NOT NULL ,
  Profundidad int2 NOT NULL,
  FOREIGN KEY (id_docto) REFERENCES DOCUMENTO (id_docto),
  FOREIGN KEY (id_token) REFERENCES TOKEN (id_token));

CREATE TABLE TOKEN (
  id_Tipo int2 NULL ,
  id_cat int2  NOT NULL ,
  id_token int8 NOT NULL ,
  Nombre char(100) NOT NULL,
  Palabrav1 char(40),
  Palabrav2 char(40),
  Bandera int2,
  BanderaH int2,
  PRIMARY KEY (id_token),
  FOREIGN KEY (id_Tipo) REFERENCES TIPO_GRAM (id_Tipo),
  FOREIGN KEY (id_cat) REFERENCES CATEGORIA (id_cat));

CREATE TABLE TIPO_GRAM (
  id_Tipo int2 NOT NULL ,
  Tipo char(50) NOT NULL ,
  Abrev char(15) NOT NULL,
PRIMARY KEY (id_tipo));

CREATE TABLE CATEGORIA (
  id_cat smallint NOT NULL ,
  Categoria char(50) NOT NULL,
PRIMARY KEY (id_cat));
```

5.4. Pasos para el desarrollo

A continuación se describe de manera detallada la secuencia de los pasos que sigue la aplicación, los pasos que se presentan se pueden realizar en orden; pero también se pueden regresar algunos pasos anteriores.

1. Obtener la colección XML en una ruta especificada. La colección a trabajar es la colección INEX la cual posee 22 carpetas, en la tabla 5.17 se presenta la cantidad de documentos que posee la colección pequeña de INEX.

Tabla 5.17: Número de documentos por directorio de la colección INEX

No.Carpeta	No. Doctos
00	30,000
01	30,000
02	30,000
03	30,000
04	30,000
05	30,000
06	30,000
07	30,000
08	30,000
09	30,000
10	30,000
11	30,000
12	30,000
13	30,000
14	30,000
15	30,000
16	30,000
17	30,000
18	30,000
19	30,000
20	30,000
21	29,388
Total	659,388

2. Obtener el número de documentos que posee la colección En este paso se cuentan todos y cada uno de los documentos XML de la colección, una vez obtenido el número total de documentos se inserta en la base de datos, en la tabla COLECCION, en el

campo No_Doctos. Para el caso de la colección pequeña de INEX cuenta con 659,388 documentos XML.

3. Generar el archivo con los números consecutivos o aleatorios de la colección. Ejemplo del archivo generado:

```
1 2 3 4 5 6 7 8 9 10 11 12 13
```

   El archivo que se genera tendrá los números consecutivos (en caso de haber seleccionado toda la colección) o bien los números aleatorios (en caso de haber seleccionado una muestra). Dicho archivo se localizará en la carpeta del proyecto, dentro de la carpeta ArchivoConf y es llamado ArchNumDoctos.txt.

4. Leer los documentos XML de la colección, de acuerdo con el archivo con los números generados.

5. Recorrer cada rama del documento con DOM.

6. Obtener el nombre del nodo y su valor.

7. Identificar si es una etiqueta, contenido o hipervínculo.

   - Si es contenido descomponerlo con TreeTagger. Dicha descomposición se realiza mediante el siguiente shell :

```
#!/bin/bash
Ruta="./Proyecto/ArchSalida/"
     echo "parametro" | cmd/tree-tagger-english > ${Ruta}TreeTSalida.txt done
exit
```

     Al momento de crear el shell se le pasa como parámetro CDATA del nodo actual del archivo XML, el cual a través de una instrucción que se ejecuta en consola se obtiene la descomposición gramatical y la cual se envía un archivo de salida TreeTSalida.txt.

     Ejemplo de la descomposición :

Location of Winters, California

Location NN location
of IN of
Winters NP Winters
, , ,
California NP California

- Generar el archivo con el valor base del contenido. Una vez que se descompone el CDATA, el resultado es enviado al archivo TreeTSalida.txt.
- Leer el archivo generado.

8. Buscar en la tabla TOKEN si se cuenta con el respectivo token, sino se cuenta se inserta.

9. Insertar en la tabla T_FRECUENCIA el token de acuerdo al documento, si ya existe registro sólo se actualiza el registro.

10. Crear matriz de frecuencias. Para la obtención de la matriz de frecuencias se parte de la tabla T_FRECUENCIA la cual posee un contenido similar al de la tabla 5.18.

Tabla 5.18: Ejemplo del contenido de la tabla T_FRECUENCIA

id_docto	**id_token**	**frecuencia**	**profundidad**
2	14	107	1
2	11	34	4
2	7	395	4
2	19	10	2
2	6	41	3

Para generar la matriz de frecuencia se necesita un archivo de texto plano el cual tenga como características documentos en las filas y los atributos en las columnas tal como se muestra en la tabla 5.19.

Tabla 5.19: Características del archivo de frecuencias

	atributo1	**...**	**atributoN**
documento1	14	...	1
...	11	...	4
documentoN	19	...	8

Para poder generar un archivo con estas características se realizó un método que implementa un cursor desde Java, el cual tratará los registros de la consulta de manera individual, es decir el cursor se utiliza para el tratamiento individual de las filas devueltas por el Sistema Gestor de Bases de Datos (SGBD), por una consulta.

El pseudocódigo del cursor se estableció de la siguiente manera:

```
 Consulta tabla DOC_COLECC, ordena por id_docto
For each id_docto
        Ejecuta consulta SELECT t. id_token, COALESCE(sum(f.frecuencia), 0) as frecuencia
                  FROM t_frecuencia f  RIGHT JOIN  token t
                  ON t.id_token = f.id_token
                  AND f.id_docto = id_docto
                  GROUP BY t.id_token
                  ORDER BY t.id_token

        For each registro
          Acumula en StringBuffer

        End for
End For
Crea archivo StringBuffer
```

Cabe destacar que se pueden generar 4 tipos de archivos : frecuencia originales, frecuencia estandarizadas por la medida Z, frecuencia normalizada por la medida mínima-máxima y frecuencia normalizada por la medida máxima, detalladas en el marco teórico y tecnológico.

Es importante enfatizar que la matriz de frecuencia que se obtiene no toma en cuenta el nivel o profundidad de los tokens.

11. Obtener la matriz de distancia. El archivo de la matriz de distancia se obtiene apartir del archivo de frecuencias, un ejemplo del archivo de la matriz de distancia es :

```
0.0
1.293917 0.0
3.6133988 4.0542054 0.0
1.8695916 1.9589162 4.730753 0.0
1.4494423 1.9185698 4.1472764 2.788593 0.0
```

Una vez generados los archivos que contienen las matrices de distancias, se pueden utilizar aplicaciones como R o Matlab, las cuales contienen diversos algoritmos de agrupamiento tales como EM-Based o Jerárquico, entre otros. Para ello es útil tener distancias precalculadas; sin embargo se necesita calcular distancias entre vectores de frecuencias y otros puntos del espacio, para poder invocar el algoritmo deseado sobre las diferentes matrices para la obtención de los grupos.

12. Generar el script con las instrucciones en R para poder aplicar los algoritmos a la colección o bien a la muestra.

    Ejemplo del script generado para el algoritmo Jerárquico :

```
read.table("./PGrafico/PGrafico_v2/ArchivoSalida/Frecuencias.txt")->frecuencias;
sink(./PGrafico/PGrafico_v2/ArchivoSalida/Resultados.txt",);
hc<-hclust(dist(frecuencias,method="euclidean"), method="ward");
plot(hc);
rect.hclust(hc,2);
grp<-cutree(hc,2);
write.table(grp,file="./PGrafico_v2/ArchivoConf/Grupos.txt", sep=" ");
```

    Ejemplo del script generado para el algoritmo K-Means :

```
 numGrupos=3
read.table("./PGrafico/PGrafico_v2/ArchivoSalida/Frecuencias.txt")->frecuencias;
sink(./PGrafico/PGrafico_v2/ArchivoSalida/Resultados.txt");
wss <- (nrow(frecuencias)-1)*sum(apply(frecuencias,2,var));
for (i in 2:numGrupos) wss[i] <- sum(kmeans(frecuencias,centers=i)$withinss);
plot(1:numGrupos, wss, type="b", xlab="Número de Grupos",ylab="Suma de cuadrados entre grupos");
fit <- kmeans(frecuencias, 2);
write.table( fit$cluster,file="./PGrafico/PGrafico_v2/ArchivoSalida/GpoKMeans.txt", sep=" ");
```

    Ejemplo del script generado para el algoritmo EM :

```
 library(clv);
library(HDclassif);
sink(./PGrafico/PGrafico_v2/ArchivoSalida/Resultados.txt");
prms<-hddc(frecuencias, K=1, 10, graph=TRUE);
cat(Grupos);
prms$class;
```

13. Ejecutar el script generado con el algoritmo deseado. Script para ejecutar el script en R:

```
#!/bin/bash
sudo -u root
cd PGrafico_v2/ArchivoConf
Rscript clusterXML.R
chmod 777 /PGrafico_v2/ArchivoConf/Rplots.pdf
```

14. Crear archivo con los grupos generados. Ejemplo del archivo generado con los grupos:

```
1
3
4
6
9
```

15. Recuperar los grupos e insertarlos en la base de datos en la tabla DOCUMENTO, en los campos Cluster_Jerarquico y Cluster_EM, respectivamente.

16. Evaluar la calidad de los grupos generados con el índice de Davies-Bouldin y el índice de Dunn.

Mayores detalles de la implementación y su articulación con la aplicación programada se publicaron en un capítulo de libro aceptado en Septiembre del 2011 [Mathieu12], (veáse anexo H, página 174).

Capítulo 6

Pruebas de agrupamiento y resultados

En este capítulo se describe la secuencia de experimentos realizados para evaluar el desempeño de la aplicación desarrollada en este trabajo.

6.1. Pruebas a la colección INEX

6.1.1. Pruebas con una muestra representativa

Trabajar sobre una muestra representativa presenta la ventaja de ahorrar tiempo y requerir recursos de cómputo reducidos; con niveles de confianza y un rango de error mínimo. Es claro que la varianza es una de las principales características que guía el proceso de generación de grupos, esto se verá más adelante al aplicar el algoritmo de K-Means. En caso de que todos los documentos fuesen iguales, es decir, que todos tuviesen los mismos valores para todos sus atributos, se observaría una varianza igual a cero. A medida que esta aumenta las diferencias entre los valores de los atributos de los documentos la varianza aumenta. En el caso de una población muy grande de documentos, con varianza cero para todos sus atributos, sería suficiente estudiar un sólo documento para conocer toda la población. A medida que las diferencias entre los documentos aumentan, la varianza aumenta, pero las diferencias pueden también aumentar entre los atributos, no necesariamente a la misma cadencia. Esto

significa que la varianza resulta diferente entre atributos y entre documentos. A medida que la varianza aumenta, resulta necesario incluir, en el estudio más documentos. Sin embargo, por limitaciones técnicas y materiales no siempre es posible considerar una gran cantidad de documentos y el estudio debe reducirse a el estudio de una o varias muestras. El estudio de las distribuciones de muestreo [Sheaffer87] de algunas cantidades muestrales (como es el caso de algunas estadísticas como el promedio o la varianza) permite conocer algunas propiedades y realizar inferencias de cantidades poblacionales desconocidas (para este caso el total de documentos constituye la población) como el promedio o la varianza. Es claro que entre una muestra y otra las cantidades muestrales, son variables (valores aleatorios), es decir, no siempre se observan los mismos valores, pero la distribución de muestreo de una cantidad como el promedio o la varianza tiende a una distribución teórica. Por ejemplo la distribución de los promedios tiende a una distribución normal. El teorema Central del Límite [Devore01] nos indica que si una variable aleatoria (X) tiene una media poblacional μ y un varianza σ^2, entonces el promedio ($\bar{X}$) de n observaciones de una muestra aleatoria sigue aproximádamente una distribución normal, con promedio μ y varianza $\frac{\sigma^2}{n}$, para un n bastante grande. En la mayoria de los casos no se conoce la varianza poblacional; sin embargo es posible obtener una estimación con la varianza muestral σ^2 con la varianza muestral S^2, representada por la siguiente fórmula :

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1} = \frac{SS}{n-1} \tag{6.1}$$

Para aplicar lo anterior se requiere establecer el tamaño de la muestra y conocer la varianza o al menos tener una idea de su magnitud. Para esto se propone realizar una o varias pruebas piloto, es decir, tomar una o varias muestras aleatorias de pequeño tamaño. En cuanto al parámetro a estimar, en nuestro caso, el número promedio de tokens (atributos) diferentes, es suficiente tener una idea del error máximo aceptado. No existe, *a priori*, ningún procedimiento que indique cuantos individuos se deben extraer en la prueba piloto [Kish75] [1]. Por lo general se obtiene una idea de la varianza por experimentación. El procedimiento se presenta en la siguiente sección.

[1] *A pesar del año de la referencia este es un libro de referencia clásico.*

6.1.2. Experimentos piloto para establecer el tamaño de una muestra representativa de documentos

Se tomó la decisión de realizar 10 experimentos aleatorios piloto cada uno con 20 documentos (hubiese sido posible tomar una sóla muestra de 200 documentos; pero por problemas técnicos y limitaciones de recursos fue más conveniente tomar pequeñas muestras) al azar del total de los documentos de la población de los 659,388. En cada muestra se contabiliza el número de tokens diferentes y se calcula la varianza. Cada prueba dio lugar a una matriz con 20 líneas y M tokens, tal como se presenta en la tabla 6.1, en donde se puede observar en la primera columna los resultados de cada documento para cada experimento, de la segunda columna y hasta la columna onceava se presentan las distintas muestras tomadas en cada experimento. Dado que cada documento presentaba un número de tokens diferentes, se consideró que ésta podría ser la variable para evaluar la variabilidad y determinar el tamaño de la muestra. Cabe aclarar que las muestras tomadas con 20 documentos cada una, no son para caracterizar todos los documentos de la colección, sino más bien para tener una idea de la varianza, y después usar una estimación para restimar un tamaño de muestra más adecuado.

Una vez obtenidas las muestras se aplica la fórmula comunmente usada para el cálculo del tamaño de la muestra, considerando que la población es muy grande :

$$n = \left(\frac{Z_{\alpha/2} * S}{E} \right)^2 \tag{6.2}$$

donde $Z_{\alpha/2}$ es el cuantíl de una distribución normal asociado a un nivel de confianza $(1 - \alpha\,\%)$, S es la desviación estándar muestral del número de tokens diferentes en la muestra de 20 documentos y E es el error. Con los resultados de los 10 experimentos aplicando para cada uno la fórmula anterior con un nivel de confianza de 95 %, con un error de 4.39, (la diferencia entre el promedio de la población y una estimación de este), se procedió a calcular el número de documentos en cada uno de los experimentos pilotos, los cuales se presentan en la la tabla 6.2.

Del estudio de estos diez experimentos se obtuvo un tamaño de muestra promedio de 942.95

Tabla 6.1: Experimentos realizados con veinte documentos.

Resultado	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5	Muestra 6	Muestra 7	Muestra 8	Muestra 9	Muestra 10
1	122	24	38	5	96	192	22	22	29	50
2	48	16	99	17	89	219	59	10	40	41
3	42	52	67	81	21	72	39	28	39	73
4	135	140	42	22	42	208	51	36	16	43
5	174	29	141	354	46	16	58	70	32	26
6	7	33	7	23	160	23	10	70	97	27
7	20	91	27	50	66	27	49	121	22	43
8	28	88	24	63	26	233	17	131	72	61
9	44	59	41	26	17	81	155	52	98	24
10	50	19	59	13	27	33	27	36	13	14
11	53	36	34	31	23	28	222	48	22	60
12	61	26	183	19	78	32	81	23	40	6
13	11	59	58	13	164	73	83	77	83	16
14	66	89	7	59	28	36	209	30	134	29
15	31	311	69	6	61	5	17	98	381	24
16	91	41	50	26	35	59	383	13	71	145
17	75	56	30	19	24	20	81	98	16	43
18	17	37	78	82	18	48	66	23	16	22
19	82	16	80	379	101	32	62	13	13	32
20	316	65	86	70	48	12	55	49	49	62
Total	1473	1287	1220	1358	1170	1449	1746	1048	1283	841
Promedio	73.65	64.35	61	67.9	58.5	72.45	87.3	52.4	64.15	42.5
Desv.Est.	71.59	66.04	43.27	105.04	44.31	75.24	91.12	36.63	82.08	30.15

Tabla 6.2: Experimentos realizados con veinte documentos y un error de 4.39.

	Muestra 1	Muestra 2	Muestra 3	Muestra 4	Muestra 5	Muestra 6	Muestra 7	Muestra 8	Muestra 9	Muestra 10	
$Z_{\frac{\alpha}{2}}$	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	1.95996398	
E	4.39	4.39	4.39	4.39	4.39	4.39	4.39	4.39	4.39	4.39	
n	1021.5905	869.451187	373.20395	82199.10438	391.384721	1128.37127	1654.92361	267.40094	1342.84342	181.251086	942.95250

(943) documentos XML para obtener una estimación del número promedio de tokens diferentes con un 4.39 de error (tabla 6.2). Considerar el error de 4.39 permite disponer de un margen y no sobre pasar de un error de 5.

Si se fija un error de 5 repitiendo los mismos cálculos, al nivel de confianza de 95 %, se obtiene un tamaño de muestra de 726.90, lo que se puede redondear a 727 documentos, el cual es un valor inferior al de 943 documentos. Si se hubiese considerado una distribución t student [Devore01], el cuantíl para un nivel de confianza del 95 % resulta 2.093024; con un error del 4.69 se hubiera requerido una muestra aleatoria de 943 documentos, mientras que con un error de 5 se requeriría tomar 829 documentos.

En este estudio el objetivo es principalmente exploratorio y no pretende construir o confirmar algún modelo por lo que no parece importante establecer un error preciso de (5, 10, etc.). Considerar un error inferior a un máximo, también evita la necesidad de completar la muestra o hacer remuestreos (en caso de problemas técnicos o por la inexistencia del individuo). En estos casos se aconseja tomar un número mayor de individuos, en relación al obtenido en la fórmula 6.2 o aplicar otros esquemas de muestreo [Kish75]. En etapas posteriores a este proyecto quizas sea interesante comparar resultados de agrupación (para una colección de documentos bien específica) aplicando diferentes esquemas de muestreo como el estratificado o por conglomerados, entre otros [Sheaffer87].

Por todo lo anterior el número de 943 documentos a seleccionar en **forma** aleatoria parece ser un valor adecuado, para la fase de evaluación de este proyecto. En etapas posteriores a este proyecto para la aplicación de este programa a una colección bien precisa, sería conveniente evaluar si la distribución de muestreo sigue una distribución normal o no para el promedio de tokens.

6.2. Exploración matriz de frecuencias

6.2.1. Primera exploración

Al aplicar el métodos de agrupamiento Jerárquico con las distintas métricas a los 943 documentos, arrojan resultados de agrupamiento poco interesantes, ya que la mayoría genera dos grupos, lo que implica que todos los documentos son muy homogéneos. Estos documentos

tienen un máximo de 12,543 tokens.

Otro problema que también se observó es la desproporcionalidad de frecuencias entre los diferentes tokens (12,543). Esto genera al aplicar los diferentes algoritmos de agrupamiento medidas de distancia desproporcionadas, generando así pocos grupos.

Para corregir ésta anomalía se procedió a normalizar los datos; sin embargo fue necesario eliminar las columnas con varianza 0. Esto no modificó los resultados de agrupamiento con la matriz no normalizada.

Una exploración más detallada permite observar que la matriz presenta únicamente ceros y unos, es decir, presencias y ausencias. Esta presencia de 0 y 1 hizo pensar que la métrica euclideana no era la más adecuada, por lo que se procedió a aplicar una métrica más apropiada, para datos binarios. La métrica binaria y el índice de similaridad de Jaccard (citados en el Marco teórico y tecnológico), resultan adecuadas para estos datos. De la aplicación de estas métricas con los métodos Jerárquico Simple y Ward, resultan las siluetas presentadas en la figura 6.1.

Al aplicar los índices de Dunn y Davies-Bouldin, para la evaluación de grupos para más detalles ver Apéndice G(página 145), a los distintos algoritmos de de agrupamiento : Jerárquico (métricas Euclideana y Jaccard, métodos simple y de Ward) y K-Means. Para el índice de Dunn se obtienen los resultados mostrados en la tabla 6.3.

De donde resulta que el mejor índice de Dunn es :

$Dunn = 9.657936$

En cuanto al índice de Davies-Bouldin se obtienen los índices presentados en la tabla 6.4:

De donde resulta que el mejor índice de Davies-Bouldin es :

$DB = 0.1035418$

De los dendogramas y de los índices de Dunn y Davies-Bouldin se puede observar que existe un buen agrupamiento en sólo 2 grupos. Estas observaciones sugieren que posiblemente

Tabla 6.3: Índices de Dunn con diferentes grupos.

Método / **Métrica**	**Simple** / **Euclideana**	**Simple** / **Jaccard**	**Ward** / **Jaccard**	K-Means
Grupos				
0	0	0	0	0
2	9.657936	7.1673892	7.0935031	9.657936
10	7.782547	0.1592864	0.1621816	1.6753263
20	6.230406	0.157715	0.11215897	1.6458254
30	5.755786	0.08041927	0.11215897	1.928777
50	4.441091	0.08041927	0.0894719	1.5848341
75	3.904634	0.06326137	0.0894719	1.7040053

Tabla 6.4: Índices de Davies-Bouldin con diferentes grupos.

Método / Métrica	Simple Euclideana	Simple Jaccard	Ward Jaccard	K-Means
Grupos				
0	0	0	0	0
2	0.103542	0.1395208	0.2097001	0.1035418
10	0.119803	0.1405597	0.7578515	0.5527422
20	0.141521	0.3460481	0.7908256	0.6158911
30	0.1525691	0.4945334	0.8373291	0.4374352
50	0.1800001	0.476655	0.8442814	0.5480018
75	0.2019422	0.592513	0.8532114	0.4980223

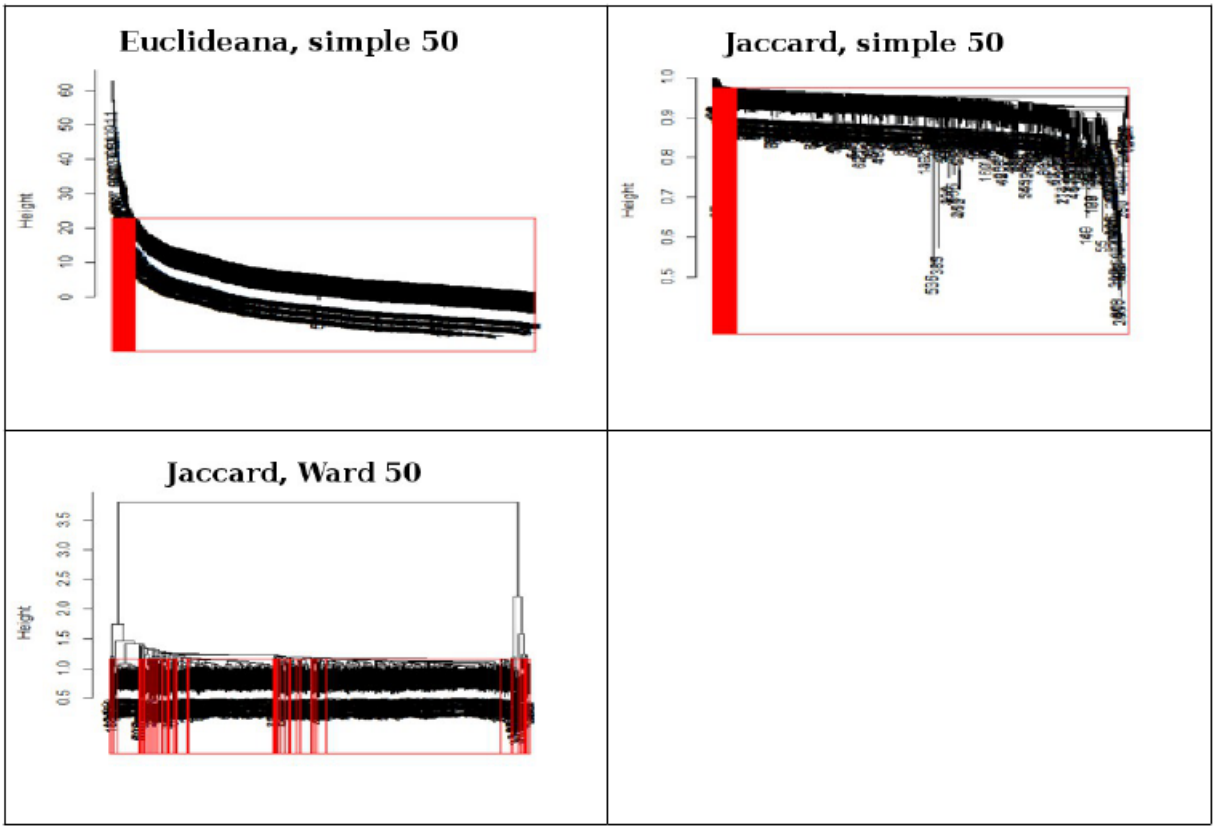


Figura 6.1: Siluetas de las Métricas Euclideana y Jaccard

existía un error en el proceso de generación de la matriz de 943.

Un estudio cuidadoso de la extracción de tokens se observó un problema de descomposición léxica por parte de Tree Tagger, ya que algunos tokens se encuentran en otro idioma distinto al inglés; otro problema era la separación de sílabas (hyphenization, en inglés), estos problemas no sólo afectaban al token en cuestión sino que también afectaban a los tokens siguientes.

6.2.2. Segunda Exploración

Después de la corrección del código se procedió a tomar nuevamente la muestra de 943, en este caso se aplicó el algoritmo de los K-Means y el Jerárquicos con distintas métricas (Binaria, Euclideana, Manhattan, Máxima y Minkowski) y métodos (Promedio, Centroide,

Completa, Mediana, Simple y Ward).

Sin embargo estos métodos requieren de conocer el número de grupos conocidos. Por lo que se hace necesario tener una estimación. Una opción es analizar la varianza, la cual está íntimamente relacionada con la suma de las diferencias de cuadrados (SS) y que se encuentra definida por la siguiente fórmula :

$$s^2 = \frac{\sum_{i=1}^{n} (X_i - \bar{X})^2}{n-1} = \frac{SS}{n-1}$$

Un experimento consiste en evaluar la SS para diferentes agrupamientos. Esto da una idea de la variabilidad dentro de cada grupo a medida que aumenta el número de grupos. Con una varianza grande se espera obtener en un sólo grupo y esta varianza puede disminuir a medida que el número de grupos aumenta, pues en teoría los grupos se hacen más homogéneos. Para visualizar la evolución de la varianza dentro de cada grupo, se realizó un experimento de agrupamiento usando el algoritmo K-Means. El K-Means es un algoritmo de agrupamiento por particiones. Para iniciar el algoritmos se requiere un número de grupos conocido (**k**). Cada grupo tiene asociado un centroide (centro geométrico del grupo). En seguida los objetos se asignan al grupo cuyo centroide esté más cerca (utilizando cualquier métrica de distancia, en nuestro caso la euclidiana). Iterativamente, se van actualizando los centroides en función de las asignaciones de puntos a grupos, hasta que los centroides dejen de cambiar. La aplicación de este procedimiento para un número inicial de grupos de 2 a 50, produce los resultados presentados en la figura 6.2. En dicha figura se pueden observar las discontinuidades de la curva, lo que sugiere el número de grupos posibles, en los cuales se pueden particionar la muestra de documentos con este criterio.

- En la figura 6.2, se observa como disminuye la varianza, con ciertas discontinuidades alrededor de 7, 20, 35, 40. Estos puntos permiten dar una idea para estimar el número de grupos. Sin embargo la varianza esta lejos de ser pequeña. Por lo que se repitió el experimento de 1 a 150, produciendo los resultados mostrados en la figura 6.3.

- En la figura 6.3, se observan las mismas discontinuidades con menos detalle, pero después, a medida que el número de grupos aumenta se observa una curva decreciente con pocas discontinuidades. En esta última gráfica se observa una discontinuidad, de la varianza, alrededor de 25 y una estabilización de la varianza alrededor de 100. Por

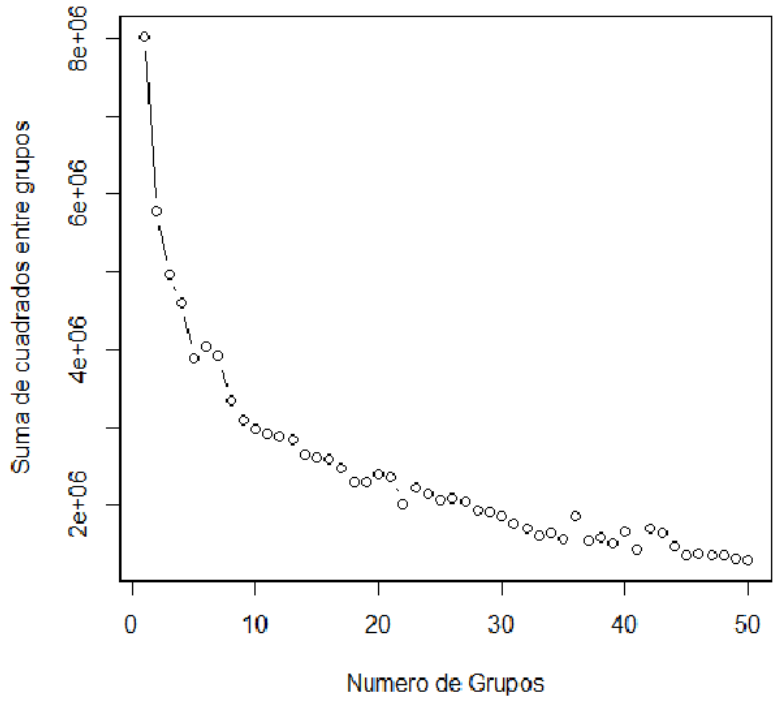


Figura 6.2: Varianza, con ciertas discontinuidades alrededor de 7, 20, 35, 40.

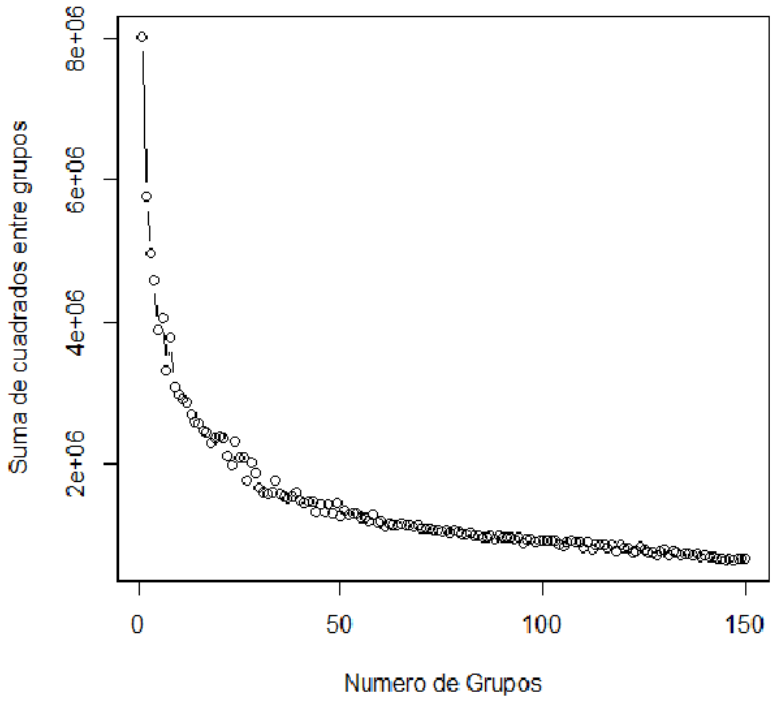


Figura 6.3: Varianza, con ciertas discontinuidades alrededor de 1 a 150.

estas razones se decidió aplicar el algoritmo Jerárquico con las distintas métricas y evaluar los resultados para 25 y 100 grupos. De lo anterior se obtiene que el mejor dendograma es el que aplica una métrica Binaria con método Ward, debido a que dicho dendograma presenta una distribución mucho más uniforme en la distancia de los elementos en cada grupo. El dendograma se puede observar en la figura 6.4.

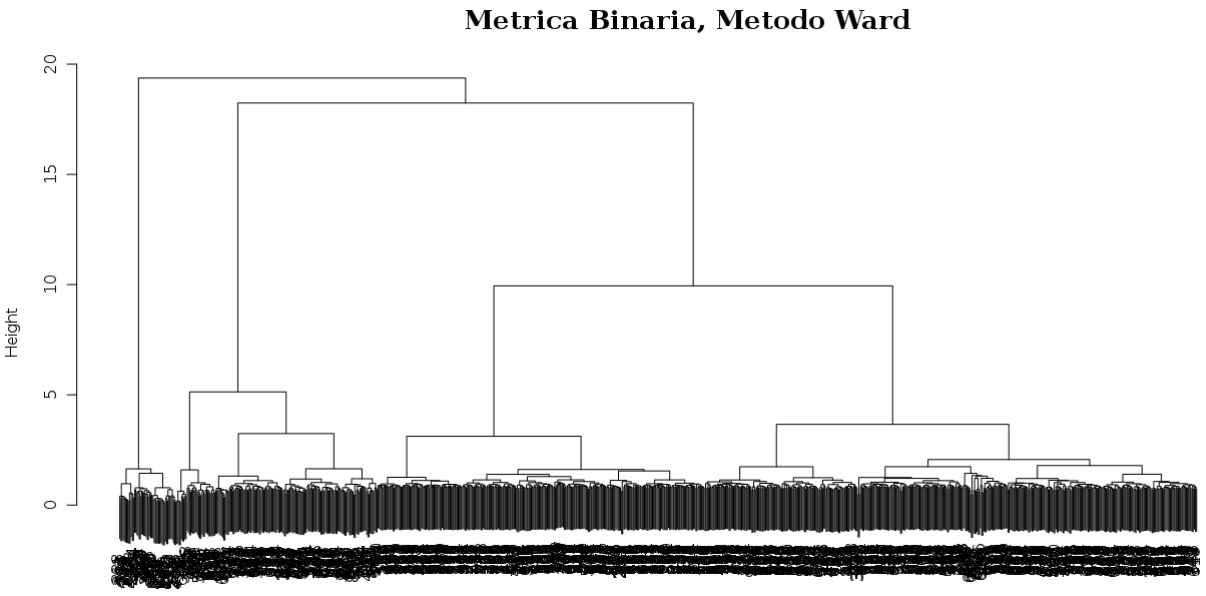


Figura 6.4: Dendograma de la métrica Binaria con el método Ward.

Los dendogramas con la evaluación del método jerárquico con la combinación de las distintas métricas, se pueden apreciar en el apéndice D .

Pruebas preliminares con una muestra aleatoria de 727 documentos, es decir, con un error de 5 muestran mínimas diferencias en los dendogramas obtenidos aplicando el método Ward con métricas Binarías y Jaccard.

6.2.3. Tercera Exploración

En esta exploración se procedió a comparar los índices de Dunn y Davies-Bouldin con diferentes los diferentes métodos y métricas aplicadas. A continuación se presentan los resultados de la evaluación de grupos usando el índice de Davies-Bouldin, para el algoritmo

Jerárquico con métrica Euclideana y estrategia Simple, con métrica de Jaccard y estrategia simple, con métrica de Jaccard y estrategia Ward y el método de los K-Means. El algoritmo de los K-Means resulta adecuado para 30 grupos, mientras que el algoritmo Jerárquico con estrategia de Ward y métrica de Jaccard resulta adecuado para 50 grupos. En los otros casos no se observa con claridad un mínimo.

Índice de Dunn

En la tabla 6.5 se presentan los índices de Dunn probado con 25 (inferior de la diagonal) y 100 (superior de la diagonal) grupos.

Una vez obtenido los distintos índices de Dunn para 25 y 100 grupos se procedió a obtener el promedio de ambos índices, por lo que resulta que el mejor promedio es aquel que se encuentra la celda sombreada. El índice de Dunn indica que el mejor método es la métrica Euclideana con el método Simple.

Índice de Davies-Bouldin

En la tabla 6.6 se presentan los índices de Davies-Bouldin probado con 25 (inferior de la diagonal) y 100 (superior de la diagonal) grupos. La celda sombreada indica cual es el mejor método y métrica aplicada a las diferentes pruebas realizadas.

Una vez obtenido los distintos índices de Davies-Bouldin para 25 y 100 grupos se procedió a obtener el promedio de ambos índices, por lo que resulta que el mejor promedio es aquel que se encuentra la celda sombreada. El índice de Davies-Bouldin indica que el mejor método es la métrica Binaria aplicada a los centroides.

Los resultados de esta exploración fueron presentados en el congreso SIMMAC 2012 (veáse anexo H, página 188). De aquí resultó interesante proponer una secuencia para la validación (figura 6.5, es decir, un conjunto de valores para distintos grupos y los algoritmos jerárquico con distintas métricas y métodos y el K-Means.

6.2.4. Cuarta exploración con el algoritmo de Expectación Máxima (EM)

Hay varias características que se deben considerar para aplicar este algoritmo. La primera es que supone que los documentos a tratar (en nuestro caso las 943 líneas) son independientes en un espacio R^p con una función de probabilidad o función de densidad :

Tabla 6.5: Índices de Dunn con 25/100 grupos respectivamente

Método / Métrica	Promedio	Centroide	Completo	Mediana	Simple	Ward
Binaria	0.3436629 / 0.11545614	2.179974 / 1.41625051	0.31674136 / 0.13353120	1.2394903 / 0.3448177	0.15898945 / 0.1561128	0.16295050 / 0.053462281
Euclideana	1.9246941 / 1.5875832	3.534545 / 1.412149	1.9761196 / 1.8255741	1.1594803 / 1.4257094	3.352225 / 1.964474	0.24957998 / 0.4529342
Manhattan	0.24957998 / 1.754071	3.534545 / 1.412149	0.6462828 / 0.9106829	2.947874 / 1.740708	2.426437 / 1.669278	0.19331574 / 0.6052401
Máxima	1.2585949 / 1.1969942	1.0487288 / 0.5827674	1.3226230 / 1.1951831	1.0791384 / 0.6880001	1.832139 / 1.089629	0.15365607 / 0.31500980
Minkowski	1.9246941 / 1.5875832	3.534545 / 1.412149	1.9761196 / 1.8255741	1.1594803 / 1.4257094	1.964474 / 1.964474	0.24957998 / 0.4529342

Tabla 6.6: Índices de Davies-Bouldin con 25/100 grupos respectivamente

Método / Métrica	Promedio	Centroide	Completo	Mediana	Simple	Ward
Binaria	0.01022427 / 0.6447279	**0.08447805 / 0.07194929**	0.800528 / 0.5422784	0.09644692 / 0.07081217	0.4784591 / 0.4085802	0.8115281 / 0.5215877
Euclideana	0.3918206 / 0.3979756	0.3258379 / 0.1150065	0.4593343 / 0.4628951	0.3129204 / 0.3114767	0.1619389 / 0.1098059	0.5647646 / 0.6334044
Manhattan	0.1196621 / 0.6334044	0.1199047 / 0.09424246	0.3926387 / 0.4825574	0.1179242 / 0.09461055	0.1185917 / 0.09561787	0.4439309 / 0.5477697
Máxima	0.4482546 / 0.4242595	0.3532889 / 0.4016776	0.5165448 / 0.5133866	0.3329509 / 0.3654624	0.1185917 / 0.09561787	0.4439309 / 0.5477697
Minkowski	0.3918206 / 0.3979756	0.3258379 / 0.1150065	0.4593343 / 0.4628951	0.3129204 / 0.3114767	0.1619389 / 0.3114767	0.5647646 / 0.6334044

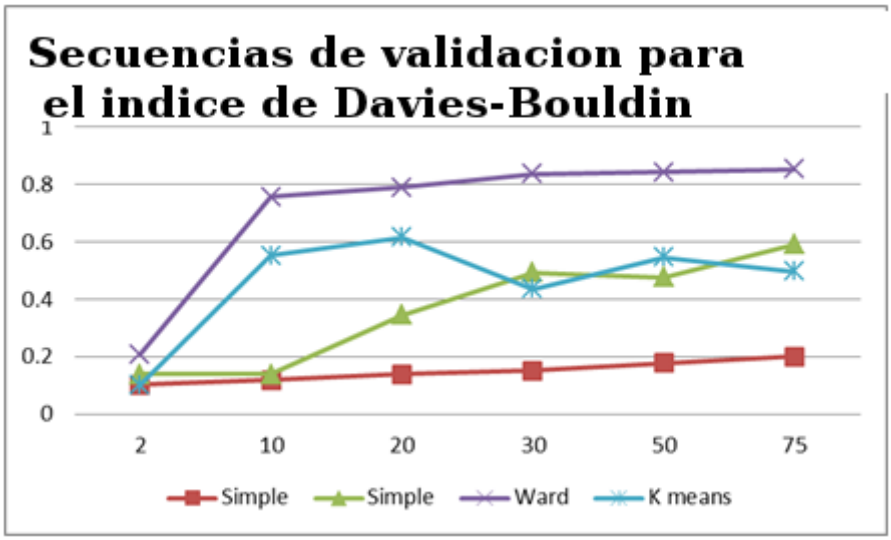


Figura 6.5: Secuencias de validación para el índice de Davies-Bouldin.

$$f(x,\theta) = \sum_{k=1}^{K} p_k \phi(x;\mu_k,\Sigma_k) \tag{6.3}$$

donde p_k es la proporción en la mezcla del componente k-ésimo y es la función de densidad con parámetros $(\mu_k, \sum_k)$, es decir, el promedio de tokens diferentes y con la matriz de covarianza.

El objetivo es estimar el número de componentes, su proporción, los promedios y la matriz de covarianzas. Para la aplicación del algoritmo EM se tienen problemas de cálculo del número de grupos, ya que el número de atributos es muy grande (número de variables "d" es mayor al número de individuos N). Es decir, que no es posible aplicar directamente el algoritmo de EM, tal como se presenta en la sección 4.6.3 y apéndice C, pues la solución inicialmente propuesta por Berge y Fraley funciona bien para el caso en que el número de líneas es mayor al número de columnas. En nuestro caso podríamos aplicar este algoritmo de EM, si tuviésemos al menos 16271 documentos, ya que cada uno contiene un máximo de 16271 columnas. Este problema ya ha sido considerado en otras investigaciones por [Berge12]. La solución propuesta es realizar particiones de la matriz para obtener los valores propios. A condición de que existan subespacios con una dimensión inferior a d (el número

de atributos). Del estudio exploratorio de nuestros datos esto parece ser el caso ya que hay muchas particiones con ceros y unos.

La búsqueda de estos subespacios y la estimación de los parámetros esta fuera de nuestros objetivos, ya que se desea estimar el número de componentes y asignar a cada documento al componente más adecuado. Afortunadamente una aplicación reciente ([Berge12]) propone una solución a este problema de estimación. Esta publicación propone el programa (de libre acceso) llamado HDclassif el cual se puede integrar en R.

Es importante señalar que la matriz de frecuencias 943 documentos y 16,271 variables son frecuencias que resultan de la contabilización de atributos nominales. Para este tipo de datos resulta adecuado aplicar métricas de tipo presencia/ausencia como las métricas de Jaccard, Binaria o muchas otras (consultar Marco teórico y tecnológico). Un estudio comparativo usando estas diferentes distancias es presentado por Finch ([Finch05]). Para este tipo de datos el algoritmo EM aplicado para la estimación de parámetros de una mezcla de modelos gaussianos, resulta poco adecuado ya que este fue desarrollado para datos de tipo intervalo y de razón.

Lo anterior significa que los resultados del algoritmo EM deben tomarse con reservas. A continuación se presenta su aplicación, con ayuda del paquete HDClassif a la matriz muestra.

Ejemplo de aplicación del algoritmo EM a la matriz muestra.

EL algoritmo HDClassif permite descomponer la matriz de covarianzas en una forma similar Fraley [Fraley06]

Los detalles del funcionamiento de esta biblioteca se encuentran en la referencia [Berge12].

Al aplicar el program HDclassif a la matriz muestra, sin ningún tratamiento se obtienen los resultados de la tabla 6.7. La primera columna muestra el modelo, la segunda columna muestra el número de grupos y la tercera columna muestra el valor del Criterio de Información Bayesiano (BIC, por sus siglas en inglés), tal como fue definido en la sección 4.6.3.

Para saber cual es el número de grupos indicados se toma el mínimo valor positivo del BIC, para nuestro ejemplo BIC = 5218894, lo que permite concluir que los documentos se agrupan en tres grupos.

Tabla 6.7: Modelo K BIC

Modelo	**Núm. Grupos**	**Valor BIC**
All	1	-24254933
AKJBKQKDK	2	-1948671
AKJBKQKDK	3	5218894
AKJBKQKDK	4	8383620
AKJBKQKDK	5	STOPPED
AKJBKQKDK	6	STOPPED
AKJBKQKDK	7	STOPPED
AKJBKQKDK	8	STOPPED
AKJBKQKDK	9	STOPPED
AKJBKQKDK	10	STOPPED

De acuerdo al criterio BIC el modelo más adecuado resulta el modelo de forma esférica.
con K= 3 agrupaciones.
Se puede visualizar en la figura 6.6, como va variando el umbral con respecto a las iteraciones.
Cada grupo representa las siguientes proporciones :

Grupo 1 30.8 %
Grupo 2 53 %
Grupo 3 16.2 %

Es decir, grupo 1 :

500 documentos

Grupo 2 : 2 290 documentos
Grupo 3 : 153 documentos

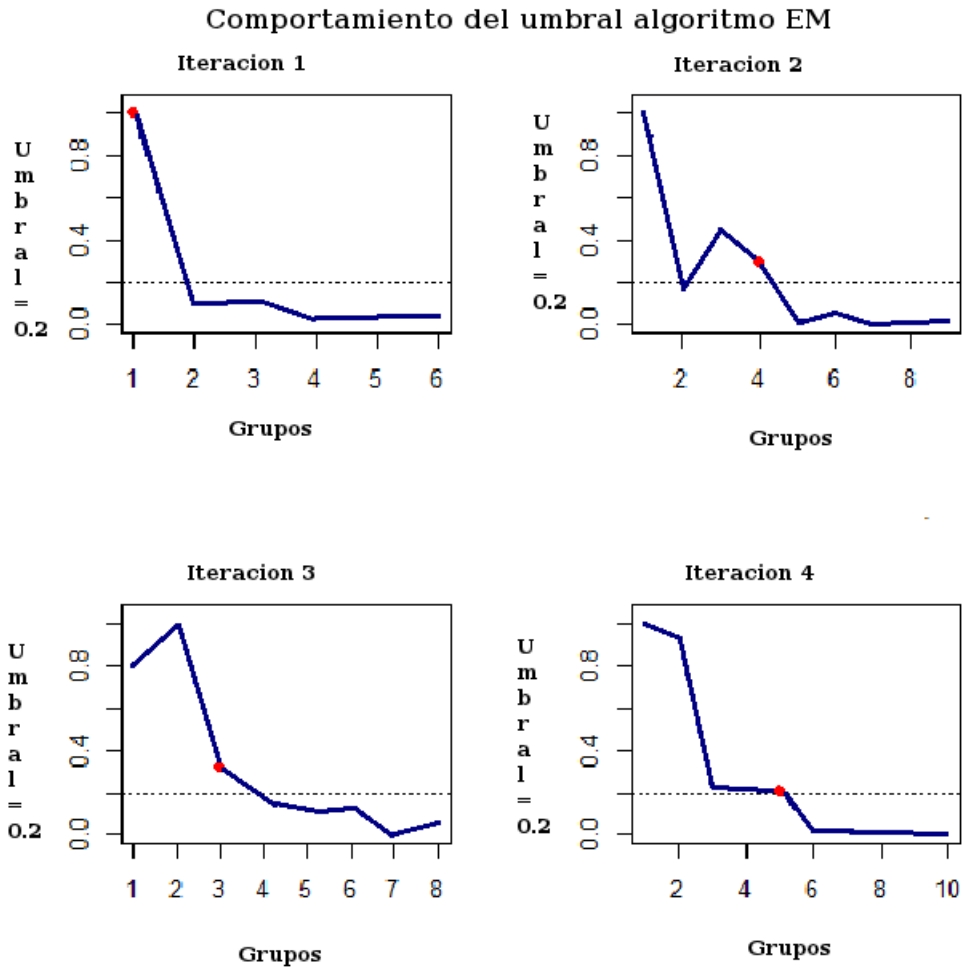


Figura 6.6: Umbral al aplicar las iteraciones EM

Con este algoritmo, a cada documento se le asignan cada una de las 3 clases de la siguiente manera :

```
  [1] 1 2 2 2 2 1 1 1 2 1 1 2 2 1 2 2 2 2 1 1 2 2 2 2 1 1 2 2 2 2 2 2 1 2 2 2 1
 [38] 2 1 1 2 1 1 3 1 1 1 1 2 1 1 1 1 3 1 2 1 1 1 1 2 1 1 1 2 1 1 1 1 1 1 3 1 2
 [75] 1 1 3 2 1 1 1 3 2 3 1 3 1 1 3 1 1 2 1 1 2 1 3 1 3 3 1 1 1 3 3 1 3 3 3 3 3
[112] 1 1 3 1 2 3 1 3 3 1 1 3 3 1 3 1 1 2 1 3 3 3 1 1 2 3 1 1 1 1 1 1 2 2 1 2 2
[149] 1 1 3 2 2 1 2 2 2 1 2 2 2 1 1 2 3 1 2 2 1 2 2 3 2 1 2 1 1 1 3 2 1 1 2 3 3
[186] 3 1 1 1 1 2 1 1 3 3 1 1 1 1 1 1 1 1 1 3 3 2 3 3 1 1 1 1 1 1 1 3 1 1 2 2 3
[223] 1 1 1 3 3 1 1 3 2 2 2 2 1 1 1 2 2 1 2 2 2 2 2 2 2 2 2 2 2 3 2 3 2 2 1 2 1
[260] 2 2 3 2 2 2 2 2 2 3 3 2 1 3 1 1 3 2 1 1 3 1 1 1 1 1 3 2 1 1 1 2 3 1 2 1 1
[297] 2 2 1 1 2 1 1 2 1 1 2 1 3 2 2 1 2 1 2 2 2 1 1 2 2 1 3 2 3 2 1 3 1 1 1 1 1
[334] 3 1 1 3 2 2 2 1 1 1 2 3 1 2 2 1 2 3 2 3 1 2 2 2 1 1 1 3 2 3 3 3 1 1 3 1 2
[371] 2 2 1 1 3 3 1 2 2 1 2 2 2 1 2 1 3 1 1 3 1 1 2 1 1 1 2 2 3 1 1 2 1 1 1 1 2
[408] 1 1 1 2 1 1 2 1 3 1 3 3 1 1 2 1 1 1 1 2 3 1 1 2 1 2 2 2 2 2 1 2 3 1 1 1 3 1
[445] 1 3 3 3 3 1 2 1 2 3 1 1 1 1 1 3 3 1 1 3 3 1 1 1 1 3 1 3 3 1 1 1 1 2 3 3 1
[482] 1 1 1 1 3 2 1 1 1 3 1 1 3 1 1 3 2 1 1 2 1 1 3 3 1 1 1 1 1 1 1 2 2 2 1 2 1
[519] 1 1 1 1 1 1 2 2 1 1 1 1 2 2 2 3 1 1 3 1 1 3 1 1 2 1 1 1 1 1 1 2 1 1 1 1 1
[556] 1 1 3 1 1 3 1 3 1 2 2 1 2 2 2 2 3 1 1 3 1 2 2 1 1 2 1 2 2 1 1 1 1 1 1 1 1
[593] 1 2 3 1 1 1 2 2 1 3 1 3 1 3 2 1 2 3 1 1 3 1 1 1 3 1 2 1 1 1 1 3 1 2 3 1 1
[630] 1 2 1 1 2 2 2 1 1 3 2 2 1 1 1 3 1 1 1 2 2 1 1 2 1 2 1 2 2 1 1 3 1 1 2 2 2
[667] 2 1 2 1 1 2 2 1 1 2 1 1 2 1 3 1 1 1 1 1 1 1 2 2 1 2 1 2 1 1 1 2 1 1 3 1 1
[704] 1 1 1 2 3 2 3 2 2 1 1 1 3 1 1 3 1 2 1 1 1 1 1 2 1 1 3 3 1 1 1 1 1 2 1 1 3
[741] 2 1 2 1 1 2 2 2 2 2 2 2 1 2 3 1 1 1 2 1 1 2 1 2 1 1 2 2 1 2 1 1 1 2 1 1 2
[778] 2 1 2 1 2 1 1 2 2 3 2 2 3 1 1 2 2 1 2 1 1 2 2 1 3 2 2 2 3 1 1 2 2 2 2 3 1
[815] 2 1 1 2 1 3 1 3 1 2 1 1 1 1 3 2 2 2 1 1 1 1 2 1 3 2 1 1 3 3 1 1 1 1 1 1 1
[852] 1 2 1 2 2 1 2 1 3 2 1 1 3 3 1 3 1 3 2 1 2 1 1 2 1 1 2 1 1 1 1 1 2 2 2 2 1
[889] 1 1 1 1 2 2 2 1 2 2 1 2 1 1 2 1 1 1 1 1 1 3 1 1 1 2 1 1 1 2 3 1 1 1 1 2 2
[926] 3 1 3 2 1 1 1 2 2 1 1 1 1 2 2 1 1 3
```

Por ejemplo el documento 524 se asigna al grupo 1.

La distribución de documentos, se observa en la figura 6.7 :

Aplicando los criterios de validación de Dunn y Davies-Bouldin resulta, la tabla 6.8, donde se presenta el índice de Dunn con las diferentes estrategias.

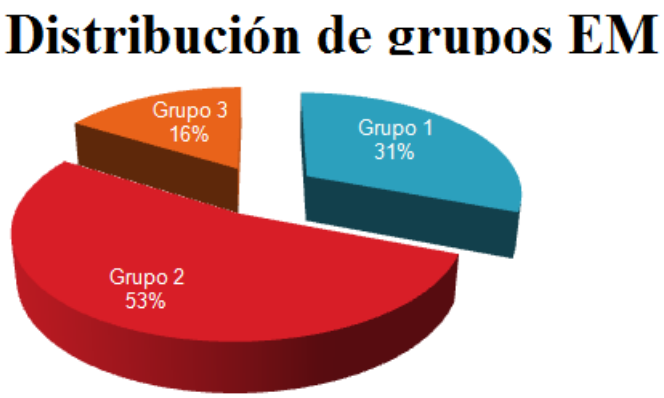


Figura 6.7: Distribución de documentos en los grupos

Tabla 6.8: Índice de Dunn

	Completo	**Promedio**	**Centroide**
Simple	0.01098765	0.04863769	0.07189888
Completo	0.30534615	1.35163826	**1.99806508**
Promedio	0.05386015	0.23841609	0.35243962
Centroide	0.02568741	0.11370729	0.16808829

Mientras que en el caso del índice de Davies-Bouldin los resultados se presentan en la tabla 6.9.

Tabla 6.9: Índice de DaviesBouldin

	Completo	**Promedio**	**Centroide**
Simple	42.022852	9.2768866	6.4000263
Completo	1.463802	0.3054058	**0.2102435**
Promedio	7.993999	1.6482565	1.1475071
Centroide	15.593353	3.2050827	2.2357455

Para mayores detalles consultar Apéndice G(página 145), comparando con los resultados de la tabla 6.3 (página 88), se observa que la calidad de esta agrupación es baja, sobre todo comparada con el algoritmo Jerárquico con el método de Ward y las métricas de Jaccard y Binaria. Esto quizás se debe a que EM es el menos adecuado para agrupar datos discontinuos, como ya se expuso anteriormente.

Se observan diferencias importantes entre los resultados de la tercera exploración con el algoritmo Jerárquicos de dónde resultan entre 30 y 50 grupos, los resultados de la cuarta exploración con el algoritmo de EM, dónde resultan 3 grupos. De aquí la importancia de tener claridad en cuanto al objetivo del agrupamiento y a la necesidad de establecer criterios de eficiencia y eficacia adecuados (consultar capítulo 4 en sección 4.8). Para esto, aún en la actualidad, la colaboración directa del usuario, o de un experto (gold estándar

[Pinto Avendaño09]) es importante, aunque sin dejar de guiarlo en el proceso de búsqueda. De lo anterior se concluye que el proceso de búsqueda de grupos esta lejos de ser automático, excepto para objetivos bien definidos. En nuestro caso, más que desarrollar una aplicación específica para una colección de datos, el interés principal fue el de ofrecer un prototipo modular y flexible que permita la exploración y estudio de datos continuos con distintos métodos para un máximo de interacción con los datos. La figura 6.8 presenta una síntesis de la secuencia seguida en este capítulo :

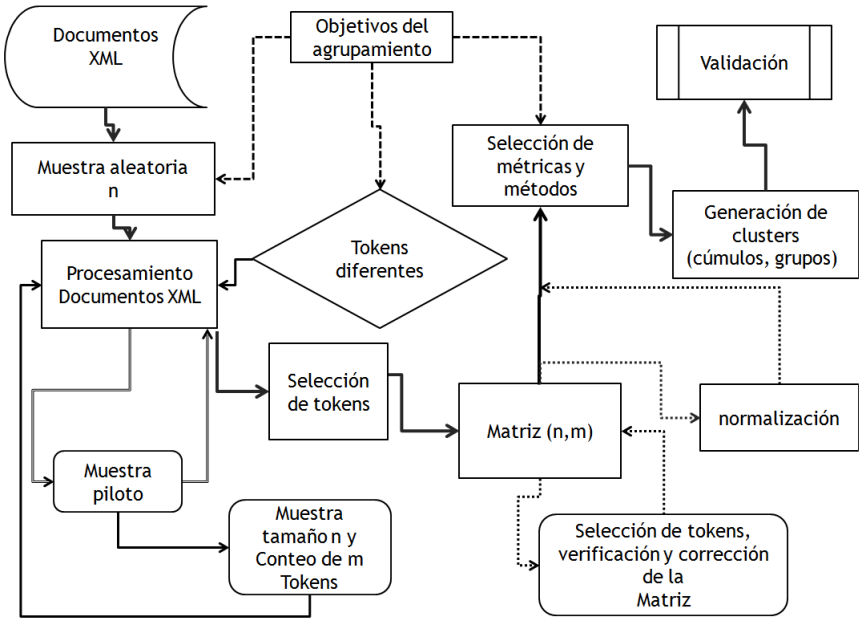


Figura 6.8: Proceso de validación

Capítulo 7

Conclusiones y perspectivas

En este capítulo se presentan los principales resultados de esta propuesta, las principales aportaciones y se proponen algunas vías para continuar su desarrollo.

7.1. Conclusiones

Durante el presente proyecto de investigación se cumplieron los objetivos establecidos en la propuesta de tesis, logrando obtener los siguientes resultados :

Se presentó la metodología, diseño e implementación de una aplicación configurable para la extracción de información contenida en documentos XML; así como para la agrupación de documentos en una colección. El diseño se especificó basándose en los distintos diagramas de UML. Es importante señalar que esta aplicación fue diseñada bajo el modelo espiral, debido a esto fue necesario hacer distintas modificaciones a nivel de las clases propuestas y del modelo relacional planteado al inicio del proceso.

En cuanto a la generación de una matriz de frecuencia, se eligió un Sistema Gestor de Base de Datos (SGBD) para el manejo de los documentos y tokens. Esto además de permitir un ahorro importante en recursos de almacenamiento, facilita el manejo y la extracción de documentos y atributos.

La aplicación ofrece la posibilidad de usar diferentes algoritmos, con diferentes estrategias y métricas, permitiendo tratar colecciones de documentos muy diversos. Por lo que se puede decir que el prototipo es una aplicación genérica, para minar colecciones XML en idioma

inglés con distintos algoritmos de agrupamiento; sin embargo no se aplicó en solucionar algún problema específico, por lo que sus aplicaciones y usos pueden ser muy diversos.

Al aplicar los algoritmos de agrupamiento se puede decir que : el algoritmo EM no es adecuado para agrupar datos discontinuos, ya que para este caso sólo produce 3 grupos, arrojando mucha variabilidad de documentos en los grupos obtenidos. Los algoritmos que mejor se adecuan son el algoritmo Jerárquico y K-Means con grupos sugeridos de 20, 30 y 40.

Esta aplicación no funciona en tiempo real ya que los documentos no son procesados en forma continua "on the fly" es decir en sincronía con la llegada de nuevos documentos a la colección. Cabe mencionar que las aplicaciones de minería se aplican una vez que se obtienen las colecciones XML.

La aplicación es un prototipo y nunca se propuso una aplicación terminada sino explorar algunas posibilidades con el fin de proponer una metodología de tratamiento de colecciones de documentos XML. En este caso, trabajar en forma asíncrona (en lotes) parece una ventaja ya que no se requiere tener comunicación continua con la fuente de documentos y facilita el proceso de experimentación y evaluación de resultados de esta aplicación.

Por supuesto que una vez realizadas las pruebas para una colección determinada, es posible evaluar la posibilidad de encapsular la aplicación y hacerla funcionar en forma continua en sincronía con la llegada de nuevos documentos a la colección.

Aunque no se presenta la noción de agrupación parcial, la aplicación puede realizar el agrupamiento con toda la colección o con una muestra representativa. Los resultados de agrupamiento con una muestra pueden ser usados como una base de entrenamiento para algoritmos de clasificación como KNN (ver apéndice H) y la aplicación de este procedimiento puede servir para clasificar los nuevos documentos a medida que se integran en la base.

7.2. Perspectivas

Para la generación de tokens de contenido, la aplicación funciona sólo para contenidos definidos en el idioma inglés. Una posibilidad sería ampliar esta aplicación para otros idiomas, en particular para el español.

En principio esta aplicación funciona independientemente de la disciplina de aplicación. Sin embargo sería interesante realizar pruebas con colecciones de documentos en diferentes disciplinas, pues parece claro que los contenidos de documentos se estructuran en distinta forma dependiendo de la disciplina. Quizás en ciertas disciplinas se encontrarían contenidos poco estructurados (más planos) y en otras disciplinas contenidos estructurados con diferentes niveles.

En términos de tiempo de procesamiento, la aplicación requiere importantes recursos en tiempo para extraer la información y generar las tablas. Los cuellos de botella se encuentran principalmente en la fase de descomposición que realiza TreeTagger y en la inserción en la base de datos. Sin embargo, el prototipo puede aplicarse a una muestra representativa.

Aunque cada uno de los módulos funciona, se requiere hacer más pruebas para evaluar su funcionamiento en forma integral. En algunos casos quizás sea necesario generar algunos archivos de configuración para que la aplicación pueda continuar, sin necesidad de recomenzar con todos los tratamientos en curso. También es deseable probar la aplicación con otras colecciones conocidas y realizar pruebas de validación.

Otro aspecto a considerar es la unificación del código en Java para evitar problemas de interoperabilidad.

A través de las diferentes perspectivas se puede apreciar que existen extensiones pendientes para mejorar y probar la aplicación.

Apéndice A

Diagramas de casos de uso

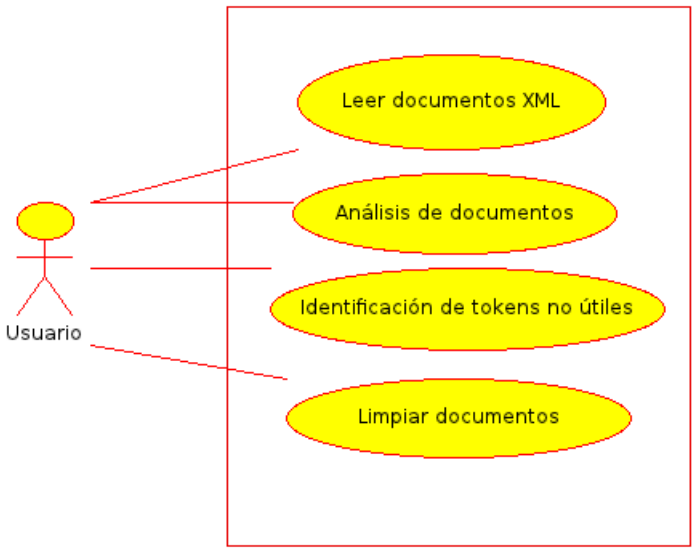


Figura A.1: Caso de uso del módulo de preprocesamiento y análisis de documentos XML

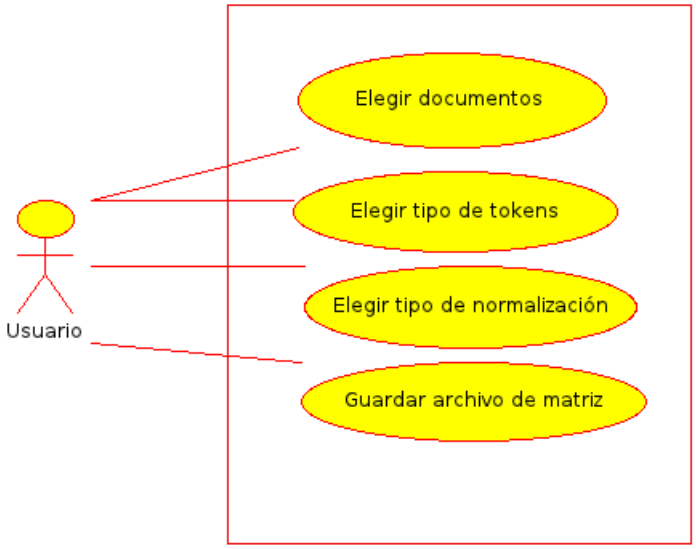


Figura A.2: Caso de uso del módulo de generación de matrices de frecuencias y normalización

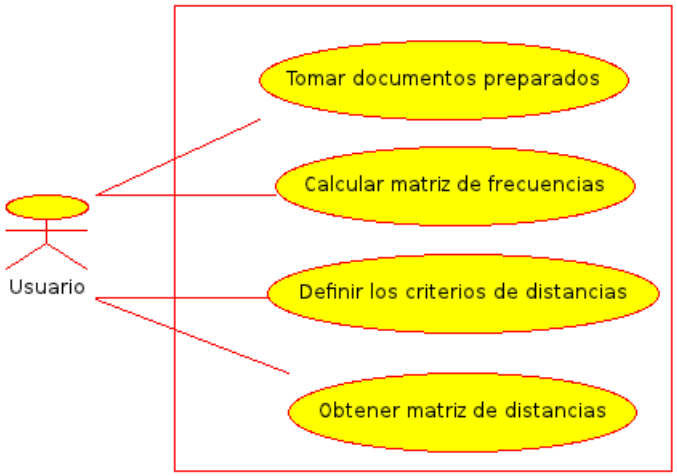


Figura A.3: Caso de uso del módulo de generación de matrices de frecuencias y cáculo de distancias

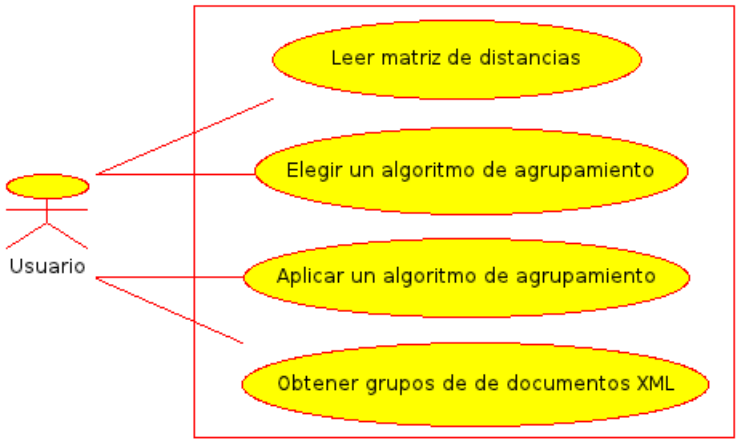


Figura A.4: Caso de uso del módulo de algoritmos de agrupamiento

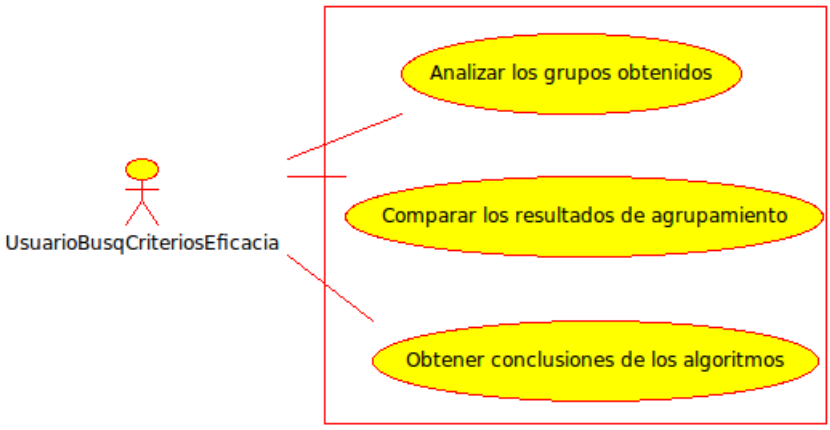


Figura A.5: Caso de uso de búsqueda de criterios de eficacia

Apéndice B

Implementación de algoritmos

B.1. Algoritmo Máxima Expectación (EM-Based)

El presente algoritmo de Máxima Expectación fue programado en SCILAB Dónde se considera que N representa el número de documentos de la colección, K número de agrupamiento, X_i el documento $i = 1, .., N$, concretamente X_i es un vector normalizado; M_j y M_bis_j con j=1, K el centro y el nuevo centro del grupo C_j, $j = 1, ..., K$, es un vector del mismo tamaño que X_i.

```
for i = 1:N
for j = 1:K
dist(i,j) = norm(X(i,:) - M(j,:));
end;
end


//calculo de la probabilidad de pertenencia :

for i = 1:N
for j = 1:K
px(i,j) = exp(-dist(i,j)^2/(2*sigma(j)));
end
end


//calculo de pxc = P(xi in Ck) :

for i = 1:N
som(i) = sum(px(i,:));
end

for i = 1:N
pxc(i,:) = px(i,:) / som(i);
```

```
end


for j = 1:K
som_pxc(j) = sum(pxc(:,j));
end

//Calculo de los nuevos centros :
for j = 1:K
for ll = 1:LL
M_bis(j,ll) = X(:,ll)'* pxc(:,j) /som_pxc(j);
end
end

//calculo de los nuevos valores de sigma :
for j = 1:K
s=0;
for i=1:N
s = s + norm(X(i,:) - M(j,:))^2*pxc(i,j);
end
sigma(j) = s/som_pxc(j);
end;


norm(M-M_bis)
M =M_bis
sigma
```

B.2. Algoritmo Jerárquico

El presente algoritmo fue programado en MATLAB, para poder comprender el algoritmo jerárquico aglomerativo.

```
clc;
clear;
close all;


X = [0 0;4 0;0 1;0 3;3 1];

Verb=0;

[n a]=size(X);
s = (n*(n-1))/2;
T = zeros(1,s);
T = nan(n,n);
%c=0;

% Calculo de distancias
 for i=1:n
     for j=i+1:n
         tmp=(X(i,1:a)-X(j,1:a));
         T(i,j)=(tmp*tmp');
     end
 end
```

```
if Verb==1
 disp(' ');
 disp (T);
end

gc= 1;
mn=0;
while isnan(mn)==0
    mn=min(min(T));
    [x tmn]=size(T);
    for i=1:tmn
        for j=i+1:tmn
            [x tmn]=size(T);
            % encuentra par con distancia minima <-- proceso central
            if T(i,j)==mn
              for l=1:tmn
                  % revisa el par con distancia minima
                  if l<i
                      tmp1=T(l,i);
                  end
                  if l>i
                      tmp1=T(i,l);
                  end
                  if l<j
                      tmp2=T(l,j);
                  end
                  if l>j
                      tmp2=T(j,l);
                  end
                  %genera la columna del nuevo grupo
                  %si es un elemento agrupado, se pasa por alto
                  if l==i || l==j || l==n+gc
                      T(l,n+gc)=nan;
                      T(n+gc,:)=nan;
                  else
                      T(l,n+gc)=min(tmp1,tmp2);
                  end
              end
              %genera un nuevo elemento en la tabla de jerarquias
              g(gc,:)=[i j mn];
              T(:,i)=nan;
              T(i,:)=nan;
              T(:,j)=nan;
              T(j,:)=nan;
              gc=gc+1;
              if Verb==1
                  disp(T);
                  disp(g);
              end
            end
        end
    end
end

H=dendrogram(g);
title('Dendrograma');
```

Apéndice C

Ejemplo del algoritmo de Expectación Máxima en R

Con el objeto de aclarar algunas dudas sobre el procedimiento de estimación de los parámetros, de acuerdo a las formulas presentadas (ver sección 4.6.3) se realizó un programa para la mezcla de dos gaussianas unidimensionales.
Primero se genera una mezcla aleatoria de 1000 observaciones con distribución gaussiana.

```
X = NULL
X = matrix(nrow=1000,ncol=1)

for (i in 1:1000) {
Z = rbinom(1,1,2/5) # se hace la mezcla de 2/5 y de 3/5
if (Z == 1) {
X[i] = rnorm(1,124,8) # de una gaussiana con promedio 124 y desviación estándar 8
} else {
X[i] = rnorm(1,157,7) # de una gaussiana con promedio 157 y desviación estándar 7
}
}
```

Las características de cada componente son :

Proporción = 2/5, promedio 124 y desviación estándar 8.
Proporción = 3/5, promedio 157 y desviación estándar 7.

Sin embargo la mezcla presenta las siguientes características :

Mínimo :101.0

Cuartíl 1:126.3

Mediana :150.2

Promedio :143.7

Cuartíl 3: 158.2

Máximo: 178.5

Varianza : 312.2382

Desviación estándar : 17.67026

En la figura C.1 se presenta el histograma de las observaciones mezcladas, donde se pueden apreciar dos modas.

Algoritmo EM para dos gaussianas unidimensionales

```
 # primeras estimaciones
prop1 = 0.2
prop2 = 0.8
mu1 = 100
mu2 = 170
sigma1 = 5
sigma2 = 4

K = 300 #  K iteraciones
for (i in 1:K) {
## Applicacion de la ecuacion #a para el caso de dos componentes de gauss
#fase E
#probabilidades apriori
pxc1th1 = prop1*dnorm(X,mean=mu1,sd=sigma1) #componente uno de mezcla
pxc1th2 = prop2*dnorm(X,mean=mu2,sd=sigma2) #componente dos de mezcla
#probabilidades a posteriori
pc1xth12 = pxc1th1 / (pxc1th1 + pxc1th2) # probas a posteriori p_{i,1}
pc1xth22 = pxc1th2 / (pxc1th1 + pxc1th2) # probas a posteriori p_{i,2}
t1x<- pc1xth12
t2x<- pc1xth22

#fase M

## Actualización de prop1 = P(Z=1 | X,Theta) :
prop1 = mean(t1x)
prop2 = 1-prop1
## Actualizacion de mu1 y mu2 :
mu1 = sum(t1x*X)/sum(t1x)
mu2 = sum(t2x*X)/sum(t2x)
## Actualizacion de sigma et sigma2 :
sigma1 = sqrt(sum(t1x*(X-mu1)^2)/(sum(t1x)))
sigma2 = sqrt(sum(t2x*(X-mu2)^2)/(sum(t2x)))
}
print (c( prop1, mu1, sigma1))
print (c(prop2, mu2, sigma2))
```

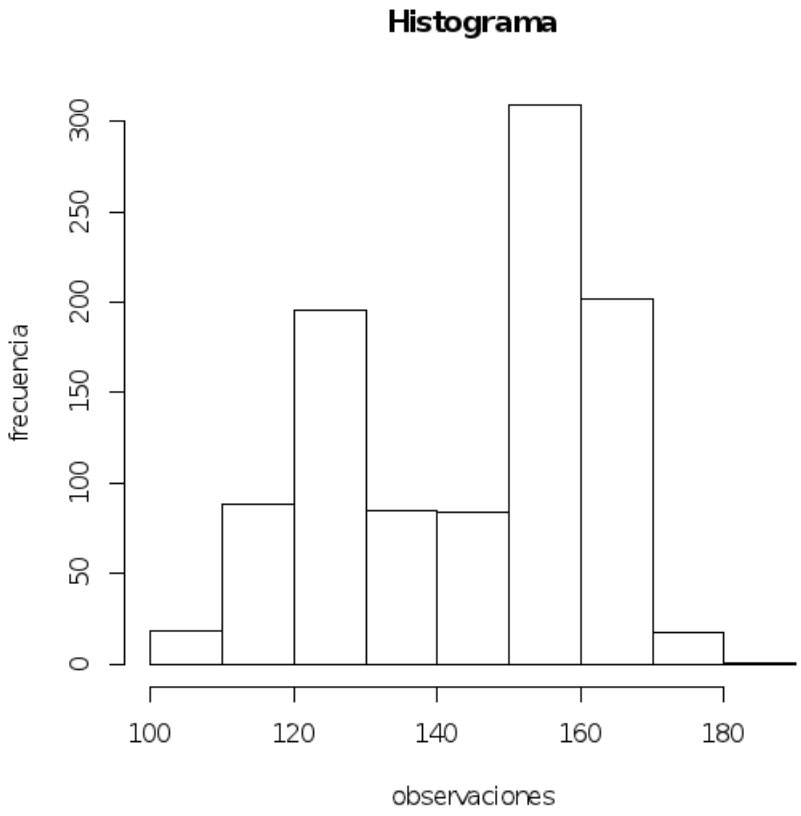


Figura C.1: Histograma de las observaciones mezcladas.

Con 300 iteraciones las estimaciones resultan. Para el primer componente :

0.4003972 124.6177001 8.1689195

Y para el segundo componente :

0.5996028 157.8224688 6.6528586

Estos valores estimados mediante el algoritmo EM no están muy lejos de los valores iniciales de los componentes, es decir :

Proporción = 2/5, promedio 124 y desviación estándar 8

Proporción = 3/5, promedio 157 y desviación estándar 7

Este programa en R está propuesto a título ilustrativo y es susceptible de muchas mejoras (ver el apéndice B donde se presenta el algoritmo EM programado en Scilab para K grupos)

Apéndice D

Dendogramas

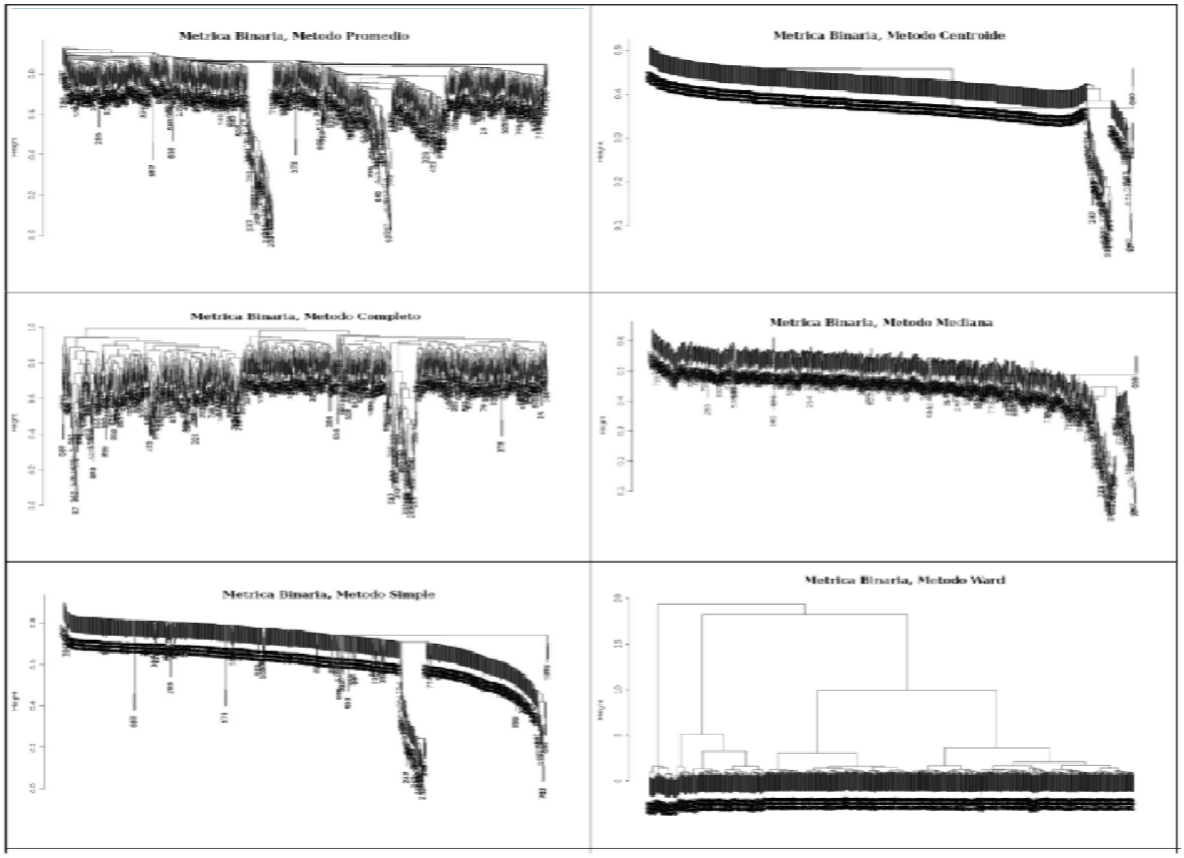


Figura D.1: Dendogramas de la métrica Binaria.

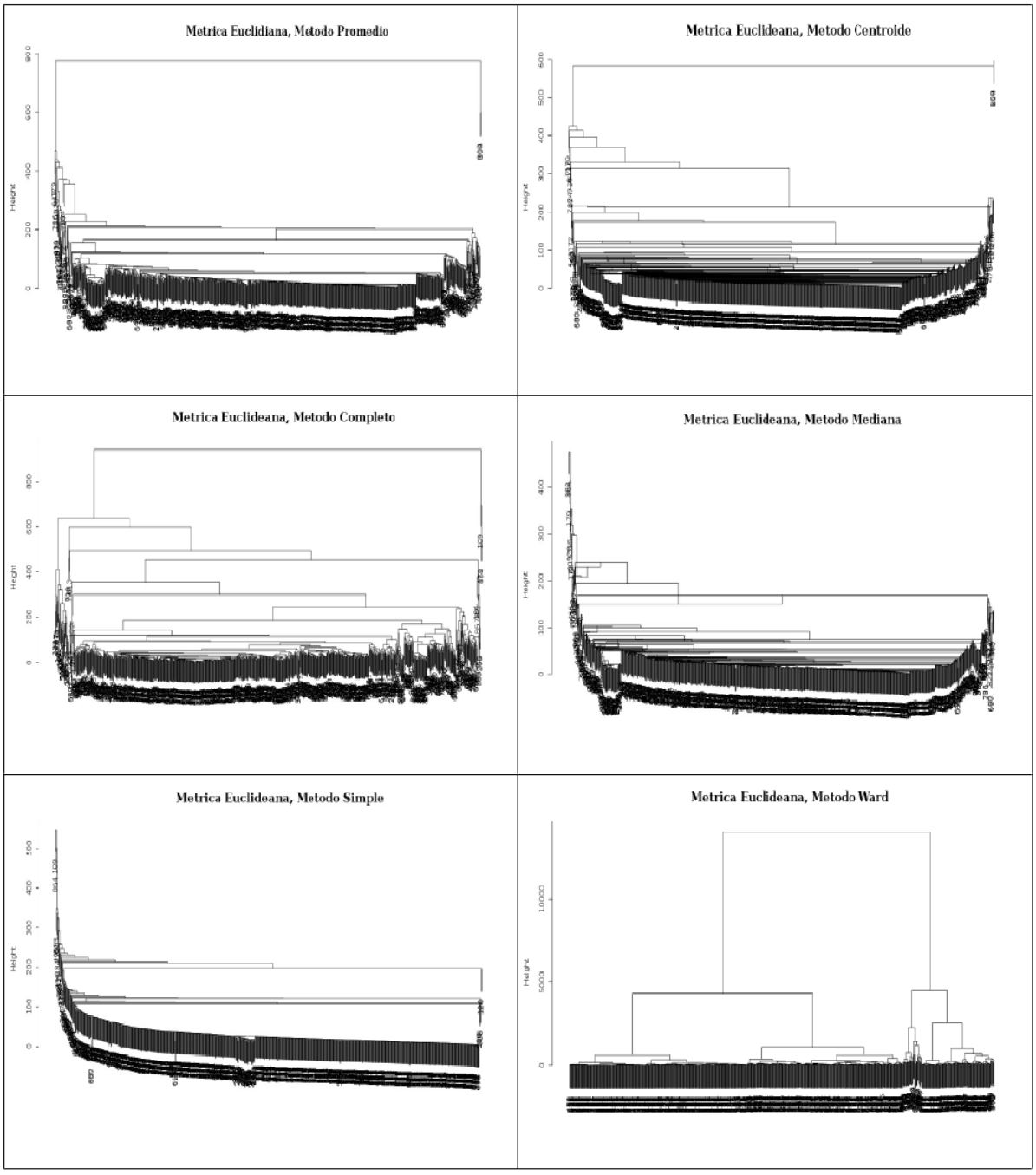


Figura D.2: Dendogramas de la métrica Euclideana.

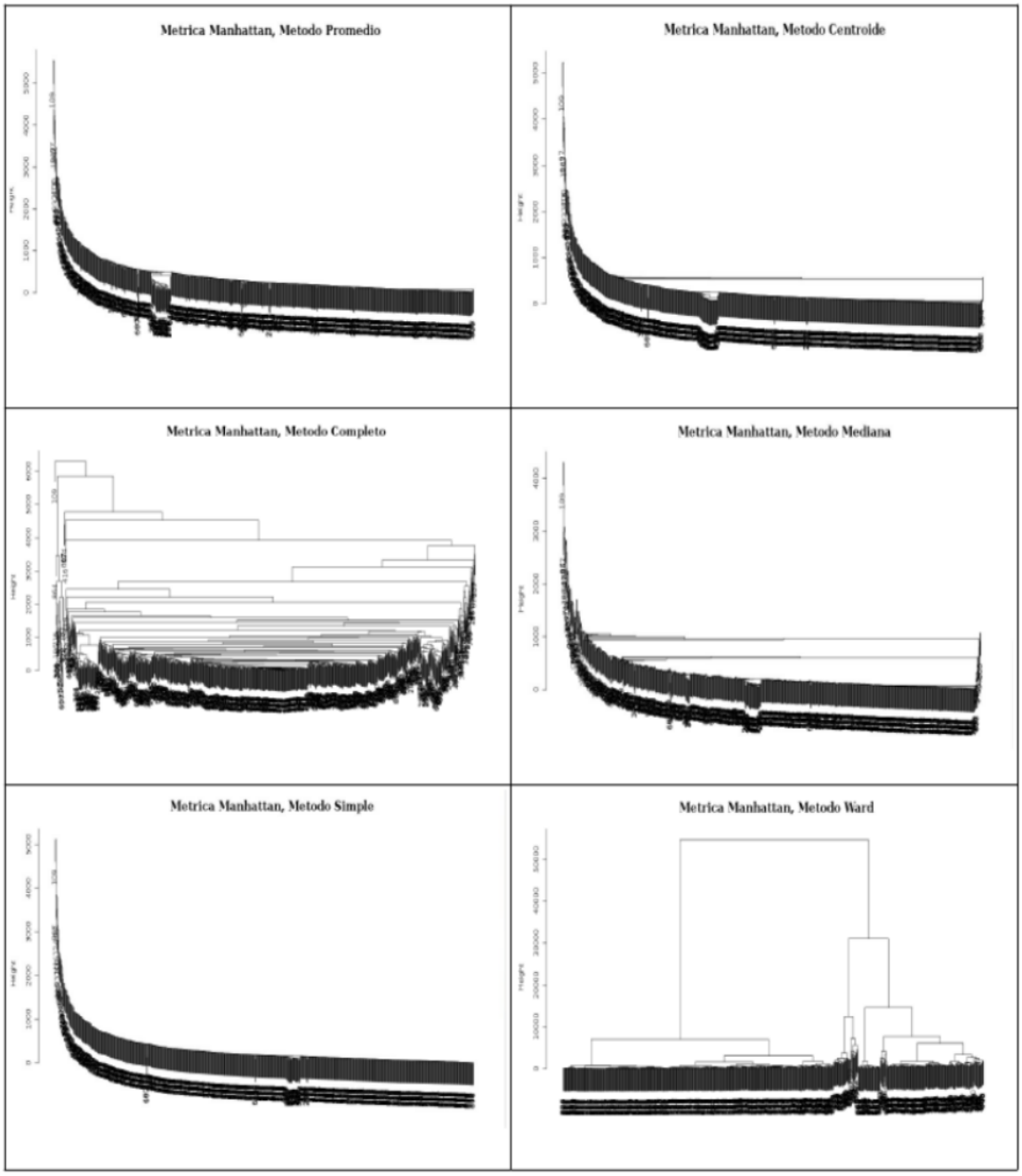


Figura D.3: Dendogramas de la métrica Manhattan.

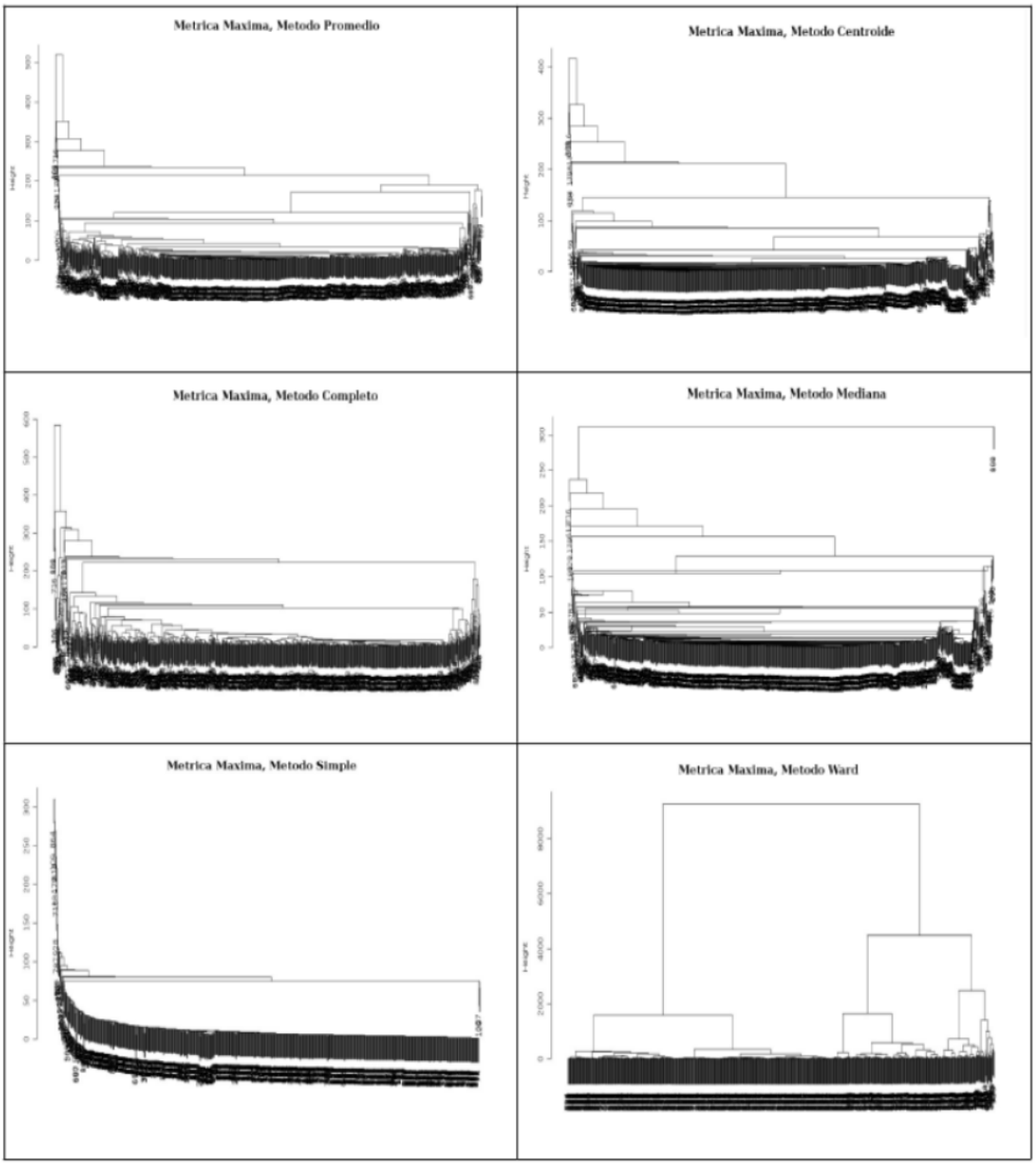


Figura D.4: Dendogramas de la métrica Máxima.

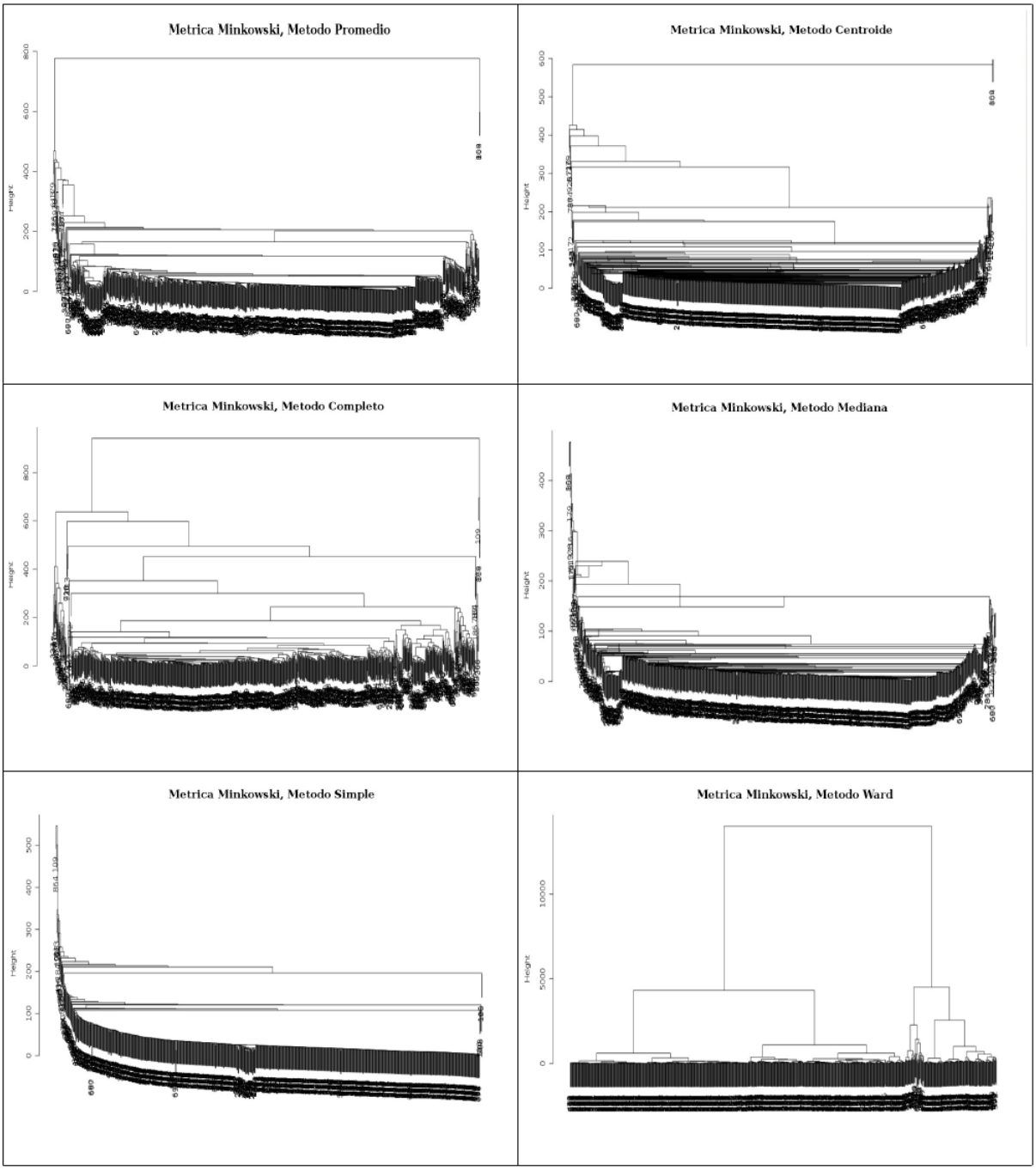


Figura D.5: Dendogramas de la métrica Minkowski.

Apéndice E

Manual de usuario

Este breve manual le permitirá aprender a utilizar de manera correcta el Asistente para la Minería XML, mediante la interfaz gráfica de usuario el uso de la aplicación es muy sencillo.

Cuando el usuario ejecute el Asistente para la Minería XML aparecerá el panel de Inicio de la aplicación, tal y como se muestra en la figura E.1, este panel le dará un breve mensaje de bienvenida.

Elección del archivo o colección a trabajar

Para poder continuar de manera correcta con el asistente, el usuario deberá de dar clic en el botón Siguiente y acontinuación se mostrará el panel de Colección, tal y como se muestra en la figura E.2. Este panel le permitirá elegir iniciar el minado XML desde un archivo de frecuencias guardado previamente o bien desde una colección.

Si el usuario desea iniciar desde un archivo de frecuencias, surgirá una pantalla como la que se muestra en la figura E.3, en donde le permitirá seleccionar la ruta del archivo.

Si el usuario elige la opción de trabajar con archivos de frecuencias previamente guardados al dar clic en el botón Siguiente, la aplicación lo llevará al panel para la elección del algoritmo de agrupamiento.

Si el usuario desea trabajar desde una colección deberá seleccionar la opción de colección, en donde se habilitará las colecciones disponibles, si no se contara con la colección deseada se puede agregar una nueva colección con el botón agregar, tal y como se muestra en la figura E.4

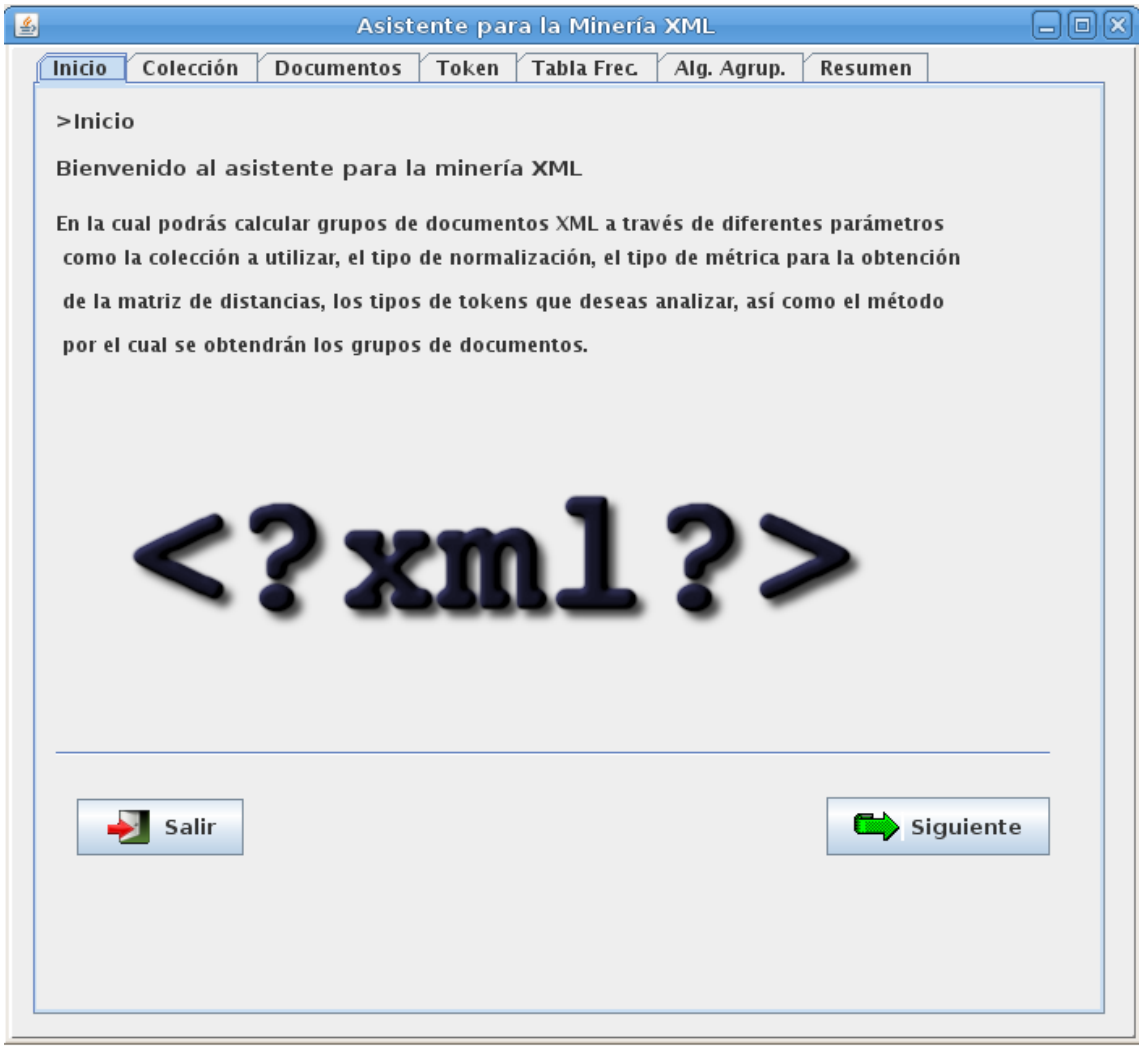


Figura E.1: Inicio del asistente para la minería XML

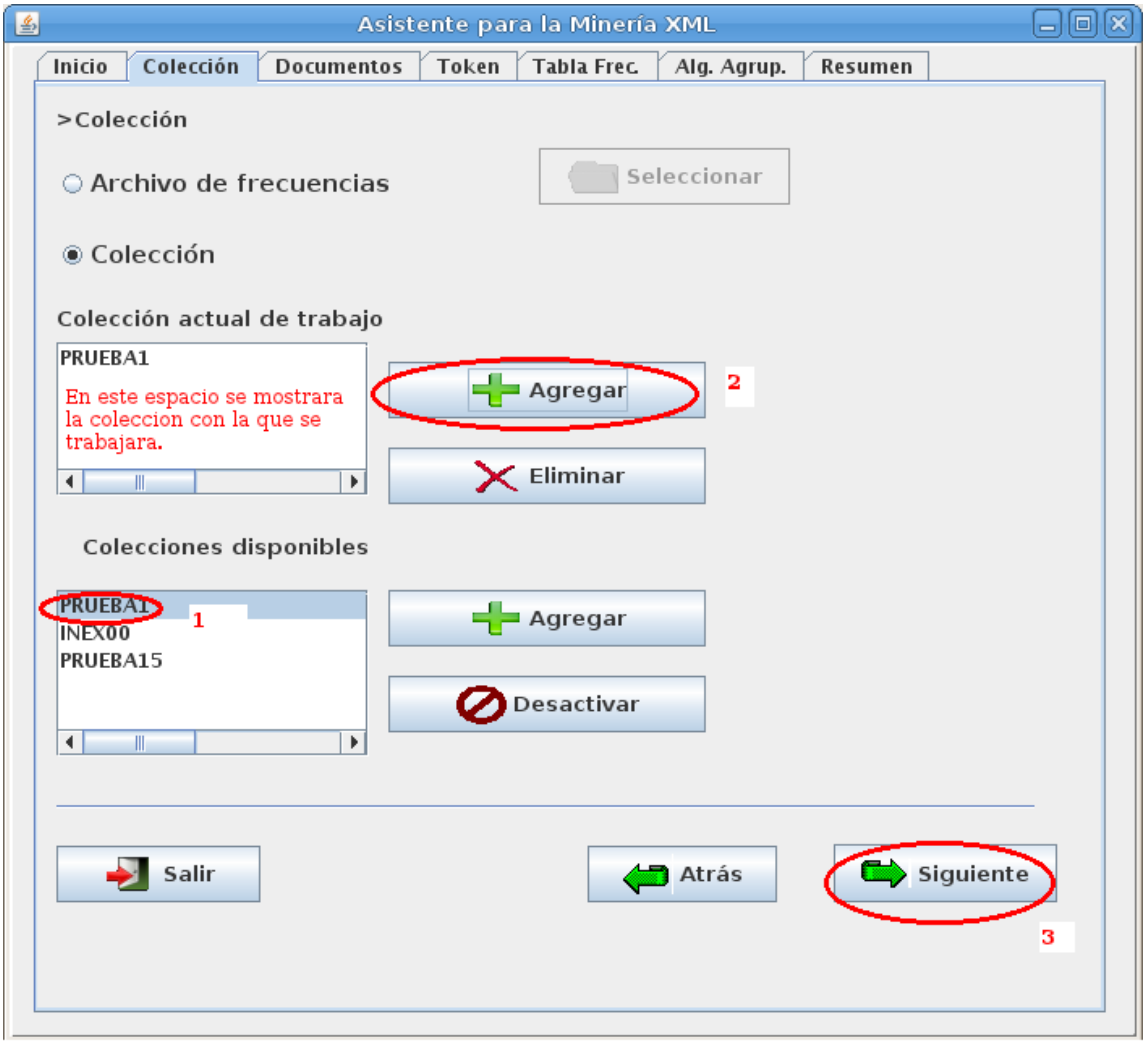


Figura E.2: Colecciones existentes

Figura E.3: Selección del archivo de frecuencias

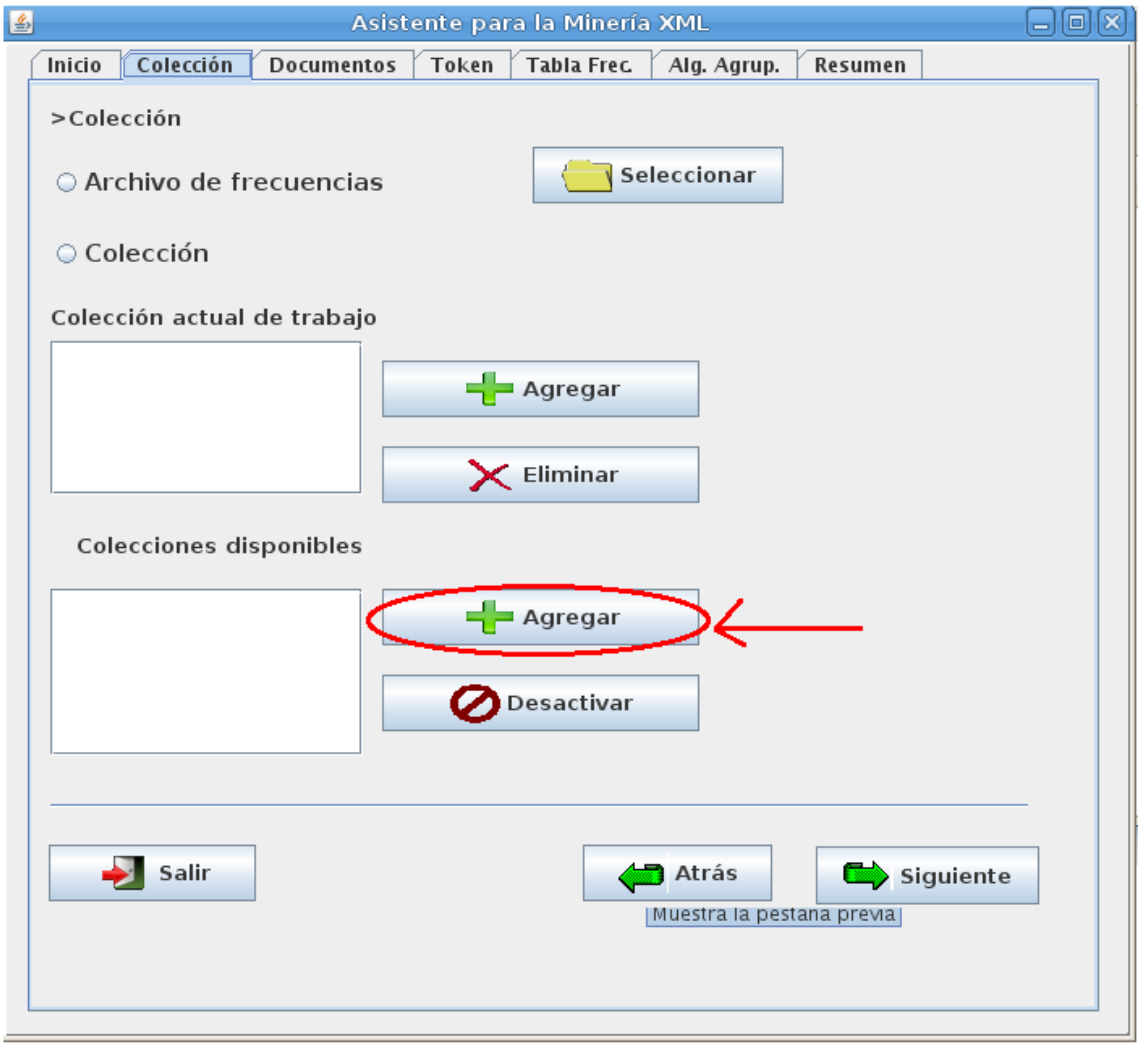


Figura E.4: Botón para agregar una nueva colección

Al dar clic en el botón aceptar emergerá una pantalla tal y como la que se muestra en la figura E.5 en donde se solicitará el nombre de la nueva colección y la ruta donde se encontrarán los archivos.

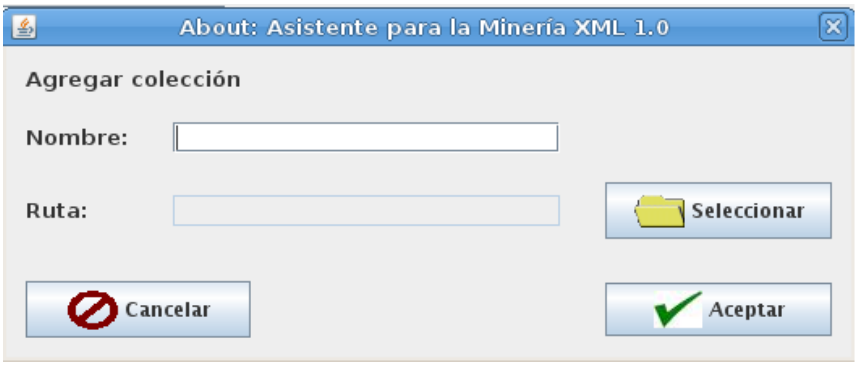


Figura E.5: Información requerida para una nueva colección

Para dar de alta una nueva colección es necesario escribir el nombre de la colección y a continuación dar clic en el botón seleccionar, en donde se mostrará una pantalla como la de la figura E.6, en donde permitirá al usuario elegir el directorio donde se encuentra la nueva colección.

Una vez que el usuario haya completado los datos de la nueva colección deberá de dar clic en el botón Aceptar, en donde se mostrará un mensaje como el que se muestra en la figura E.7.

Si el usuario desea cancelar el proceso de alta de una colección basta con dar clic en el botón Cancelar, en donde se mostrará un mensaje preguntando si ¿Desea cancelar la agregación de una nueva colección?, tan sólo basta con dar clic en el botón Si.

Si el usuario desea desactivar alguna colección existente, tan sólo basta con seleccionar la colección y dar clic en el botón Desactivar.

Para habilitar una colección de minado es necesario seleccionar una colección disponible y posteriormente dar clic en el botón Agregar, la colección de minado aparecerá en el área de colección actual de trabajo; tal y como se muestra en la figura E.8.

Cabe aclarar que solamente es posible minar una colección a la vez.

Elegir toda la colección o una muestra

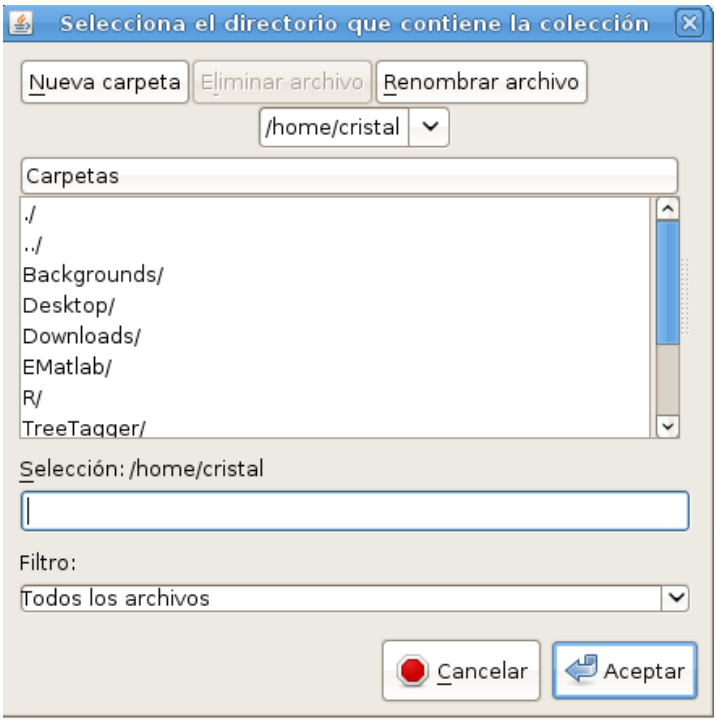


Figura E.6: Selección del directorio de una nueva colección

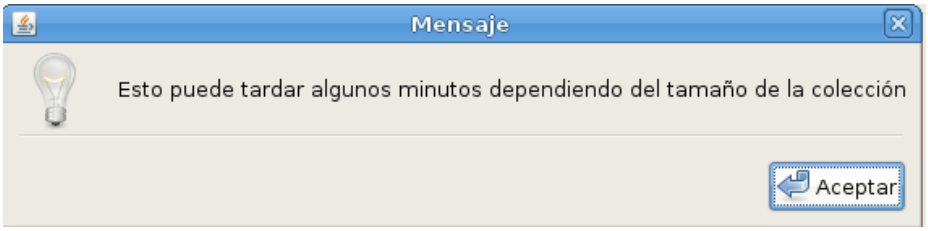


Figura E.7: Mensaje al dar de alta una nueva colección

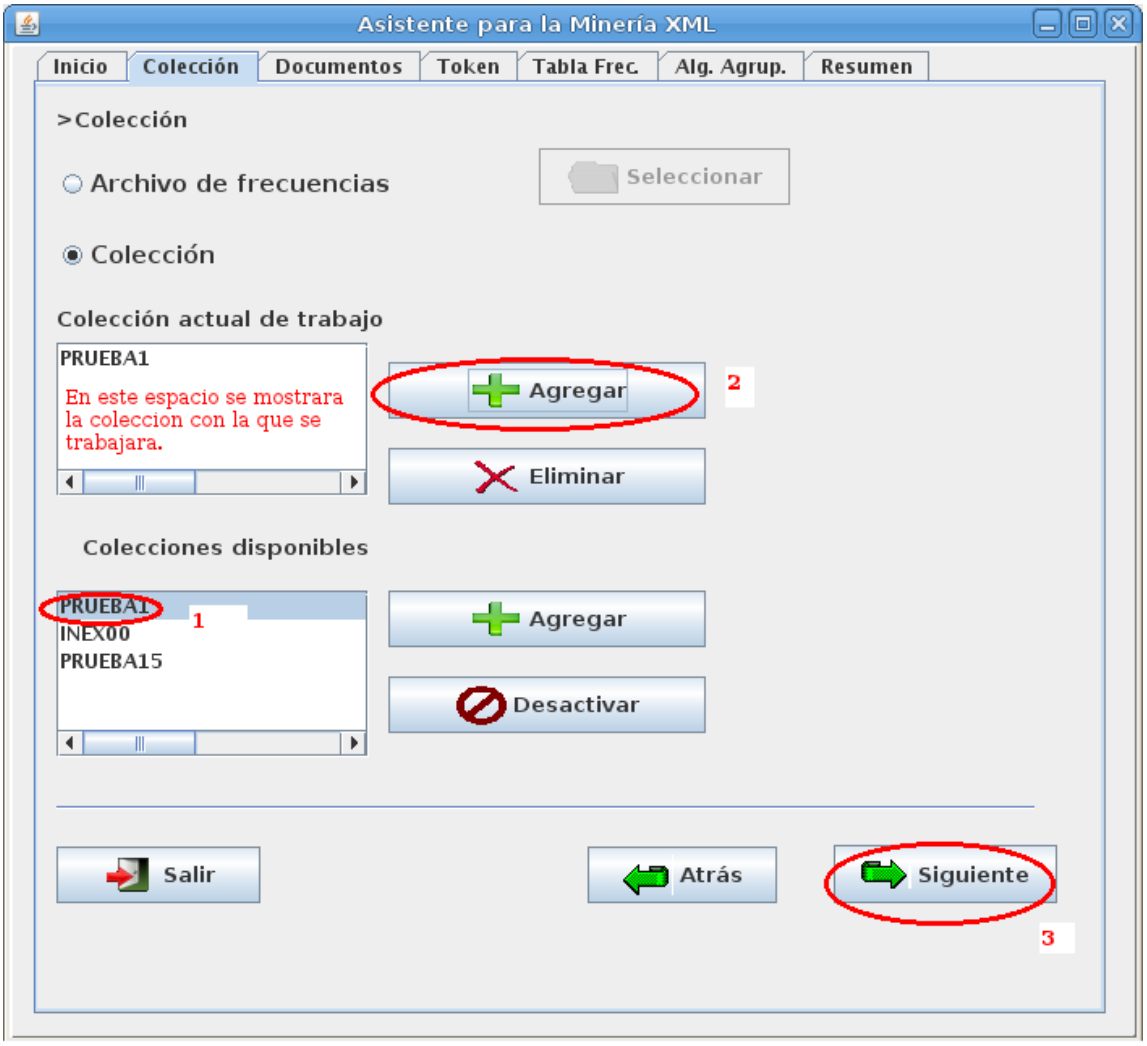


Figura E.8: Pasos para elegir el minado de una colección

Al haber elegido la colección a minar es necesario dar clic en el botón Siguiente, el cual mostrará el panel que se muestra en la figura E.9.

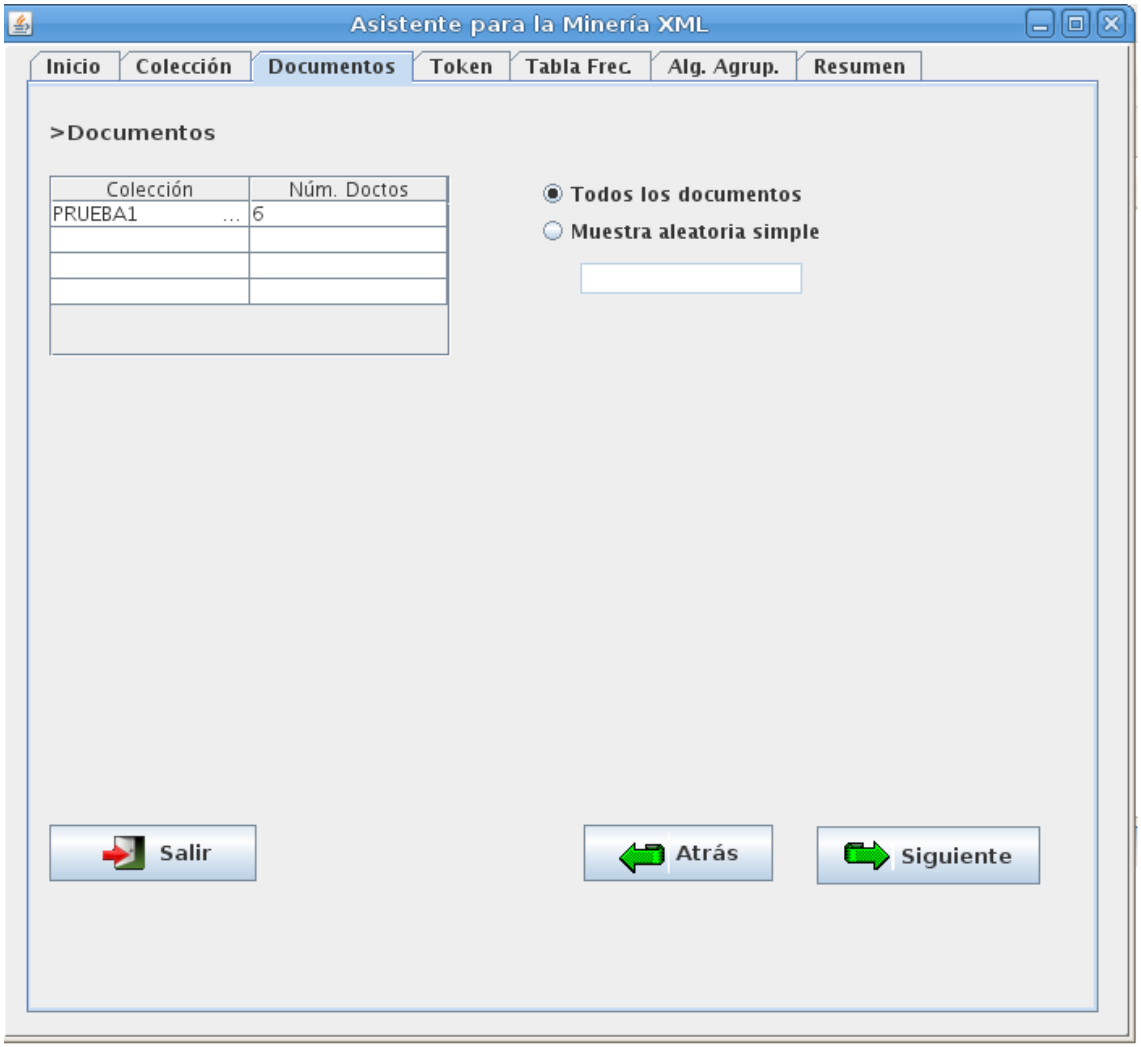


Figura E.9: Número de documentos que conforman la colección

En donde se muestra el número de documentos de la colección y donde el usuario podrá elegir si desea trabajar con toda la colección o bien con una muestra.

Si el usuario desea trabajar con una muestra, deberá de introducir el número de documentos con los que desea trabajar. Esta cifra deberá de ser menor o igual al número total de documentos con los que cuenta la colección. La figura E.10 muestra el lugar donde el usuario

deberá introducir el número de la muestra.

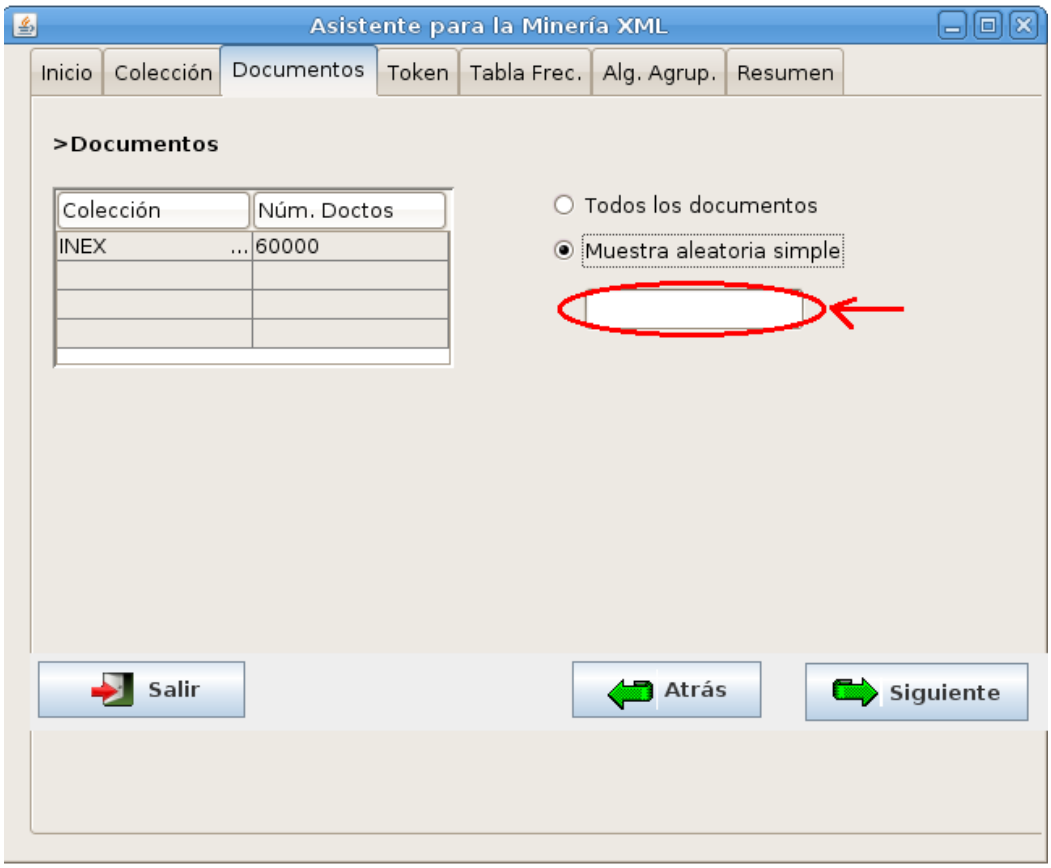


Figura E.10: Lugar donde el usuario debe introducir el número de la muestra

De no ser así la aplicación emitirá un mensaje como el que se muestra en la figura E.11, el cual informará que el número excede.

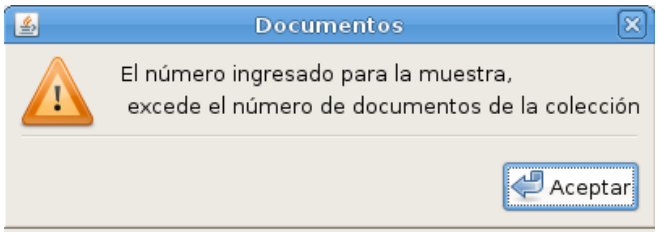


Figura E.11: Aviso al exceder el número de documentos contenidos en la colección

Elegir tipos de tokens

Posteriormente el usuario deberá dar clic en el botón Siguiente, el cual mostrará un panel como se muestra en la figura E.12.

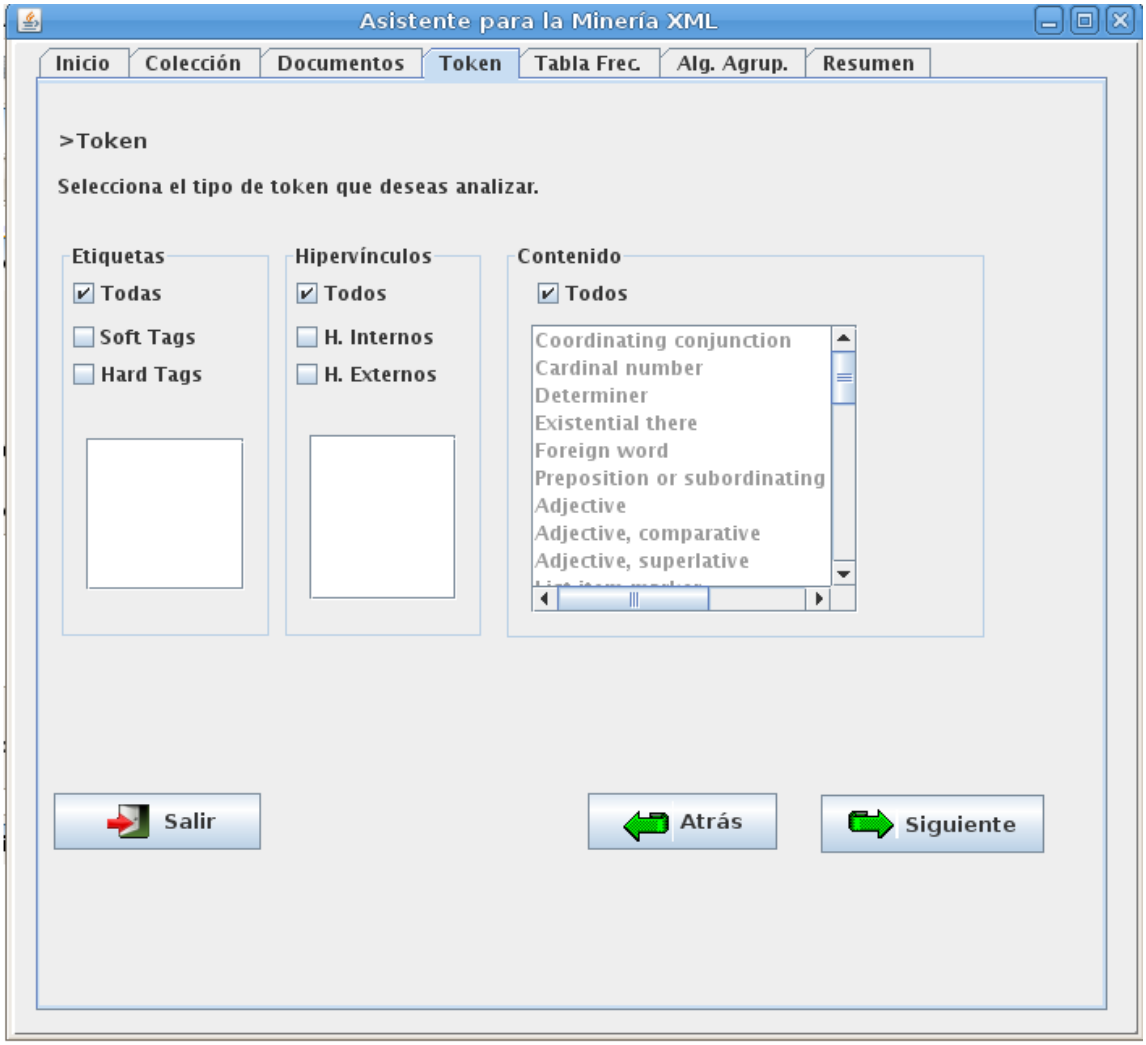


Figura E.12: Tipos de tokens para ser elegidos

En esta pantalla le permitirá al usuario elegir los diferentes tokens a minar, estos tokens se pueden agrupar en etiquetas, hipervínculos y contenido. En etiquetas sólo le permitirá al usuario elegir etiquetas soft (forma) o hard (definición), en hipervínculos le permitirá al usuario elegir hipervínculos internos y externos. En la selección de tokens de contenido, si

el usuario desea minar todos los tokens basta elegir todos de lo contrario, el programa le permitirá realizar una selección múltiple de los tokens que desea analizar.

Elegir tipo de normalización

Al haber realizado la selección de tokens a analizar el usuario deberá de presionar el botón Siguiente, el cual mostrará el panel que se muestra en la figura E.13.

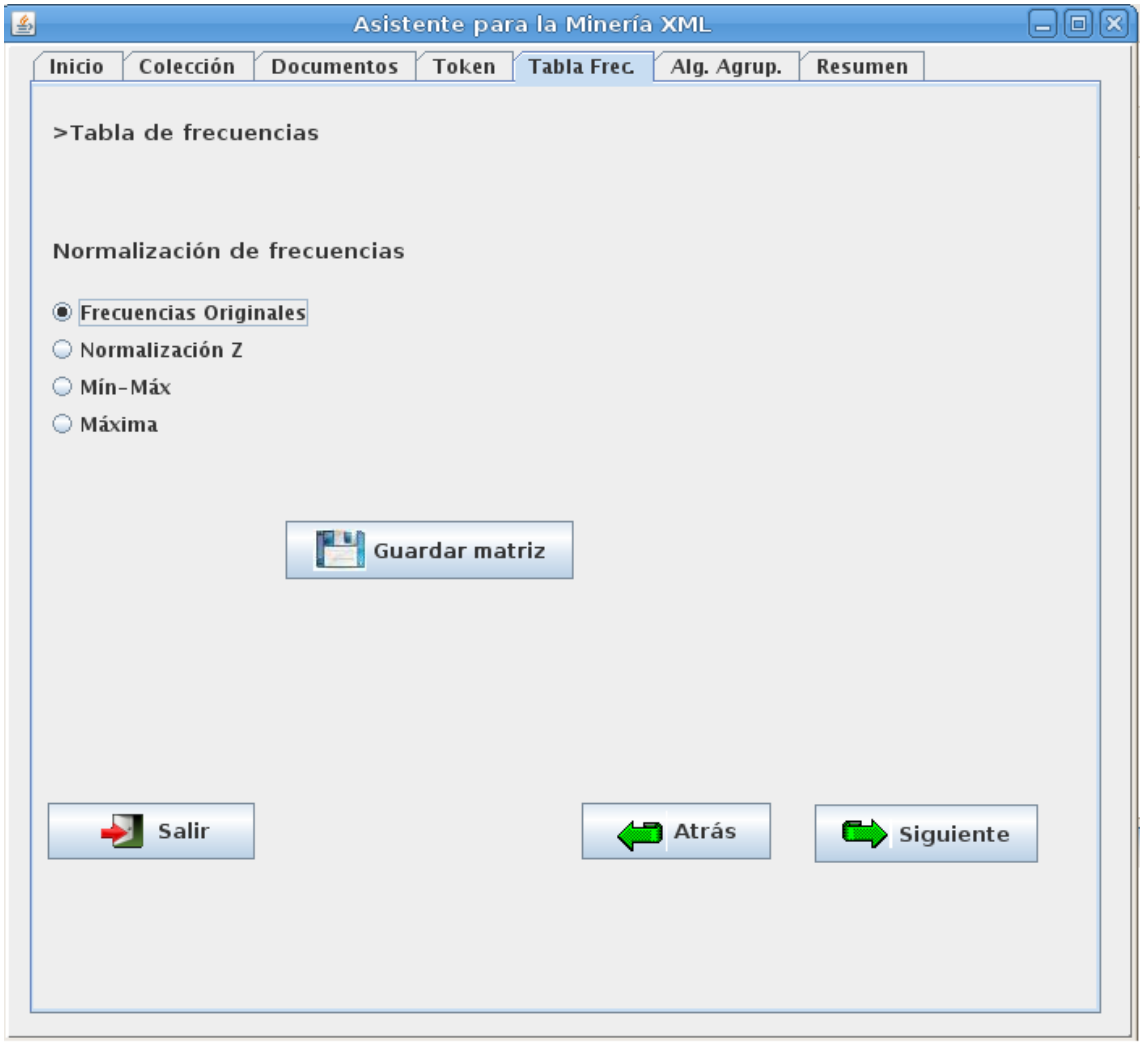


Figura E.13: Métodos para normalizar las frecuencias

En esta pantalla el usuario puedrá elegir la normalización que desea aplicar a la matriz de frecuencias, si el usuario desea guardar dicha matriz, lo puede realizar dando clic en el

botón Guardar matriz, en donde le mostrará al usuario el panel para elegir la ruta donde se guardará el archivo.

Elegir algoritmos de agrupamiento

Para poder continuar es necesario que el usuario de clic en el botón Siguiente, el cual mostrará un panel como el que se muestra en la figura E.14.

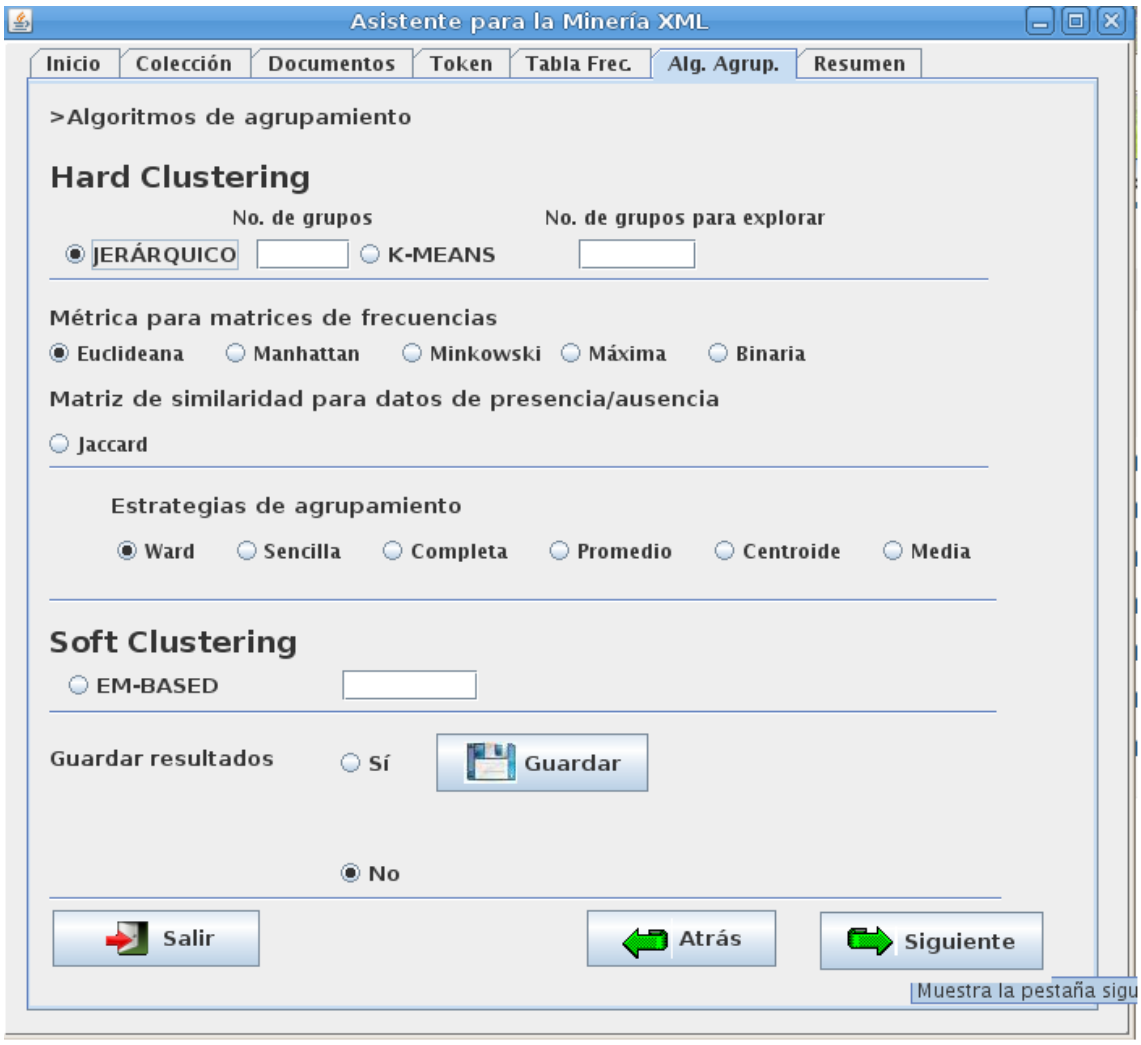


Figura E.14: Métodos de agrupamiento disponibles

En esta pantalla el usuario podrá elegir el método para la agrupación, así como la métrica de distancia que podrá aplicar a la matriz de frecuencias, en caso de elegir un método hard

(jerárquico).

Para continuar con el asistente el usuario deberá de dar clic en el botón Siguiente, el cual presentará la pantalla de Resumen como la que se muestra en la figura E.15.

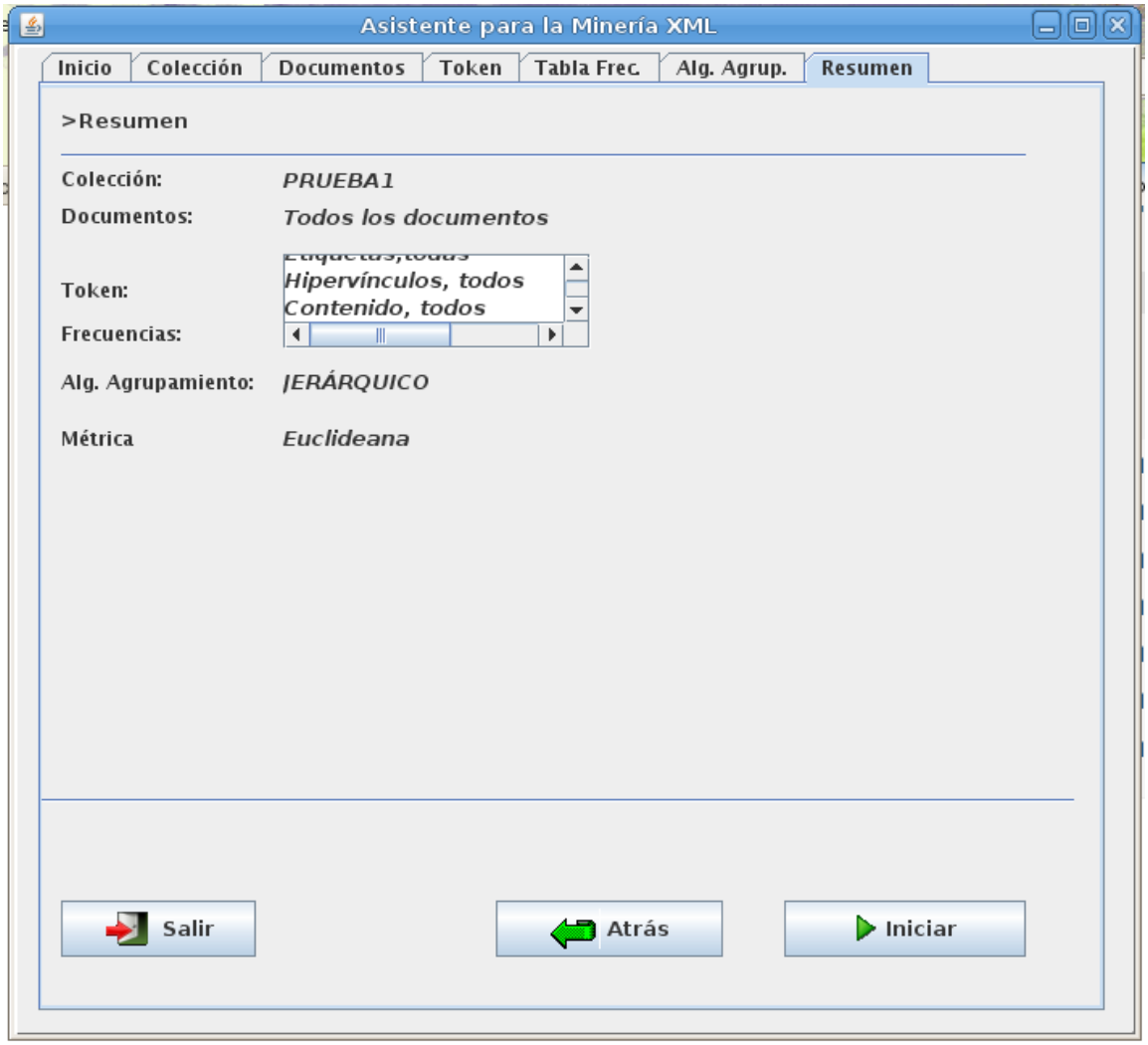


Figura E.15: Resumen de los parámetros de configuración del asistente

En esta pantalla tal como lo indica su nombre, muestra al usuario los parámetros que el usuario ha configurado. Para poder dar inicio al proceso de agrupación basta con dar clic en el botón Iniciar, si el usuario desea cancelar el proceso deberá dar clic en el botón Salir.

Apéndice F

Manual de instalación

Para poder ejecutar de manera correcta el proyecto es necesario seguir los siguientes pasos:

1. Contar con sistema operativo Linux específicamente con la distribución Arch Linux [1].
2. Instalar la base de datos PostgreSQL, la versión 9.0.2.
3. Tener instalado TreeTagger [2], para ello es necesario que se instale con el usuario con el que se trabaje en Linux y no como root; pues de lo contrario marca errores de permiso
4. Tener instalado Netbeans y el JDK de Oracle [3], ya que si se instala Open JDK marca errores, para poder revisar que versión de Java se tiene instalado, se hace en consola con el comando *java -version*
5. Correr los scripts para crear las tablas de la siguiente manera: coleccion, rutas, documento, doc_coleccion, tipo_gram, categoria, token y por último t_frecuencia.
6. Ejecutar en PostgreSQL el script de llenado de tablas.
7. Crear las siguientes secuencias: sec_categoria, sec_coleccion, sec_docto, sec_rutas, sec_tipogram y sec_token.
8. Abrir el proyecto con Netbeans, verificar los datos de conexión a la BD.

[1] *http://www.archlinux.org/*
[2] *http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/*
[3] *http://www.oracle.com/technetwork/java/javase/downloads/index.html*

9. La clase que se debe ejecutar es la clase Main.java

En la fig.F.1 se muestra el diagrama de flujo de los pasos para ejecutar el proyecto.

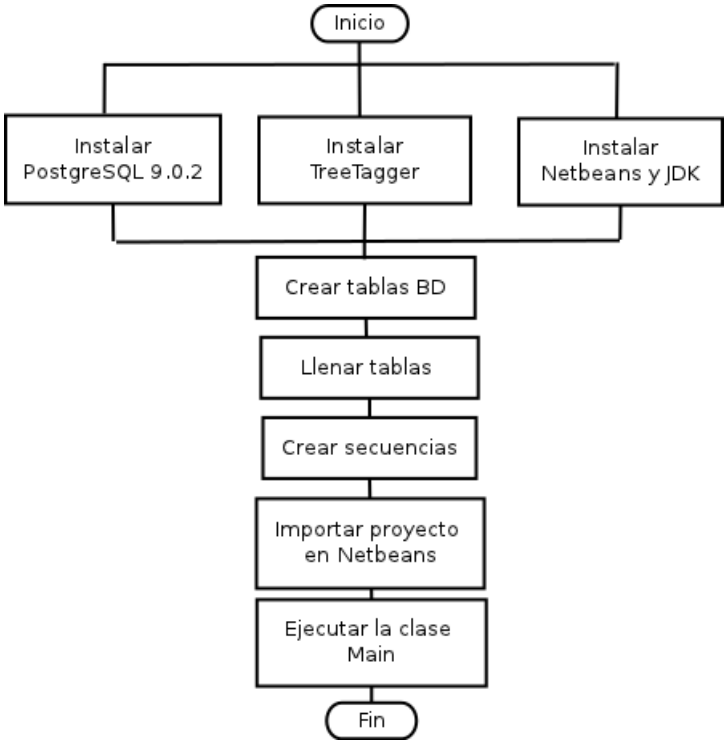


Figura F.1: Diagrama que muestra los pasos para ejecutar el proyecto

Apéndice G

Cálculo de índices de Dunn y Davies-Bouldin

Con el fin de presentar el cálculo de los índices de evaluación de grupos presentados en la sección 4.7 *"Criterios de Evaluación de Agrupamiento"*, se presenta a continuación un ejemplo de cálculo del Indice de Dunn y de Davies-Bouldin para la evaluación de 5 grupos que resultan de la aplicación del Método Jerarquíco con la estrategia de Ward y la métrica Euclidiana. En el caso de 5 grupos se identifican de acuerdo a la figura G.1 las distancias entre los grupos y los diámetros de cada uno de los cinco grupos. La figura G.1 (página 146) muestra el diagrama del enunciado anterior.

Los diámetros de cada grupo están representados por :

$$[\Delta(X_1), \Delta(X_2), \Delta(X_3), \Delta(X_4), \Delta(X_5)] \tag{G.1}$$

Estos diámetros, también denominados distancias intragrupo, se cálculan aplicando una métrica para medir la distancia (máxima, promedio y promedio al centroide). El centroide también se denomina baricentro.

En lo que respecta a las distancias entre grupos, estas se pueden obtener mediante distintos criterios (distancia mínima entre grupos, distancia máxima entre grupos, distancia promedio entre grupos y distancia entre los centroides). Estas distancias se pueden representar mediante la siguiente matriz.

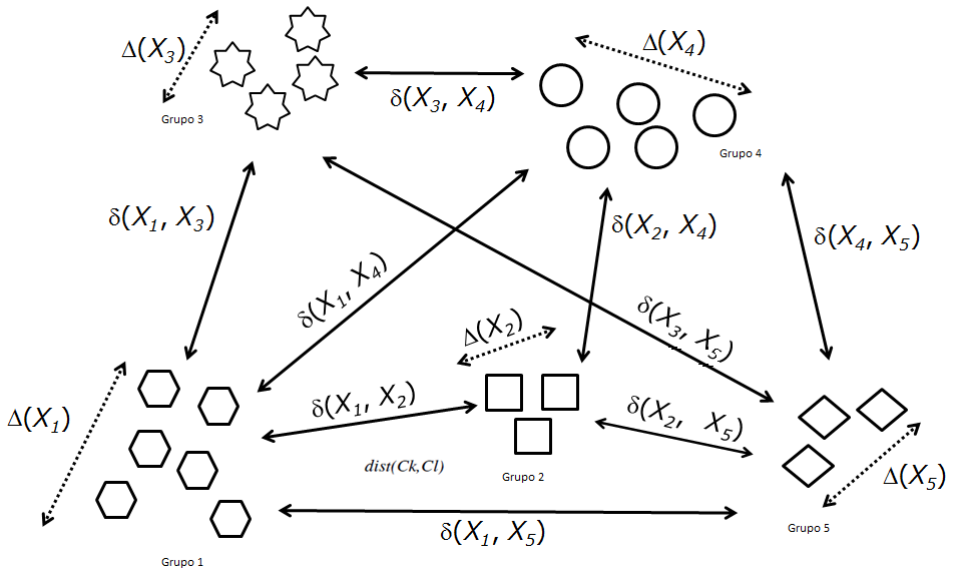


Figura G.1: Distancias entre los grupos y los diámetros de cada uno de los cinco grupos

$$\begin{bmatrix} \delta(X_1,X_1) & \delta(X_1,X_2) & \delta(X_1,X_3) & \delta(X_1,X_4) & \delta(X_1,X_5) \\ \delta(X_2,X_1) & \delta(X_2,X_2) & \delta(X_2,X_3) & \delta(X_2,X_4) & \delta(X_2,X_5) \\ \delta(X_3,X_1) & \delta(X_3,X_2) & \delta(X_3,X_3) & \delta(X_3,X_4) & \delta(X_3,X_5) \\ \delta(X_4,X_1) & \delta(X_4,X_2) & \delta(X_4,X_3) & \delta(X_4,X_4) & \delta(X_4,X_5) \\ \delta(X_5,X_1) & \delta(X_5,X_2) & \delta(X_5,X_3) & \delta(X_5,X_4) & \delta(X_5,X_5) \end{bmatrix} \tag{G.2}$$

Esta matriz contiene 25 distancias intergrupo (entre grupos). Aunque dado que es una matriz simétrica y las distancias entre el mismo grupo son cero, en la matriz se representan 10 distancias. Las distancias descritas permiten calcular el índice de Dunn con la siguiente fórmula :

$$Dunn = \min_{1\leq i\leq c} \left\{ \min_{\substack{1\leq j\leq c \\ j\neq i}} \left\{ \frac{\delta(x_i,x_j)}{\max\limits_{1\leq k\leq c}\{\Delta(x_k)\}} \right\} \right\} \tag{G.3}$$

Dónde :

- x_i, x_j; es la distancia entre los grupos X_i, X_j.

- x_k; es la distancia intragrupo de X_k.

- c; es el número de grupos de la partición U.

Cabe señalar que los valores elevados del índice de Dunn indican un buen agrupamiento. Para ejemplifar la secuencia de los cálculos vamos a realizar el agrupamiento Jerárquico con el método de Ward y la métrica Euclidiana a la tabla de frecuencias con 943 documentos XML de muestra y con 16271 variables o atributos.
De aquí se obtiene la siguiente asignación de las 943 observaciones en 5 grupos.

```
[1] 1 1 1 1 1 2 2 2 1 2 2 1 1 1 1 1 1 1 2 2 1 1 1 1 1 2 1 1 1 1 1 1 2 1 1 1 1
[38] 1 2 1 1 2 1 3 2 2 2 1 1 1 1 2 2 2 1 1 2 2 2 3 1 1 2 2 1 2 2 2 3 2 2 3 1 1
[75] 2 1 3 1 2 1 2 3 1 4 2 3 2 3 3 2 3 1 2 2 1 2 4 3 3 4 1 2 2 3 3 2 4 3 4 3 3
[112] 2 1 4 2 1 3 2 4 3 2 2 3 4 3 4 2 1 1 2 3 3 4 2 2 1 3 2 2 1 2 2 4 1 1 1 1 1
[149] 2 2 2 1 1 2 1 1 1 1 1 1 1 2 2 1 2 3 1 1 1 1 1 3 1 1 1 2 1 2 4 1 2 1 1 4 3
[186] 4 1 2 1 2 1 1 2 3 3 1 2 1 2 2 2 2 2 2 2 4 1 3 4 2 2 3 3 1 2 1 3 3 2 1 1 2
[223] 2 2 2 3 3 1 2 3 5 5 2 5 2 2 2 5 5 1 5 5 5 5 5 5 2 5 5 5 5 3 5 5 5 5 2 5 3
[260] 5 5 3 5 5 5 5 5 2 3 3 1 1 3 2 2 2 1 2 1 3 2 2 1 2 2 2 2 1 1 3 2 1 3 2 1 2 2
[297] 1 1 2 1 1 2 2 1 2 2 2 1 2 3 1 1 3 1 2 1 1 1 1 1 1 1 2 3 1 4 1 2 3 2 2 2 1 2
[334] 3 2 1 4 5 5 5 2 1 3 1 3 2 5 5 3 1 3 5 5 3 5 5 5 2 2 2 5 1 5 5 3 2 2 4 2 5
[371] 1 5 2 1 2 3 1 1 1 1 1 1 1 2 1 2 3 2 1 3 2 2 1 1 2 1 1 1 4 1 2 1 2 2 2 2 1
[408] 2 2 1 1 2 1 1 1 4 2 3 3 2 1 1 2 2 1 2 1 4 2 2 1 2 1 1 1 1 2 1 2 2 2 2 2 1
[445] 1 3 3 3 3 2 1 3 1 5 2 2 1 2 2 3 2 2 1 3 3 1 2 2 2 4 1 3 3 2 2 2 2 1 3 5 3
[482] 2 2 2 2 3 1 2 2 2 4 1 2 3 2 2 3 1 2 2 1 2 2 3 3 2 1 2 1 2 1 1 1 1 1 1 1 2
[519] 2 1 2 2 2 1 1 1 2 1 2 2 1 1 1 2 2 1 3 3 1 3 1 4 1 2 2 1 1 2 1 1 2 1 2 2 1
[556] 1 2 3 1 2 2 1 4 2 1 1 1 1 1 1 1 3 2 2 4 2 1 1 2 2 1 1 1 1 2 2 2 1 1 2 2 1
[593] 2 1 4 2 2 1 1 1 2 3 2 3 1 3 1 2 1 3 2 2 4 1 2 2 3 1 1 1 2 2 1 2 2 1 3 2 1
[630] 3 1 2 1 1 1 1 2 2 3 1 1 2 1 2 3 1 3 2 1 1 1 3 1 2 1 1 1 1 2 2 2 2 1 1 1 1
[667] 1 1 1 1 2 1 1 1 2 1 2 2 1 4 3 2 2 2 2 2 2 2 1 1 1 1 3 1 2 2 4 1 2 2 3 2 2
[704] 1 2 2 1 4 1 3 1 1 2 3 2 4 2 2 3 2 1 2 1 2 3 2 1 2 2 3 2 2 2 1 2 1 1 1 1 3
[741] 1 1 1 2 1 1 1 1 1 1 1 1 2 1 2 1 2 1 1 2 1 1 1 1 2 2 1 1 2 1 1 2 3 1 1 2 1
[778] 1 2 1 2 1 2 2 1 1 4 1 1 3 2 2 1 1 1 1 2 2 1 1 2 4 1 1 1 2 2 2 1 1 1 1 3 2
[815] 1 2 1 1 1 3 2 4 2 1 2 2 2 1 3 1 1 1 2 2 2 3 1 2 4 1 1 1 3 3 2 2 2 2 2 1 2
[852] 1 1 2 1 1 1 1 2 2 1 2 1 4 4 2 4 2 3 1 2 1 2 1 1 1 2 1 1 2 1 2 2 1 1 1 1 1
[889] 2 2 2 1 1 1 1 2 1 1 1 1 2 2 1 2 2 2 2 1 1 3 2 2 2 1 2 1 1 1 2 2 3 2 2 1 1
[926] 3 1 4 1 1 1 2 1 1 3 2 1 1 1 1 2 1 2
```

Es decir se agrupan 394 documentos en el grupo 1, 347 documentos en el grupo 2, 17 documentos en el grupo 3, 41 documentos en el grupo 4 y 44 documentos en el grupo 5. Los cuales se distinguen en la figura G.2 (correspondiente al histograma de los grupos), que se presenta en la página 148.

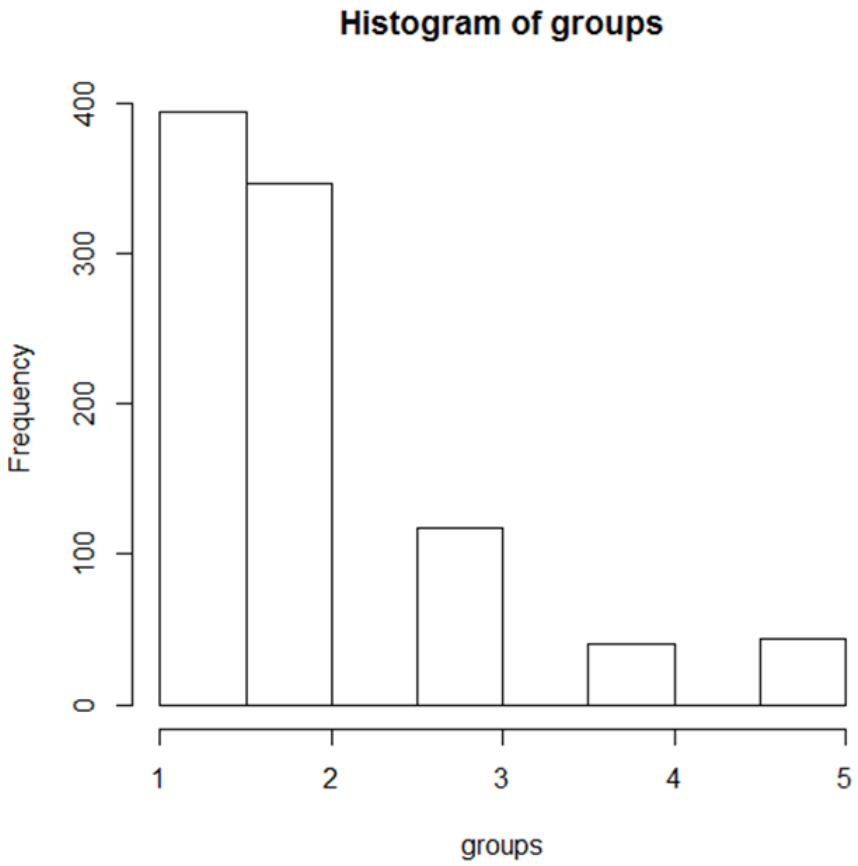


Figura G.2: Histograma de los grupos.

En la tabla G.1 se puede observar más específicamente la distribución de documentos XML por grupos mostrada en el histograma.

En la figura G.3, que se presenta en la página 150, se muestra el dendograma correspondiente al agrupamiento jerárquico.

Para calcular el índice de Dunn se deben obtener los 5 diámetros para los cada uno de los grupos.

Las 5 distancias intragrupos se pueden obtener mediante la distancia máxima, la distancia promedio entre todos los objetos o bien el promedio de las distancias al centro del grupo.

El diámetro completo representa la distancia entre 2 de los objetos más alejados dentro del mismo grupo.

Tabla G.1: Distribución de documentos por grupo.

	Grupo1	**Grupo2**	**Grupo3**	**Grupo4**	**Grupo5**
Núm. Elementos	394	347	117	41	44

Tabla G.2: Distancia Euclidiana con diferentes criterios.

Criterio \ **Diámetros**	$\Delta(X_1)$	$\Delta(X_2)$	$\Delta(X_3)$	$\Delta(X_4)$	$\Delta(X_5)$
Completo	51.77	148.38	310.97	941.43	96.68
Promedio	20.86	54.02	125.05	377.47	35.39
Centroide	14.52	37.53	86.88	260.34	24.72

$$diam1(C) = max\left\{d(x,y) :\ (x,y)\ pertenecen\ al\ grupo\ C\right\} \tag{G.4}$$

El diámetro promedio define la distancia promedio entre todos los objetos del mismo grupo.

$$diam2(C) = \frac{1}{|C|(|C|-1)} * sum\left\{\forall(x,y)\ que\ pertenezcan\ al\ grupo\ C\ y\ x! = y\right\} d(x,y) \tag{G.5}$$

El diámetro centroide refleja el doble promedio de la distancia entre todos los objetos y el centro del grupo ($y(C) - centro\ del\ grupo$).

$$diam3(C) = \frac{1}{|C|} * sum\left\{\forall(x)\ que\ pertenezcan\ al\ grupo\ C\right\} d(x, y(C)) \tag{G.6}$$

Mientras que la matriz de distancias entre los grupos esta representada por la expresión G.7 :

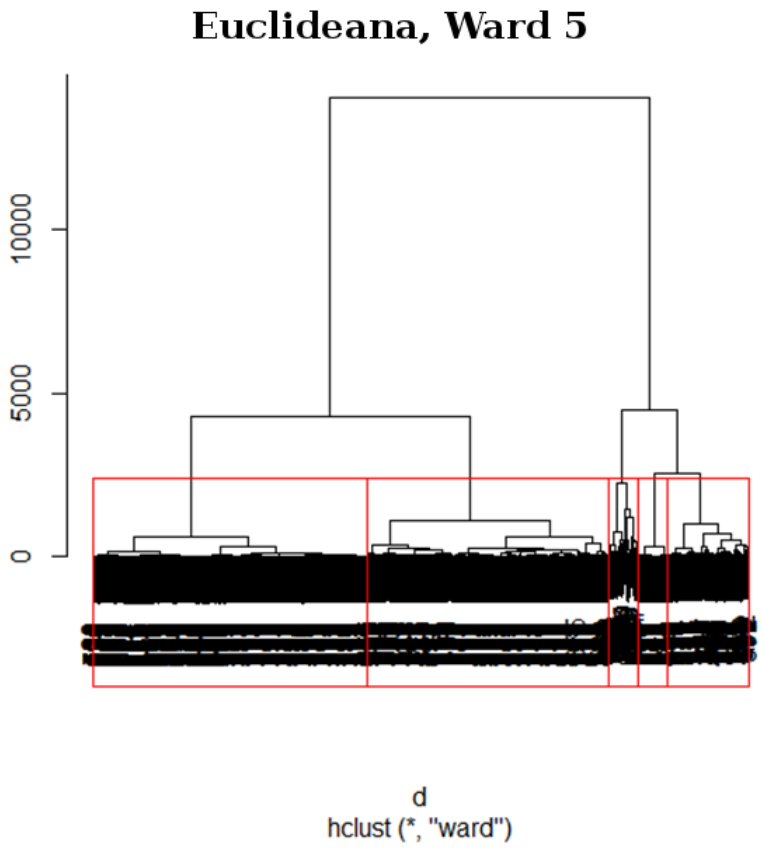


Figura G.3: Dendograma del agrupamiento jerárquico métrica Euclidiana, método Ward.

$$\begin{bmatrix} \delta(X_1,X_1) & \delta(X_1,X_2) & \delta(X_1,X_3) & \delta(X_1,X_4) & \delta(X_1,X_5) \\ \delta(X_2,X_1) & \delta(X_2,X_2) & \delta(X_2,X_3) & \delta(X_2,X_4) & \delta(X_2,X_5) \\ \delta(X_3,X_1) & \delta(X_3,X_2) & \delta(X_3,X_3) & \delta(X_3,X_4) & \delta(X_3,X_5) \\ \delta(X_4,X_1) & \delta(X_4,X_2) & \delta(X_4,X_3) & \delta(X_4,X_4) & \delta(X_4,X_5) \\ \delta(X_5,X_1) & \delta(X_5,X_2) & \delta(X_5,X_3) & \delta(X_5,X_4) & \delta(X_5,X_5) \end{bmatrix} \tag{G.7}$$

La tabla G.2 presenta las diámetros de cada uno de los cinco grupos mediante tres criterios (completo, promedio y centroide).

La siguiente matriz presenta los valores de las distancias entre grupos obtenidas para el

ejemplo dado :

$$\begin{bmatrix} & c1 & c2 & c3 & c4 & c5 \\ c1 & 0.0000 & 121.9590 & 273.3222 & 871.1544 & 158.0222 \\ c2 & 121.9590 & 0.0000 & 275.4487 & 864.6878 & 167.1227 \\ c3 & 273.3222 & 275.4487 & 0.0000 & 849.9847 & 247.5035 \\ c4 & 871.1544 & 864.6878 & 849.9847 & 0.0000 & 829.2961 \\ c5 & 158.0222 & 167.1227 & 247.5035 & 829.2961 & 0.0000 \end{bmatrix} \tag{G.8}$$

También se puede representar de la siguiente forma la matriz de distancia:

$$\begin{bmatrix} 0 & 121.9590 & 273.3222 & 871.1544 & 158.0222 \\ - & 0.0000 & 275.4487 & 864.6878 & 167.1227 \\ - & - & 0.0000 & 849.9847 & 247.5035 \\ - & - & - & 0.0000 & 829.2961 \\ - & - & - & - & 0.0000 \end{bmatrix} \tag{G.9}$$

Existen 10 valores de distancias, ya que la matriz es simétrica y en la diagonal las distancias son cero, ya que la distancia es una función simétrica la distancia entre el mismo objeto es cero. La tabla G.3, muestra los valores de la distancia entre grupos.

Tabla G.3: Valores de la distancia entre grupos.

1	2	3	4	5	6	7	8	9	10
c1 y c2	**c1 y c3**	**c1 y c4**	**c1 y c5**	**c2 y c3**	**c2 y c4**	**c2 y c5**	**c3 y c4**	**c3 y c5**	**c4 y c5**
121.96	273.32	871.15	158.02	275.45	864.69	167.12	849.98	247.5	829.3

En base a la tabla G.2, se elige como criterio el promedio para calcular el diámetro de cada grupo, obteniendo la tabla G.4.

Tabla G.4: Promedio de la distancia entre todos los objetos y el centro del grupo.

14.52	37.53	86.88	260.34	24.72

El valor máximo de los cinco valores resulta : 260.34

Con los 10 valores de distancia entre grupos mostrados en la tabla G.3 y el mayor diámetro se obtienen diez valor.

Se aplica la siguiente fórmula, la cual nos dará como resultado los índices de Dunn :

$$\frac{\delta(X_i, X_j)}{\max\limits_{1\leq k\leq c}\{\Delta(x_k)\}} \tag{G.10}$$

De este análisis resulta que el índice con un mejor valor del índice de Dunn es :

$Dunn = 0.47$

Índice de Davies-Bouldin

Considerando la fórmula del índice de Davies-Bouldin DB(U) la cual se puede definir como :

$$DB(U) = \frac{1}{c}\sum_{i=1}^{c} \max_{i\neq j}\left\{\frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i, X_j)}\right\} \tag{G.11}$$

Dónde los valores pequeños corresponde a los mejores grupos, es decir los grupos son compactos y sus centros estan alejados de los demás grupos. La configuración de grupos que minimiza DB es tomada como el óptimo número de grupos, C.

Para nuestro ejemplo se retoma la tabla G.4, que representa el el promedio doble de la distancia entre todos los objetos del grupo.

Para calcular los diez valores del numerador en la siguiente fórmula :

$$\max_{i\neq j}\left\{\frac{\Delta(X_i) + \Delta(X_j)}{\delta(X_i, X_j)}\right\} \tag{G.12}$$

Se obtiene la tabla G.5 a partir de la tabla G.2 y se divide entre las 10 distancias entre los 5 grupos (tabla G.6). De esto resulta la tabla G.7.

Tabla G.5: $\Delta(X_i) + \Delta(X_j)$

11	2	52.04
1	3	101.4
1	4	274.86
1	5	39.23
2	3	124.41
2	4	297.87
2	5	62.24
3	4	347.23
3	5	111.6
4	5	285.06

entre la tabla G.6:

Como resultado se obtiene la tabla G.7:

El mejor valor del índice de Davies-Bouldin resulta :

Índice de $DB = 0.45$.

Tabla G.6: $\delta(X_i, X_j)$

1	c1 y c2	121.96
2	c1 y c3	273.32
3	c1 y c4	871.15
4	c1 y c5	158.02
5	c2 y c3	275.45
6	c2 y c4	864.69
7	c2 y c5	167.12
8	c3 y c4	849.98
9	c3 y c5	247.5
10	c4 y c5	829.3

Tabla G.7: $\frac{\Delta(X_i)+\Delta(X_j)}{\delta(X_i,X_j)}$

0.43
0.37
0.32
0.25
0.45
0.34
0.37
0.41
0.45
0.34

Apéndice H

Artículos desarrollados

H.1. Artículo : Desarrollo de una aplicación destinada a la clasificación de información textual y su evaluación por simulación.

El artículo *Desarrollo de una aplicación destinada a la clasificación de información textual y su evaluación por simulación* fue publicado en la revista Administración y Organizaciones, perteneciente a la Univerisidad Autónoma Métropolitana, Unidad Xochimilco. A continuación se presenta la referencia completa :

Cristal Karina Galindo Durán, Carlos Avilés Cruz, Mihaela Juganaru-Mathieu y Héctor Javier Vázquez. **"Desarrollo y simulación de una aplicación destinada a la toma de decisiones basada en un clasificador de información textual"**, Revista No. 25, ISSN 1665-014X, UAM-Xochimilco, indexada en latindex. 2012.

http://bidi.xoc.uam.mx/fasciculos_revista.php?id_revista=9

Administración

Desarrollo de una aplicación destinada a la clasificación de información textual y su evaluación por simulación

Cristal Karina Galindo Durán[1]
Mihaela Juganaru-Mathieu[2]
Carlos Áviles Cruz[3]
Héctor Javier Vázquez[4]

Resumen

En el presente trabajo se propone una aplicación de cómputo destinada a la clasificación de documentos textuales, basada en el algoritmo de los K vecinos más próximos. Después de presentar brevemente el diseño y las funcionalidades de la aplicación, se presentan los resultados de la simulación de pruebas de clasificación de documentos.

[1] Maestría en Ciencias de la Computación, División de Ciencias Básicas e Ingeniería (DCBI), Universidad Autónoma Metropolitana, Unidad Azcapotzalco (UAM-A). Ave. San Pablo 180, Col. Reynosa Tamaulipas, México D.F., C.P. 02200, Tel. 55-26-72-66-76, cdgalindod@gmail.com
[2] Laboratoire en Sciences et Technologies de l'Information; Institut H. Fayol, École Nationale Supérieure des Mines de St Étienne, 42023 ST ÉTIENNE Cedex 2, France, mathieu@emse.fr
[3] Depto. de Electrónica,DCBI, UAM-A. Ave. San Pablo 180, Col. Reynosa Tamaulipas, México D.F., C.P., 02200, Tel. 5318-9550 (ext 1026), Fax 5394 6843, caviles@correo.azc.uam.mx ,
[4] Depto. de Sistemas, DCBI, UAM-A. Ave. San Pablo 180, Col. Reynosa Tamaulipas, México D.F., C.P. 02200, Tel. 5318-9532 (ext 109), Fax 5394 4534, hjv@correo.azc.uam.mx

Abstract

In the present work a software application is proposed to perform classification of textual documents, based on the use of the K Nearest Neighbors. The design and the functionalities of the software application are presented as well as the results of simulation tests to classify documents.

Palabras clave: clasificación, información textual, K-vecinos más próximos, simulación
Key words: classification, textual information, K-nearest neighbors, simulation

I. Introducción

Es difícil imaginar una organización sin un sistema de información formal o informal. Hoy en día, muchos sistemas de información se han computarizado con la finalidad de que diferentes niveles de una organización compartan gran cantidad de información relacionada con sus diferentes actores (internos o externos) y con sus múltiples actividades, en particular aquellas relacionadas con la toma de decisiones.

Para el desarrollo de estos sistemas de información, diversos autores (Ackoff, 2003, Martin, 2009; Moreno, 2000) resaltan la importancia de obtener claridad en cuanto a los objetivos del sistema de información y a los distintos conceptos relacionados con la información y su manejo; ya que esto resulta fundamental para obtener una mayor definición de las funciones, funcionalidades e, incluso, de la arquitectura del sistema de información. Vázquez (2007), apoyado en la propuesta de Ackoff (1989), recuerda la importancia de distinguir entre datos, información y conocimiento. Los datos son símbolos, o unidades "atómicas", producto de la observación; los mismos representan objetos, eventos y propiedades; al establecer asociaciones entre los datos, por ejemplo preguntando, ¿quién?, ¿qué?, ¿dónde?, ¿cuándo? y ¿cuánto?, se genera información útil para decidir qué hacer, pero no, para decidir cómo hacerlo. Las respuestas a ¿cómo? constituyen el conocimiento. Sin embargo, es importante aclarar que no necesariamente a partir de la información o del conocimiento se pueden obtener los datos o la información que permitieron su construcción (Carlisle, 2007), pues al relacionar, esto es, al interactuar, los datos o la información entre sí, puede surgir un nuevo subsistema conceptual con nuevas propiedades, no siempre identificables en los elementos aislados que lo formaron.

En lo que respecta al manejo de información, los procedimientos de clasificación y de agrupación resultan fundamentales en los procesos de toma de decisiones, ya que ayudan a identificar, distinguir, e incluso a establecer, criterios para evaluar distintas alternativas para tratar determinada situación o problemática. Un ejemplo de clasificación es cuando se cuenta con una base de datos o información de perfiles de clientes en la que se consideran distintos atributos relacionados con su capacidad de pago. Cada perfil es una clase. Si otros clientes solicitan un préstamo hipotecario, el proceso de clasificación consiste en determinar la clase a la que pertenecen y decidir si se les otorga o no el préstamo. Si no se cuenta con información previa el proceso consiste en agrupar las nuevas solicitudes de préstamo tomando en cuenta los atributos relacionados con su capacidad de pago, identificar los grupos y establecer una jerarquía entre los grupos.

Del entendimiento de los conceptos relacionados con la información y su manejo puede surgir, por ejemplo, un sistema destinado sólo a la memorización de datos, un sistema de administración de base de datos que permita generar información, un sistema de clasificación, un sistema de comunicación o bien un sistema "experto" que pueda responder algunas preguntas de tipo ¿cómo? En este artículo se propone desarrollar una aplicación para el manejo de datos e información textual.

En la actualidad existen grandes avances en cuanto a los sistemas de información para el manejo de datos e información numérica. Sin embargo, aún se requieren

esfuerzos para desarrollar sistemas para el manejo de información textual. En una organización, es frecuente, observar gran cantidad de datos, información y conocimientos en forma textual dispersos en documentos impresos, muchos de éstos olvidados en los archivos de la empresa. Cuando se encuentran documentos en forma digital, resulta que algunos de éstos fueron hechos con programas que ya no existen, con formatos obsoletos o archivados en soportes magnéticos distintos, esto es, no es fácil recuperarlos. Por esta dispersión de la información y por la poca integración de los sistemas de información se generan procesos de decisión poco eficaces y eficientes (Ackoff, 2003).

En particular, en el presente trabajo, se propone una aplicación para el manejo de información destinada a la clasificación de documentos.

En la sección II se presenta información general sobre el proceso de clasificación y en particular sobre el algoritmo de los K vecinos más próximos.

En la sección III se presenta el diseño de la aplicación propuesta y las diferentes etapas del proceso de clasificación:

- Creación de la base de entrenamiento
- Ingreso de los documentos a clasificar
- Aplicación del algoritmo de los K-vecinos más próximos para clasificar cada documento en las clases predefinidas en la base de entrenamiento

En la sección IV se estudian opciones para evaluar el desempeño del clasificador. Es decir, para un conjunto de documentos dado, evaluar si éstos quedan en las clases preestablecidas. Sin embargo es necesario considerar, para su validación, distintos factores, como el número de documentos en la base de entrenamiento, la cantidad de clases de éstos y el total de los vecinos más próximos. De aquí la necesidad de realizar pruebas por simulación. En este artículo se entiende por simulación el proceso de diseño del modelo de un sistema y la realización de experiencias para evaluar y comprender el funcionamiento de éste (Shannon, 1988).

II. Proceso de clasificación

La clasificación es un proceso de categorización de información. Para establecer las categorías, se requiere elaborar una base de entrenamiento previa, la cual contiene la información de las diferentes clases. Si se cuenta con un objeto y se desea saber si pertenece a una de las clases, se identifican los atributos más significativos y se evalúa qué tan semejantes son a los atributos de la base de entrenamiento. En el caso que existan varias opciones de clases, se establece una jerarquía para poder decidir la clase a la que será asignado el objeto (Hernández, 2004). Para la identificación del elemento por clasificar, se aplican diferentes métricas (medidas de distancia) que indican qué tan similar es un objeto con respecto de alguna clase de la base de entrenamiento. Si las características son numéricas, existen diversas métricas; la más utilizada es la distancia euclidiana; sin embargo, en el caso de información textual en donde sólo es posible contar las propiedades, éstas son discretas y se necesita establecer otras funciones (Feldman y Sanger, 2007).

A lo largo de este proceso, el usuario interviene y da seguimiento a las diferentes etapas. Es decir, el proceso se produce de manera supervisada.

Los diferentes algoritmos de clasificación se pueden agrupar en tres grandes tipos:

- Los clasificadores paramétricos se basan en la estimación de parámetros de las distribuciones de probabilidad que representan las clases de estudio mediante distribuciones multinomiales, mezcla de multinomiales o por ejemplo una combinación de modelos (Bernouilli, gaussiano y multinomial) (Mesa, 2008) .
- Los clasificadores no paramétricos son aquellos que se basan en la estimación directa sobre la probabilidad a posteriori de pertenecer a una clase. Uno de los clasificadores más simples es el clasificador de los vecinos más próximos, llamado comúnmente KNN (*K-Nearest Neighbors*). Éste consiste en establecer el número de vecinos más próximos del objeto por clasificar (Moreno, 2004)
- Los clasificadores artificiales son clasificadores que se basan en la aplicación de diferentes técnicas de inteligencia artificial para el reconocimiento de patrones; el más conocido es el modelo del perceptrón generalizado o multicapa (Barandela, 2001)

En este trabajo se usará el clasificador no paramétrico de los K-vecinos más próximos, debido a que es uno de los algoritmos más sencillos.

El método consiste en establecer la cantidad de vecinos más próximos del objeto por clasificar dentro de la base de entrenamiento. Cuando el nuevo objeto se presenta al sistema de aprendizaje, éste se clasifica según la distancia más cercana (Mora, 2008). Los vecinos más próximos a un objeto se obtienen, en caso de atributos numéricos, mediante diferentes distancias sobre los *n* posibles atributos. La mejor elección de los *k* depende fundamentalmente de los datos; generalmente, valores grandes de los *k* reducen el efecto de ruido en la clasificación, pero crean límites entre clases parecidas (Clark y Boswell, 2000).

III. Desarrollo de la aplicación de clasificación

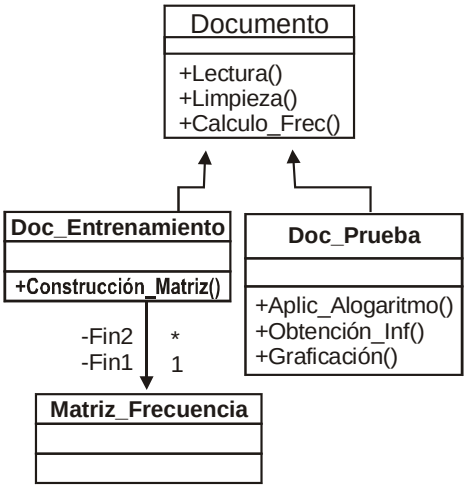
Para realizar esta aplicación se utilizó Matlab,[5] el cual es un *software* matemático que ofrece un entorno de desarrollo integrado (IDE) con un lenguaje de programación interpretado propio (llamado lenguaje M). El programa Matlab permite manejar diferentes estructuras de datos, generar resultados en forma gráfica de calidad, disponer de subrutinas probadas y bibliotecas de funciones preestablecidas. Esto resulta útil para el desarrollo rápido de los algoritmos y demostración de aplicaciones de cómputo. Es importante señalar que esta aplicación forma parte del estudio de factibilidad para el desarrollo, en lenguaje Java,[6] de una aplicación de manejo de información a gran escala para 60 000 documentos (Galindo, 2011).

En esta aplicación se propone un diseño considerando tres clases: **Documento**, **Doc_Entrenamiento**, y **Doc_Prueba** las cuales se relacionan de acuerdo con la Figura 1.

- La clase **Documento** posee un método de *Lectura* que permite leer los archivos de texto plano (*Lectura*) y un método de *Limpieza*, el cual permite eliminar símbolos y caracteres poco relevantes para el usuario. El método de cálculo de frecuencias (*Calculo_Frec*) invoca al

[5] http://www.mathworks.com/
[6] htp://www.java.com y un IDE http://netbeans.com

Figura 1.
Diagrama de clases con sus principales métodos.



método construcción de la matriz de frecuencias de la clase **Doc_Entrenamiento.**

- La clase **Doc_Entrenamiento** contiene el método construcción de la matriz de frecuencias (*Construccion_Matriz*) a partir del cual se aplica el método para generar la matriz de frecuencias.
- La clase **Doc_Prueba** posee el método *Aplic_Algoritmo* el cual permite construir la matriz de distancias y aplicar el método de K vecinos más próximos. En el método *Obtencion_Inf* se define a que clase pertenece el objeto a clasificar (en este caso un documento). El método de *Graficacion* permite visualizar los resultados en forma de grupos.

En la figura 2 se presenta el funcionamiento de la aplicación tomando como base estas clases y la definición del proceso de clasificación. Con el fin de presentar las principales funcionalidades de esta aplicación, y por la facilidad de acceso a documentos, se decidió usar documentos de texto de una organización editorial, la cual desea clasificar artículos de revistas en distintas categorías. Para fines demostrativos se supone que desean clasificar esos artículos (definición de atributos), tomando como atributos los signos de puntación y signos aritméticos, en una de las siguientes categorías: Sociales (clase 1), Ingeniería (clase 2) y Medicina (clase 3).

Figura 2.
Proceso de clasificación

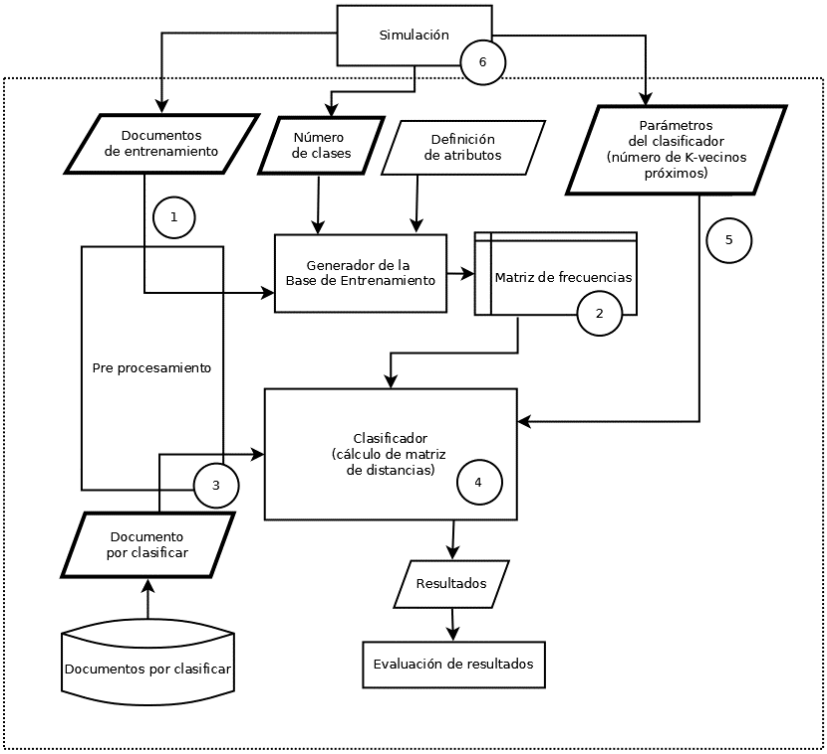


Tabla 1.
Matriz de frecuencias

Documentos	**.**	**,**	**“**	**(**	**)**	**?**	**!**	**-**	**:**	**;**	**<**	**>**	**+**	*****	**/**	**=**	***# Pala***	***Clase***
1	413	694	0	229	228	0	0	65	71	81	3	5	31	2	5	87	8170	1
2	315	736	0	46	46	4	0	54	23	37	0	0	0	0	13	0	8179	1
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
50	483	866	0	113	116	1	0	51	46	28	0	0	0	2	11	0	10871	1
51	217	183	0	57	59	0	0	72	25	11	0	0	0	0	7	6	6450	2
52	738	345	0	243	244	0	0	33	68	26	1	1	0	0	48	24	7399	2
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
100	507	275	0	110	118	0	0	09	29	11	0	1	2	18	10	10	9053	2
101	588	379	0	110	115	1	0	35	86	44	27	2	0	60	27	53	5503	3
102	390	305	0	55	55	3	0	43	59	25	0	0	0	2	16	6	5695	3
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.	.
150	377	356	0	46	57	3	0	93	83	77	0	0	0	0	4	0	5196	3

En la primera etapa (etapa 1 en la figura 2), que se realiza sólo una vez, la aplicación inicia con la lectura de la información previa (documentos de entrenamiento) y el procesamiento supervisado de los documentos. Una vez que el usuario considera que los documentos están listos se obtiene la matriz de frecuencias (etapa 2 en la figura 2), desarrollada en la tabla 1, con las características establecidas. Esta matriz de frecuencias es la base de conocimiento previo, llamada comúnmente “base de entrenamiento”.

En la segunda etapa (etapa 3 en la figura 2), que se repite para cada documento por clasificar, el usuario proporciona el documento por clasificar y se generan las frecuencias, descritas en la tabla 2 tomando en cuenta las mismas características usadas para construir la base de entrenamiento.

Tabla 2.
Frecuencias del documento por clasificar

.	**,**	**“**	**(**	**)**	**?**	**!**	**-**	**:**	**;**	**<**	**>**	**+**	*****	**/**	**=**	**# Pala**
422	374	0	120	120	0	0	297	54	46	0	0	2	0	235	49	6645

Una vez obtenida la matriz de frecuencias y las frecuencias del documento por clasificar se procede al cálculo de la matriz de distancias (etapa 4 en la figura 2) descrita en la tabla 3.

Tabla 3.
Matriz de distancias

Distancia	***Clase***
0.2445	1
0.1311	1
.	.
.	.
.	.
0.5610	2
0.5450	2
.	.
.	.
.	.
0.5363	3
0.5362	3

Posteriormente se ingresa el número de vecinos más próximos que se desea obtener (etapa 5 en la figura 2) y en base al número indicado, se procede a obtener las distancias menores. Una vez seleccionados, se contabiliza, para decidir en qué clase aparece el mayor número de atributos y se obtiene así la clase en la que se clasifica el documento

Tabla 4.
Matriz de distancia con los 10 vecinos más cercanos

Distancia (x 10^{-3})	***Clase***
0	**3**
306.8241	**3**
347.7959	1
369.1734	2
389.6858	2
404.6863	1
410.8771	3
421.5780	1
426.9344	3
439.1070	2

Para este ejemplo se propusieron 10 vecinos (aunque se recomienda usar un número de vecinos impar para evitar ambigüedades), los cuales se muestran en la tabla 4. Como resultado, el programa nos indica a qué clase pertenece el documento, es decir, si pertenece a Sociales, Ingeniería o Medicina. Para este ejemplo en particular el documento pertenece a la clase 3, es decir, a Medicina.

IV. Evaluación de la aplicación

Para realizar la evaluación del desempeño de un clasificador existen diversas estrategias, las más utilizadas son (Mesa, 2008):

- Re sustitución
- Validación simple
- Validación cruzada (*Cross- Validation*)

1.- Re sustitución

Es una técnica de evaluación de desempeño de la clasificación en donde la misma base de datos de aprendizaje se usa para la prueba. Este tipo de evaluación proporciona una medida optimista de clasificación con un mínimo error.

2.- Validación simple

Esta técnica (también se conoce como *hold out*.) de evaluación de desempeño es una parte integral del proceso de entrenamiento, el cual consiste en dividir los datos en tres grupos:

- Conjunto de datos de entrenamiento
- Conjunto de datos de validación
- Conjunto de datos de prueba

3.- Validación cruzada

En esta técnica, también llamada "deja los k fuera" (*leave k out*) o "manten los k fueras" (*hold k out*), se toma un porcentaje en forma aleatoria (del total de la base), para el aprendizaje y resto para la prueba. Generalmente el porcentaje tanto para el aprendizaje como para la prueba es del 50%. El tiraje y prueba aleatorio se repite un cierto número de veces. El resultado final será el promedio sobre las n-realizaciones.

Como resultado de aplicar un método de validación, se obtiene una matriz de confusión. La matriz de confusión ideal presenta una matriz con "unos" en la diagonal, como se presenta en la tabla 5.

Tabla 5.
Matriz de confusión ideal

Clase 1	*Clase 2*	...	*Clase n*
1	**0**	...	**0**
0	**1**	...	**0**
0	**0**	...	**0**
0	**0**	...	**1**

V. Pruebas de simulación y resultados

Las pruebas se realizan mediante simulación (punto 6 en la figura 2), utilizando los mismos documentos de prueba. Para el experimento se consideró el número de documentos en la base de entrenamiento, el número de clases de éstos y el total de vecinos más próximos. En estas pruebas, la selección de los documentos de prueba se realiza en forma secuencial, sin embargo también es posible realizarla en forma aleatoria. El método utilizado para la evaluación del desempeño de esta aplicación es el método de validación cruzada, el cual toma para cada una de las clases 30 documentos de cada clase para la realización de la prueba.

Un ejemplo de la matriz de confusión obtenida se presenta en la tabla 6.

Tabla 6.
Matriz de confusión obtenida con 80 documentos de entrenamiento, 15 vecinos y 3 clases

	Clase 1	**Clase2**	**Clase 3**
Clase 1	0.86	0	0.14
Clase 2	0	1	0
Clase 3	0.07	0	0.93

Tabla 7.
Resultados de la clasificación

	Documentos en cada clase de la base de entrenamiento																	
	50						**80**						**125**					
Clases de entrenamiento	**2** (sociales e ingeniería)			**3** (sociales, ingeniería y medicina)			**2** (sociales e ingeniería)			**3 (**sociales, ingeniería y medicina)			**2** (sociales e ingeniería)			**3** (sociales, ingeniería y medicina)		
Vecinos más próximos del objeto por clasificar	**5**	**15**	**25**	**5**	**15**	**25**	**5**	**15**	**25**	**5**	**15**	**25**	**5**	**15**	**25**	**5**	**15**	**25**
Resultado	0.75	0.8	**0.90**	0.74	0.82	**0.90**	0.85	**0.91**	**0.92**	0.81	**0.90**	**0.91**	0.83	**0.92**	**0.94**	0.86	**0.90**	**0.93**

Los resultados, modificando el número de documentos de la base de entrenamiento, la cantidad de clases de la base de entrenamiento y el total de vecinos más próximos del documento por clasificar, se presentan en la tabla 7.

El resultado se mide con la razón del número de documentos bien clasificados (es decir aquellos que se encuentran en la clase a la que pertenecen) respecto del número total de documentos por clasificar. Por ejemplo, si se consideran dos clases y se desea clasificar 60 documentos (en teoría 30 para cada clase), considerando 5 vecinos más próximos, se obtiene, después de aplicar el algoritmo, que sólo 50 fueron bien clasificados. El resultado aparece entonces dividiendo 50 entre 60, es decir, 0.75.

De los resultados de esta simulación, ver tabla 7, se observa que, entre mayor sea el tamaño de la base de entrenamiento y mayor el número de vecinos más próximos, se obtiene una mejor clasificación de los documentos.

VI. Conclusiones

El presente trabajo presenta una aplicación enfocada a la clasificación de documentos de texto no estructurado o texto plano. Este es un caso que no considera la jerarquía o la importancia del contenido de los documentos; en realidad la información puede presentarse en diferentes niveles de importancia y no necesariamente tiene el mismo valor en la organización. Para incluir el nivel de importancia de la información contenida en los documentos, sería necesario considerar un formato de texto, como el formato XML (Harold, 2004). Este formato es ampliamente usado en la actualidad por los beneficios que presenta; por ejemplo, la posibilidad de estructurar contenido con distintos niveles de importancia y distinguir así entre las diferentes jerarquías de información textual.

Es importante resaltar que la clasificación requiere de información previa, la cual no siempre está disponible

en una organización. En este caso el proceso de agrupación resulta más adecuado ya que no necesita conocimiento previo.

En lo que respecta a las pruebas de simulación, se observa que, para poder llevar a cabo una buena clasificación, es importante partir de una buena base de entrenamiento y para ello se aconseja contar con el mismo número de elementos en cada una de las clases, ya que si alguna clase cuenta con mayor número de elementos, el algoritmo de clasificación parece ser menos efectivo, favoreciendo a la clase que posea mayor número de elementos.

Por supuesto, hay que considerar los resultados de esta simulación con reserva, ya que los documentos de prueba se seleccionaron de forma secuencial, y esto genera resultados deterministas. Debido a la dificultad de disponer de datos de entrenamiento representativos para simular todas las situaciones posibles, es importante incluir aleatoriedad en el proceso: por ejemplo tomar documentos de manera aleatoria, definir atributos en forma aleatoria, etc. Otra opción sería comparar los resultados usando otros métodos de validación.

En la bibliografía, se reportan diversas aplicaciones para la clasificación de documentos (Téllez, 2005). Sin embargo, esta experiencia ha permitido realizar una aplicación funcional, obtener una mayor comprensión del proceso de construcción de matrices de frecuencias y de distancias a partir de información textual; y contar con una aplicación que permita realizar experimentos de simulación para evaluar los efectos de diversos parámetros sobre el resultado de la clasificación. Estos resultados son una etapa importante en el estudio de factibilidad y es parte fundamental para el desarrollo de una aplicación de gran escala para tratar documentos XML que posean una estructura interna (Galindo, 2011).

Agradecimientos

Mihaela Juganaru-Mathieu agradece el apoyo de la Doctora Silvia González Brambila, Coordinadora de la Maestría en Ciencias de la Computación, de la División de Ciencias Básicas e Ingeniería (DCBI), de la Universidad Autónoma Metropolitana, Unidad Azcapotzalco (UAM—A), así como el del Doctor Nicolás Domínguez Vergara, profesor y ex jefe del Departamento de Sistemas, de la misma División.

Héctor Javier Vázquez agradece al doctor Russell L. Ackoff (1919, 2009) por todas sus enseñanzas sobre los diferentes conceptos de sistemas, por sus publicaciones, conferencias, cursos, investigaciones, pláticas y conversaciones personales. Así mismo agradece al profesor Germán Sergio Monroy Alvarado por su amistad, su paciencia y motivación continua para estudiar las metodologías de sistemas. Finalmente, Héctor Javier Vázquez no deja de agradecer, por el apoyo otorgado para realizar sus estudios de Maestría y Doctorado en Francia. En particular agradece, a la Universidad Autónoma Metropolitana, Unidad Azcapotzalco, por las distintas licencias sin goce de sueldo, concedidas; y por otro lado, agradece, por el apoyo económico otorgado, al Ministerio de Educación Superior y de la Investigación (MESR) y a la Agencia Nacional del Empleo de Francia (ANPE), ambas instituciones del Gobierno de Francia.

Bibliografía

• Ackoff, R. L. (1989). *From Data to Wisdom*, Journal of Applied Systems Analysis, Vol. 16.

• Ackoff, R. L. (2003). *Redisigning Society*, Stanford University Press, Stanford.

• Barandela, G. E. (2001). *Corrección de la muestra para el aprendizaje del perceptrón multicapa*. Revista Iberoamericana de Inteligencia Artificial, 2-9.

• Galindo, D. C. K., Juganaru-Mathieu M. y Vázquez H. J. (2011). *Specification Design for an XML Mining Configurable Application*, International MultiConference of Engineers and Computer Scientists (aceptada para su presentación y publicación, http://www.iaeng.org/IMECS2011/publication.html).

• Carlisle, J. P. (2007). *A Look into the Relationship Between Knowledge Management and the Knowledge Hierarchies*, Proceedings of the 40th Hawaii International Conference on System Sciences (http://www.computer.org/portal/web/csdl/doi/10.1109/HICSS.2007.19).

• Clark, P., y Boswell, R. (2000). *Data Mining. Practical Machine Learning Tools and Tecniques with Java Implementations*. Morgan Kaufmann Publishers.

• Feldman, R., y Sanger, J. (2007). *The Text Mining Handbook. Advanced Aproaches in Analyzind Unstructured Data*. Cambridge University Press.

• Harold, E. R., y Means, W. S. (2004). *XML in a Nutshell*. O'Reilly Media.

• Hernández, J., Ramírez Quintana, M. J., y Ramírez F. C. (2004). *Introducción a la Minería de Datos*. Pearson Prentice Hall.

• Martin, J. N. (2009). *Knowledge Generation in the Enterprise Using Information and Data Systems*. Second International Symposium on Engineering Systems, MIT, Cambridge, Massachusetts, Junio 15-17, (http://esd.mit.edu/symp09/submitted-papers/martin-paper.pdf).

• Mesa, F. D. (2008). *Algoritmos de Aprendizaje Continuo Mediante Selección de Prototipos para Clasificadores Basados en Distancias*. Tesis de la Universitat Jaume I, Castellón, España.

• Mora, F., J., Morales España, G., y Barrera Cárdenas, R. (2008). *Evaluación del clasificador basado en los K Vecinos más Cercanos para la Localización*. Ingeniería e Investigación , 81-86.

• Moreno, F. S. (2004). *Clasificadores Eficaces Basados en Algoritmos Rápidos de Búsqueda del Vecino más Cercano*. Tesis de Doctorado de la Universidad de Alicante, España.

• Moreno, O. A. (2000). *Diseño e Implementación de un Lexicón Computacional para Lexicografía y Traducción Automática*. Estudios de Lingüística del Español, Vol 9. http://elies.rediris.es/

• Shannon, R. E. (1988) *Simulación de Sistemas*, Trillas, Ciudad de México, D.F.

• Téllez, V. A. (2005) *Extracción de Información con Algoritmos de Clasificación,* Tesis de Maestría del Inaoe (http://ccc.inaoep.mx).

• Vázquez H. J., Martínez A. F. J., Monroy A. G. S. (2007). *Más allá del Conocimiento: un Enfoque Sistémico*. Administración y Organizaciones, 23-38.

H.2. Artículo : Specification Design for an XML Mining Configurable Application.

El artículo *Specification Design for an XML Mining Configurable Application fue presentado en The International MultiConference of Engineers and Computer Scientists* (IMECS) en 2011. A continuación se presenta la referencia completa :

Cristal Karina Galindo Durán, Mihaela Juganaru-Mathieu y Héctor Javier Vázquez. **"Specification Design for an XML Mining Configurable Application"**. The International MultiConference of Engineers and Computer Scientists 2011 llevada acabo en Hong Kong, 16-18 de Marzo, 2011, http://www.iaeng.org/IMECS2011/

Specification Design for an XML Mining Configurable Application

Cristal Karina Galindo Durán, Mihaela Juganaru-Mathieu and Héctor Javier Vázquez

***Abstract*—This work presents a methodology for XML mining centered on the process of extracting document features related to structure and content. This information is used to obtain similarity measures and to cluster XML documents. A conceptual framework is proposed to design an application with the primary goal of implementing a modular and easily configurable tool for mining large XML document collections.**

***Index Terms*—conceptual framework, clustering, modularity, XML mining, UML.**

I. Introduction

CUrrently XML [1], [2] plays a very important role in the development of applications in which textual information is abundant, because it allows for storage of structured and/or semi-structured information [3] independently of any final presentation. For situations in which textual information is more abundant than numbers, traditional techniques of information storage and retrieval, such as relational databases, are poorly adapted. XML also offers compatibility between different systems and platforms and in different applications; it provides the ability to share information in a reliable and easy manner; it is the basis for the development of online applications and e-commerce; and it can be used for creating collections of documents.

However, direct use of an XML collection is difficult because it requires an understanding of technologies such as XSLT (eXtensible Stylesheet Language Transformations) and CSS (Cascading Style Sheets). To manage information stored in XML and to extract information not explicit in the collections of data, research has been undertaken on a process known as "XML mining"

The aim of the present work is to propose a framework for the design of a modular and configurable application for XML mining, in particular for document clustering in large collections such as INEX[1] (INitiative for the Evaluation of XML Retreival), composed of semi-structured textual data. This framework is intended to lead to development of a modular and configurable application–modular in the sense that it will be easy to change modules, functions or algorithms and configurable in that it will allow the user to:

- mine various large collections;
- choose different attributes about structure, content or both;
- count frequencies or count only the presence/absence of structural or content features; and
- choose different mining algorithms.

In this last case it will enable, depending of the counting method, the use of different metrics of similarity, generate different similarity matrices, and permit change of the clustering algorithms. The first part of this paper presents a brief description of XML and an overview of XML mining. Then a conceptual design is proposed. The second part of this paper uses the conceptual diagram as a basis for implementation of the application, highlighting the use of case diagrams, classes structure, and modules of the application.

II. XML Description, Collections and Mining

XML is a standard meta-language that uses extensible tags and was developed by the World Wide Web Consortium. It is a simplification and adaptation of SGML (Standard Generalized Markup Language) and defines the specific language grammar (just as HTML is itself a language defined by SGML). Therefore, XML is not a programming language but is a language of description of the information and also a way to define other languages (XHTML, SVG, MathML) for different needs. Some advantages of using XML are:

- It is easily processable (data management) by both people and by applications.
- Content and presentation are independent.
- It is designed to be used in any language or with any alphabet.
- Parsing is easy because of the strict rules governing composition of a document.
- It uses a hierarchical structure.
- The number of names of tags is unlimited.
- There are powerful tools for linking (XLink) to other XML or other type of documents.

Considering these advantages, XML is an ideal storage format for collections of text and data containing links to other documents (within the same collection or in another collection) and references to images and other multimedia files. For example, INEX provides a large collection of articles from Wikipedia in 2007, which focuses primarily on XML documents. It currently consists of two collections: a

Manuscript received December 20, 2010; revised February 07, 2011. The authors acknowledge the financial support given to the Promep group Analysis and Information Management and to the Universidad Autónoma Metropolitana, Unidad Azcapotzalco (UAM-A) for the financial support given to the project, Study and Systems Modelling, Number: 2270217 División de Ciencias Básicas e Ingeniería, UAM A. C. K. Galindo Durán acknowledges the Consejo Mexiquense de Ciencia y Tecnología (COMECyT) for the financial support received from February to July 2010.

Cristal Karina Galindo Durán, Master's Computer Science Program, Universidad Autónoma Metropolitana, Unidad Azcapotzalco, Avenida San Pablo 180, Col. Reynosa Tamaulipas, México D.F. C.P. 02200, México, email:cdgalindod@gmail.com.

Mihaela Juganaru-Mathieu, Laboratoire en Sciences et Technologies de l'Information, Institut H. Fayol, Ecole Nationale Supérieure des Mines de Saint Étienne, 158, cours Fauriel, 42023, SAINT ÉTIENNE Cedex 2, France, email: mathieu@emse.fr.

Héctor Javier Vázquez, Departamento de Sistemas, Universidad Autónoma Metropolitana, Unidad Azcapotzalco, Avenida San Pablo 180, Col. Reynosa Tamaulipas, México D.F. C.P. 02200, México, email: hjv@correo.azc.uam.mx.

[1]http://www.inex.otago.ac.nz/

large one that has about 2.7 million documents and a smaller one that contains approximately 60,000 documents. This collection includes also bags of words that are commonly found in documents and frequencies of various structures in the form of XML tags, trees, links and names of entities.

The focus of XML mining is to develop and use techniques to extract information from XML documents. XML mining has been influenced by other fields, such as data mining, information retrieval, pattern recognition, and artificial intelligence. XML mining [4] can be defined as the process that enables discovery of new knowledge not explicit in the documents analyzed, but that is revealed once relationships are established among the contents of several structured documents.

The extraction of information is achieved by taking into account information about the structure of documents themselves, and/or their content, either textual or numerical. This does not mean that traditional techniques used for data mining in general [5] will not be useful, but it does mean that XML's multidimensional structure allows the management of information in different ways [6]. XML research has led to the development of various techniques and tools [7], [8] dealing with data frequency patterns as well as frequency patterns within parse trees.

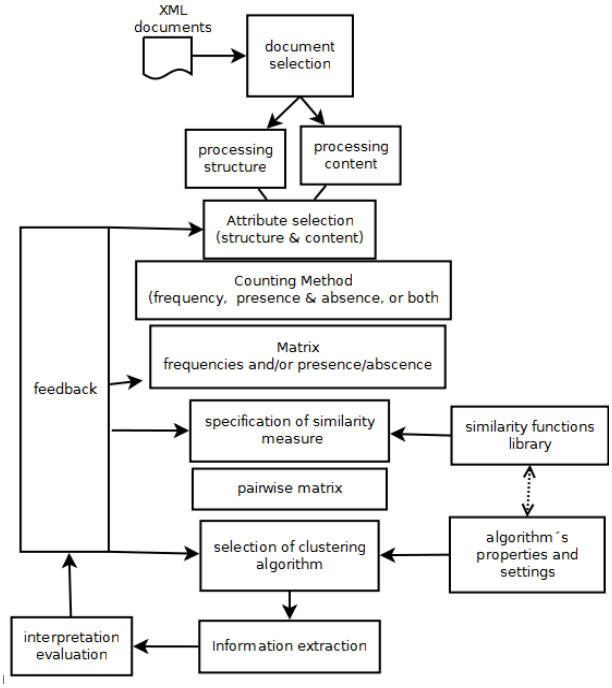


Fig. 1. Conceptual Model

III. Conceptual Design for XML Mining

The following is an overview of the process of extracting information from XML:

1) Selection set of XML documents.
2) Pre-processing of XML documents (discovery of the structure, identification of similar items, and/or removal of common characteristics).
3) Selecting and applying the XML mining technique, such as clustering.
4) Information extraction.
5) Interpreting, evaluating and, eventually, reconfiguring procedures.

Pre-processing of XML documents is necessary to identify their structure and content. For example, structure is revealed throught document parsing. This operation generates a parse tree of the XML document. Once the structure is known, content is identified by locating similar items and/or removing items with similar characteristics. During this process, all or a subset of attributes can be selected and counted (frequency and presence and/or absence). These counts can be integrated in a single table or organized in different count or frequency matrices [7]; for example, one frequency count matrix to register all documents and information about their structure (tags, markers, token path, depth or any another feature related to structure) and another matrix to register counts of tokens about documents content (composed mainly of text). Depending on the nature of the counts, there are different kinds of similarity measures, whether they are frequencies or binary data. For example, in the case of frequency counts, it is common to use the Euclidean, Mahalanobis, Minkowski and χ^2 distances [9]. In the case of the presence or absence of particular features, it is possible to assign more or less importance to presences than to absences: the Jaccard and Dice-Sorensen indexes are most often used for this purpose. However, if both presences and absences are considered to have the same importance then, the Sokal and Michener, Sokal and Sneath, Hamming, or Roux [9] indexes may be used. Once the metric is chosen, similarity matrices are obtained and then different cluster procedures can be applied such as expectation maximum (EM) based or Hierarchical algorithms.

The proposed application will be oriented to clustering and not to classification [10]. Clustering is the process of partitioning data into a set of groups, clusters, or subclasses. This process is performed without any information about possible groups and without examples or learning sets. Once the process finished, all data in the cluster share common traits [11], [12]. Given that the user is not directly involved until the end of the process, clustering is considered to be unsupervised classification. Conversely, in supervised classification, which is used to predict group membership for data instances, the user intervenes and monitors the process.

Once the clusters have been obtained, they can be parsed to find data structures and explore their characteristics,which can be interpreted and evaluated, and the mining process eventually can be repeated. The Figure 1 depicts this process with the goal of using it as a specification to develop a mining application.

IV. Software Design

From the description of the conceptual model, it might seem that XML mining is a linear procedure, however; the process must be guided by the user during selection of attributes, similarity measure alternatives, cluster algorithms and other settings. Therefore, a valuable, user-oriented application would be one that gives the user the ability to choose among different types of tokens, different types of similarity distances, and even different types of clustering algorithms. All these options can be applied first on a large collection or on a test collection. At the end of the process and during user interpretation of results, the application would "feed"

user's knowledge into the XML mining process. The Unified Modeling Language (UML[2]) is used to present different elements of the proposed design. In the following sections, use case, class diagrams and implementation of an entity relational model to store information are presented.

A. Use Case Diagrams

As shown in the conceptual model, a use case for each step (eight) and a class diagram are proposed. The use case diagrams illustrate specific aspects of interactions of the user with the application. As an example, the four user case diagrams related to the most critical functionalities are:

- Use case to import XML documents: This case presents the activities to obtain the XML collection. The user connects to the INEX site, saves the list of zip files, then extracts and stores XML documents.
- Use case for pre-processing documents: Documents should be cleaned (like removing undesired features) under expert or user predefined rules. After the structure is revealed (for example, with a parse tree) and content is studied, the next step is to obtain tokens and count them.
- Use case for pairwise matrix generation: Frequency or presence/absence counts are established for tokens about content and structure. From this information, a mixed frequency matrix is obtained, pairwise distances are calculated, and the pairwise matrix is built.
- Use case for clustering: This includes all the actions to obtain clusters. Different cluster algorithms can be used, without the necessity of changing the other modules.

The remaining use case diagrams present interactions with results evaluation and their management.

B. Class Diagram

In the class diagram, classes are first defined for the XML collection and its documents. These two classes are directly associated with a multiplicity [1,*], given that one collection has several documents. An important class is the "Distribution_frequency" which is associated with several documents and several tokens. Tokens are the objects identified in the document and may have several categories (for example, structural or grammatical). The remaining classes are related to the mining process, which applies cluster algorithms, based on a given metric to the frequency matrix (revealed by classes: Token and Distribution_frequency).

The classes proposed are: **Collection_C, Document_C, Distribution_Frequency_C, Token_C, Criteria_C, Choice_C, Metrics_C, Apply_C, Cluster_C, and Algorithm_C.**

Figure 2 shows a static view of the class diagram, with the ten classes and their associations, proposed for this application.

The design process was separated into two packages, as follows:

- package one to obtain documents, to tokenize them and store them, and to obtain the frequency matrix
- package two to deal with the mining process itself.

For the first package it is proposed to use a database management system (DBMS).

[2]http://www.omg.org/uml

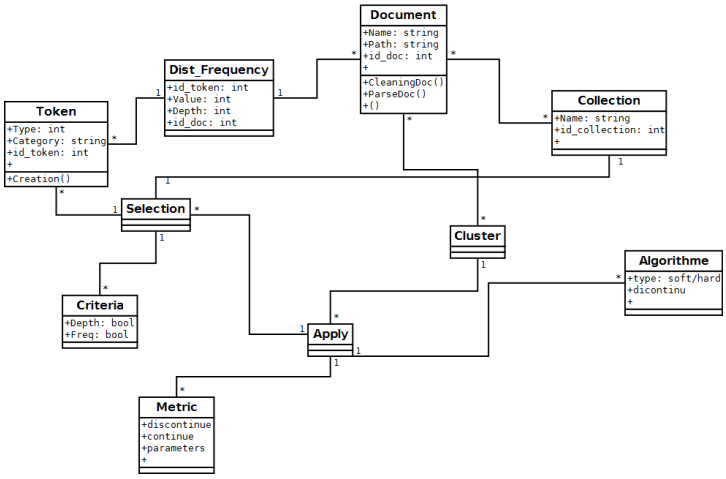


Fig. 2. Class Diagram

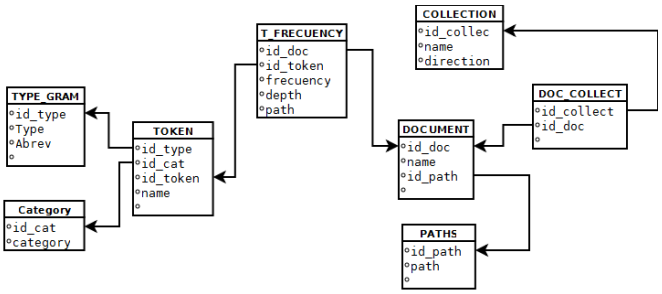


Fig. 3. Relational Model

C. Relational Model

The eight tables of the relational model are presented in Figure 3, as follows:

1) The **COLLECTION** table with a tuple for each XML document collection.
2) The **DOCUMENT** table references **PATH** table and contains a tuple for each XML document.
3) The **DOC_COLLECTION** table indicates the composition of collections. It references the **COLLECTION** table and the **DOCUMENT** table.
4) The **PATH** table indicates the computer path where the XML files are physically stored.
5) The **TOKEN** table stores a unique tuple for each possible token and references the tables **TYPE_GRAM** and **CATEGORY**.
6) The **CATEGORY** table contains a tuple for each category of a token: word, tag, link or other.
7) The **TYPE_GRAM** table indicates the possible grammatical type for a token word.
8) The **T_FREQUENCY** table indicates frequency of tokens in documents and has two references: one to the **DOCUMENT** table and a second one to the **TOKEN** table.

D. Implementation

Our major decision about the first part of this application was to store, on a DBMS, all the information about documents, tokens and frequencies in a data base. The purpose is to avoid calculating all feature frequencies (structure and content) each time the user analyzes different sets of attributes. Frequency matrices and presence/absence matrices will be directly extracted with aggregated SQL queries from the tables of the database. PostgreSQL is used for the database implementation of the DBMS.

Another decision was to use a grammatical invariable generator such as Tree Tagger[3], a part of speech tagger that obtains the grammatical category and invariable form of every word [13].

The least, but not the less important, decision was to use an object oriented language (in our case Java) to implement the whole application. This allowed us to introduce and consider new distance measures or mining algorithms. Java also offers the possibility of working with various APIs for DBMS or XML, to interface with Tree Tagger or to call mining procedures or libraries written in GNU applications such as R[4] or other languages.

The first steps are to obtain the XML collection, assign a number and identify the computer path where each document is stored. This information is stored in the DOCUMENT and PATH tables.

To track and access information stored in XML documents, there are several mechanisms. Basic mechanisms are implemented in parsers such as SAX (Simple API for XML[5]) and DOM (Document Object Model[6]). SAX tracks the document sequentially in depth while DOM randomly accesses the document and builds, in memory, a tree of the XML document, permitting the document to be parsed, in depth or in breadth, or its elements to be modified. Because DOM is less complex, it is used to obtain tags, hyperlinks and paths tags. After parsing, Tree Tagger is used to obtain content tokens. From this, a token frequency matrix is created, which forms the basis of the content and structure frequency matrices. Token frequency matrix generation requires extra storage, but, as explained previously, it was decided to implement it, given that it is simple to obtain whole tokens counts, and because it later gives the user a choice of generating the frequency or the presence/absence matrix. From another perspective, this option actually saves space, because it is not necessary to save features in text form nor separate frequency matrices (structure and content) each time the user analyzes different sets of attributes.

V. Conclusion

In this paper, we presented a UML-based software model design for implementing a modular and configurable application for XML mining and clustering in a large collection of documents. The conceptual model is a framework for understanding and following user actions to obtain the information and process it. Concerning the generation of a token frequency matrix, this choice saves space and it allows, if required, to obtain one or two separate frequency matrices, one for structure and other for content, given that each token category can be easily identified in a single table.

Althought this work is not new, it highlights the importance of developing a configurable and modular application; the research works have excluded the possibility of modifying parts of the pre-processing and mining application or changing its configuration.

The first stage (package one) of development of the software application has been completed and uses Java and PostgreSQL for implementing the database management system. This implementation has been tested with 60,000 documents, and we have been able to obtain the token frequencies. These frequencies will be the basis for generation of frequency matrices, which in turn are the basis for the second package, which focuses on the mining process.

[3]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html

[4]http://www.r-project.org/

[5]http://www.saxproject.org/

[6]http://www.w3.org/DOM/

Acknowledgments

The authors acknowledge Dr. Nicolas Dominguez Vergara (former head of the Systems Department) and Dr. Silvia González Brambila (Coordinator of the Master's Computer Science Program at the UAM Azcapotzalco) for their support and encouragement. They also acknowledge Peggy Currid for her advice in copyediting.

References

[1] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau, "Extensible markup language (XML) 1.0 (fifth edition), w3c recommendation," Available: http://www.w3.org/TR/REC-xml/, 2008.

[2] E. R. Harold and W. S. Means, *XML in a Nutshell, Third Edition*. O'Reilly Media, 2004.

[3] S. Abiteboul, P. Buneman, and D. Suciu, *Data on the Web: From Relations to Semistructured Data and XML*. Morgan Kaufmann, 2001.

[4] L. Candillier, L. Denoyer, P. Gallinari, M. Rousset, A. Termier, and A. Vercoustre, "Mining XML documents," in *Data Mining Patterns: New Methods and Applications*, P. Poncelet, F. Masseglia, and M. Teisseire, Eds. Information Science Reference, 2007, ch. 8, pp. 198–219.

[5] J. Hernández-Orallo, M.-J. Ramírez-Quintana, and C. Ferri-Ramírez, *Introducción a la Minería de Datos*. Pearson - Prentice Hall, 2004.

[6] T. Tran, R. Nayak, and P. Bruza, "Document clustering using incremental and pairwise approaches," in *Focused Access to XML Documents 6th International Workshop of the Initiative for the Evaluation of XML Retrieval, INEX 2007*. Springer Verlag, 2008, pp. 222–233.

[7] ——, "Combining structure and content similarities for XML document clustering," in *7th Australasian Data Mining Conference*, 2008.

[8] J.-W. Lee and S.-S. Park, "Computing similarity between XML documents for XML mining," in *Engineering Knowledge in the Age of the Semantic Web, 14th International Conference, EKAW 2004*. Springer Verlag, 2004, pp. 492–493.

[9] R. Maurice, *Algotithmes de Classification*. Ed. Masson, Paris, France, 1985.

[10] "Oracle text application developer's guide release 1 (10.1)," Available: http://www.stanford.edu/dept/itss/docs/oracle/10g/text.101/b10729/classify.htm, 2010.

[11] M. Garre, J.-J. Cuadrado, M. A. Sicilia, D. Rodríguez, and R. Rejas, "Comparación de diferentes algoritmos de clustering en la estimación de coste en el desarrollo de software," *Revista Española de Innovación, Calidad e Ingeniería del Software*, vol. 3, no. 1, pp. 7–22, 2007.

[12] A. K. Jain, "Data clustering: 50 years beyond k-means," *Pattern Recognition Letters*, vol. 31, no. 8, pp. 651–666, 2010.

[13] H. Schmid, "Probabilistic part-of-speech tagging using decision trees," in *New Methods In Language Processing*, D. B. Jones and H. Somers, Eds. Routledge, 1997.

H.3. Artículo : Configurable Application Designed for Mining XML Document Collections.

El artículo *Configurable Application Designed for Mining XML Document Collections* fue publicado en el libro Intelligent Control and Innovative Computing de la editorial Springer en el año 2012. A continuación se presenta la referencia completa :

Mihaela Juganaru-Mathieu, Cristal Karina Galindo Durán y Héctor Javier Vázquez. **"Configurable Application Designed for Mining XML Document Collections"**. Intelligent Control and Innovative Computing. Volumen 110, ISBN : 978-1-4614-1695-1. Ed. Springer US. Año 2012. http://www.springerlink.com/content/g000gh4672356wm2/

Chapter 18
Configurable Application Designed for Mining XML Document Collections

Mihaela Juganaru Mathieu, Cristal Karina Galindo-Durán, and Héctor Javier Vázquez

Abstract In this chapter we present a flexible and configurable application for mining large XML document collections. This work is centered on the process of extracting document features related to structure and content. From this process, an attribute frequency matrix is generated and, depending on the cluster algorithm, it is transformed and/or used to obtain similarity measures.

Keywords Clustering • Conceptual framework • Methodology • Modularity • UML • XML mining

1 Introduction

The eXtensible Markup Language XML [1, 2] is a self-describing language for data representation, exchange and storage of information in structured and/or semi-structured form [3], without the need to specify how information will be viewed. This signifies that content and presentation are independent. XML also provides the means to share information in a reliable and easy manner and offers compatibility between different systems and platforms and between different applications. These

M.J. Mathieu
Laboratoire en Sciences et Technologies de l'Information, Institut H. Fayol, Ecole Nationale Supérieure des Mines de Saint Etienne, 158, cours Fauriel, 42023, SaintEtienne Cedex 2, France
e-mail: mathieu@emse.fr

C.K. Galindo-Durán (✉)
Computer Science, MSE Program, Universidad Autónoma Metropolitana, Unidad Azcapotzalco
e-mail: cdgalindod@gmail.com

H.J. Vázquez (✉)
Departamento de Sistemas, Universidad Autónoma Metropolitana, Unidad Azcapotzalco, Avenida San Pablo 180, Col. Reynosa Tamaulipas, Mexico D.F. C.P. 02200, Mexico
e-mail: hjv@correo.azc.uam.mx; statfor@yahoo.com

S.I. Ao et al. (eds.), *Intelligent Control and Innovative Computing*, Lecture Notes in Electrical Engineering 110, DOI 10.1007/978-1-4614-1695-1_18,
© Springer Science+Business Media, LLC 2012

reasons make XML a preferred language to encode text and non-text data, to build Web servers and Web applications and for creating collections of documents. Given that most non text data are stored in databases, XML is generally used to store collections of text and data containing links to other documents (within the same collection or in another collection) and references to images and other multimedia files.

The aim of the first part of this chapter is to present the methodological and design principles for building an application to mine XML documents, highlighting specific issues such as parsing and tokenization. This methodology is based on a general data mining process model and a specification design [4] that resulted in a modular and configurable XML mining application. It is modular in the sense that it is easy to change modules, functions or algorithms and is configurable in that it allows the user to:

- Mine various large collections.
- Choose all documents, or a random sample of them from a given collection.
- Choose, as items to mine, different attributes about structure, content and/or hyperlinks (within and/or outside documents).
- Count frequencies or count only the presence/absence of structural or content features.
- Produce different data transformations from the frequency matrix.
- Generate different metrics of similarity as well as similarity matrices, depending of the counting method used in the previous step.
- Choose from different mining algorithms.

The second part of this chapter discusses basic software design elements and gives details about the implementation of the application, in particular parsing and tokenization.

2 Methodology and Design for an XML Mining Application

For general data mining, two well-known process models [5] are classically presented: an eight-step model proposed by Anand and Buchner in 1998 [6] and a nine-step model proposed by Fayyad et al. in 1998. The latter, presented by Minei [7], might be summarized as follows:

1. To understand the problem in order to define the tasks to solve and goals to pursue, i.e., association, classification, cluster analysis, sequential pattern discovery, temporal modeling, regression or deviation detection.
2. To identify the data target and the variables to achieve the goal.
3. To pre-process and to clean data by removing noise, identifying outliers, identifying missing values and selecting methods for correcting and handling them.
4. To reduce data dimensionality or to transform it to find invariant representations of data.

5. To choose the data mining task depending on the goal, i.e. description or prediction.
6. To select methods and algorithms for searching and extraction of data patterns, including parameter setting.
7. To generate patterns of interest; the result might take the form of classification rules, decision trees, regression models or trends.
8. To evaluate and/or to interpret the mined patterns. This refers to reconfiguring procedures and then considering a possible return to one or more of the previous steps.
9. To evaluate the performance of the system and to resolve potential conflicts with previous or a priori beliefs or extracted knowledge.

Each step is discussed, in detail, in texts about data mining [8]. Most of the time, however, definitions of terms such as *data, information or knowledge* are not clarified. Although it is not the goal of this chapter to discuss issues about those terms, it is important to understand and differentiate them [9].

The application is specified and designed to mine collections of XML documents (the data target) oriented to clustering and not to classification [10]. Clustering is the process of partitioning data into a set of groups, clusters or subclasses [11]. This process is performed without any information about possible groups and without examples or learning sets. Once the process finished, all data in the cluster share common traits. Given that the user is not directly involved until the end of the process, clustering is considered to be unsupervised classification. Conversely, in supervised classification, which is used to predict group membership for data instances, the user intervenes and monitors the process.

In the case of information coded in XML, the general process model does not clarify how to take into account document's content (data variables) and its structure (attributes about structure) [12, 13].

An XML document is in fact a graph representing a collection of atomic and complex objects in the form of an ordered and labeled tree, where each node represents an XML element and each element might have one or more XML attributes. Attributes are also provided to annotate elements. An XML element is made up of a start tag, an end tag, and content in between. The start and end tags describe the content within the tags (the value of the element). The content may be a Character Data (CDATA) section or an ordered set of XML elements. Complexity increases because the same tag may exist at different levels of the tree. This gives a multidimensional structure to an XML document and creates the need to manage information in different ways [14].

In the pre-processing phase, in addition to the tasks listed in step 3 of the nine step model, an extra task should be included: that is, to consider tags, their nesting and data references to reveal their structure in a XML document. Most of the methods work on the trees representing documents [15, 16]. Some of them compare trees using metric distances based on: the edit distance, the minimum number of mutations required to change a tree into another one or the number of paths they share. Other methods focus on sub-trees inside the document.

To reveal structure, it is necessary to include a parsing step and eventually some particular strategies for XML retrieval and extraction of data patterns from XML documents, such as partition of XML documents into non-overlapping indexing units or mapping to a set of lexicalized sub-trees [17]. On the other hand redundancy caused by nested elements makes it necessary to establish restrictions, such as discarding small elements, removing or keeping some types defined by users or collapsing several nested elements in the same result.

Fortunately, if the objective is just to mine the data, as in the case of our application, some tasks such as identifying tokens (content words, hyperlinks, tags and references, as well as their nesting level), can be reduced.

A token in a XML document is the smallest data element that can be processed to revel a meaning. As part of a CDATA section, a word is a token. A CDATA section separates any kind of character (including XML markup) that should be ignored by the parser. In well formed XML documents, an open tag always has a closed tag, so we can just consider the open tags. If we need to indicate a token's position in the XML tree, it's sufficient to consider the path of the node. In the case of a token inside a CDATA section, the path of the CDATA is just considered.

The simplest form of tokenization (obtaining the tokens) is to consider all the tags and all the words as they are presented in the textual parts. However, another problem might emerge: As, number of tokens increases data dimensionality also increases. This can be solved, as in text mining [18], by taking the invariant form of the word in its lexical category. It's also possible to perform a supervised tokenization and to take into account some external information, such as a list of the type of tags, indications about the importance of a type of code or ontology, lists of synonyms or specialized vocabulary indicating the "important" words.

An option for reducing and managing data dimensionality is to remove irrelevant characteristics, such as elements containing references to images, sound files and other multimedia formats. In the case of both internal and external links, just their number, not their content, might be considered.

Moreover this, for mining purposes it might be sufficient to consider the frequency or just simply the presence of tokens. All or a subset of tokens can be selected and counted (frequency and presence and/or absence). These counts can be integrated in a single table or organized in different count or frequency matrices; for example, one frequency count matrix to register tokens about document-structure (tags, markers, token path, depth or any another feature related to structure) and another matrix to register counts of tokens about document-content (composed mainly of text). The final result of this is a matrix containing frequencies about tokens (frequency matrix). In this way, XML mining research has led to the development of various techniques and tools [19, 20] dealing with data frequency patterns as well as frequency patterns within parse trees.

The frequency matrix, in its original or transformed form, is the basis for different well-known statistical techniques oriented to data prediction or just to description. Transformations or normalizations may be applied, for example to standardize values or to stabilize variance. For the present application a hard clustering and soft clustering technique are proposed. In hard clustering, each document belongs only to one cluster while in soft clustering the same document may belong, at

different degree, to different clusters. The expectation maximum (EM) algorithm (soft clustering) can be applied directly on the frequency matrix [21], while for algorithms such as hierarchical clustering (hard clustering) [11], a metric should be chosen to obtain similarity matrices.

Depending on the nature of the counts (i.e., frequencies or binary data), there are different kinds of similarity measures. For example, in the case of frequency counts, it is common to use the Euclidean, Mahalanobis, Minkowski and $\chi 2$ distances [22]. In the case of the presence or absence of particular features, it is possible to assign more or less importance to presences than to absences: the Jaccard and Dice–Sorensen indexes are most often used for this purpose. However, if both presences and absences are considered to have the same importance, then the Sokal and Michener, Sokal and Sneath, Hamming or Roux indexes may be used [22].

We could also consider another type of frequency measure, a frequency function inspired by information retrieval and text mining: Tf-idf (Term frequency–inverse document frequency) is a measure that evaluates the importance of a word in a document or a set of documents [18]. This measure takes into account the direct frequency of a word in the document as well as the inverse frequency giving less weight to words that are present in many documents. For this type of frequency, we use similarity measures such as product cosine or Jaccard. Figure 18.1 provides a view of the sequence of steps for pre- processing document structure and content. This forms part of the conceptual model proposed to design this application [4].

Once the clusters have been obtained, they can be parsed to find data structures and explore their characteristics, which can then be interpreted and evaluated, and the mining process eventually repeated.

3 Software Description

User-oriented, flexibility and modularity are the basic principles that guide the design of this application. The application should be user-oriented and flexible in the sense that it gives the user the ability to choose among different types of tokens, different frequencies, different data transformations, different types of similarity distances and even different types of clustering algorithms. All these options can be applied on a large collection or on a random sample of the collection. At the end of the process and during user interpretation of results, the application would "feed" the user's knowledge into the XML mining process. Modularity means that the application can be partitioned to allow easy implementation and maintenance. This would allow a user to modify a part of the application or to integrate a new procedure without redesigning the entire application. In a first level, the design process was separated into two packages, as follows:

- Package one, to obtain documents, to tokenize them and store them, and to obtain the frequency matrix.
- Package two, to deal with the mining process itself.

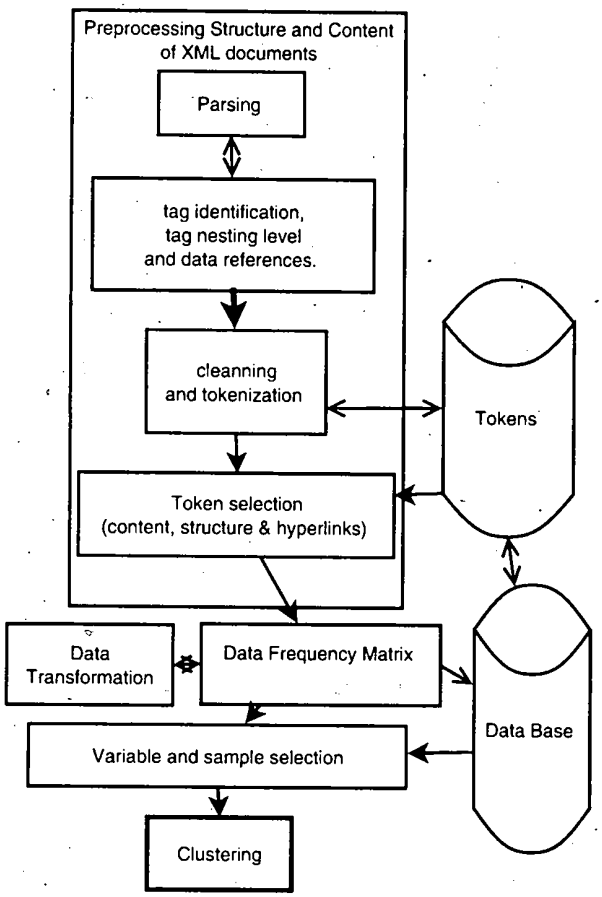


Fig. 18.1 Conceptual model

The unified modeling language (UML[1]) is used to present different elements of the proposed design such as use case and class diagrams. In the following section, class diagrams and implementation of an entity relational model to store information are presented. Details of the use case diagrams are presented in [4].

3.1 Class Diagram

In the class diagram, classes are first defined for the XML collection and its documents. These two classes are directly associated with a multiplicity [1, *],

[1]http://www.uml.org [Accessed November 8, 2011].

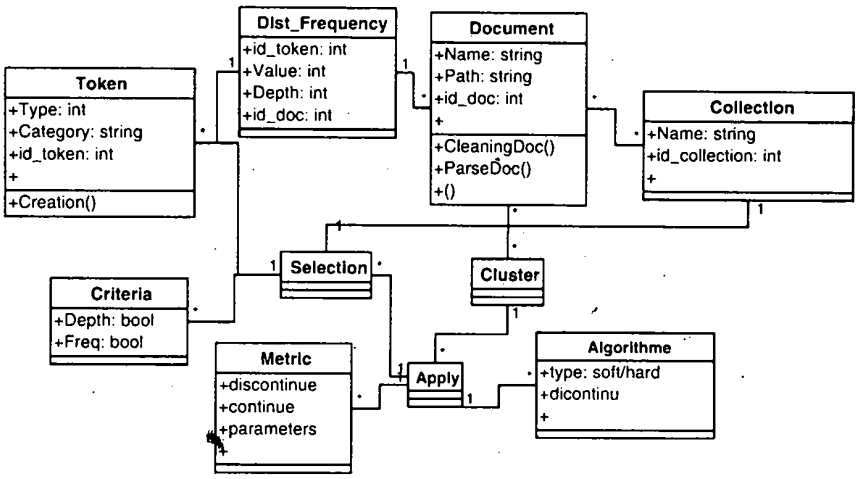


Fig. 18.2 Class diagram

given that one collection has several documents. An important class is "Distribution_frequency", which is associated with several documents and several tokens. Tokens are the objects identified in the document and may have several categories (for example, structural or grammatical). The remaining classes are related to the mining process, which applies cluster algorithms, based on a given metric to the frequency matrix (revealed by classes: Token and Distribution_frequency). The classes proposed are: **Collection_C, Document_C, Distribution_Frequency_C, Token_C, Criteria_C, Choice_C, Metrics_C, Apply_C, Cluster_C and Algorithm_C.**

Figure 18.2 shows a static view of the class diagram, with the ten classes and their associations proposed for this application.

3.2 Implementation

The INEX collection[2] is used as a data target to implement this application. It currently consists of two collections: a large one that has about 2.7 million documents and a smaller one that contains approximately 463,000 XML documents. However, for testing purposes only 60,000 documents are considered. This collection also includes bags of words that are commonly found in documents and frequencies of various structures in the form of XML tags, trees, links and names of entities. One feature is that documents are well formed and there is no need for a document type

[2]This collection contains an image of Wikipedia articles, and was extracted from a 2007 copy.

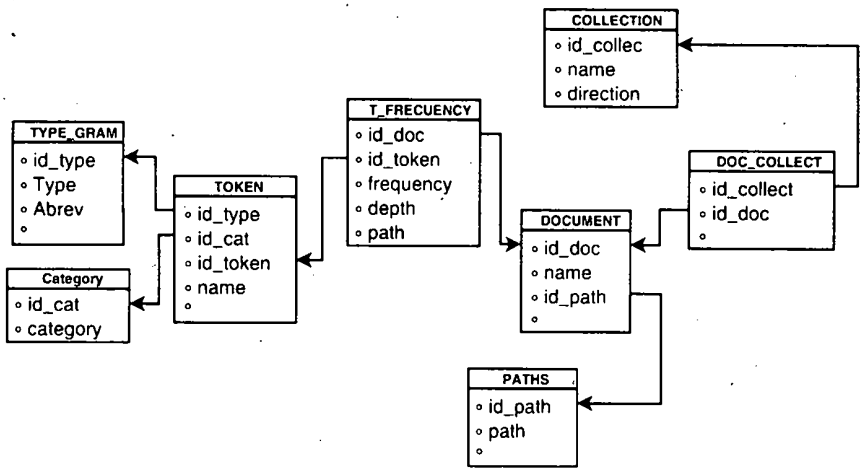


Fig. 18.3 Relational model

definition (DTD). The collection is grouped in 22 packages, with a mean number of five branches in each document. With these characteristics, work is greatly reduced because each document has a unique node on the first level, and an open tag is always closed in a coherent order. This also means that it is known where to begin parsing and that it is not necessary to check whether the tag is closed.

For package one, our major decision was to store in a DBMS (database management system) all the information about documents, tokens and frequencies in a database. The purpose is to avoid calculating all feature frequencies (structure, content and hyperlinks) each time the user analyses different sets of attributes. Frequency matrices and presence/absence matrices are directly extracted with aggregated SQL queries from the tables of the database. PostgreSQL[3] is used for the database implementation of the DBMS. The eight tables of the relational model are presented in Fig. 18.3, as follows:

1. COLLECTION table with a tuple for each XML document collection.
2. DOCUMENT table references PATH table and contains a tuple for each XML document.
3. DOC_COLLECTION table indicates the composition of collections.
4. PATHS table to indicate where the XML files are physically stored.
5. TOKEN table stores a unique tuple for each possible token.
6. CATEGORY table contains a tuple for each category of a token: word, tag, link or other.
7. TYPE_GRAM table indicates the grammatical type for a token word.
8. T_FREQUENCY table indicates frequency of tokens in documents.

[3]http://www.postgresql.org/ [Accessed November 8, 2011].

A first step is to obtain the XML collection, assign a number and identify the path where each document is stored. This information is stored in the DOCUMENT and PATH tables. There are several mechanisms for tracking and accessing information stored in XML documents. Basic mechanisms are implemented in parsers such as SAX (simple API for XML[4]) and DOM (document object model[5]). SAX tracks the document sequentially in depth while DOM randomly accesses the document and builds, in memory, a tree of the XML document, permitting the document to be parsed, in depth or in breadth, or its elements to be modified. Because DOM is less complex, it is used to obtain tags, hyperlinks and paths tags. The pseudocode is the following:

```
/* pseudocode for Parsing and Tokenization */
for each document D of a collection
    if D was tokenized then break;
    insert into DOCUMENT table;
    count[...] < −0$;
    let id_doc be the key into DOCUMENT table;
    depth < −0;
    obtain with DOM the XML tree
    /* treat all the nodes starting with the root */
    for each node n of XML tree
      if n is <outsidelink> or <collectionlink> node n is a link
      else
        n is a node to treat
        let <x> be the tag of the node;
        if <x> not exists in TOKEN table then
          insert <x> into TOKEN;
          indicate if is a hard or a soft tag;
        end if
        let textual be the CDATA of n;
call TreeTagger for textual: the output is: list of (word,
                    lexical_category, invariable_form)
  for each (word, lexical_category, invariable_form)
      if (word, lexical_category, invariable_form) ∉ TOKEN table
then insert into TOKEN table
          end if
          let be id_token the key value in TOKEN table
          count[id_token, depth] ++
        end for
        depth ++;
        recursive call to treat the sons of n, if any;
      end if
```

[4]http://www.saxproject.org/ [Accessed November 8, 2011].

[5]http://www.w3.org/DOM/ [Accessed November 8, 2011].

```
    end for
    for each count[id_t, depth] <> 0 do
        insert (id_doc, id_t, depth, count[id_t, depth]) into
      FREQUENCY
              table
    end for
end for
```

Another decision was to use a grammatical invariable generator such as Tree Tagger[6], a part of speech tagger that obtains the grammatical category and invariant form of every word [23]. After parsing, Tree Tagger is used to obtain content tokens.

From this, a token frequency matrix is created, which forms the basis of the content and structure frequency matrices. Token frequency matrix generation requires extra storage, but, as explained previously, it was decided to implement it, given that it is simple to obtain whole tokens counts, and because it later provides the user a choice of generating the frequency or the presence/absence matrix.

In this implementation with INEX, 23 soft tags (such as emph, normal list and item) and 7 hard tags (such as article, name and body) were considered, as well as, 2 types of links (collection link and outside link) and 36 tokens for content (such as cardinal number, determiner, existential and foreign word). This provides at least 66 variables for each document.

This architecture of the DBMS offers enough flexibility to center on a subset of the variables. For example, if the user wants to cluster documents based only on the words without tags and links, a filter is introduced in the CATEGORY table. Also, if the user wants to consider only the words having a meaning (excepting for example the stop words[7]), it is easy to filter in the TYPE_GRAM table to exclude determinants, conjunctions and prepositions. It is also possible in the TOKEN table to flag the stop word linking to external information.

Another characteristic of this application is its ability to work on the whole collection or just a random sample of it. As an example for a small collection with 545 XML documents, the most important tables have 8,336 rows for the TOKEN table and 156,793 for the T_FREQUENCY table. The smallest document (6 kb) has about 16 tokens, and the larger document (30 kb) has around 1,846.

Java[8], an object oriented language, was used to implement this application. Using Java allows us to introduce and consider new distance measures or mining algorithms. Java also offers the possibility of working with various APIs for DBMS or XML, to interface with Tree Tagger or to call mining procedures or libraries written in GNU applications such as R[9] or other languages.

---

[6]http://www.ims.uni-stuttgart.de/projekte/corplex/TreeTagger/DecisionTreeTagger.html [Accessed November 8, 2011].

[7]A stop word is a word frequently appearing in a text, such as "the", "and", "a".

[8]http://www.oracle.com/technetwork/java/javase/downloads/index.html [Accessed November 8, 2011].

[9]http://www.r-project.org/ [Accessed November 8, 2011].

4 Conclusion and Future Work

In this chapter, we presented the methodology and implementation for a modular and configurable application designed for XML mining and clustering in a large collection of documents. The specified design was based on UML, and the particularities of XML documents were fundamental to understand the mining process.

Concerning the generation of a token frequency matrix, the choice of a DBMS saves space and allows, if required, the creation of one or two separate frequency matrices one for structure and other for content, given that each token category can be easily identified.

This work highlights the importance of developing a configurable and modular application based on a specified design; other research has excluded the possibility of modifying parts of the pre-processing and mining application or changing its configuration.

The first stage (package one) of development of the software application has been completed and uses Java and PostgreSQL for implementing the database management system. The information saved in the database allows the user to mine: structure or content, hyperlinks (within and/or outside documents) and also to choose the type of the frequency (direct, presence/absence, standardized, transformed or Tf-idf) and, depending of the clustering method, to apply a similarity function.

This implementation is being tested with a representative sample of 537 from 60,000 documents to obtain the token frequencies and then the frequency matrix.

The frequency matrix is the basis for the second package, which focuses on the numerical part of the data mining process. Hierarchical clustering (hard clustering) [11] and expectation maximization (EM) algorithm (soft clustering) [21] are proposed for cluster analysis. They can be implemented in Java or linked with subroutines programmed in other applications such as R or in other mining tools [5].

Acknowledgments The authors acknowledge financial support given to the Promep group "Analysis and Information Management" and to the Universidad Autónoma Metropolitana, Unidad Azcapotzalco (UAM-A) for the financial support given to the project, "Study and Systems Modelling," Number: 2270217. They acknowledge Dr. Nicolas Dominguez Vergara (former head of the Systems Department) and Dr. Silvia Gonzalez Brambila (former Coordinator of the Master's Computer Science Program at the UAM Azcapotzalco) for their support and encouragement. They also acknowledge Peggy Currid for her advice in copyediting.

References

1. Bray T, Paoli J, Sperberg-McQueen CM, Maler E, Yergeau F (2008) eXtensibleMarkup Language (XML) 1.0 (5th Edition), X3C recommendation. http://www.w3.org/TR/REC-xml/ [Accessed November 8, 2011]
2. Harold ER, Scott Means W (2004) XML in a nutshell. O'Reilly Media, Sebastopol, CA, USA

3. Abiteboul S, Buneman P, Suciu D (2001) Data on the web: from relations to semi structured data and XML. Morgan Kaufmann, Series in data management systems, San Francisco, USA
4. Duran CKG, Juganaru-Mathieu M, Vazquez HJ (2011) Specification design for an xml mining configurable application, lecture notes in engineering and computer science. In: Proceedings of the international multiconference of engineers and computer scientists 2011, IMECS 2011, Hong Kong, 16–18 March 2011. (best paper award). http://www.iaeng.org/publication/IMECS2011/IMECS2011_pp378-381.pdf [Accessed November 8, 2011]
5. Mikut R, Reischl M (2011) Data mining tools, WIREs data mining knowledge discovery. 1(5):431–443. http://onlinelibrary.wiley.com/doi/10.1002/widm.24/pdf [Accessed November 8, 2011]
6. Büchner AG, Mulvenna MD, Anand SS, Hughes JG (1999) An internet-enabled knowledge discovery process (best paper award). In: Proceedings of the 9th international database conference, Hong Kong, July 1999. http://mcbuchner.com/HTML/Research/PDF/IDC99.pdf [Accessed November 8, 2011]
7. Minaei-Bidgoli B (2004) Data mining for a web-based educational system, PhD thesis, Michigan State University. http://www.lon-capa.org/papers/BehrouzThesisRevised.pdf [Accessed November 8, 2011]
8. Cios KJ, Pedrycz W, Swiniarski RW, Kurgan LA (2007) Data mining: a knowledge discovery approach, Springer Science+Business Media, New York, USA
9. Ackoff RL (1989) From data to wisdom. J Appl Syst Anal 16:3–9
10. Oracle text application developer's guide release 1 (10.1) (2010). http://www.stanford.edu/dept/itss/docs/oracle/10g/text.101/b10729/classify.htm [Accessed November 8, 2011]
11. Grabmeier J, Rudolph A (2002) Techniques of cluster algorithms in data mining, in data mining and knowledge discovery. Kluwer Academic Publishers, 6(4):303–360, Netherlands. http://www.springerlink.com/content/d6ekxxcu0d2ngamj/fulltext.pdf [Accessed November 8, 2011]
12. Candillier L, Denoyer L, Gallinari P, Rousset MC, Termier A, Vercoustre AM (2007) Mining XML documents. In: Poncelet P, Masseglia F, Teisseire M (ed) Data mining patterns: new methods and applications. pp 198–219
13. Büchner AG, Baumgarten M, Mulvenna MD, Böhm R, Anand SS (2000) Data mining and XML: current and future issues. In: Proceedings of the 1st international conference on web information systems engineering, Hong Kong. http://mcbuchner.com/HTML/Research/PDF/WISE00.pdf [Accessed November 8, 2011]
14. Tran T, Nayak R, Bruza P (2008) Document clustering using incremental and pairwise approaches. In: Focused access to XML documents 6th international workshop of the initiative for the evaluation of XML retrieval, INEX-2007. pp. 222–233. Springer-Verlag, Berlin Heidelberg
15. Candillier L, Tellier I, Torre F (2005) Transforming XML trees for efficient classification and clustering. Workshop on mining XML documents, INEX-2005, Schloss Dagstuhl, 28–30 November 2005. http://www.grappa.univ-lille3.fr/_candillier/publis/INEX05.pdf [Accessed November 8, 2011]
16. Dalamagas T, Cheng T, Winkel K-J, Sellis Timos (2003) A methodology for clustering XML documents by structure. Inform Syst 31:187–228
17. Manning CD, Raghavan P, Schtze H (2009) An introduction to information retrieval. Cambridge University Press, Cambridge, pp 201–210. http://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf [Accessed November 8, 2011]
18. Feldman R, Sanger J (2007) The text mining handbook: advanced approaches in analyzing unstructured data. Cambridge University Press, Cambridge
19. Tran T, Nayak R, Bruza P (2008) Combining structure and content similarities for XML document clustering. In: Proceedings 7th Australasian data mining conference, Glenelg, South Australia. CRPIT, 87. Roddick JF, Li J, Christen P, Kennedy PJ, (ed). http://crpit.com/confpapers/CRPITV87TranT.pdf [Accessed November 8, 2011]
20. Lee JW and Park SS (2004) Computing similarity between XML documents for XML mining. In Engineering Knowledge in the Age of the Semantic Web, 14th International Conference, EKAW, pp 492–493. Springer Verlag

21. Dempster AP, Laird NM, Rubin DB (1977) Maximum likelihood from incomplete data via the EM algorithm. J R Stat Soc B 39(1):1–38. Royal Statistical Society, London, http://web.mit.edu/6.435/www/Dempster77.pdf [Accessed November 8, 2011]
22. Roux M (1985) Algorithmes de Classification. Masson, Paris
23. Schmid H (1997) Probabilistic part-of-speech tagging using decision trees. In: Jones DB, Somers H (ed) New methods in language processing. Routledge, London

H.4. Artículo : Evaluación de Resultados de Agrupamiento de una Aplicación de Minería para Documentos en Formato XML.

El artículo *Evaluación de Resultados de Agrupamiento de una Aplicación de Minería para Documentos en Formato XML.* fue presentado en el XVIII Simposio Internacional de Métodos Matemáticos Aplicados a las Ciencias (XVIII SIMMAC) es el evento de matemática aplicada más importante de América Central. Es organizado por el Centro de Investigación en Matemática Pura y Aplicada (CIMPA) de la Universidad de Costa Rica cada dos años, con la colaboración de la Escuela de Matemática. A continuación se presenta la referencia completa :

Héctor Javier Vázquez y Cristal Karina Galindo Durán y Mihaela Juganaru-Mathieu. **"Evaluación de Resultados de Agrupamiento de una Aplicación de Minería para Documentos en Formato XML"**. XVIII Simposio Internacional de Métodos Matemáticos Aplicados a las Ciencias (XVIII SIMMAC) llevado a cabo en Costa Rica, 21 al 24 de febrero 2012.

Cabe mencionar que se anexa la carta de aceptación; así como la presentación realizada dicho Simpio.

XVIII SIMMAC
Simposio Internacional de Métodos Matemáticos Aplicados a las Ciencias
International Symposium on Mathematical Methods Applied to the Sciences
San José, Costa Rica 21-24 Febrero/*February*, 2012
E-Mail: simmac.cimpa@ucr.ac.cr Fax: +(506) 2511 4918 http://www.cimpa.ucr.ac.cr/simmac/
CIMPA – Escuela de Matemática, Universidad de Costa Rica, 2060 San José, Costa Rica

San José, December, 22nd, 2011

Hector Javier Vázquez
Departamento de Sistemas, Universidad Autónoma Metropolitana, Unidad Azcapotzalco
México D.F., México
hjv@correo.azc.uam.mx, statfor@yahoo.com, cdgalindod@gmail.com,mathieu@emse.fr

On behalf of the Scientific Committee of the *XVIII International Symposium on Mathematical Methods Applied to the Sciences* (SIMMAC), that will take place in San José, Costa Rica, February, 21-24, 2012, I am pleased to inform you that your communication:

"Evaluación de resultados de agrupamiento de documentos en formato XML"

has been accepted for presentation in the Symposium.

Please, consult the instructions for the authors at the website http://www.cimpa.ucr.ac.cr/simmac/, for the final version of your communication. This abstract should be sent by January 6th, 2012 to the Chairman of the Scientific Committee, simmac.cimpa@ucr.ac.cr. The abstract must fulfil the following conditions:

- In Latex 2e, following our format
- Complete title
- Complete names of the authors
- Complete E-Mails of all authors
- Country and university of the authors
- Keywords in English (keywords in Spanish are optional).

The one who presents the communication at the SIMMAC should be the first author.

In case you send us the abstract in another format (PDF, Word, text,...), we would translate it into LaTeX if you pay a fee of US$10 (payable with your registration).

Those authors willing to publish the complete article should bring it to the SIMMAC in digital and paper (3 hard copies), with a maximum of 10 pages. Please, follow the rules indicated in our website.

We would appreciate it if you can confirm your attendance in the Symposium no later than January 10th, 2012. Please follow the registration instructions at our website.

Sincerely yours,

Mario Villalobos
Scientific Chairman
XVIII SIMMAC

Evaluación de Resultados de Agrupamiento de una Aplicación de Minería para Documentos en Formato XML

H.J. Vázquez (1), C.K. Galindo-Durán (2) y M.J. Mathieu (3)
Departamento de Sistemas, Universidad Autónoma Metropolitana, Unidad Azcapotzalco,
Avenida San Pablo 180, Col. Reynosa Tamaulipas, México D.F. C.P. 02200, México
e-mail: hjv@correo.azc.uam.mx ,statfor@yahoo.com[1]
Computer Science, MSE program, Universidad Autónoma Metropolitana, Unidad Azcapotzalco.
e-mail: cdgalindod@gmail.com[2]
Laboratoire en Sciences et Technologies de l'Information, Institut H. Fayol, Ecole Nationale Superieure des Mines de Saint Etienne, 158, cours Fauriel, 42023, Saint Etienne Cedex 2, France.
e-mail: mathieu@emse.fr[3]

13 de enero de 2012

Resumen

Keywords: clustering, cluster analysis, XML documents, data mining.
Palabras clave: agrupamiento, documentos XML, minería de datos.

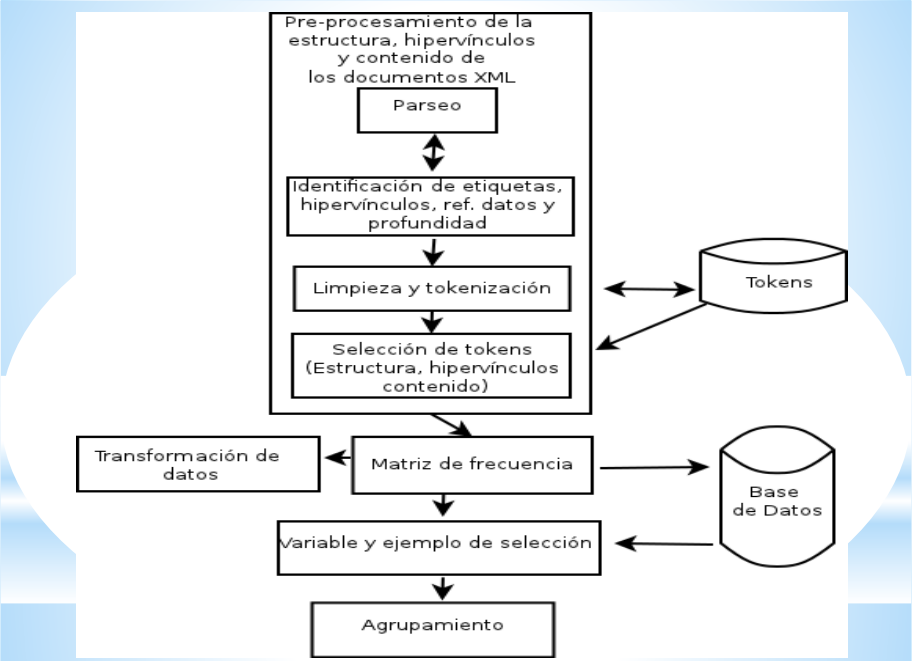
Actualmente se desarrolla y evalúa una aplicación para la minería de grandes volúmenes de documentos en formato XML [1]. Esta aplicación permite agrupar documentos de colecciones XML de acuerdo a características definidas en el contenido y en su estructura, esta última definida por etiquetas. En este trabajo se evalúan resultados, mediante los indices de Davies-Boulding y Dunn, considerando diferentes algoritmos, métricas y distancias de corte sobre una muestra representativa obtenida de una población de 659,388 documentos XML de la colección INEX.

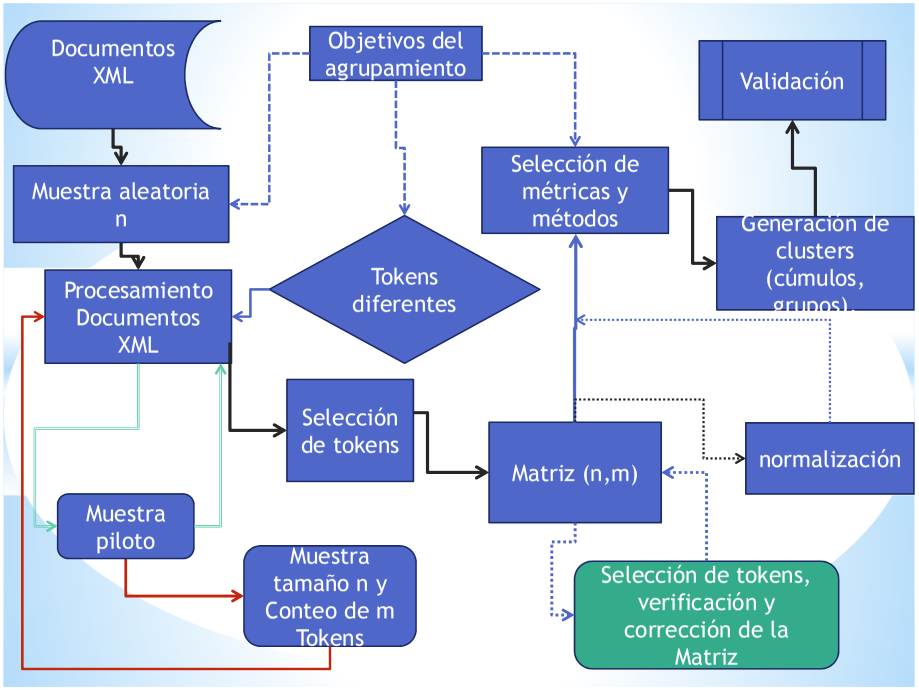
Referencias

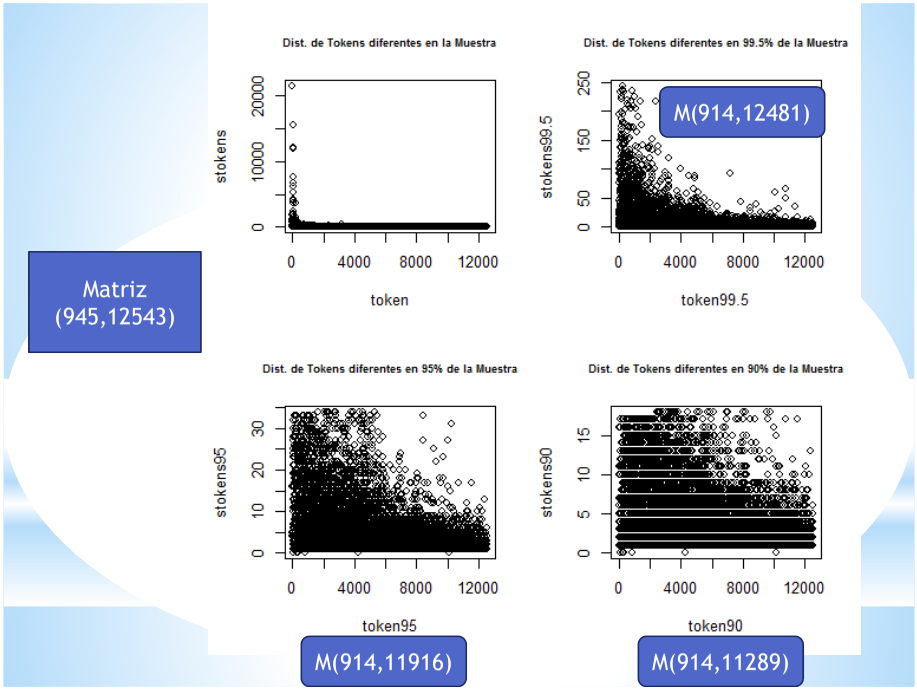
[1] Durán CKG, Juganaru-Mathieu M, Vazquez HJ (2011), *Specification design for an xml mining configurable application*, lecture notes in engineering and computer science. In: Proceedings of the international multiconference of engineers and computer scientists 2011, IMECS 2011, Hong Kong, 16–18 March 2011.

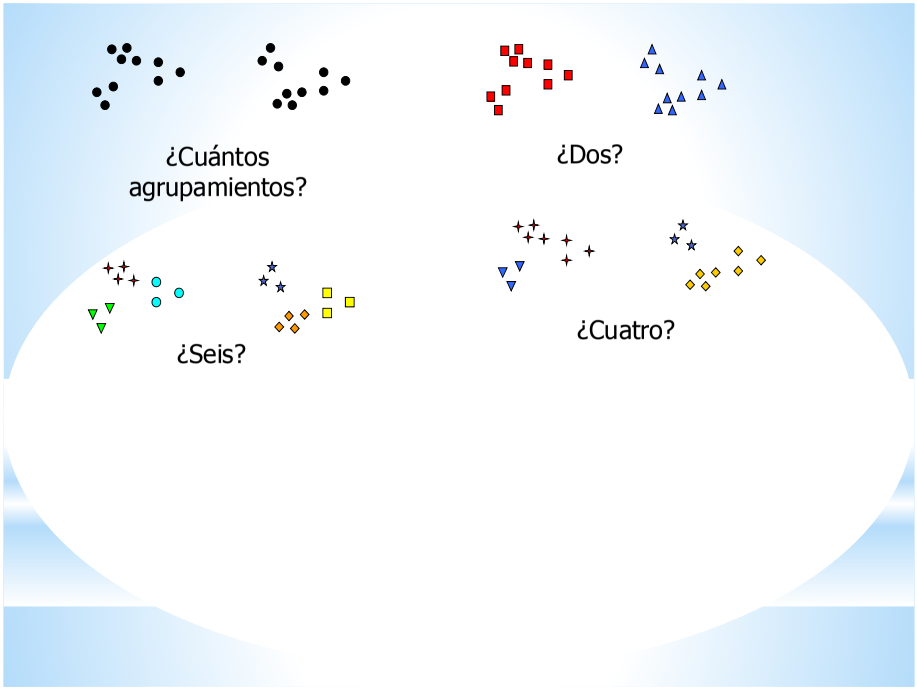
Evaluación de resultados de agrupamiento de documentos en formato XML

H. J. Vázquez, C.K. Galindo-Durán
M.J. Mathieu
México-Francia
Departamento de Sistemas
Universidad Autónoma Metropolitana



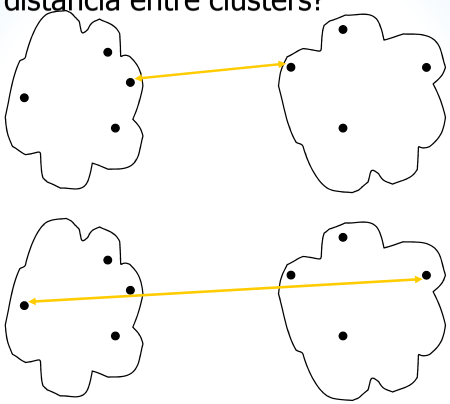

Documentos XML
Objetivos del agrupamiento
Validación
Muestra aleatoria n
Selección de métricas y métodos
Generación de clusters (cúmulos, grupos)
Tokens diferentes
Procesamiento Documentos XML
Selección de tokens
Matriz (n,m)
normalización
Muestra piloto
Muestra tamaño n y Conteo de m Tokens
Selección de tokens, verificación y corrección de la Matriz


Matriz (945,12543)
Dist. de Tokens diferentes en la Muestra
stokens
token
Dist. de Tokens diferentes en 99.5% de la Muestra
stokens99.5
token99.5
M(914,12481)
Dist. de Tokens diferentes en 95% de la Muestra
stokens95
token95
M(914,11916)
Dist. de Tokens diferentes en 90% de la Muestra
stokens90
token90
M(914,11289)



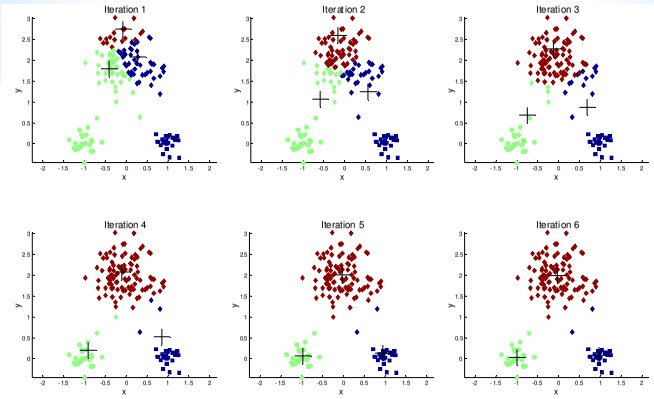
Método Jerárquico:
¿Cómo medir la distancia entre clusters?

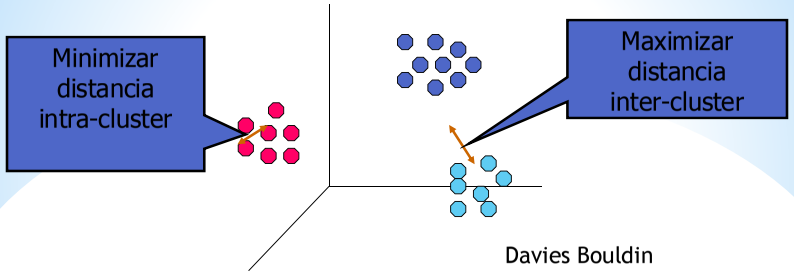
- MIN simple
- MAX completa



$$\delta^2(x,y) = \frac{m_x m_y}{m_x + m_y} \|x - y\|^2$$
$$= \frac{m_x m_y}{m_x + m_y} d^2(x-y)^2$$
$$= \frac{m_x m_y}{m_x + m_y} \sum_i (x_i - y_i)^2$$

k-Means





Davies Bouldin

$$DB = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i, \text{ where}$$

$$R_i = \max_{j=1\ldots nc, i\neq j}\left(R_{ij}\right), \; i = 1\ldots n_c$$

$$D = \min_{i=1\ldots n_c}\left\{\min_{j=i+1\ldots n_c}\left(\frac{d(c_i,c_j)}{\max_{k=1\ldots n_c}(diam(c_k))}\right)\right\}, \text{where}$$

$$d(c_i,c_j) = \min_{x\in c_i, y\in c_j}\{d(x,y)\} \text{ and } diam(c_i) = \max_{x,y\in c_i}\{d(x,y)\}$$

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}$$

$$d_{ij} = d(v_i, v_j), \; s_i = \frac{1}{\|c_i\|}\sum_{x\in c_i} d(x, v_i)$$

Validación y Objetivos

Métricas (15) =
manhattan", "euclidean", "canberra", "bray", "kulcz
ynski", "jaccard", "gower", "altGower", "morisita", "
horn", "mountford", "raup" , "binomial", "chao

Métodos Jerárquicos (7)=
"ward", "single", "complete", "average", "mcquitty
", "median","centroid"

normalización (11)= distintas normalizaciones

clusters (2,10, 20, 30, 50, 75))=

Compacto

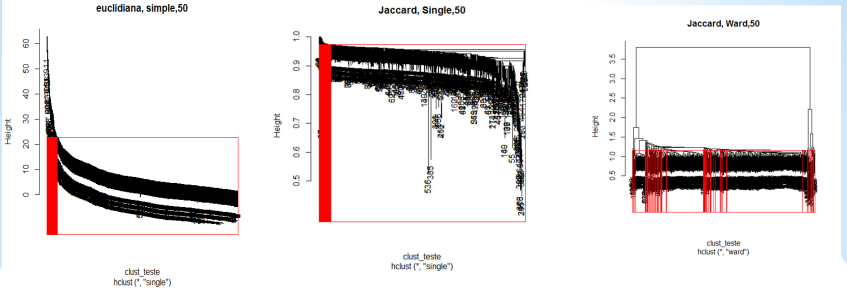
Agrupamiento

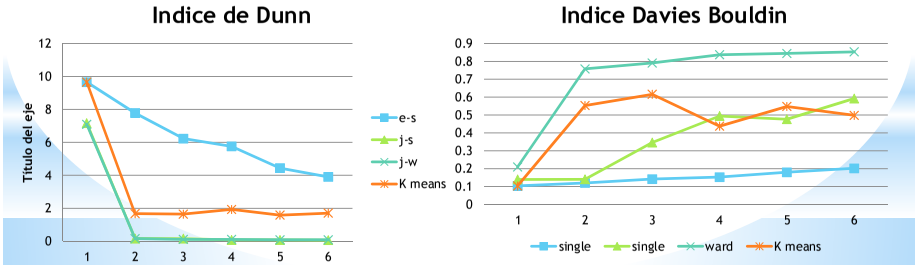
Separación

Conexión

Compacto

6930 combinaciones
48 días





clusters	e-s	j-s	j-w	K means
2	9.657936	7.1673892	7.0935031	9.657936
10	7.782547	0.1592864	0.1621816	1.6753263
20	6.230406	0.157715	0.11215897	1.6458254
30	5.755786	0.08041927	0.11215897	1.928777
50	4.441091	0.08041927	0.0894719	1.5848341
75	3.904634	0.06326137	0.0894719	1.7040053
	euclidiana	jaccard	jaccard	
clusters	single	single	ward	K means
2	0.103542	0.1395208	0.2097001	0.1035418
10	0.119803	0.1405597	0.7578515	0.5527422
20	0.141521	0.3460481	0.7908256	0.6158911
30	0.1525691	0.4945334	0.8373291	0.4374352
50	0.1800001	0.476655	0.8442814	0.5480018
75	0.2019422	0.592513	0.8532114	0.4980223

n0 - without normalization
n1 - standardization ((x-mean)/sd)
n2 - Weber standardization ((x-Me)/MAD)
n3 - unitization ((x-mean)/range)
n4 - unitization with zero minimum ((x-min)/range)
n5 - normalization in range <-1,1> ((x-mean)/max(abs(x-mean)))
n6 - quotient transformation (x/sd)
n7 - quotient transformation (x/range)
n8 - quotient transformation (x/max)
n9 - quotient transformation (x/mean)
n10 - quotient transformation (x/sum)
n11 - quotient transformation (x/sqrt(SSQ))

Índice

Referencias

[Abiteboul01] Abiteboul, S., Buneman, P., y Suciu, D. Data on the Web: From Relations to Semistructured Data and XML. Morgan Kaufmann, 2001.

[Ackoff03] Ackoff, R. L. Redisigning Society. Stanford University Press, 2003.

[Antonellis08] Antonellis, P., Makris, C., y Tsirakis, N. XEdge: Clustering Homogeneous and Heterogeneous XML Documents Using Edge Summaries. Proceedings of the 2008 ACM Symposium on Applied Computing ACM New York, 2008.

[Berge12] Berge, L., Bouveyron, C., y Girard, S. HD-classif: An R Package for Model-Based. Clustering and Discriminant Analysis of High-Dimensional Data, Journal of Statistical Software, 46(6), 2012.
URL `http://www.jstatsoft.org/`

[Bilmes98] Bilmes, J. A. A Gentle Tutorial of the EM Algorithm and its Application to Parameter Estimation for Gaussian Mixture and Hidden Markov Models, 1998.
URL `http://ssli.ee.washington.edu/people/bulyko/papers/em.pdf`

[Candillier07] Candillier, L., Denoyer, L., Gallinari, P., Rousset, M., Termier, A., y Vercoustre, A. Mining XML Documents. En P. Poncelet, F. Mas-

saglia, y M. Teisseire, eds., Data Mining Patterns: New Methods and Applications, págs. 198–207. Information Science Reference, 2007.

[Dalamagas04a] Dalamagas, T., Cheng, T., Winkel, K., y Sellis, T. Clustering XML documents by structure. Hellenic Conference on Artificial Intelligence, 2004.

[Dalamagas04b] Dalamagas, T., Cheng, T., Winkel, K., y Sellis, T. Clustering XML Documents by using Structural Summaries. EDBT Workshops, págs. 547–556, 2004.

[Dalamagas06] Dalamagas, T., Cheng, T., Winkel, K., y Sellis, T. A Methodology for Clustering XML Documents by Structure. Elsevier, (31):187–228, 2006.

[Dempster77] Dempster, A. y Laird N.M.and Rubin, D. Maximum likelihood from incomplete data via the em algorithm. Journal of the Royal Statistical Society, 39(1):1–38, 1977.

[Devore01] Devore, J. L. Probabilidad y estadística para ingeniería y ciencias. Thomson Learning, ed. Quinta, 2001.

[DOM10] Document Object Model (DOM), 2010.
URL `http://www.w3.org/DOM/`

[Finch05] Finch, H. Comparison of Distance Measures in Cluster Analysis with Dichotomous Data. Journal of Data Science , (3):85–100, 2005.

[Fraley06] Fraley, C. y Raftery, A. E. MCLUST : Model-based cluster analysis. R port by Ron Wehrens. R Foundation for Statistical Computing, 2006.

[Galindo-Durán11] Galindo-Durán, C. K., Mathieu, M. J., y Vázquez, H. J. Specification design for an xml mining configurable application. Lecture Notes in Engineering and Computer Science:Proceedings of the International Multiconference of Engineers and Computer Scientists 2011, págs. 378–381. 2011. ISBN 978-988-18210-3-4.
URL `http://www.iaeng.org/publication/IMECS2011/`

[Gan09] Gan, G. Data Clustering in C++ : An Object Oriented Aproach. CRC Pres, 2009.

[Garre07] Garre, M., Cuadrado, J. J., Sicilia, M., Rodríguez, D., y Rejas, R. Comparación de diferentes algorimos de clustering en la estimación de coste en el desarrollo de software. Revista Española de Innovación, Calidad e Ingeniería del Software, 3(1), 2007.

[Gutiérrez94] Gutiérrez, R., González, A., Torres, F., y Gallardo, F. Técnicas de Análisis de datos Multivariable. Tratamiento Computacional. Universidad de Granada, 1994.

[Halkidi01] Halkidi, M., Batistakis, Y., y Vazirgiannis, M. On Clustering Validation Techniques. Journal of Intelligent Information Systems, (17):107–145, 2001.

[Harold04] Harold, E. R., Means, W. S., et al. XML in a Nutshell. O'Reilly Media, 3ª ed$^{\underline{\text{ón}}}$., 2004.

[Huang08] Huang, A. Similarity Measures for Text Document Clustering. Proceedings of the Sixth New Zealand Computer Science Research Student Conference (NZCSRSC2008), Christchurch, New Zealand, págs. 49–56, 2008.

[Jain10] Jain, A. K. Data clustering: 50 years beyond K-means. Pattern Recognition Letters, (31):651–666, 2010.

[Jajuga00] Jajuga, K. y Walesiak, M. Classification and information processing at the turn of the millennium. SpringerVerlag, 2000.

[Jung-Wong04] Jung-Wong, L. y Scung-Soo, P. Computing Similarity Between XML Documents for XML Mining. En Engineering Knowledge in the Age of the SemanticWeb. INEX 2004, págs. 492–493. Springer Berlin Heidelberg, 2004.

[Kish75] Kish, L. Muestreo de encuestas. Trillas, México, 1975.

[Kutty08] Kutty, S., Tran, T., Nayak, R., y Li, Y. Clustering XML Documents Using Closed Frequent Subtrees: A Structural Similarity Approach. En Focused Axes to XML Document. INEX 2007, págs. 183–194. Springer-Verlag Berlin Heidelberg, 2008.

[Larose07] Larose, D. T. Discovering Knowledge in Data. Wiley, 2007.

[Lebart84] Lebart, L., Morineau, A., y Warkwick, K. Multivariate Descriptive Analysis. Wiley, 1984.

[Mathieu12] Mathieu, M. J., Galindo-Durán, C. K., y Vázquez, H. J. Configurable application designed for mining xml document collections. En S. I. Ao, O. Castillo, y X. Huang, eds., Intelligent Control and Innovative Computing, tomo 110 de Lecture Notes in Electrical Engineering, págs. 233–245. Springer US, 2012. ISBN 978-1-4614-1695-1.

[Pinto Avendaño09] Pinto Avendaño, D. E. Tratamiento de Textos Cortos : Agrupamiento y Evaluación. Tesis Doctoral, Universidad Politécnica de Valencia, 2009.

[Ramírez04] Ramírez, M. J. y Ferri, C. Introducción a la Minería de Datos, tomo 1. Pearson Prentice Hall, 1ª edón., 2004.

[Rodríguez01] Rodríguez, M. E., Álvarez, S., y Bravo, E. Coeficientes de Asociación, Primera Edición. Universidad Autónoma Metropolitana - Iztapalapa, 2001.

[Roux85] Roux, M. Algorithmes de Classification. Editions Masson, 1985.

[Rumbaugh99] Rumbaugh, J., Jacobson, I., y Booch, G. El Lenguaje Unificado de Modelado. Addisson-Wesley, 1999.

[SAX10] Simple API for XML (SAX), 2010.
URL `http://www.saxproject.org/`

[Sheaffer87] Sheaffer, R. L., Mendenhall, W., y Ott, L. Elementos de muestreo. Iberoaméricana, México, 1987.

[Srivastava09] Srivastava, A. N. y Mehran, S. Text Mining Classification, Clustering, and Applications. CRC Press Taylor & Francis Group, 2009.

[Team11] Team, R. D. C. R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria, 2011.
URL `http://www.r-project.org/`

[Tran08a] Tran, T., Nayak, R., y Bruza, P. Combining Structure and Content Similarities for XML Document Clustering. Seventh Australasian Data Mining Conference (AusDM 2008), 2008.

[Tran08b] Tran, T., Nayak, R., y Bruza, P. Document Clustering Using Incremental and Pairwise Approaches. En Focused Access to XML Documents. INEX 2007, págs. 222–233. Springer-Verlag Berlin Heidelberg, 2008.

[Usama Fayyad96] Usama Fayyad, G. P.-S. y Smyth, P. From data mining to knowledge discovery in databases. AI Magazine, 17(3):37–54, 1996.

[Vázquez07] Vázquez, H. J., Martínez, A. F. J., y Monroy, A. G. S. Más allá del conocimiento un enfoque sistémico. Administración y Organizaciones, (19), 2007.
URL `http://bidi.xoc.uam.mx/tabla_contenido_fasciculo.php?id_fasciculo=371`

[Wu09] Wu, X. y Vipin, K. The Top Ten Algorithms in Data Mining. CRC Pres, 2009.

[XML10] Xml. el nuevo lenguaje universal, 2010.
URL `http://www.bibliociencias.cu/gsdl/collect/eventos/index/assoc/HASH0104/f016d031.dir/doc.pdf`