



UNIVERSIDAD DE GUANAJUATO

---

---

DIVISIÓN DE INGENIERÍAS  
CAMPUS IRAPUATO - SALAMANCA

“DESARROLLO DE UN SISTEMA PARA  
MEDIR SIMILITUD ENTRE CLASES”

TESIS PROFESIONAL

QUE PARA OBTENER EL TÍTULO DE:  
INGENIERO EN SISTEMAS COMPUTACIONALES

PRESENTA:

JUAN CARLOS RUIZ GONZÁLEZ

ASESORES:

DR. RAFAEL GUZMÁN CABRERA  
DR. MIGUEL TORRES CISNEROS

SALAMANCA, GUANAJUATO

ABRIL 2015



# Agradecimientos

*Pendientes agradecimientos*

# Resumen

El incremento continuo de información en formato digital obliga a contar con nuevos métodos y técnicas para acceder, recopilar y organizar estos volúmenes de información textual. Una de las técnicas más utilizadas para organizar las información es la clasificación de documentos. Los sistemas de clasificación automática de textos tienen una baja eficiencia cuando las clases son muy parecidas, y en este caso es muy importante el poder identificar aquellos atributos que nos permiten separar una clase de otra. En otras palabras: el poder utilizar únicamente en el sistema de clasificación aquellos atributos relevantes que permiten la separación entre clases. En este trabajo se presenta un sistema para generar gráficas de similitud entre documentos pertenecientes a clases de un corpus dado, tarea previa al proceso de clasificación automática. Estas gráficas son utilizadas como un método de refinamiento auxiliándose de las similitudes entre los documentos no clasificados. Mediante la herramienta de desarrollo Qt5, se diseña una interfaz de usuario la cual facilita la manipulación de un corpus de documentos seleccionado y a su vez, da la oportunidad de seleccionar entre diferentes métricas de similitud para la generación de gráficas en las que se aprecia qué tan similar es una clase con respecto de la otra. Con esto se busca poder anticipar el desempeño de un método de clasificación automática.

# Índice general

<b>Agradecimientos</b>	<b>3</b>
Resumen . . . . .	I
<b>1. Antecedentes</b>	<b>1</b>
1.1. Descripción del problema . . . . .	1
1.2. Objetivo de la tesis . . . . .	3
1.3. Estructura de la tesis . . . . .	4
<b>2. Clasificación automática de textos</b>	<b>5</b>
2.1. Aprendizaje automático . . . . .	5
2.2. Clasificación de textos . . . . .	8
2.2.1. Pre-procesamiento . . . . .	9
2.2.2. Representación de documentos . . . . .	9
2.2.3. Esquemas de pesado . . . . .	10
2.2.4. Reducción de dimensionalidad . . . . .	11
Umbral de Frecuencia . . . . .	11
Ganancia de información . . . . .	12
2.3. Métodos de clasificación . . . . .	12
2.3.1. Clasificador por Naive Bayes . . . . .	13
2.3.2. Clasificador por Vecinos más Cercanos (KNN) . . . . .	14
2.3.3. Clasificador por Prototipos . . . . .	15
2.3.4. Clasificador de Máquinas de Vectores de Soporte (SVM) . . . . .	17
2.3.5. Medidas de evaluación . . . . .	19
Exactitud . . . . .	19
Precisión . . . . .	20
Recuerdo . . . . .	20
F-measure( $F_\beta$ ) . . . . .	21
Micropromedio . . . . .	21

<b>ÍNDICE GENERAL</b>	<b>III</b>
Macropromedio . . . . .	21
<b>3. Desarrollo</b>	<b>22</b>
3.1. Medidas de similitud . . . . .	22
3.1.1. Manhattan . . . . .	22
3.1.2. Dice . . . . .	23
3.1.3. Coseno . . . . .	23
3.1.4. Jaccard . . . . .	23
3.2. Programación . . . . .	23
3.2.1. Selección de las tecnologías de desarrollo . . . . .	24
C++ . . . . .	24
STL . . . . .	25
Bibliotecas Boost . . . . .	25
3.2.2. Abstracción de procesos . . . . .	26
Lectura de documentos . . . . .	26
Procesamiento de documentos . . . . .	27
Cálculo de similitud . . . . .	28
3.2.3. Modelado de las clases . . . . .	29
Document . . . . .	29
Corpus . . . . .	30
Corpora . . . . .	31
Preprocesor . . . . .	31
Metrics . . . . .	33
3.3. Desarrollo de Interfaz Gráfica . . . . .	35
3.3.1. Framework Qt . . . . .	35
3.3.2. QCustomPlot . . . . .	35
3.3.3. Diseño de módulos . . . . .	36
Ventana principal del programa . . . . .	36
Módulo de histogramas de frecuencia . . . . .	40
Módulo de gráficas de similitud . . . . .	41
Módulo auxiliar de procesamiento de XML . . . . .	42
<b>4. Pruebas y resultados</b>	<b>43</b>
4.1. Corpus de Estudio . . . . .	43
4.2. Gráficas de similitud . . . . .	44
4.2.1. Gráficas de 1600 documentos . . . . .	44
4.2.2. Gráficas de 4000 documentos . . . . .	69
4.3. Pruebas de Clasificación . . . . .	94

<b>ÍNDICE GENERAL</b>	<b>IV</b>
4.3.1. Clasificación 1600 documentos . . . . .	94
4.3.2. Clasificación 4000 documentos . . . . .	96
<b>5. Conclusiones</b>	<b>97</b>
<b>Trabajo Futuro</b>	<b>98</b>
A. Bibliotecas Boost	99
B. Registros de resultados	101
C. Registros de resultados	110

# Índice de figuras

2.1.	Estructura básica de un sistema de aprendizaje automático. . . . .	6
2.2.	Proceso de clasificación de documentos. . . . .	8
2.3.	Clasificación por vecinos más cercanos. . . . .	15
2.4.	Clasificación por prototipos. . . . .	16
2.5.	Problema de clasificación linealmente separable. . . . .	17
2.6.	Un par de hiperplanos y sus márgenes de riesgo de error. . . . .	18
2.7.	Transformación a un espacio de alta dimensionalidad [18] . . . . .	19
3.1.	Diagrama de flujo del proceso del programa. . . . .	26
3.2.	Procesamiento de documentos. . . . .	27
3.3.	Diagrama de clase: Document. . . . .	29
3.4.	Diagrama de clase: Corpus. . . . .	30
3.5.	Diagrama de clase: Corpora. . . . .	31
3.6.	Diagrama de clase: Preprocesor. . . . .	32
3.7.	Diagrama de clase: Metrics. . . . .	34
3.8.	Logotipo del framework Qt. . . . .	35
3.9.	Logotipo de la biblioteca QCustomPlot. . . . .	36
3.10.	Ejemplo de mapa de color generado con QCustomPlot. . . . .	36
3.11.	Ventana principal de la aplicación. . . . .	37
3.12.	Cuadro de diálogo para selección de palabras de paro. . . . .	38
3.13.	Ejemplo de directorio de corpus. . . . .	38
3.14.	Ejemplo de subdirectorios de las clases. . . . .	38
3.15.	Ejemplo de formato de documentos de una clase. . . . .	39
3.16.	Módulo de visualización de histogramas. . . . .	40
3.17.	Módulo de visualización de gráficas de similitud. . . . .	41
3.18.	Sub-módulo de matriz de confusión. . . . .	42
3.19.	Módulo auxiliar XMLParser. . . . .	42

## ÍNDICE DE FIGURAS

VI

4.1. Similitud entre clases resultante de cada métrica. . . . .	45
4.2. Métricas aplicadas al conjunto de 1600 documentos. . . . .	46
4.3. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento). . . . .	47
4.4. Métricas aplicadas al subconjunto de entrenamiento. . . . .	48
4.5. Similitud entre clases resultante de cada métrica (subconjunto de prueba). . . . .	49
4.6. Métricas aplicadas al subconjunto de prueba. . . . .	50
4.7. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 2. . . . .	51
4.8. Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 2. . . . .	52
4.9. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 2. . . . .	53
4.10. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 2. . . . .	54
4.11. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 2. . . . .	55
4.12. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 2. . . . .	56
4.13. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 3. . . . .	57
4.14. Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 3. . . . .	58
4.15. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 3. . . . .	59
4.16. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 3. . . . .	60
4.17. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 3. . . . .	61
4.18. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 3. . . . .	62
4.19. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 4. . . . .	63
4.20. Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 4. . . . .	64
4.21. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 4. . . . .	65

4.22. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 4. . . . .	66
4.23. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 4. . . . .	67
4.24. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 4. . . . .	68
4.25. Similitud entre clases resultante de cada métrica. . . . .	70
4.26. Métricas aplicadas al conjunto de 4000 documentos. . . . .	71
4.27. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento). . . . .	72
4.28. Métricas aplicadas al subconjunto de entrenamiento. . . . .	73
4.29. Similitud entre clases resultante de cada métrica (subconjunto de prueba). . . . .	74
4.30. Métricas aplicadas al subconjunto de prueba. . . . .	75
4.31. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 2. . . . .	76
4.32. Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 2. . . . .	77
4.33. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 2. . . . .	78
4.34. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 2. . . . .	79
4.35. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 2. . . . .	80
4.36. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 2. . . . .	81
4.37. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 3. . . . .	82
4.38. Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 3. . . . .	83
4.39. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 3. . . . .	84
4.40. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 3. . . . .	85
4.41. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 3. . . . .	86
4.42. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 3. . . . .	87

## *ÍNDICE DE FIGURAS*

VIII

4.43. Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 4. . . . .	88
4.44. Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 4. . . . .	89
4.45. Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 4. . . . .	90
4.46. Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 4. . . . .	91
4.47. Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 4. . . . .	92
4.48. Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 4. . . . .	93
4.49. Exactitud de los clasificadores para 1600 documentos. . . . .	95
4.50. Exactitud de los clasificadores para 4000 documentos. . . . .	96

# Índice de tablas

1.1.	Ejemplo de tabla de frecuencias. . . . .	2
1.2.	Ejemplo de matriz de similitud de documentos. . . . .	3
2.1.	Matriz de confusión de la clase $c_i$ . . . . .	20
2.2.	Micropromedio y macropromedio a partir de la matriz de confusión. . . . .	21
3.1.	Equivalencias entre símbolos XML y HTML. . . . .	32
4.1.	Resultados clasificación 1600 documentos. . . . .	95
4.2.	Resultados clasificación 4000 documentos. . . . .	96
A.1.	Listado de algunas bibliotecas Boost y su funcionalidad. . . .	100

# Capítulo 1

## Antecedentes

### 1.1. Descripción del problema

En la actualidad el almacenamiento de contenido digital se ha vuelto más abundante y menos costoso. Esto ha provocado que la cantidad de información digital generada por compañías de diferentes rubros crezca a una gran velocidad, generando de esta forma, grandes repositorios de conocimiento. Sin embargo, esto ha provocado la necesidad de crear técnicas para poder clasificar de manera automática estos volúmenes de datos [1].

En el caso de clasificación documentos de texto, los documentos son convertidos de su contenido original, a arreglos de información los cuales representan el contenido de esos documentos. Una de las técnicas más utilizadas para la representación de los documentos es la de usar la característica de frecuencia de aparición de una palabra o frase en el documento [1]. Generando de esta manera una tabla la cual representa el vocabulario del documento.

En la tabla 1.1 se muestra una representación esquemática del vocabulario de una clase, donde  $w_{ki}$  representa la  $i - \text{esima}$  palabra con una frecuencia de aparición  $f_{ki}$  en el vocabulario.

Siendo de esta manera posible realizar el proceso de clasificación mediante diferentes técnicas, como pueden ser:

- Clasificadores Bayesianos.
- Clasificadores de vecinos más cercanos.

Tabla 1.1: Ejemplo de tabla de frecuencias.

$\omega_k$	$f_k$
$\omega_{k1}$	$f_{k1}$
$\omega_{k2}$	$f_{k2}$
$\vdots$	$\vdots$
$\vdots$	$\vdots$
$\omega_{kn}$	$f_{kn}$

- Clasificadores por prototipos.
- Clasificadores de máquinas de vectores de soporte.

No obstante, estos procesos de clasificación requieren un gran número de ejemplos de entrenamiento manualmente etiquetados para poder aprender adecuadamente. La tarea de etiquetado a menudo debe ser realizada por un experto en el tema, este proceso consume una gran cantidad de tiempo [2].

Además de la necesidad de ejemplos para el entrenamiento, un problema recurrente al que se enfrenta un clasificador, es el de la similitud entre los documentos de diferentes clases, este problema consiste en qué tan parecido es un documento de una clase  $a$  con respecto de una clase  $b$ . La similitud, por lo general está representada por una escala numérica entre 0 y 1, donde 0 representa que no existe similitud alguna y 1 representa que el documento es el mismo. Para una mejor visualización de esta información, se utiliza una matriz de similitud de documentos.

En la tabla 1.2 se muestra una representación de la matriz de similitud de documentos. Donde 0 representa que las palabras entre los documentos, son totalmente diferentes. Mientras que 1 representa que los documentos son 100 % similares.

Por lo que contar con una herramienta que permita saber desde un inicio qué grado de similitud entre los documentos existe, así como el poder eliminar tanto las llamadas palabras vacías (stopwords) como las palabras recurrentes, puede ser una buena forma de saber de manera anticipada la exactitud del clasificador.

Tabla 1.2: Ejemplo de matriz de similitud de documentos.

	$d_{11}$	$d_{12}$	$\cdots$	$d_{21}$	$d_{22}$	$\cdots$	$d_{nm}$
$d_{11}$	1	0	$\cdots$	0	0	$\cdots$	0
$d_{12}$	0	1	$\cdots$	0	0	$\cdots$	0
$\cdots$	0	0	$\ddots$	0	0	$\cdots$	0
$d_{21}$	0	0	$\cdots$	1	0	$\cdots$	0
$d_{22}$	0	0	$\cdots$	0	1	$\cdots$	0
$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\cdots$	$\ddots$	$\cdots$
$d_{nm}$	0	0	$\cdots$	0	0	$\cdots$	1

## 1.2. Objetivo de la tesis

El presente trabajo de tesis tiene como objetivo principal la creación de una herramienta para generación y visualización de gráficas de similitud de documentos entre dos o más clases. La herramienta debe tener, además, las siguientes características:

- Una forma intuitiva de cargar el corpus de documentos.
- Permitir el filtrado de las palabras de paro al momento de la lectura de los documentos.
- Poseer una interfaz gráfica que permita visualizar las tablas de frecuencia procesadas.
- Poder aplicar un umbral de frecuencia el cual elimine las palabras poco frecuentes en los documentos.
- Permitir visualizar los histogramas de frecuencia generados por cada una de las clases, así como de un documento en específico.
- Implementación de diferentes métricas de similitud entre documentos.
- Capacidad para guardar la información generada por la herramienta, en un formato que se pueda utilizar en otros entornos como Matlab, R o similares.

Con estas características, se busca que la herramienta pueda ser utilizada en diferentes problemáticas como un auxiliar previo a la tarea de categorización, permitiendo de este modo, saber con anticipación, cómo será el comportamiento del clasificador automático y con esto incrementar la exactitud de clasificación al seleccionar los mejores atributos.

### 1.3. Estructura de la tesis

El presente documento está organizado de la siguiente manera:

En el capítulo dos se introducen los conceptos básicos sobre la clasificación de textos, incluyendo el pre-procesamiento de la información, la forma de representación de documentos, esquemas de pesado, los métodos de clasificación relacionados y las medidas de evaluación para determinar el éxito de la clasificación.

En el capítulo tres se presenta el desarrollo de la herramienta, se abordan las diferentes métricas de similitud implementadas dentro del proyecto, se abordan los conceptos aplicados para la programación de las métricas y las características del programa y finalmente se aborda la creación de la interfaz gráfica encargada de la interacción con el usuario.

En el capítulo cuatro se muestran las pruebas y resultados obtenidos del uso del programa, se presenta el corpus de estudio seleccionado, la realización de las pruebas de clasificación y los resultados obtenidos y finalmente se presenta el cálculo del rendimiento del clasificador con respecto a la información entregada por el programa.

Finalmente en el capítulo cinco, se presentan las conclusiones de la tesis y el trabajo futuro.

# Capítulo 2

## Clasificación automática de textos

La clasificación automática de documentos es la tarea en la cual los documentos, en formato electrónico, son categorizados dentro de un conjunto de categorías (clases) previamente definidas utilizando información etiquetada como entrenamiento. La clasificación automática de documentos tiene múltiples aplicaciones, como son la clasificación automática de opiniones [3], la clasificación automática de patentes [4], entre otros. En este capítulo se presentan las principales tareas relacionadas con la clasificación de textos.

### 2.1. Aprendizaje automático

El Aprendizaje automático es una rama de la inteligencia artificial que tiene como objetivo desarrollar técnicas que permitan a las computadoras aprender a desarrollar tareas que los seres humanos hacemos de forma natural y rápida, como por ejemplo, reconocer imágenes, entender el lenguaje natural, tomar decisiones, etc.

Las técnicas de aprendizaje computacional se han utilizado frecuentemente para resolver problemas donde se manejan grandes cantidades de información y es necesario encontrar un patrón que permita determinar el comportamiento de dicha información. El objetivo del aprendizaje computacional es desarrollar modelos que sean capaces de aprender de la experiencia previa de los eventos que se presentan, a partir de conjuntos de datos [5]. La finalidad de los modelos es extraer información implícita dentro de los datos para poder hacer predicciones y tomar decisiones sobre nuevos datos.

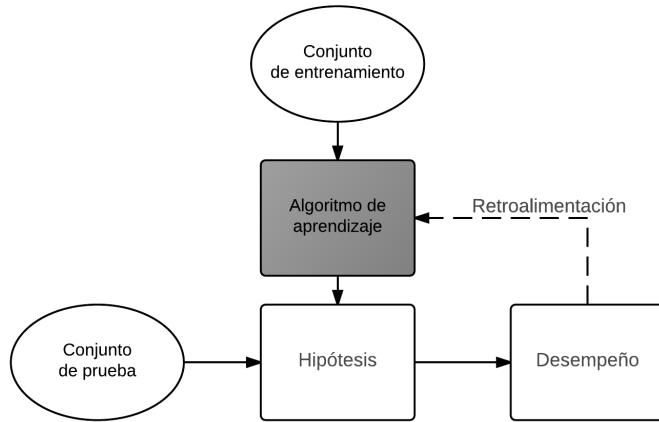


Figura 2.1: Estructura básica de un sistema de aprendizaje automático.

Una definición comúnmente utilizada es la siguiente: “*El aprendizaje automático es el estudio de algoritmos computacionales que van mejorando automáticamente su desempeño a través de la experiencia*” [6]. Más formalmente se puede enunciar: “*se dice que un programa de computadora aprende de una experiencia E con respecto a un grupo de tareas T y una medida de rendimiento P, si su desempeño en tareas T medidas por P mejoran con la experiencia E*”. En la figura 2.1 se presenta el esquema básico de un sistema de aprendizaje automático

Existen varias tareas que se pueden abordar con sistemas de aprendizaje. Éstas pueden clasificarse como:

1. *Tareas de predicción*: De manera general, estas tareas se pueden dividir en dos, clasificación y regresión.
  - La *clasificación* es una tarea básica en el análisis de datos y en el reconocimiento de patrones. La clasificación o categorización se define como la tarea de asignar objetos de un universo a dos o más clases predefinidas. Se conocen como clases o categorías a las opciones con las que se puede asignar una etiqueta. Para tomar una decisión de la clase a la que pertenecen los objetos, es necesario conocer las características particulares de cada clase [5].
  - La *regresión o estimación* tiene como objetivo inducir un modelo para poder predecir el valor futuro de una variable dados los va-

lores presentes y pasados de los atributos. Por ejemplo, estimar la producción de gasolina en una refinería [5].

2. *Tareas descriptivas:* Son usadas para el análisis preliminar de los datos. Buscan derivar descripciones concisas de características de los datos, por ejemplo, medias y desviaciones estándares, entre otras, que permitan describir a un conjunto de datos [7].
3. *Tareas de segmentación:* tratan de buscar una separación de los datos en subgrupos o categorías de acuerdo a un cierto criterio. Las categorías pueden ser exhaustivas y mutuamente excluyentes o jerárquicas. En esta tarea es común utilizar algoritmos de clustering, principalmente cuando no se cuenta con un conjunto de entrenamiento [8].
4. *Tareas de análisis de dependencias:* el valor de un elemento puede usarse para predecir el valor de otro. La dependencia puede ser probabilística, por medio de una red de dependencias o puede ser funcional [5].
5. *Tareas de detección de desviaciones de casos extremos o anomalías:* permiten detectar los cambios más significativos en los datos con respecto a valores pasados o normales y filtra grandes volúmenes de datos que son menos probables de ser seleccionados. El problema central de esta tarea se encuentra en determinar cuándo una desviación es significativa para ser de interés [7].
6. *Tareas de aprendizaje en base a experiencia:* se utiliza información y retroalimentación de soluciones para mejorar el desempeño de un sistema basado en aprendizaje automático, entre más se utiliza el sistema, mayor será la experiencia adquirida y por tanto tendrá un mejor desempeño [5].
7. *Tareas de búsqueda:* se utilizan principalmente para resolver algún problema de optimización. Involucran el uso de algoritmos genéticos y técnicas de búsqueda local, con la finalidad de encontrar información relevante a una petición hecha por el usuario [9].

En base tanto a la definición de aprendizaje automático, como a las tareas que con él se pueden llevar a cabo utilizando sistemas de aprendizaje, podemos decir que ocurre aprendizaje automático en un programa, si éste puede modificar aspectos de sí mismo, de tal modo que en una ejecución

subsecuente con la misma entrada, se produce un resultado mejor. Uno de los tipos de aprendizaje automático es el aprendizaje supervisado donde al algoritmo de aprendizaje se le proporciona un conjunto de entrada con las correspondientes salidas correctas, y a partir de éstas el algoritmo “aprende” comparando su salida con la correcta, con esto sabe el error, luego entonces, se modifica para corregir [5]. Un ejemplo de aplicación de este tipo de aprendizaje es la clasificación automática de documentos.

## 2.2. Clasificación de textos

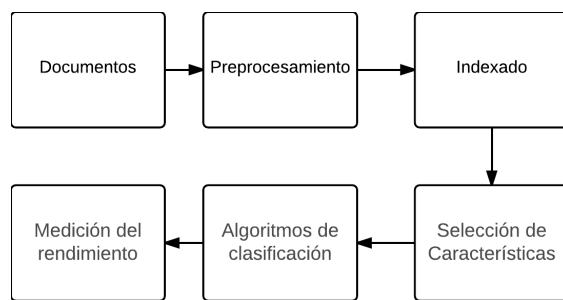


Figura 2.2: Proceso de clasificación de documentos.

En la clasificación de textos el conjunto de objetos a clasificar  $O = \{o_1, \dots, o_n\}$  es un grupo de documentos  $D = \{d_1, \dots, d_n\}$ . El conjunto de documentos  $D_E$ , al cual le fue asignado cada valor de  $\phi : (d_j, c_i)$ , se conoce como corpus de entrenamiento; usualmente el etiquetado se realiza manualmente por expertos que deben leer los documentos antes de asignar una clase.

El conjunto de prueba  $D_P$  contiene datos que no han sido utilizados durante el entrenamiento, es decir  $D_E \cap D_P = \emptyset$ . El objetivo de poseer conjuntos de prueba es saber qué tan bien trabaja un modelo particular con ejemplos nuevos. Por ello los datos de prueba y de entrenamiento deben ser diferentes para que la prueba sea válida.

Una vez que se cuenta con los dos conjuntos de documentos etiquetados que servirán como entrenamiento y prueba, los documentos deben ser preparados a fin de ser utilizados por el clasificador, para ello es necesario aplicar ciertos pasos previos. A continuación se presentan algunos procesos relacionados a la representación de los documentos.

En la figura 2.2 se muestra el flujo del proceso de clasificación de textos. Primeramente los documentos son preprocesados, removiendo datos poco relevantes para el proceso, posteriormente los documentos son indexados convirtiéndolos a un formato más manejable para el proceso de clasificación. Una vez indexados, se realiza el proceso de selección de características con las que se aplicará la clasificación. Posteriormente se aplican los algoritmos de clasificación seleccionados y se hace una comparación de rendimiento entre estos. En las siguientes secciones se describe cada una de estas etapas.

### 2.2.1. Pre-procesamiento

El primer paso en la clasificación de documentos es la transformación de los documentos, los cuales típicamente son cadenas de caracteres, en una representación que sea compatible con algún algoritmo de aprendizaje utilizado para llevar a cabo la tarea de clasificación de los documentos. La transformación textual usualmente consiste en las siguientes acciones:

- *Remover etiquetas*: Las etiquetas tanto en HTML como en XML, entre otras, se utilizan normalmente para organizar las colecciones tanto de entrenamiento como de prueba bajo diferentes categorías o rubros como puede ser temática, fecha o el nombre del autor. Sin embargo, éstas deben ser removidas para poder llevar a cabo la tarea de clasificación utilizando sólo la información del texto en cuestión.
- *Eliminar palabras vacías*: También conocidas como palabras de paro o stopwords; son las palabras de aparición frecuente como los artículos, pronombres, preposiciones, etc. Estas palabras tienen poco impacto a la hora de determinar la clase, por lo que es conveniente eliminarlas desde un inicio.
- *Usar un lematizador*: Consiste en obtener las raíces morfológicas de las palabras. De esta manera, por ejemplo, la palabra caminar, camino, caminaré o caminé, serán llevados a una misma raíz léxica.

### 2.2.2. Representación de documentos

La representación de documentos consiste en mapear los documentos a una forma compacta la cual pueda utilizarse en el clasificador. La forma más

común para representar cada documento es un vector con términos ponderados como entradas, concepto tomado del modelo de espacio vectorial usado en recuperación de información [10]. Es decir, si la colección de documentos está representada por  $D = \{d_1, d_2, \dots, d_n\}$ , donde  $d_i$  es un documento y  $n$  es el número de documentos en la colección, para cada documento se forma una representación matricial, por ejemplo el  $k$ -ésimo estará representando por  $d_j = \{w_{1k}, w_{2k}, \dots, w_{mk}\}$ , donde  $m$  es el número de términos en la colección.

Existen varias maneras de determinar el peso de  $w_{ij}$  del término  $i$  en el documento  $j$ , el peso se asocia comúnmente con la importancia que el término en cuestión tiene para distinguir una categoría de otra. En el próximo apartado se abordan algunos de los esquemas de pesado más comunes.

### 2.2.3. Esquemas de pesado

Sea  $\#(t_k, d_j)$  el número de ocurrencias de un término  $t_k$  en el documento  $d_j$ ,  $|D|$  el número de documentos del conjunto y  $\#D(t_k)$  el número de documentos en los que aparece el término  $t_k$ .

- **Pesado Booleano:** Asigna un 1 si la palabra aparece en el documento, y un 0 si no aparece.

$$w_{kj} = \begin{cases} 1 & \text{si } \#(t_k, d_j) \geq 1 \\ 0 & \text{en otro caso} \end{cases} \quad (2.1)$$

- **Frecuencia del Término (TF):** Asigna el número de veces que aparece la palabra en el texto y 0 si no aparece.

$$w_{kj} = \#(t_k, d_j) \quad (2.2)$$

- **TFxIDF:** Term Frequency x Inverse Document Frequency. Establece una relación entre la frecuencia del término en el documento y la frecuencia de éste en el resto de los documentos de la colección.

$$tfidf(t_k, d_j) = \#(t_k, d_j) \cdot \log\left(\frac{|D|}{\#D(t_k)}\right) \quad (2.3)$$

Existen otros esquemas de pesado que consideran factores adicionales, por ejemplo, el tamaño de los documentos, la existencia de clases desbalanceadas, es decir, las clases cuentan con un número significativamente diferentes de instancias de entrenamiento.

#### 2.2.4. Reducción de dimensionalidad

Un problema central en la clasificación automática de textos es la alta dimensionalidad del espacio de características o atributos. La dimensionalidad está dada por el número de palabras distintas, típicamente miles, que tiene una colección de documentos. Si se utilizan las técnicas estándar de clasificación, cuando se tiene un conjunto de características muy grande, esto implica un alto costo computacional. Sin embargo, este costo, la mayoría de las veces, no representa una mejora significativa en los resultados obtenidos. De aquí surge entonces la necesidad de reducir la dimensionalidad del espacio de características.

En la clasificación automática de documentos la alta dimensionalidad del espacio de términos puede ocasionar un sobre-ajuste en el proceso de aprendizaje [10], lo cual provoca problemas de efectividad debido a que el sistema de clasificación tiende a comportarse mejor sobre los datos con los que ha sido entrenado y no conserva la tendencia en aquéllos no vistos.

Además la alta dimensionalidad también se refleja en la eficiencia, haciendo el problema menos tratable para el método de aprendizaje. Para disminuir el problema se selecciona un subconjunto de  $m$  de atributos. Este proceso, que comúnmente se le conoce como *selección de características*, permite reducir significativamente la dimensionalidad, es decir, su efecto es reducir el tamaño del vector de características de  $m$  a  $m'$ , siendo  $m' \ll m$ ; donde el conjunto  $m'$  es llamado *conjunto de términos reducido*. La reducción se hace calculando una función de calidad para los términos, y seleccionando aquéllos con mayor calificación. Existen varias técnicas para poder aplicar la reducción de dimensionalidad, a continuación se presentan algunas de estas:

##### Umbral de Frecuencia

La reducción por umbral de frecuencia es una de las más sencillas, si el número de documentos en los cuales ocurre un término es menor a cierto umbral predefinido, dicho término es removido. Este criterio se basa en la

suposición de que las palabras que rara vez ocurren en una colección no son informativas para la predicción de la clase y por lo tanto no tienen influencia en el desempeño global del clasificador [10].

### Ganancia de información

La Ganancia de Información (IG por sus siglas en inglés Information Gain) mide la entropía del sistema con la presencia o ausencia de cada palabra en un documento, es decir, mide si el grado de desorden del sistema se incrementa o reduce al conocer el valor de un atributo determinado. Con ello, asigna un ‘valor de información’ a cada atributo, calculándose como:

$$\begin{aligned} IG(t_k) &= \sum_{i=1}^{|C|} P(c_i) \log P(c_i) \\ &+ P(t_k) \sum_{i=1}^{|C|} P(c_i|t_k) \log P(c_i|t_k) \\ &+ P(\bar{t}_k) \sum_{i=1}^{|C|} P(c_i|\bar{t}_k) \log P(c_i|\bar{t}_k) \end{aligned} \quad (2.4)$$

Donde  $P(c_i)$  es la probabilidad de la clase  $c_i$  y se estima con la cantidad de documentos de la colección total que pertenecen a la clase  $c_i$ ;  $P(t_k)$  es la probabilidad de seleccionar un documento que contenga el término  $t_k$  y se estima que la porción de documentos en los cuales ocurre el término;  $P(c_i|t_k)$  es la probabilidad condicional de que un documento pertenezca a la clase  $c_i$  dado que el documento contiene al término  $t_k$ , se obtiene con los documentos de la clase  $c_i$  en los que el término ocurre al menos una vez; de la misma forma,  $P(c_i|\bar{t}_k)$  es la probabilidad condicional de que un documento pertenezca a la clase  $c_i$  dado que el documento no contiene el término  $t_k$ .

La ganancia de información se calcula para cada palabra del conjunto de entrenamiento y aquéllas cuya ganancia es menor a determinado umbral son eliminadas. Típicamente se conservan aquellas palabras cuya IG es positiva.

### 2.3. Métodos de clasificación

Dentro del aprendizaje computacional existen múltiples métodos de clasificación. Sin embargo, para el manejo de texto los que han obtenido mejores

resultados son Naive Bayes, Máquinas de Vectores de Soporte, Vecinos más Cercanos y Rocchio. En esta sección se describe el funcionamiento de cada uno de ellos.

### 2.3.1. Clasificador por Naive Bayes

El método bayesiano o probabilístico ha sido ampliamente usado para la clasificación de documentos [11]. Este método usa la probabilidad conjunta de las palabras y las categorías para estimar la probabilidad  $P(c_i|d_j)$  de cada categoría, dado un documento.

Si se tiene un conjunto de documentos  $D = \{d_1, d_2, \dots, d_m\}$  asociado a las clases previamente definidas  $C = \{c_1, c_2, \dots, c_n\}$ , cada documento es representado por un vector  $d_j = (w_{1j}, w_{2j}, \dots, w_{\tau j})$  donde  $\tau$  es el conjunto de términos que pertenecen a  $c_i$ ; el método bayesiano estima la probabilidad a posteriori de cada categoría  $c_i$  dado el documento  $d_j$  de la siguiente manera:

$$P(c_i|d_j) = \frac{P(c_i)P(d_j|c_i)}{P(d_j)} \quad (2.5)$$

Donde  $P(d_j)$  es la probabilidad de que se elija aleatoriamente el documento  $d_j$  (esta probabilidad es independiente de las clases, por lo que se puede omitir) y  $P(c_i)$  es la probabilidad de que el documento elegido pertenezca a la categoría  $c_i$ . Debido a que el número de posibles documentos  $d_j$  es muy grande, se vuelve complicado el cálculo de  $P(d_j|c_i)$ .

Para simplificar el cálculo de  $P(d_j|c_i)$  es común asumir que la probabilidad de un término dado es independiente de los otros términos que aparecen en el mismo documento. Aunque a primera vista esto puede ser visto como una simplificación exagerada, el método Naive Bayes representa resultados comparables con los obtenidos por métodos más elaborados [12].

Usando esta simplificación es posible determinar  $P(d_j|c_i)$  con el producto de probabilidades de cada término que aparece en el documento, de la siguiente manera:

$$P(d_j|c_i) = \prod_{t=1}^{\tau} P(w_{tj}|c_i) \quad (2.6)$$

De las dos expresiones anteriores, tenemos que la probabilidad de que el documento  $d_j$  elegido aleatoriamente pertenezca a la categoría  $c_i$  es:

$$P(d_j|c_i) = P(c_i) \prod_{t=1}^{|\tau|} P(w_{tj}|c_i) \quad (2.7)$$

Con  $P(c_i)$  calculado como:  $P(c_i) = \frac{N_{c_i}}{N}$ ; es el número de documentos de la categoría  $c_i$  y  $N$  es el total de documentos en el conjunto de entrenamiento. Por su parte  $P(w_{tj})$  puede ser calculado como:

$$P(w_{tj}|c_i) = \frac{1 + \text{count}(w_{tj}, c_i)}{N_{c_i} + |\tau|} \quad (2.8)$$

Donde  $\text{count}(w_{tj}, c_i)$  es el número de veces que el término  $w_{tj}$  aparece en los documentos de la categoría  $c_i$ . Para resolver el problema de probabilidad cero se usa la estimación de Laplace, conocida como: Add-One Smoothing [13].

De esta manera a  $d_j$  se le asigna la categoría  $c_i$  donde  $P(c_i|d_j)$  es máxima. El método de Naive Bayes es muy popular en el área de clasificación de documentos, siendo ampliamente utilizado en múltiples investigaciones ([11, 14]).

### 2.3.2. Clasificador por Vecinos más Cercanos (KNN)

El clasificador de vecinos más cercanos (KNN, por sus siglas en inglés, K-nearest neighbors), es un clasificador de enfoque estadístico, el cual ha sido estudiado intensivamente en el área de reconocimiento de patrones, por alrededor de cuatro décadas [15].

El algoritmo KNN es bastante simple: Dado un documento de prueba a ser clasificado, el algoritmo busca los  $k$  vecinos más cercanos a través de los documentos previamente clasificados mediante alguna medida de similitud <sup>1</sup> y ordena esos documentos en base a sus valores de similitud. Las clases de los  $K$  vecinos más cercanos son utilizadas para predecir la clase del documento de prueba, utilizando los valores de similitud de los vecinos como un esquema de pesado entre clases candidatas, si más de un vecino pertenece a la misma categoría se utiliza la suma de sus ponderaciones como el peso de la clase. Finalmente, la clase con la mayor ponderación es a la que se le asignará el

---

<sup>1</sup>En el próximo capítulo se abordan con más detalle dichas medidas.

documento de prueba. En caso de un empate el proceso de asignación se dificulta teniendo que recurrir a realizar pruebas con  $k$  vecinos diferentes hasta lograr la clase más conveniente. Otra desventaja de este algoritmo, es la complejidad computacional necesaria para poder recorrer todos los documentos de entrenamiento. En la figura 2.3 se muestra un ejemplo del funcionamiento de este clasificador.

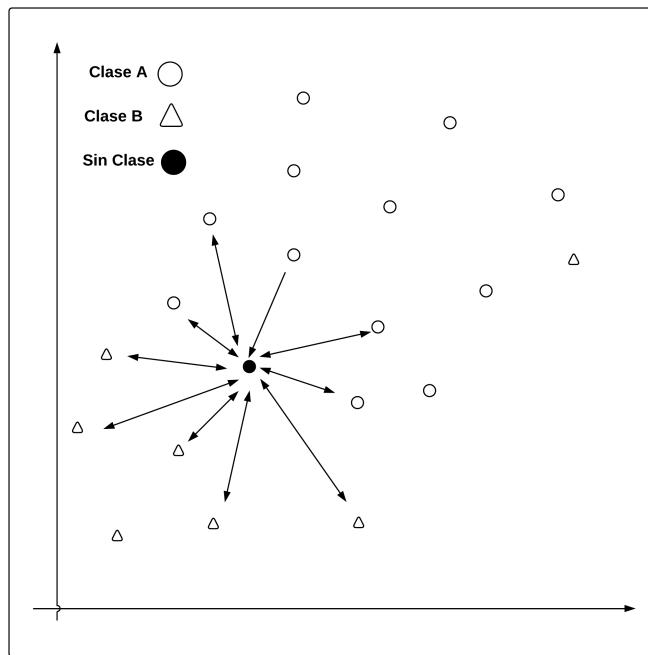


Figura 2.3: Clasificación por vecinos más cercanos.

### 2.3.3. Clasificador por Prototipos

El clasificador Rocchio es uno de los varios tipos de clasificadores que trabajan con prototipos. La idea general es construir un prototipo  $a_i$  de cada clase  $c_i \in C$ , de forma que, cuando un ejemplo nuevo  $d$  debe ser clasificado, solamente se compare con los prototipos y se asigne la clase de aquel que sea más similar. No es necesario que el ejemplo del prototipo exista, en algunos casos se calcula mediante promedio, suma normalizada o alguna otra medida de sus atributos. En la figura 2.4 se muestra un ejemplo del clasificador Rocchio, como se puede apreciar, a diferencia del clasificador KNN (ver fi-

gura 2.3) sólo se realiza la comparación contra un solo elemento de la clase (prototipo), mientras que el KNN realiza este proceso contra cada uno de los elementos de la clase.

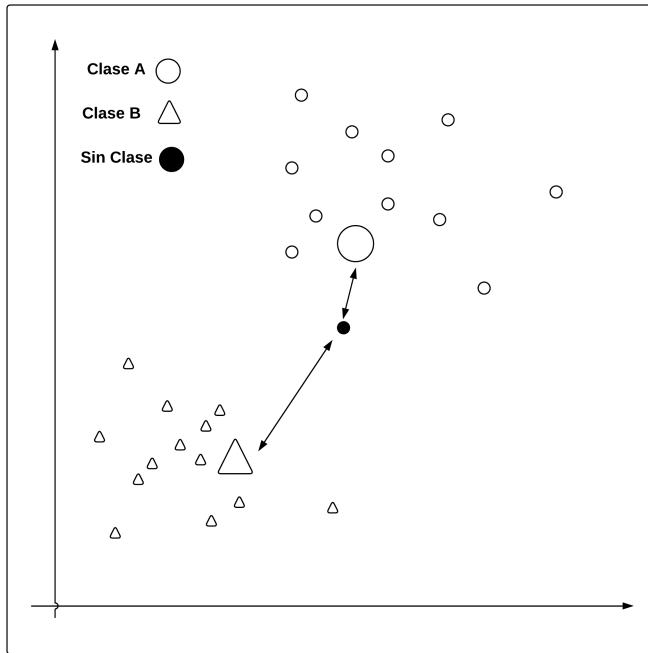


Figura 2.4: Clasificación por prototipos.

Específicamente en el clasificador Rocchio, para obtener el vector prototípico, se hace una suma ponderada de los documentos etiquetados con la clase  $c_i$  en el conjunto de entrenamiento menos la suma ponderada de los no pertenecientes a la clase  $c_i$ .

$$a_i = \beta \sum_{d_j \in c_i} d_j - \gamma \sum_{d_j \notin c_i} d_j \quad (2.9)$$

Donde  $\beta$  y  $\gamma$  son parámetros para dar peso a la Suma.

### 2.3.4. Clasificador de Máquinas de Vectores de Soporte (SVM)

Las máquinas de vectores de soporte fueron presentadas en 1995 por Vapnik [16], y fueron aplicadas por primera vez a la categorización de textos por Joachims [17]. Mientras la mayoría de los algoritmos de aprendizaje se centran en reducir los errores cometidos por el modelo generado, SVM (Support Vector Machines) no busca reducir el riesgo cometiendo pocos errores, sino que pretende construir modelos confiables [5]. Esta técnica tiene raíces en la teoría de aprendizaje estadístico. Básicamente mapea los documentos en un espacio de atributos de alta dimensionalidad e intenta aprender hiperplanos de margen máximo entre dos categorías de documentos. Además, representa los límites de decisión usando un subconjunto de ejemplos de entrenamiento, conocidos como vectores de soporte.

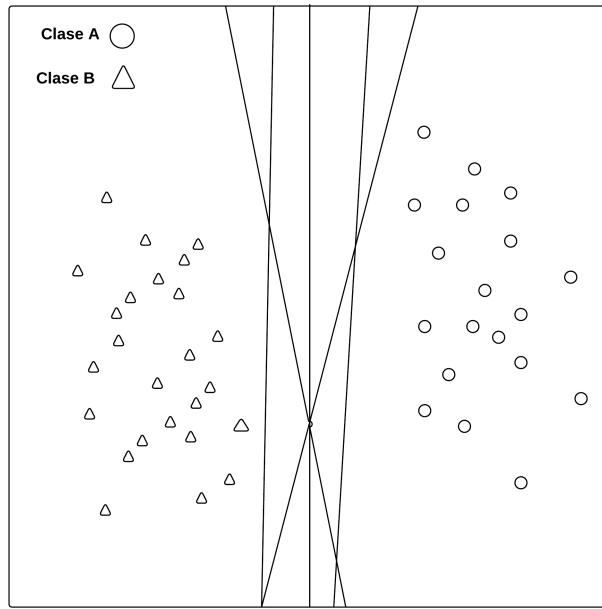


Figura 2.5: Problema de clasificación linealmente separable.

La Figura 2.5 muestra una gráfica de un conjunto de ejemplos de entrenamiento que pertenecen a dos diferentes categorías representadas por triángulos y círculos. Los datos son linealmente separables, es decir, podemos encontrar un hiperplano tal que todos los triángulos estén de un lado

del hiperplano y todos los círculos queden en el otro lado. Sin embargo, hay una infinidad de posibles hiperplanos, como se puede apreciar en la figura. El hecho de que estos hiperplanos no tengan ningún error al separar los ejemplos de entrenamiento, no garantiza que con nuevos documentos suceda lo mismo.

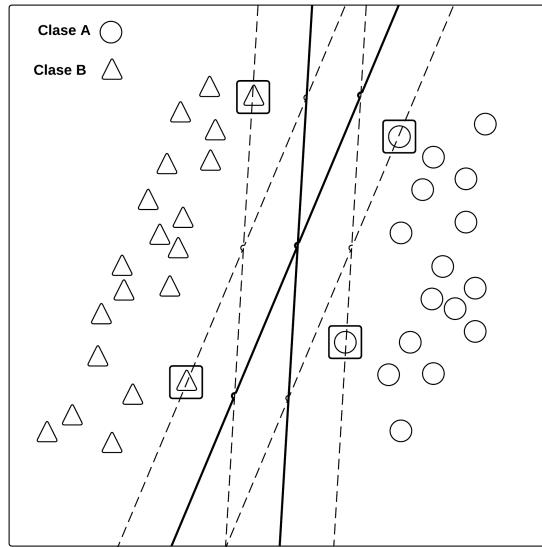


Figura 2.6: Un par de hiperplanos y sus márgenes de riesgo de error.

La Figura 2.6, muestra dos hiperplanos y sus márgenes de riesgo de error. Entre mayores son los márgenes, menor será el riesgo de que un documento nuevo sea clasificado de manera errónea. En esta figura los cuadros indican los ejemplos que son tomados como vectores de soporte. Esto es para el caso en que los conjuntos son linealmente separables.

SVM es considerado el primer método kernel porque, para problemas que no son linealmente separables, SVM usa funciones de convolución o Kernels. Aplicar kernels consiste en hacer una transformación del problema original mediante una representación de información distinta llevándolo a un espacio de alta dimensionalidad, donde los documentos transformados son linealmente separables (ver figura 2.7).

Este método es aplicable a problemas clasificación binaria, pero puede ser

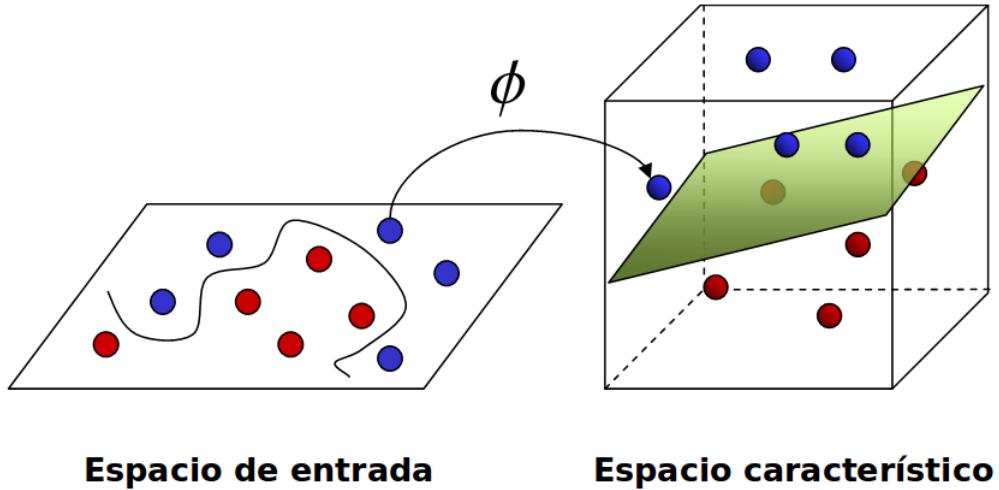


Figura 2.7: Transformación a un espacio de alta dimensionalidad [18]

extendido a problemas de más de dos categorías. Para estos casos, de tener  $n$  clases es necesario construir  $n - 1$  clasificadores [19].

### 2.3.5. Medidas de evaluación

La evaluación de los clasificadores de textos se conduce de forma experimental usualmente midiendo la exactitud del clasificador, es decir, la capacidad de tomar decisiones correctas.

Un clasificador genera una matriz de confusión como se muestra en la Tabla 2.1, donde  $TP_i$  indica el número de verdaderos positivos, es decir, cuántos documentos fueron correctamente clasificados bajo la clase  $c_i$ , de forma similar,  $FP_i$  indica el número de falsos positivos, aquéllos que fueron clasificados erróneamente como positivos.  $FN_i$  corresponde al número de falsos negativos y  $TN_i$  corresponde al número de verdaderos negativos. Con base en estos valores, se pueden calcular Exactitud ( $e$ ), Precisión ( $\pi$ ) y Recuerdo ( $\rho$ ), medidas que permiten saber qué tan exitoso es el clasificador.

#### Exactitud

La exactitud es una medida global ya que se refiere a la capacidad del clasificador para acertar en la clasificación, considerando correctamente clasifi-

Tabla 2.1: Matriz de confusión de la clase  $c_i$ .

Clase $c_i$		Decisión del experto	
		Si	No
Decisión del clasificador	Si	$TP_i$	$FP_i$
	No	$FN_i$	$TN_i$

ficados tanto a los ejemplos positivos como negativos. En breve, la exactitud es un valor entre 0 y 1 que representa el porcentaje de documentos correctamente clasificados y se obtiene mediante:

$$e = \sum_{i=1}^{|C|} \frac{TP_i + TN_i}{TP_i + TN_i + FP_i + FN_i} \quad (2.10)$$

Un inconveniente de esta medida de evaluación es que, si las clases están desbalanceadas y existe tendencia del clasificador a predecir la clase mayoritaria, entonces no reflejará la calidad real del clasificador ya que el asignar la clase más frecuente reflejará buenos resultados con esta medida.

## Precisión

La precisión indica la probabilidad de que el documento asignado a cierta clase por el clasificador, efectivamente pertenezca a esa clase. De la matriz de confusión, la precisión se obtiene por la ecuación:

$$\pi = \frac{TP_i}{TP_i + FP_i} \quad (2.11)$$

## Recuerdo

El recuerdo estima la probabilidad de que un documento que pertenezca a cierta clase, sea correctamente asignado durante el proceso de clasificación. De la matriz de confusión, el recuerdo se obtiene por:

$$\rho = \frac{TP_i}{TP_i + FN_i} \quad (2.12)$$

### F-measure( $F_\beta$ )

Comúnmente utilizada para englobar la precisión ( $\pi$ ) 2.11 y el recuerdo ( $\rho$ ) 2.12 se obtiene con:

$$F_\beta = \frac{(1 + \beta^2)\pi\rho}{\beta^2\pi + \rho} \quad (2.13)$$

Donde  $\beta$  es el parámetro que controla la importancia relativa entre las dos medidas. Usualmente se fija el valor  $\beta = 1$  para dar igual importancia a ambos valores.

Las definiciones dadas por 2.10, 2.11 y 2.12 son medidas de evaluación por categoría, para obtener datos globales se calculan los promedios. Existen dos formas de realizar el cálculo global, el micropromedio y el macropromedio.

### Micropromedio

El micropromedio calcula los parámetros  $TP_i, FP_i, FN_i$  y  $TN_i$  para todas las categorías obteniendo una medida global única, con la cual otorga el mismo peso a cada documento dando a las categorías una importancia proporcional al número de ejemplos positivos que le corresponden.

### Macropromedio

Es el promedio obteniendo las medidas de evaluación por categoría y posteriormente obtiene un promedio global. El macropromedio da el mismo peso a cada categoría. La diferencia entre micro y macro promedio debe considerarse si las clases son desbalanceadas. La Tabla 2.2 muestra cómo se obtienen de la matriz de confusión.

Tabla 2.2: Micropromedio y macropromedio a partir de la matriz de confusión.

	Micropromedio	Macropromedio
Precisión ( $\pi$ )	$\pi = \frac{\sum TP_i}{\sum(TP_i + FP_i)}$	$\pi = \frac{\sum(\frac{TP_i}{TP_i + FP_i})}{ C }$
Recuerdo ( $\rho$ )	$\rho = \frac{\sum TP_i}{\sum(TP_i + FN_i)}$	$\rho = \frac{\sum(\frac{TP_i}{TP_i + FN_i})}{ C }$

# Capítulo 3

## Desarrollo

Como se mencionó en el apartado 1.2, el objetivo de la tesis consiste en crear un sistema que genere y visualice gráficas de similitud entre documentos. En este capítulo se abordan las medidas de similitud implementadas en el sistema desarrollado para el presente trabajo, así como la programación y diseño de la interfaz de usuario de la herramienta.

### 3.1. Medidas de similitud

Algunos métodos de clasificación requieren determinar qué tan semejante es un documento con otro. Las medidas de similitud son capaces de expresar una cantidad numérica entre 0 y 1, donde 0 es nada semejante y 1 es la representación de la condición de igualdad. Sean  $d_i$  y  $d_j$  documentos representados de la forma  $d_j = (w_{1j}, w_{2j}, \dots, w_{mj})$ , entonces algunas medidas de similitud o distancia se definen en las siguientes subsecciones.

#### 3.1.1. Manhattan

La medida Manhattan es la llamada medida de bloques o calles, ya que se suman las distancias horizontales y verticales desde el punto de inicio hasta el punto final [20].

$$manhattan(d_i, d_j) = \sum |w_{ki} - w_{kj}| \quad (3.1)$$

### 3.1.2. Dice

El coeficiente de Dice determina la similitud entre dos documentos pesados dando importancia a los atributos de la intersección [21].

$$dice = (d_i, d_j) = \frac{2 \sum (w_{ki} \times w_{kj})}{\sum w_{ki} + \sum w_{kj}} \quad (3.2)$$

### 3.1.3. Coseno

La medida cosenoidal es una de las más populares para determinar la similitud de los documentos. El objetivo es determinar el ángulo entre dos vectores, en este caso los vectores de representación de los documentos [22].

$$coseno(d_i, d_j) = \frac{\sum w_{ki} \times w_{kj}}{\sqrt{\sum w_{ki}^2} \times \sqrt{\sum w_{kj}^2}} \quad (3.3)$$

### 3.1.4. Jaccard

El coeficiente de similitud de Jaccard o índice de Jaccard mide la similitud entre dos conjuntos de muestras. Aunque originalmente fue utilizado para comparar tipos de flores en un ecosistema [23], ha tenido buena aceptación en el campo del análisis de documentos [24].

$$jaccard(d_i, d_j) = \frac{\sum (w_{ik} \times w_{jk})}{(\sum w_{ik}^2 + \sum w_{jk}^2) - (\sum w_{ik} \times \sum w_{jk})} \quad (3.4)$$

## 3.2. Programación

Una vez sentadas las bases teóricas del proceso de extracción de información y clasificación de documentos, pasamos a la etapa de programación de la herramienta. En esta etapa se seleccionan las tecnologías a utilizar para la tarea, se realiza la abstracción de información necesaria para el modelado de clases y finalmente se programan las clases con las que posteriormente se desarrollará la interfaz de usuario.

### 3.2.1. Selección de las tecnologías de desarrollo

Para el desarrollo de la herramienta se optó por utilizar el lenguaje **C++** junto con las bibliotecas **STL** y **Boost** para la programación de los algoritmos, y el framework **Qt5** para el diseño de la interfaz de usuario, de éste ultimo se hablará en la sección siguiente.

**C++** es un lenguaje de programación de propósito general, el cual posee la cualidad de orientación a objetos y posee múltiples bibliotecas para acelerar el proceso de desarrollo. Para el desarrollo de este trabajo, se utilizan el conjunto de bibliotecas STL y las bibliotecas Boost, a continuación se presenta una descripción detallada de el lenguaje y las bibliotecas utilizadas.

#### C++

**C++** es un lenguaje de programación creado por Bjarne Stroustrup en el año 1983 [25]. Fue concebido como una extensión del lenguaje C. Tiene como principal característica, el soporte para orientar a objetos, así como la capacidad de manipular plantillas abstractas para la llamada programación genérica.

Se seleccionó este lenguaje para el desarrollo del programa, debido a que no es un lenguaje propietario; por lo que permite poder distribuir el programa y el código fuente sin tener que pagar alguna regalía a una empresa, como sería el caso si el desarrollo fuera hecho con Visual C++® (Microsoft [26]) o Matlab® (MathWorks [27]).

Otra de las razones para seleccionar C++ para el modelado del programa, es su capacidad para ser portado entre plataformas, de esta manera no se limita a que el programa corra sólo bajo la plataforma Microsoft Windows® [28], sino que mientras la programación se mantenga dentro del estándar, se puede ejecutar en plataformas basadas en Unix, como son las distribuciones Linux [29] y la plataforma Mac OS de la empresa Apple® [30].

Finalmente se escogió C++ como la mejor opción para el desarrollo por sus bibliotecas STL [31] así como por su bibliotecas Boost [32], las cuales contribuyen considerablemente a mejorar el tiempo de desarrollo de la aplicación.

## STL

La biblioteca de plantillas estándar (Standar Template Library), es un conjunto de bibliotecas de contenedores, algoritmos e iteradores, los cuales tienen como objetivo fomentar una programación genérica, a continuación se describen cada una de las partes que conforman la biblioteca.

- *Contenedores:* Permiten almacenar y organizar datos en la memoria, estos contenedores tienen la cualidad de almacenar diferentes tipos de datos y a su vez fomentan una mejor organización del código. Algunos de los contenedores incluidos son: arreglos, vectores, listas, colas, pilas, tablas hash, con sus respectivas variantes [33].
- *Algoritmos:* Son una colección de funciones, las cuales sirven para procesar la información contenida en los contenedores. Algunas de las operaciones principales son: ordenamiento (sort), copiado (copy), búsqueda (search), entre otras. Estas funciones están programadas de manera de plantilla por lo que se pueden utilizar con cualquier tipo de dato o clase; además estas funciones se pueden usar tanto con contenedores STL como con arreglos sencillos de C++ o contenedores propios del usuario [31, 34].
- *Iteradores:* Son la generalización del concepto de punteros, sirven para apuntar a los elementos de los contenedores. Se pueden incrementar de la misma manera que se incrementa un puntero. Los iteradores son pieza clave en la biblioteca STL, esto debido a que permiten conectar contenedores con algoritmos [31].

## Bibliotecas Boost

Boost es un conjunto de bibliotecas de código abierto, las cuales tienen como finalidad incrementar el alcance del lenguaje C++ [35]. Tienen una muy buena integración con las bibliotecas STL; al igual que STL, Boost posee un conjunto de contenedores, algoritmos e iteradores, los cuales permiten resolver tareas de una manera eficiente.

En el Apéndice A se presenta la lista de las librerías que engloba Boost. Para el desarrollo de la aplicación se utilizó “*Boost.Regex*”, la cual simplifica la tarea de utilizar expresiones regulares para la detección de patrones en los documentos.

### 3.2.2. Abstracción de procesos

El primer paso para poder programar las clases que componen el programa, es analizar el proceso que se desea realizar. En este caso el programa tiene como meta generar una gráfica de similitud entre los grupos de documentos que se le entreguen. En la figura 3.1 se muestra el proceso que realizará el programa en un esquema general.

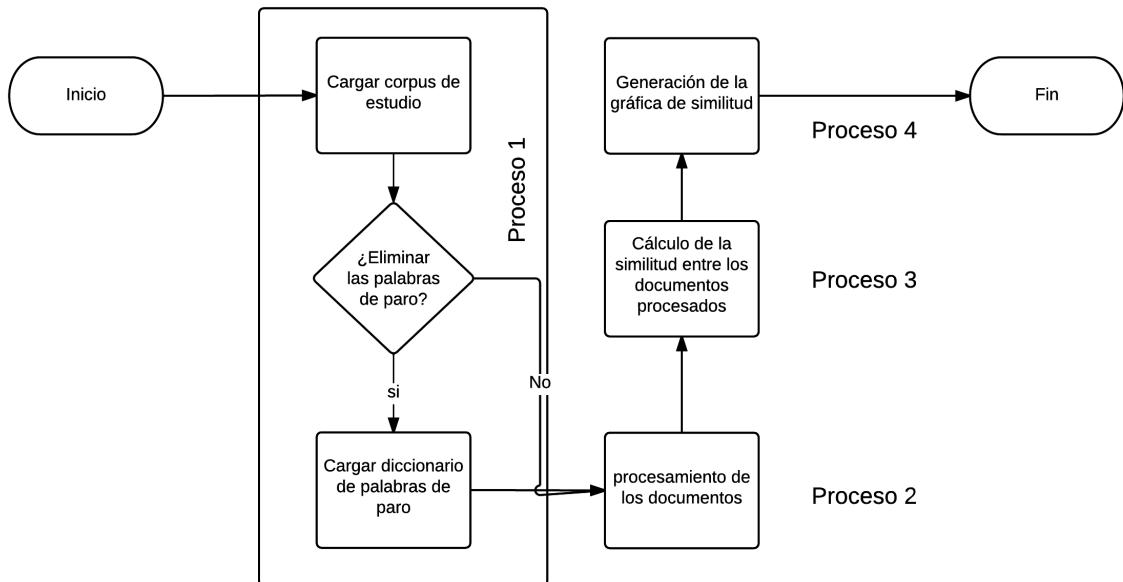


Figura 3.1: Diagrama de flujo del proceso del programa.

Una vez visualizado el proceso que se desea realizar, el siguiente paso es detectar los subprocessos que se realizarán dentro del objetivo principal de la aplicación. Para esta aplicación se tienen cuatro subprocessos principales a cubrir; la carga del corpus de estudio y las palabras de paro, el procesamiento de los documentos para poder compararlos, el cálculo de la similitud entre estos y finalmente la presentación de la gráfica; este último se abordará en la siguiente sección.

#### Lectura de documentos

En esta primer etapa, se carga el corpus de estudio el cual se procesará más adelante, además se contempla la opción de poder cargar una la lista de

palabras de paro las cuales se utilizarán en la siguiente etapa.

### Procesamiento de documentos

Una vez cargados los documentos, se busca convertir los documentos a una representación más adecuada para los fines del programa (véase sección 2.2.2). En este proceso se engloba la etapa de *preprocesamiento* de los documentos (véase sección 2.2.1), donde se eliminan los elementos que no son útiles para la tarea. En la figura 3.2 se muestran los pasos a realizar dentro de este proceso.

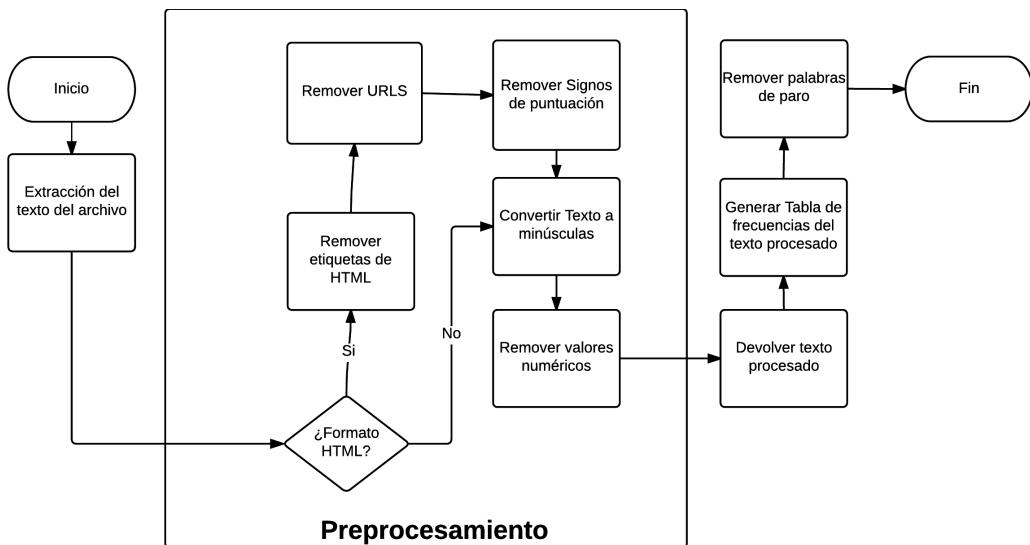


Figura 3.2: Procesamiento de documentos.

### Cálculo de similitud

En la etapa de cálculo de similitud se utilizan los documentos procesados previamente para generar la gráfica de similitud entre documentos a través de las métricas planteadas previamente (véase sección 3.1). Los pasos a seguir para calcular la similitud entre documentos son:

1. Balancear la tabla de frecuencias del documento  $i$  con respecto del documento  $j$ .
2. Calcular la similitud de documentos en base a la métrica seleccionada.
3. Devolver el valor de similitud (0 – 1).

Estos pasos deben seguirse para todos los documentos del corpus, hasta tener completa la gráfica de similitud.

### 3.2.3. Modelado de las clases

Una vez que se cuenta con una abstracción clara de los procesos lógicos que realiza el programa, se puede realizar el modelado de los atributos y métodos correspondientes a cada una de las clases que requeriremos para operar. A continuación se presentan las clases que representan la abstracción del problema, en la siguiente sección se abordarán las clases que representan el modelado de la interfaz de usuario.

#### Document

La clase *Document* tiene como finalidad el poder manipular la información de cada uno de los documentos que conforman un corpus. Posee atributos para almacenar el nombre del archivo, el contenido y su contenido ya procesado en una tabla de frecuencias. Para el manejo de la tabla de frecuencias, se utiliza un contenedor **STL map** el cual tiene como par ordenado la palabra y su frecuencia **<string,int>**, en la figura 3.3 se presenta el diagrama de la clase con sus respectivos atributos y métodos.

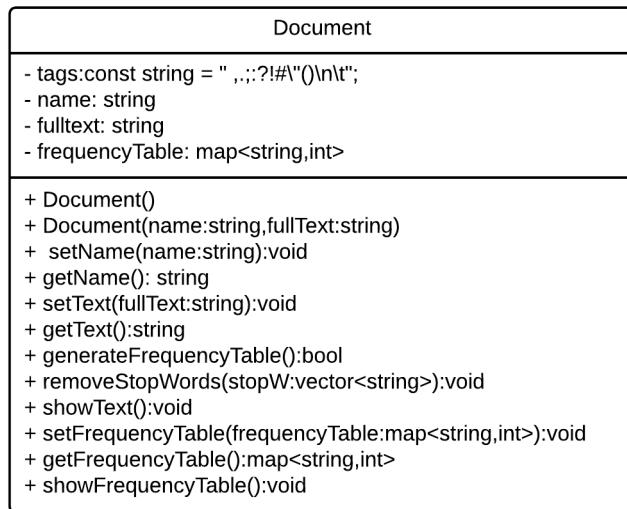


Figura 3.3: Diagrama de clase: Document.

## Corpus

La clase *Corpus* se encarga de almacenar una categoría específica del grupo de documentos. En caso de que los documentos no estén etiquetados, todos los documentos pertenecerán a la misma categoría. La clase posee atributos para poder identificar la categoría del corpus, un arreglo de documentos modelado en **STL vector** y una tabla de frecuencias que representa a toda la categoría. La figura 3.4 muestra el diagrama de la clase con todos los atributos y métodos necesarios.

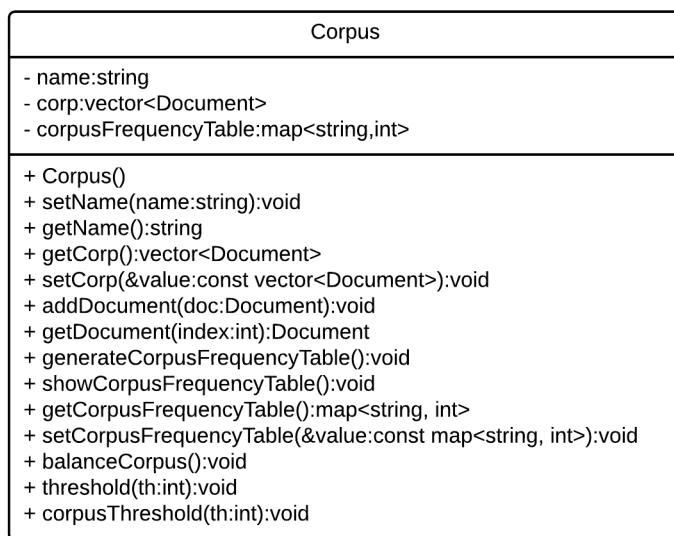


Figura 3.4: Diagrama de clase: Corpus.

De esta clase adicional a las operaciones de set y get, cabe destacar las siguientes operaciones:

- *balanceCorpus*: Agrega las palabras inexistentes en cada documento con respecto de los otros con valor de 0.
- *threshold*: Remueve las palabras que están por debajo del umbral en cada uno de los documentos y posteriormente regenera la tabla de frecuencias de la categoría.
- *corpusThreshold*: Remueve las palabras que están por debajo del umbral en la tabla de frecuencias de la categoría, sin alterar las tablas de frecuencias de los documentos.

## Corpora

Esta clase tiene como finalidad almacenar las categorías almacenadas en la clase corpus, se puede considerar como el global de los documentos. La clase posee un arreglo de categorías almacenadas en un **STL vector** y un **STL map** el cual almacena la tabla de frecuencias global del corpus de estudio. En la figura 3.5 se muestran los atributos y métodos que componen la clase.

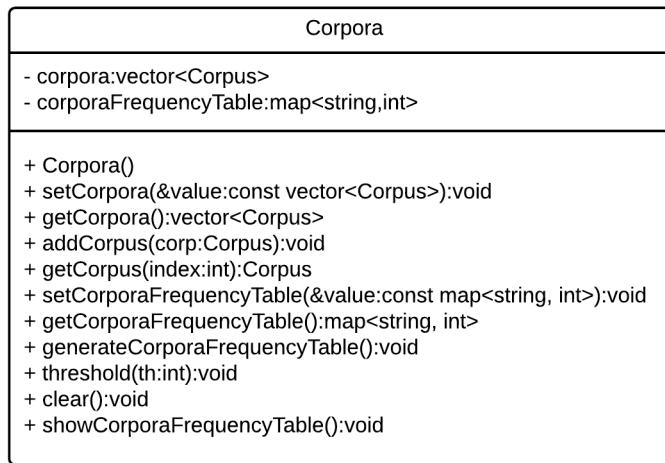


Figura 3.5: Diagrama de clase: Corpora.

En esta clase, el método de *threshold* actúa a manera de cascada invocando al método *threshold* de cada uno de los elementos del arreglo de categorías y posteriormente llamando a **generateCorporaFrequencyTable**, el cual genera una tabla de frecuencias global.

## Preprocesor

La clase *Preprocesor*, representa la mayor parte del proceso mostrado en la figura 3.2, las tres clases anteriores nos sirven como contenedores para poder manipular la información del corpus, mientras que esta clase, nos provee métodos para poder eliminar aquellos elementos del texto que no nos son útiles a la hora de aplicar las métricas de similitud. Esta clase se apoya de la biblioteca **Boost.Regex** para poder aplicar patrones generados con expresiones regulares de una manera más eficiente. En la figura 3.6 se muestran los métodos que posee la clase.

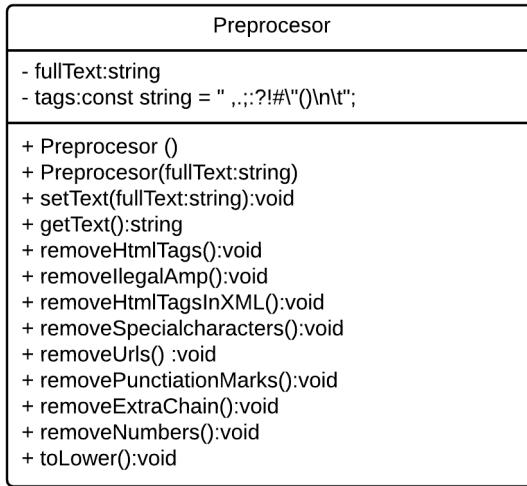


Figura 3.6: Diagrama de clase: Preprocesor.

Adicional a las operaciones de set y get, la clase Preprocesor nos permite hacer las siguientes operaciones de filtrado:

- *removeHtmlTags*: Permite remover las etiquetas de HTML mediante la expresión regular <[^>]\*>, en caso de que las etiquetas se encuentren en formato de XML, el método primero realizará una conversión a formato XML.

XML	HTML
&gt;	>
&lt;	<
&amp;	&
&quot;	”

Tabla 3.1: Equivalencias entre símbolos XML y HTML.

- *removeIllegalAmp*: Permite remover caracteres “&” ilegales en el texto del documento.
- *removeSpecialcharacters*: Permite remover caracteres especiales los cuales respondan al patrón "\032".

- *removeUrls*: Permite remover urls que se encuentren en el documento. Esto se logra utilizando la expresión regular:

```
((([A-Za-z]{3,9}:(?:\/\/)?)(?:[-;:&=\+\$,\\w]+@)?[A-Za-z0-9\\.\\-]+
|(?:www\\.|[-;:&=\+\$,\\w]+@)[A-Za-z0-9\\.\\-]+)((?:\\/[\\+\%\\\/.\\w\\-_]*?)?
\\?\\?(?:[-\+&;%@\\.\\w_]*#?(?:[\\.\\!\\/\\\\w]*))?)
```

- *removePunctuationMarks*: Tiene como función el remover signos de puntuación, como son puntos, comas, acentos, etc. Para lograrlo se utiliza la siguiente expresión regular: [^\w\sáéíóúñ].
- *removeExtraChain*: Elimina todos los espacios extra dentro del documento utilizando la expresión regular : [\w\W]\*
- *removeNumbers*: Tiene como función el remover toda expresión numérica que se encuentre en el documento, para ello se utiliza la expresión regular:

```
(\+|-)?[0-9]*(\.( [0-9] [0-9]*))?( (E|e)(\+|-)?[0-9]+)?
```

- *toLower*: Transforma todo el texto a minúsculas.

## Metrics

La clase Metrics posee la implementación de las diferentes métricas de similitud seleccionadas y posee varios métodos para manipular las tablas de frecuencias y entregar la matriz de similitud del corpus. La clase posee en su mayoría métodos privados, aprovechando de esta manera la propiedad de encapsulamiento de la programación orientada a objetos, permitiendo al usuario de la clase el uso sólo de los métodos cuyo fin último es el generar la matriz. En la figura 3.7 se muestran todos los métodos que conforman esta clase.

Metrics
<pre> - balance(&amp;tf1:map&lt;string,int&gt; ,&amp;tf2:map&lt;string,int&gt;):void - show(map&lt;string,int&gt; tf1):void - copyDocuments( *docs:vector&lt;Document&gt;, c:Corpora):void - copyfrequencyTables(*fts:vector&lt;map&lt;string, int&gt; &gt;, c:Corpora):void - transpose(mat:vector&lt;vector&lt;float&gt; &gt;):vector&lt;vector&lt;float&gt; &gt; - rotate(mat:vector&lt;vector&lt;float&gt; &gt; ):vector&lt;vector&lt;float&gt; &gt; - operationMode(&amp;mode:int,&amp;c:Corpora,&amp;docs:vector&lt;Document&gt;, &amp;fts:vector&lt;map&lt;string,int&gt; &gt;,&amp;matrix:vector&lt;vector&lt;float&gt; &gt;):void - copyDocsAndAllocate(&amp;c:Corpora,&amp;docs:vector&lt;Document&gt;, &amp;matrix:vector&lt;vector&lt;float&gt; &gt;):void - copyFtsAndAllocate(&amp;c:Corpora, &amp;fts:vector&lt;map&lt;string, int&gt; &gt;, &amp;matrix:vector&lt;vector&lt;float&gt; &gt;):void - manhattan(d1:Document,d2:Document):float - manhattan(doci:map&lt;string, int&gt;, docj:map&lt;string, int&gt;):float - dice(d1:Document,d2:Document):float - dice(doci:map&lt;string, int&gt;, docj:map&lt;string, int&gt;):float - cosMetric(d1:Document, d2:Document):float - cosMetric( doci:map&lt;string, int&gt;, docj:map&lt;string, int&gt;):float - jaccard( d1:Document, d2:Document):float - jaccard( doci:map&lt;string, int&gt;, docj:map&lt;string, int&gt;):float + Metrics() + generateManhattan( c:Corpora, mode:int):vector &lt; vector &lt;float &gt; &gt; + generateDice( c:Corpora, mode:int):vector &lt; vector &lt;float &gt; &gt; + generateCos( c:Corpora, mode:int):vector &lt; vector &lt;float &gt; &gt; + generateJaccard( c:Corpora, mode:int):vector &lt; vector &lt;float &gt; &gt; + multiplyByScalar(mat:vector&lt;vector&lt;float&gt; &gt; , scalar:float):vector&lt;vector&lt;float&gt; &gt; + negativeMatrix(mat:vector&lt;vector&lt;float&gt; &gt; , val:float):vector&lt;vector&lt;float&gt; &gt; + normalizeMatrix(mat:vector&lt;vector&lt;float&gt; &gt;):vector&lt;vector&lt;float&gt; &gt; </pre>

Figura 3.7: Diagrama de clase: Metrics.

### 3.3. Desarrollo de Interfaz Gráfica

En la sección anterior, se aborda meramente el desarrollo del programa a un nivel lógico. En esta sección se plantea el desarrollo de la interfaz de usuario la cual utiliza las clases previamente modeladas.

#### 3.3.1. Framework Qt



Figura 3.8: Logotipo del framework Qt.

Qt es un framework multiplataforma para desarrollo de aplicaciones de interfaz gráfica, permite utilizar C++ o QML, CSS y JavaScript como lenguajes de desarrollo. Qt puede usarse sin costo para proyectos cuyo licenciamiento sea de código abierto, o bien se puede pagar una licencia de uso comercial para proyectos privados [36]. Se eligió Qt sobre otros frameworks para desarrollo de interfaces de usuario, debido a su calidad de ser multiplataforma y permitir desarrollar en C++, lenguaje seleccionado para el desarrollo general de la aplicación.

#### 3.3.2. QCustomPlot

QCustomPlot es un Widget de código abierto C++ para Qt, el cual permite graficar y visualizar información. No requiere dependencias y está totalmente documentado. Esta biblioteca se enfoca en generar gráficas 2D y ofrece un alto rendimiento para visualización de aplicaciones en tiempo real [37]. QCustomPlot ofrece la posibilidad de exportar a varios formatos como PDF, PNG, JPG y BMP.



Figura 3.9: Logotipo de la biblioteca QCustomPlot.

Se eligió esta biblioteca, debido a su alta flexibilidad para la generación de gráficas de mapas de color, donde el color se utiliza como una tercera dimensión (ver figura 3.9), como es el caso de las gráficas de similitud. Además de que al ser una herramienta de código abierto se puede utilizar para el desarrollo sin tener que pagar licenciamiento alguno.

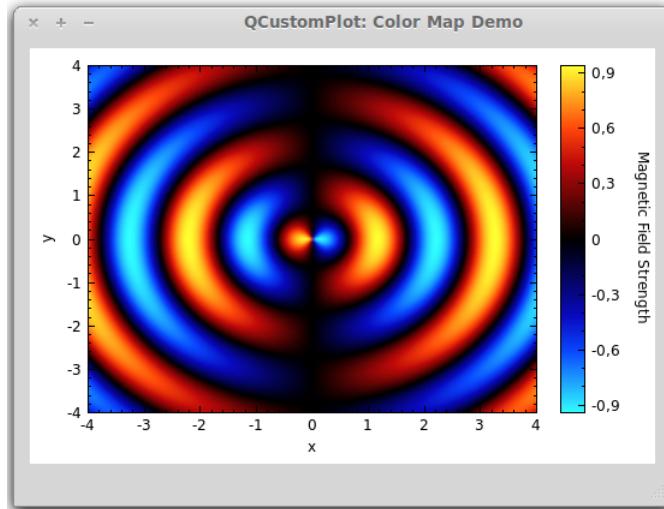


Figura 3.10: Ejemplo de mapa de color generado con QCustomPlot.

### 3.3.3. Diseño de módulos

La finalidad de la interfaz de gráfica es proveer una manera intuitiva de uso y manipulación de la información generada por el corpus, así como la posibilidad de exportar esta información en un formato legible para otras plataformas, como Matlab,R project, etc. A continuación se presentan los módulos que componen la interfaz con su respectiva descripción de uso.

### Ventana principal del programa

Este módulo consiste en el hilo principal de la aplicación. En él se puede realizar la carga del corpus a analizar y visualizar las clases, archivos y tablas de frecuencia, así como aplicar filtros, y llamar a los demás módulos del programa.

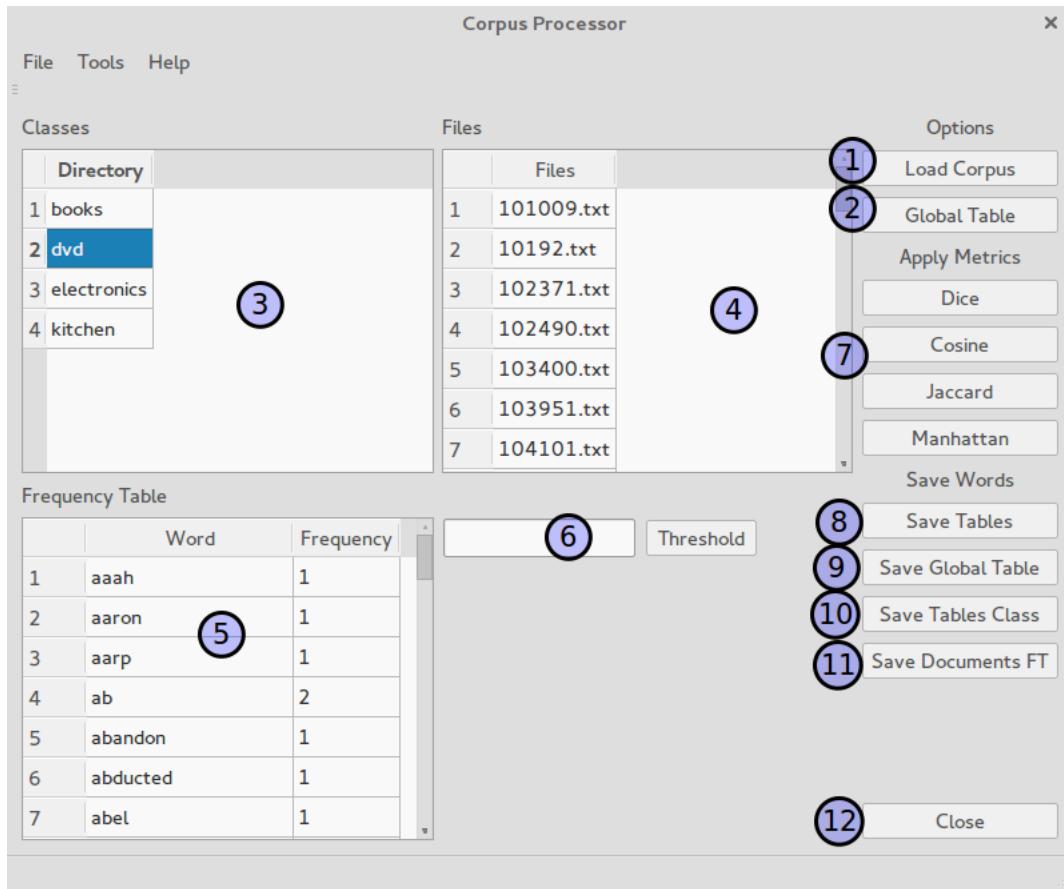


Figura 3.11: Ventana principal de la aplicación.

En la figura 3.10 se muestran los elementos que conforman la ventana, a continuación se presenta una descripción detallada de cada uno de estos.

1. *Load Corpus*: Sirve para realizar la carga de las palabras de paro (ver figura 3.11) y del corpus de estudio, para poder procesarlo, el corpus debe de presentarse de la siguiente manera:

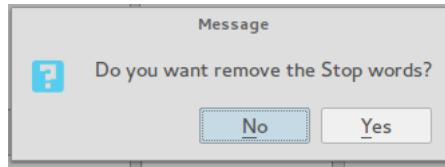


Figura 3.12: Cuadro de diálogo para selección de palabras de paro.

- El corpus deberá estar en un directorio (ver figura 3.12).

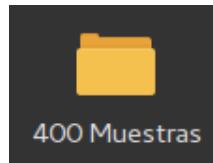


Figura 3.13: Ejemplo de directorio de corpus.

- Dentro del directorio se deberán presentar los directorios de las clases que lo conforman (ver figura 3.13).



Figura 3.14: Ejemplo de subdirectorios de las clases.

- Dentro de cada directorio deberán de estar cada una de las conversaciones en formato txt (ver figura 3.14).

De esta manera al cargar el corpus al sistema este pre-procesa cada uno de los documentos, calcula las tablas de frecuencias a todos los niveles (documentos, clases, corpus) y finalmente presenta la información como se muestra en la figura 3.10.

2. *Global Table*: El botón de Global Table, despliega en el elemento de Frequency Table (5) la tabla de frecuencias global de todas las palabras procesadas en los documentos.

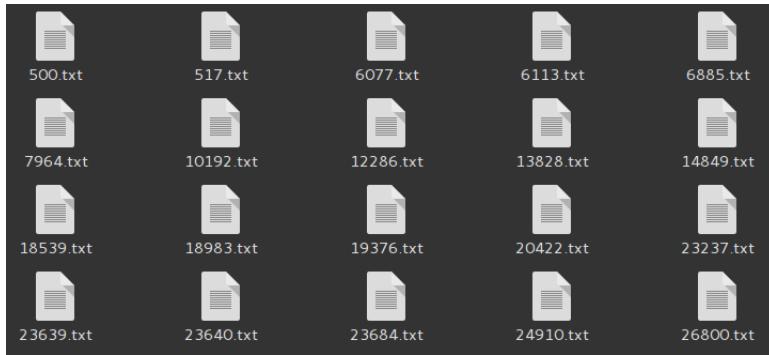


Figura 3.15: Ejemplo de formato de documentos de una clase.

3. *Classes*: En este elemento se muestran los diferentes directorios que conforman las clases del corpus, en caso de que el corpus no se encuentre clasificado, sólo se mostrará un directorio de clases.
4. *Files*: Muestra los documentos que conforman una clase dada, este elemento se actualiza dependiendo de la clase seleccionada en el elemento (3).
5. *Frequency Table*: Despliega las tablas de frecuencias que contiene la aplicación, puede mostrar tanto la tabla del corpus completo, así como la tabla de una clase seleccionada o de un solo documento seleccionado.
6. *Threshold*: Aplica un umbral de frecuencia a todos los documentos del corpus y posteriormente regenera las tablas de frecuencias a nivel de clase y global.
7. *Métricas*: Los botones contenidos en esta sección llaman a cada uno de los algoritmos implementados en la clase *Metrics*, y posteriormente invocan al módulo que muestra las gráficas de similitud.
8. *Save Tables*: Guarda en el directorio indicado, todos los documentos procesados a manera de tabla de frecuencias, la tabla de frecuencias de cada una de las clases y la tabla de frecuencias global del corpus, todo esto a manera de archivos en formato CSV.
9. *Save Global Table*: Genera un archivo en formato CSV de la tabla de frecuencias del corpus.

10. *Save Table Class*: Genera un archivo con las tablas de frecuencias en formato CSV por cada clase que se encuentre en el corpus.
  11. *Save Documents FT*: Crea un directorio donde almacena todos los documentos procesados a manera de tabla de frecuencias en formato CSV.
  12. *Close*: Cierra la aplicación.

Una función adicional de la ventana principal, es el desplegar el histograma de frecuencias referente a cada clase o a cada documento, este módulo se muestra ante el evento de dar doble clic en alguna de las clases o alguno de los documentos.

## Módulo de histogramas de frecuencia

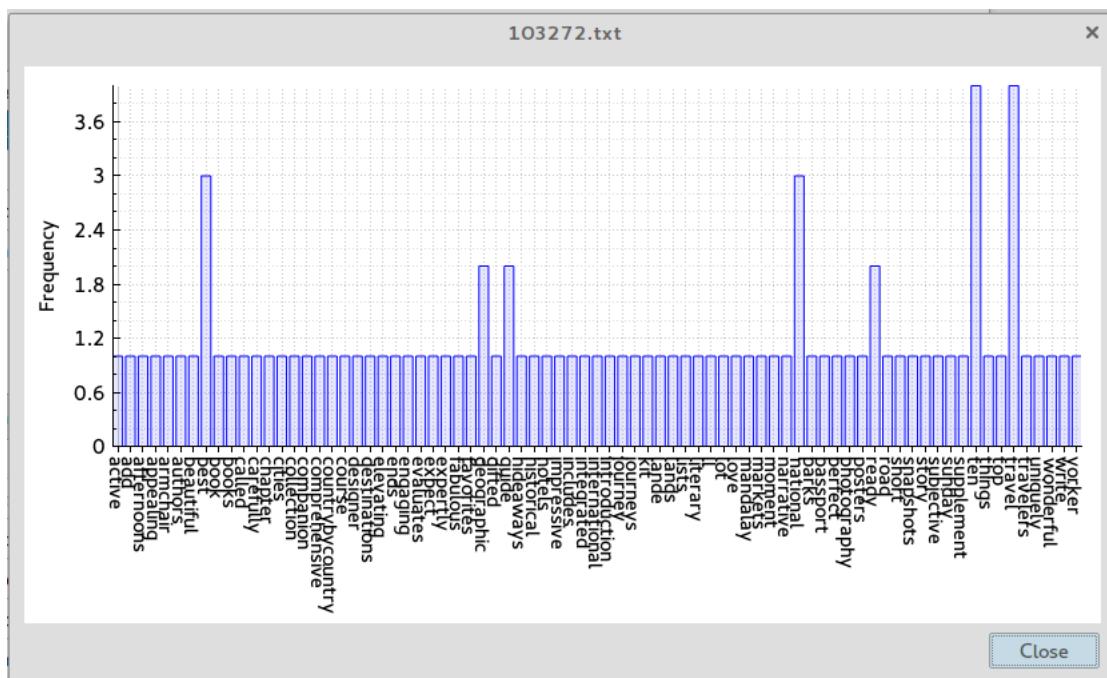


Figura 3.16: Módulo de visualización de histogramas.

El módulo de visualización de histogramas de frecuencia, tiene como finalidad entregar una asistencia visual al usuario para el análisis de las tablas

de frecuencias entregadas por el programa. Cuenta con la operación de acercar/alejar y funciona tanto para mostrar los histogramas de las clases, como de los documentos únicos.

### Módulo de gráficas de similitud

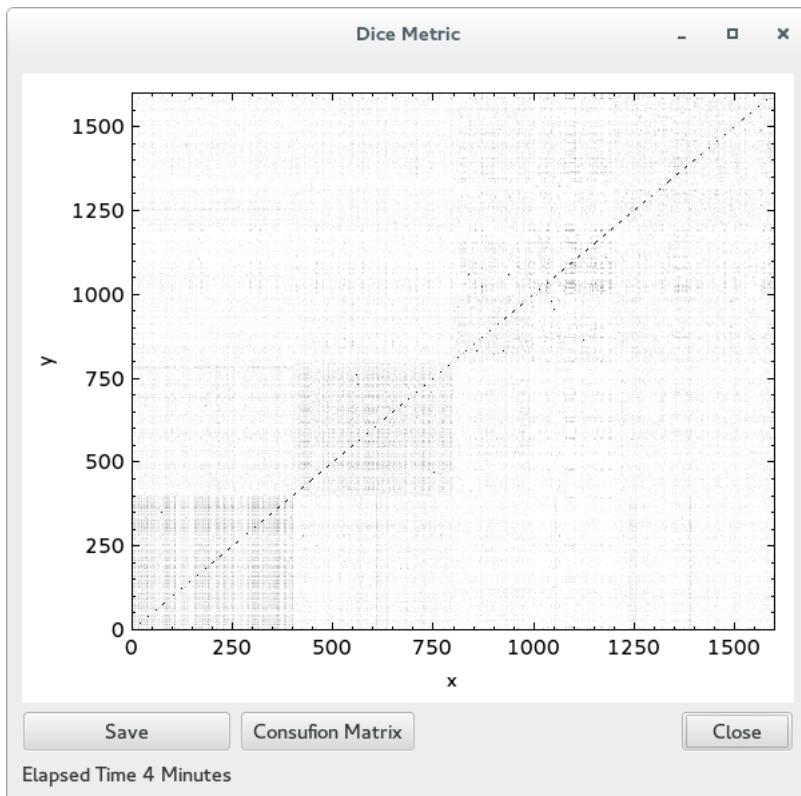


Figura 3.17: Módulo de visualización de gráficas de similitud.

El módulo de visualización de gráficas de similitud, es invocado mediante los botones de llamada para las métricas de similitud (ver figura 3.10). Permite visualizar la matriz de similitud generada por la clase *Metrics*, posee las funciones de acercar y alejar, y la función de guardar la imagen generada; si se utiliza esta última, el programa guarda la imagen en el formato solicitado y además genera un archivo formato CSV con la matriz de similitud que representa el documento. Por otra parte, el módulo posee la llamada a un sub-módulo el cual despliega la matriz de confusión entre las clases, en éste

se puede apreciar qué tan similar es una clase con respecto de otra y se puede exportar dicha matriz a un archivo CSV.

Confusión Matrix				
	books	dvd	electronics	kitchen
kitchen	0.260456	0.32886	0.653019	1
electronics	0.195024	0.270707	1	0.653019
dvd	0.367332	1	0.270707	0.32886
books	1	0.367332	0.195024	0.260456

**Save**      **Close**

Figura 3.18: Sub-módulo de matriz de confusión.

### Módulo auxiliar de procesamiento de XML

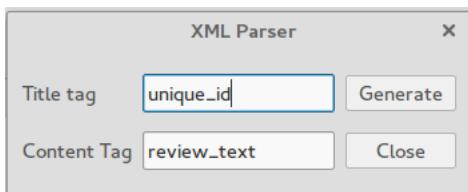


Figura 3.19: Módulo auxiliar XMLParser.

Este módulo se encuentra en el menú Tools de la ventana principal (ver figura 3.10). Fue diseñado pensando en los casos en el que el corpus de estudio es entregado como un solo archivo XML, el cual posee todos los documentos, se encarga de procesar cada uno de los contenidos y convertirlos a archivos con formato TXT. De esta manera, si el archivo XML posee 400 conversaciones, el módulo generará 400 archivos en formato TXT con las etiquetas que se le indiquen para mantener el título y el contenido del documento.

# Capítulo 4

## Pruebas y resultados

En el capítulo anterior se presentó el desarrollo del programa para calcular la similitud entre documentos, se definieron las medidas de similitud seleccionadas para implementar en el programa, se seleccionó el lenguaje de programación a utilizar, la abstracción de los procesos del programa, el modelado de las clases y el desarrollo de la interfaz de usuario. En este capítulo se presenta el corpus de estudio seleccionado para las pruebas, las pruebas de similitud entre clases y los resultados obtenidos en la clasificación.

### 4.1. Corpus de Estudio

El corpus de estudio utilizado para las pruebas es el “*Multi-Domain Sentiment Dataset (version 2.0)*” [38]. El corpus esta compuesto de opiniones de 4 diferentes categorías de productos tomados de la base de datos de Amazon [39]. Las cuatro categorías seleccionadas son:

- Libros. Compuesta de 1463 opiniones positivas y 1039 opiniones negativas.
- DVD's. Compuesta de 1391 opiniones positivas y 1396 opiniones negativas.
- Electrónicos. Compuesta de 948 opiniones positivas y 1014 opiniones negativas.
- Artículos de cocina. Compuesta de 832 opiniones positivas y 922 opiniones negativas.

En la siguiente sección, se presentan las pruebas realizadas utilizando diferentes muestras y umbrales de frecuencia.

## 4.2. Gráficas de similitud

Para la realización de las pruebas se crearon 2 conjuntos de documentos, uno compuesto de 1600 archivos (400 archivos en cada clase) y otro compuesto de 4000 archivos (1000 archivos en cada clase). Estos conjuntos a su vez, se subdividieron en un conjunto de entrenamiento (70 % del conjunto original) y un conjunto de prueba (30 % del conjunto original) los cuales se utilizaron para realizar el proceso clasificación el cual se presenta en la siguiente sección.

Se calculó la similitud de los conjuntos, utilizando cada unas de las métricas implementadas y se hicieron pruebas con diferentes umbrales de frecuencia. A continuación se presentan las gráficas resultantes agrupadas por métrica aplicada.

### 4.2.1. Gráficas de 1600 documentos

Las siguientes gráficas representan las pruebas realizadas al conjunto de 1600 documentos para los casos donde:

- Sólo se eliminan palabras de paro.
- Se aplica un umbral de frecuencia de 2.
- Se aplica un umbral de frecuencia de 3.
- Se aplica un umbral de frecuencia de 4.

Para cada caso, se muestran las gráficas del conjunto completo, y las gráficas de sus subconjuntos de prueba y entrenamiento.

	books	dvd	electronics	kitchen
kitchen	0.249855	0.402015	0.717055	1
electronics	0.204545	0.379918	1	0.717055
dvd	0.320256	1	0.379918	0.402015
books	1	0.320256	0.204545	0.249855

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.340418	0.427214	0.72559	1
electronics	0.311293	0.429679	1	0.72559
dvd	0.357125	1	0.429679	0.427214
books	1	0.357125	0.311293	0.340418

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.142763	0.251576	0.558913	1
electronics	0.113924	0.234506	1	0.558913
dvd	0.190657	1	0.234506	0.251576
books	1	0.190657	0.113924	0.142763

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.256555	0.562191	1
electronics	0.0317623	0.307405	1	0.562191
dvd	0.127063	1	0.307405	0.256555
books	1	0.127063	0.0317623	0

(d) Manhattan

Figura 4.1: Similitud entre clases resultante de cada métrica.

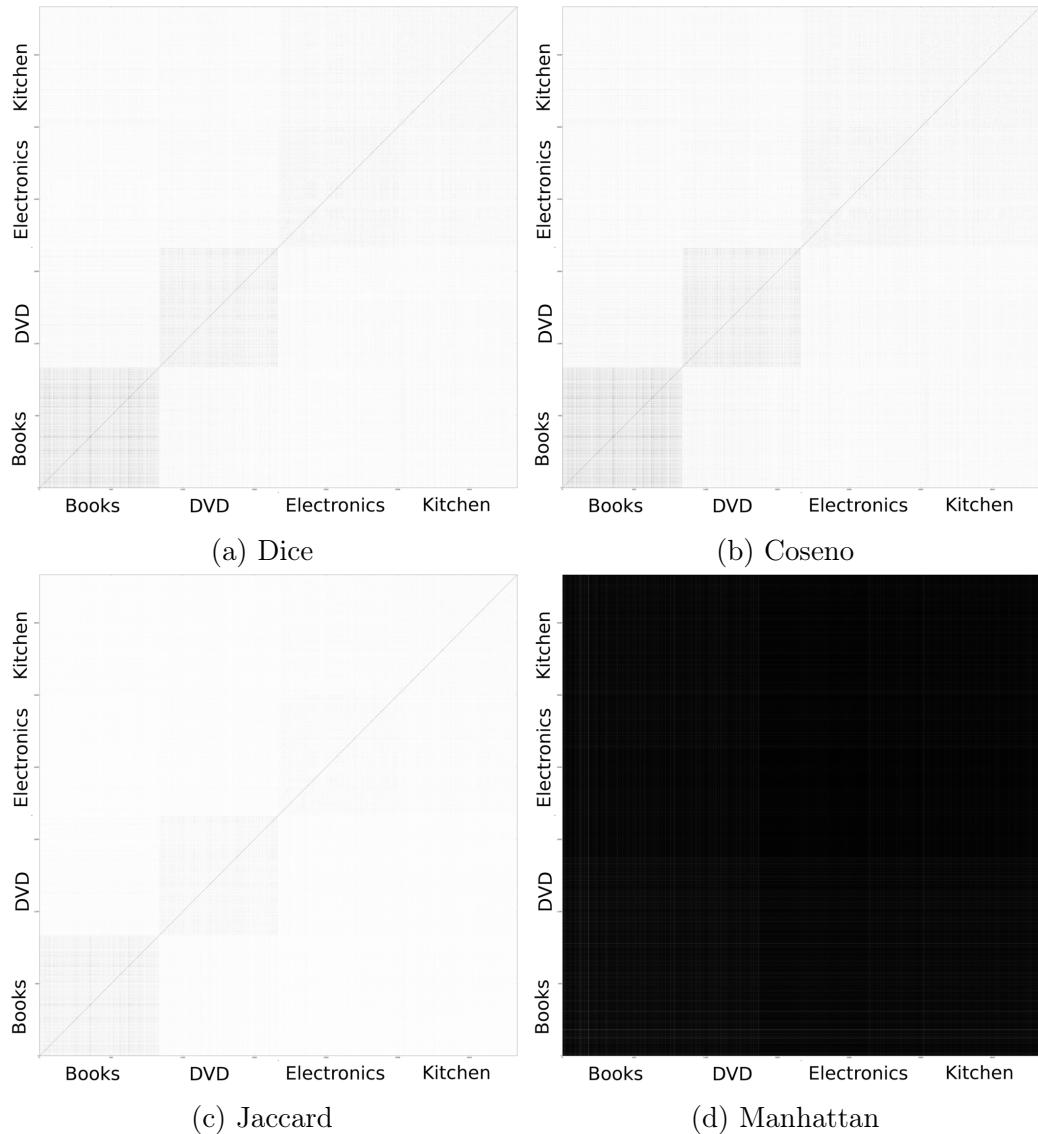


Figura 4.2: Métricas aplicadas al conjunto de 1600 documentos.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.345867	0.42206	0.693498	1
electronics	0.317042	0.41283	1	0.693498
dvd	0.370053	1	0.41283	0.42206
books	1	0.370053	0.317042	0.345867

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.146972	0.229722	0.5303	1
electronics	0.129527	0.219013	1	0.5303
dvd	0.214326	1	0.219013	0.229722
books	1	0.214326	0.129527	0.146972

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	5.96046e-...	0.146517	0.532455	1
electronics	0.0128672	0.163673	1	0.532455
dvd	0.0759009	1	0.163673	0.146517
books	1	0.0759009	0.0128672	5.96046e-...

(d) Manhattan

Figura 4.3: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento).

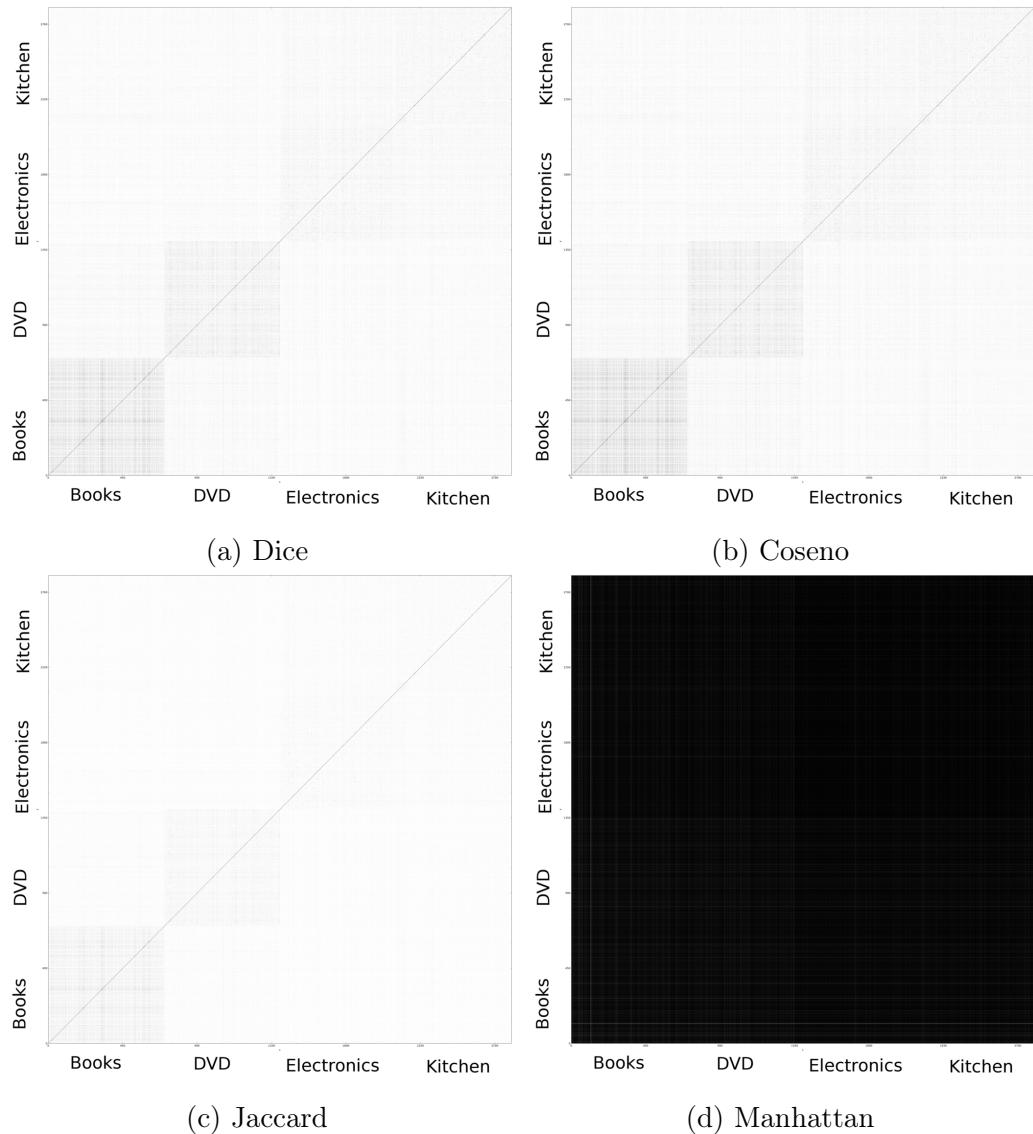


Figura 4.4: Métricas aplicadas al subconjunto de entrenamiento.

	books	dvd	electronics	kitchen
kitchen	0.218417	0.368144	0.595709	1
electronics	0.130192	0.325317	1	0.595709
dvd	0.178653	1	0.325317	0.368144
books	1	0.178653	0.130192	0.218417

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.296939	0.372652	0.653267	1
electronics	0.248067	0.338133	1	0.653267
dvd	0.271683	1	0.338133	0.372652
books	1	0.271683	0.248067	0.296939

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.122597	0.225598	0.424206	1
electronics	0.0696286	0.194256	1	0.424206
dvd	0.0980885	1	0.194256	0.225598
books	1	0.0980885	0.0696286	0.122597

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.46163	0.553622	1
electronics	0.0880362	0.647664	1	0.553622
dvd	0.110915	1	0.647664	0.46163
books	1	0.110915	0.0880362	0

(d) Manhattan

Figura 4.5: Similitud entre clases resultante de cada métrica (subconjunto de prueba).

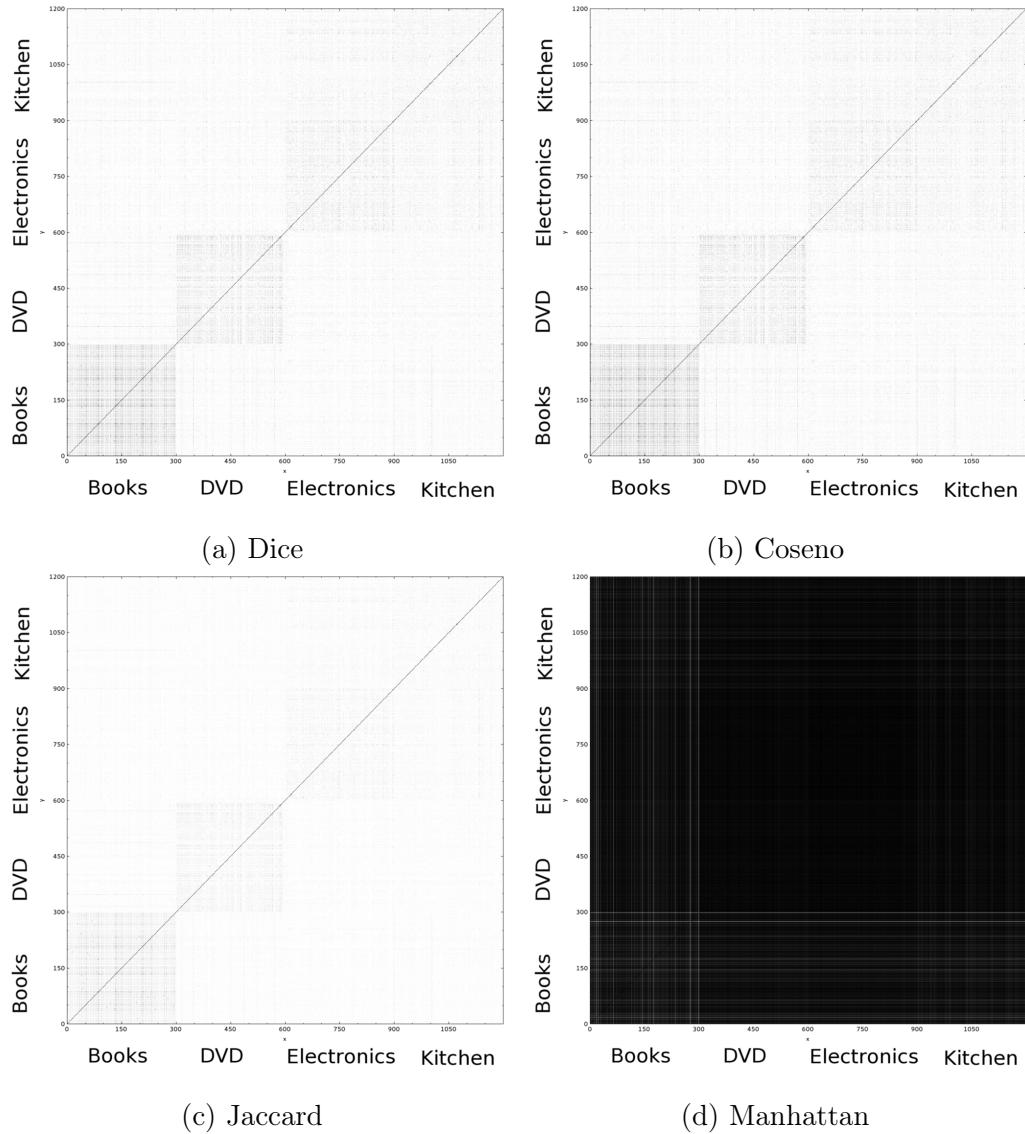


Figura 4.6: Métricas aplicadas al subconjunto de prueba.

	books	dvd	electronics	kitchen
kitchen	0.0681353	0.142873	0.420928	1
electronics	0.0500725	0.153189	1	0.420928
dvd	0.110637	1	0.153189	0.142873
books	1	0.110637	0.0500725	0.0681353

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.139786	0.176653	0.430826	1
electronics	0.124711	0.218162	1	0.430826
dvd	0.13653	1	0.218162	0.176653
books	1	0.13653	0.124711	0.139786

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0352692	0.0769325	0.266567	1
electronics	0.0256792	0.0829481	1	0.266567
dvd	0.0585581	1	0.0829481	0.0769325
books	1	0.0585581	0.0256792	0.0352692

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	5.96046e-...	0.276365	0.559805	1
electronics	0.0711843	0.359656	1	0.559805
dvd	0.0314453	1	0.359656	0.276365
books	1	0.0314453	0.0711843	5.96046e-...

(d) Manhattan

Figura 4.7: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 2.

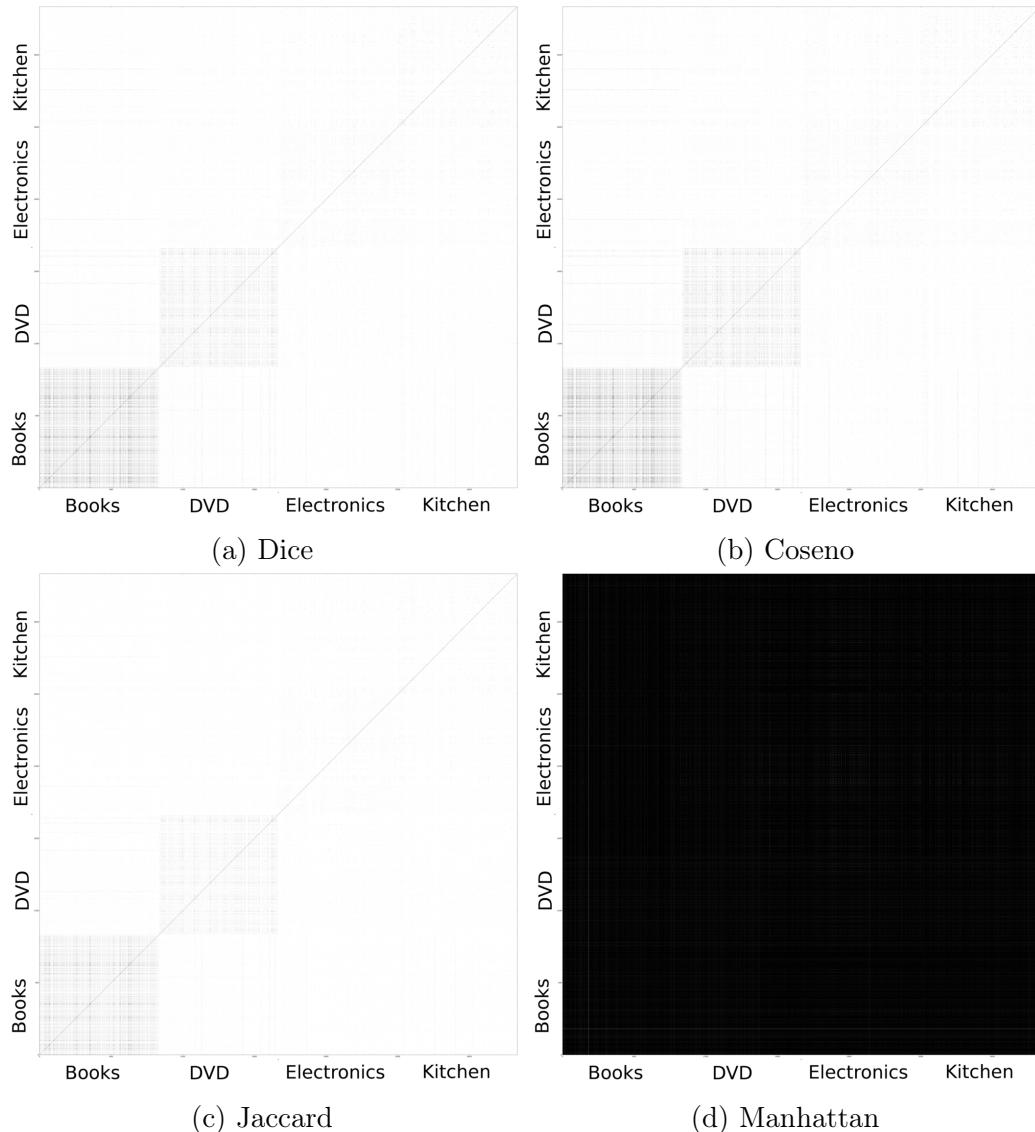


Figura 4.8: Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.147383	0.177523	0.403614	1
electronics	0.129357	0.209045	1	0.403614
dvd	0.145791	1	0.209045	0.177523
books	1	0.145791	0.129357	0.147383

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0400242	0.0706805	0.252374	1
electronics	0.0333284	0.0810087	1	0.252374
dvd	0.0704988	1	0.0810087	0.0706805
books	1	0.0704988	0.0333284	0.0400242

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0496819	0.195597	0.53297	1
electronics	0.0709886	0.229728	1	0.53297
dvd	0	1	0.229728	0.195597
books	1	0	0.0709886	0.0496819

(d) Manhattan

Figura 4.9: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 2.

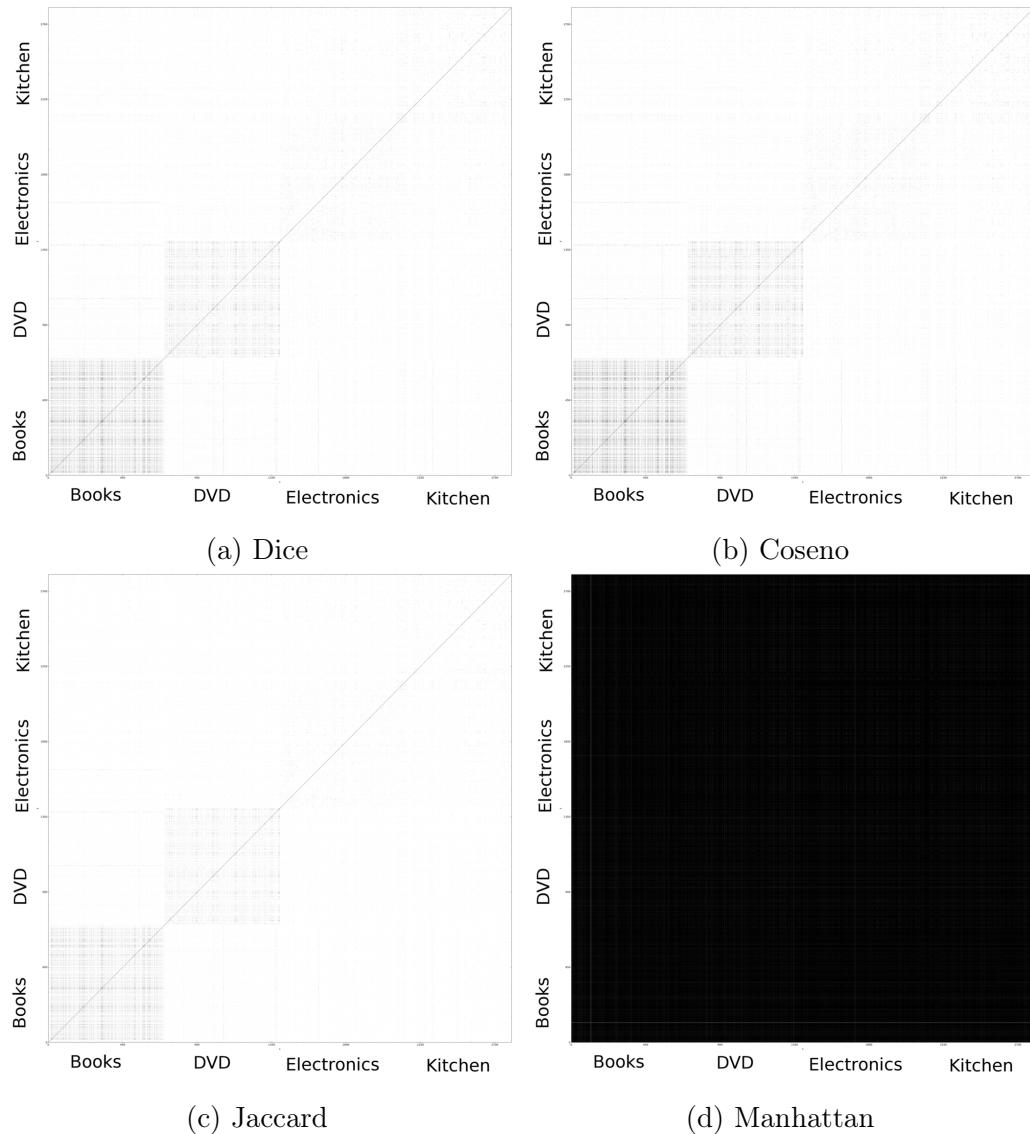


Figura 4.10: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.0434153	0.0942482	0.224436	1
electronics	0.0178012	0.0998605	1	0.224436
dvd	0.041529	1	0.0998605	0.0942482
books	1	0.041529	0.0178012	0.0434153

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0945472	0.0942781	0.320371	1
electronics	0.0904164	0.140027	1	0.320371
dvd	0.0924931	1	0.140027	0.0942781
books	1	0.0924931	0.0904164	0.0945472

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0221893	0.0494546	0.126403	1
electronics	0.008980...	0.0525543	1	0.126403
dvd	0.0212048	1	0.0525543	0.0494546
books	1	0.0212048	0.008980...	0.0221893

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.570443	0.63026	1
electronics	0.200282	0.81689	1	0.63026
dvd	0.167206	1	0.81689	0.570443
books	1	0.167206	0.200282	0

(d) Manhattan

Figura 4.11: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 2.

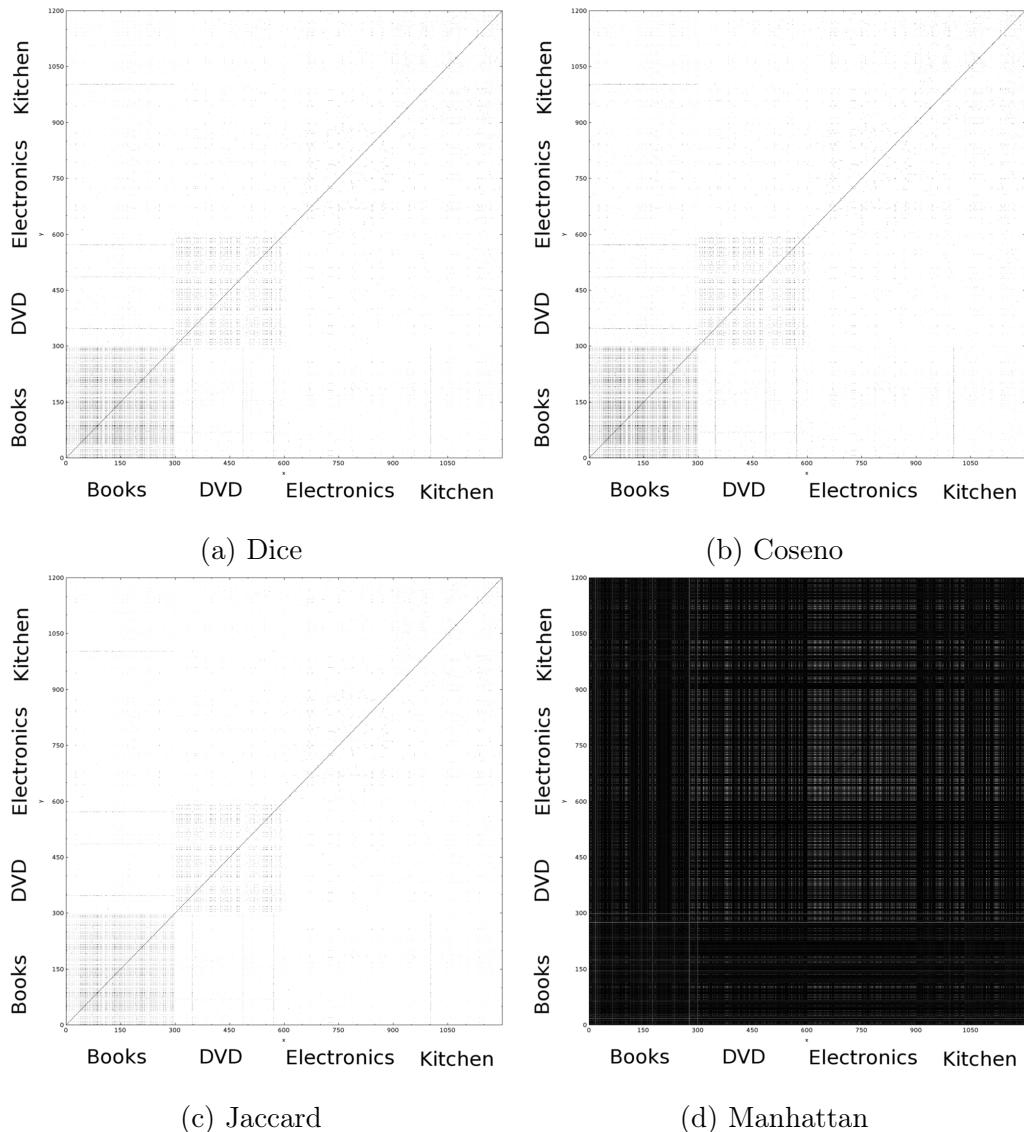


Figura 4.12: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.0234966	0.0536781	0.211011	1
electronics	0.0195469	0.07976	1	0.211011
dvd	0.0571288	1	0.07976	0.0536781
books	1	0.0571288	0.0195469	0.0234966

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0561667	0.0676669	0.223935	1
electronics	0.0646661	0.128455	1	0.223935
dvd	0.0769456	1	0.128455	0.0676669
books	1	0.0769456	0.0646661	0.0561667

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.011888	0.0275793	0.11795	1
electronics	0.0098699	0.0415365	1	0.11795
dvd	0.0294043	1	0.0415365	0.0275793
books	1	0.0294043	0.0098699	0.011888

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0390278	0.324128	0.59683	1
electronics	0.135188	0.42212	1	0.59683
dvd	0	1	0.42212	0.324128
books	1	0	0.135188	0.0390278

(d) Manhattan

Figura 4.13: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 3.

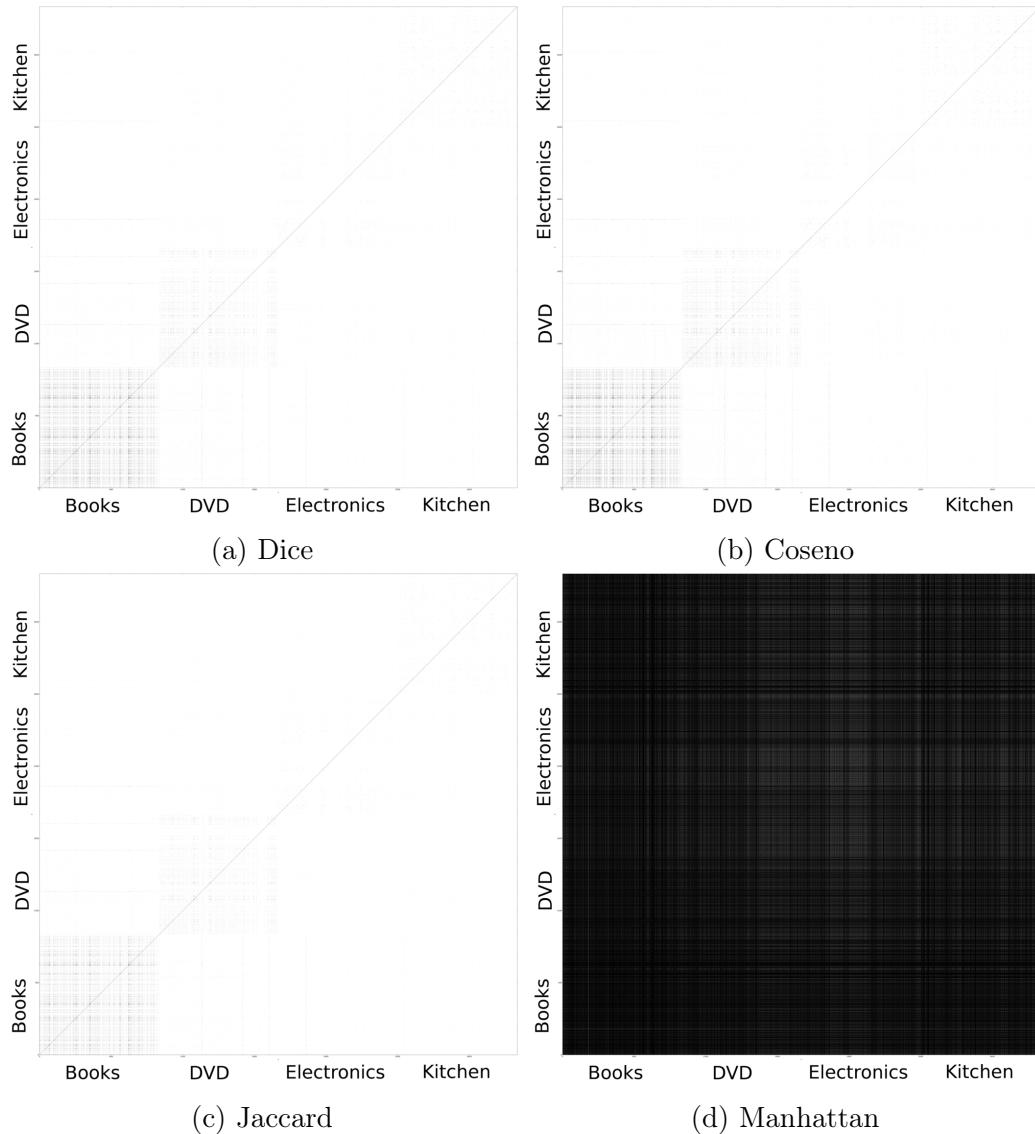


Figura 4.14: Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0605693	0.0629285	0.210102	1
electronics	0.071197	0.127156	1	0.210102
dvd	0.07975	1	0.127156	0.0629285
books	1	0.07975	0.071197	0.0605693

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0142889	0.0231868	0.115865	1
electronics	0.0146407	0.0426981	1	0.115865
dvd	0.035861	1	0.0426981	0.0231868
books	1	0.035861	0.0146407	0.0142889

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.128249	0.275308	0.590971	1
electronics	0.176642	0.3237	1	0.590971
dvd	0	1	0.3237	0.275308
books	1	0	0.176642	0.128249

(d) Manhattan

Figura 4.15: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 3.

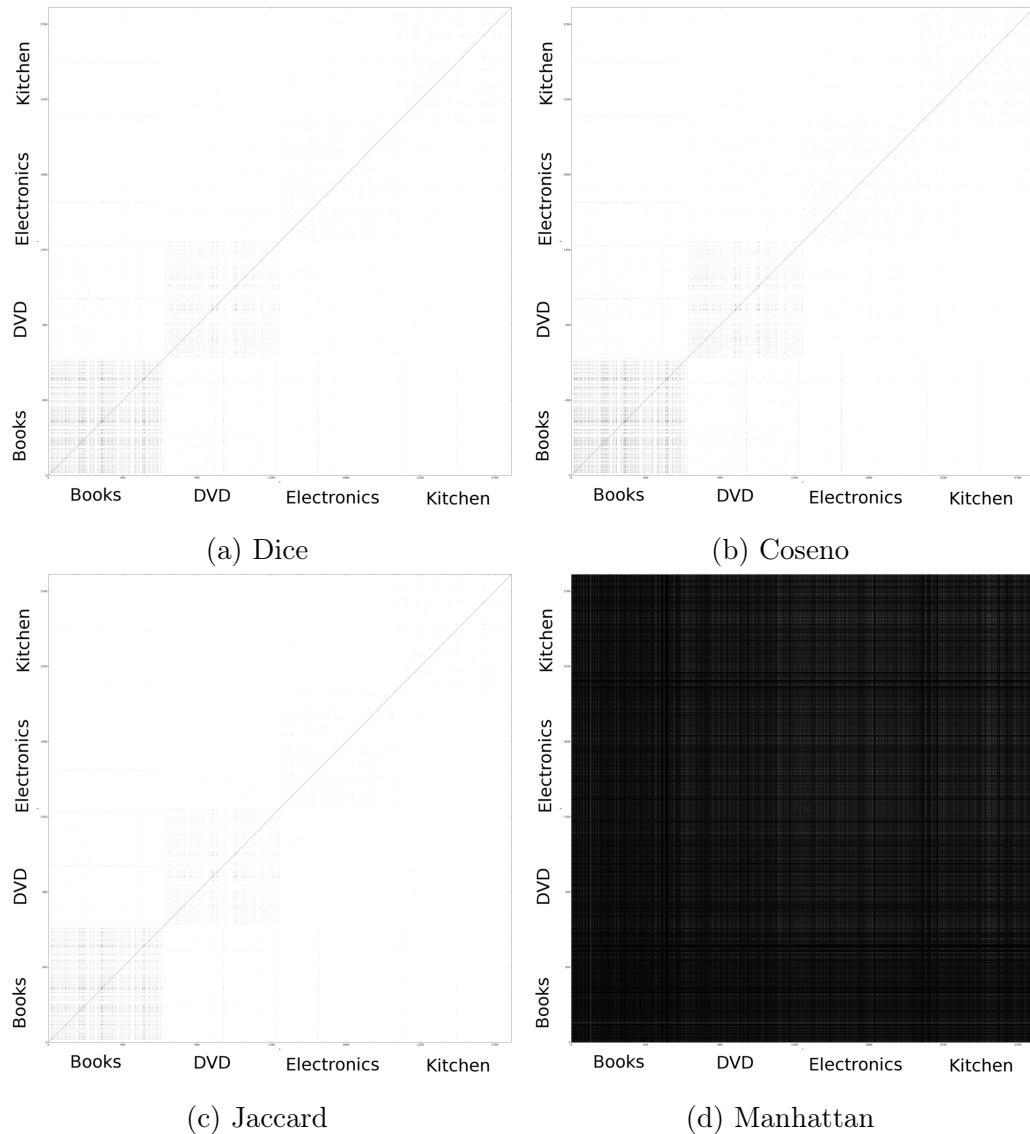


Figura 4.16: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.009383...	0.0337758	0.0203533	1
electronics	0.001668...	0.0381001	1	0.0203533
dvd	0.0171281	1	0.0381001	0.0337758
books	1	0.0171281	0.001668...	0.009383...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0244991	0.0366384	0.0560295	1
electronics	0.0223113	0.0726978	1	0.0560295
dvd	0.0658712	1	0.0726978	0.0366384
books	1	0.0658712	0.0223113	0.0244991

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004713...	0.017178	0.0102813	1
electronics	0.000834...	0.01942	1	0.0102813
dvd	0.008638	1	0.01942	0.017178
books	1	0.008638	0.000834...	0.004713...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.649064	0.696843	1
electronics	0.234423	0.911428	1	0.696843
dvd	0.200615	1	0.911428	0.649064
books	1	0.200615	0.234423	0

(d) Manhattan

Figura 4.17: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 3.

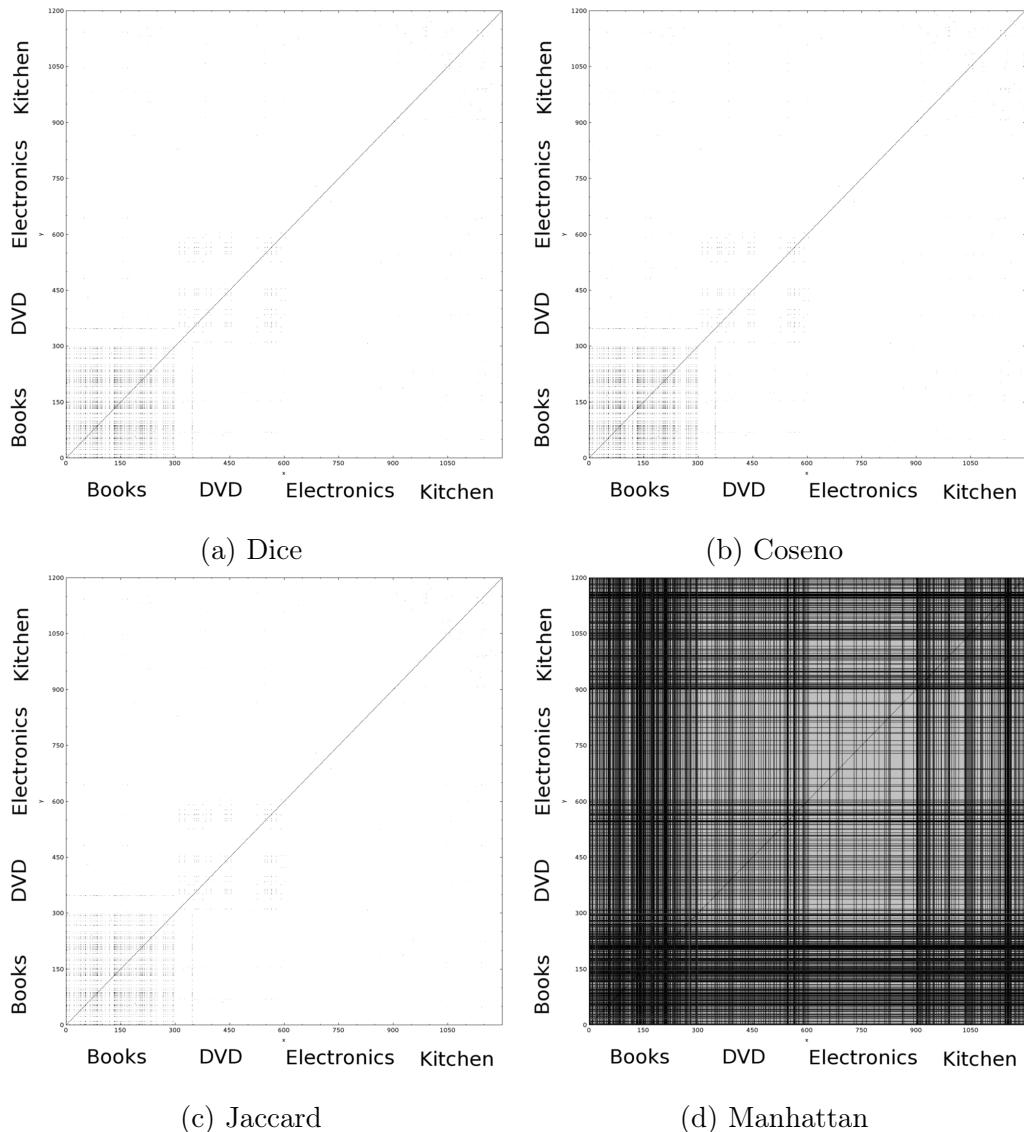


Figura 4.18: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.008259...	0.0279491	0.0945509	1
electronics	0.0069263	0.0370565	1	0.0945509
dvd	0.0375526	1	0.0370565	0.0279491
books	1	0.0375526	0.0069263	0.008259...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0201228	0.0354525	0.107024	1
electronics	0.0272603	0.0685419	1	0.107024
dvd	0.0509446	1	0.0685419	0.0354525
books	1	0.0509446	0.0272603	0.0201228

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004147...	0.0141726	0.0496213	1
electronics	0.003475...	0.018878	1	0.0496213
dvd	0.0191356	1	0.018878	0.0141726
books	1	0.0191356	0.003475...	0.004147...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0989162	0.402571	0.654802	1
electronics	0.182958	0.485338	1	0.654802
dvd	0	1	0.485338	0.402571
books	1	0	0.182958	0.0989162

(d) Manhattan

Figura 4.19: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 4.

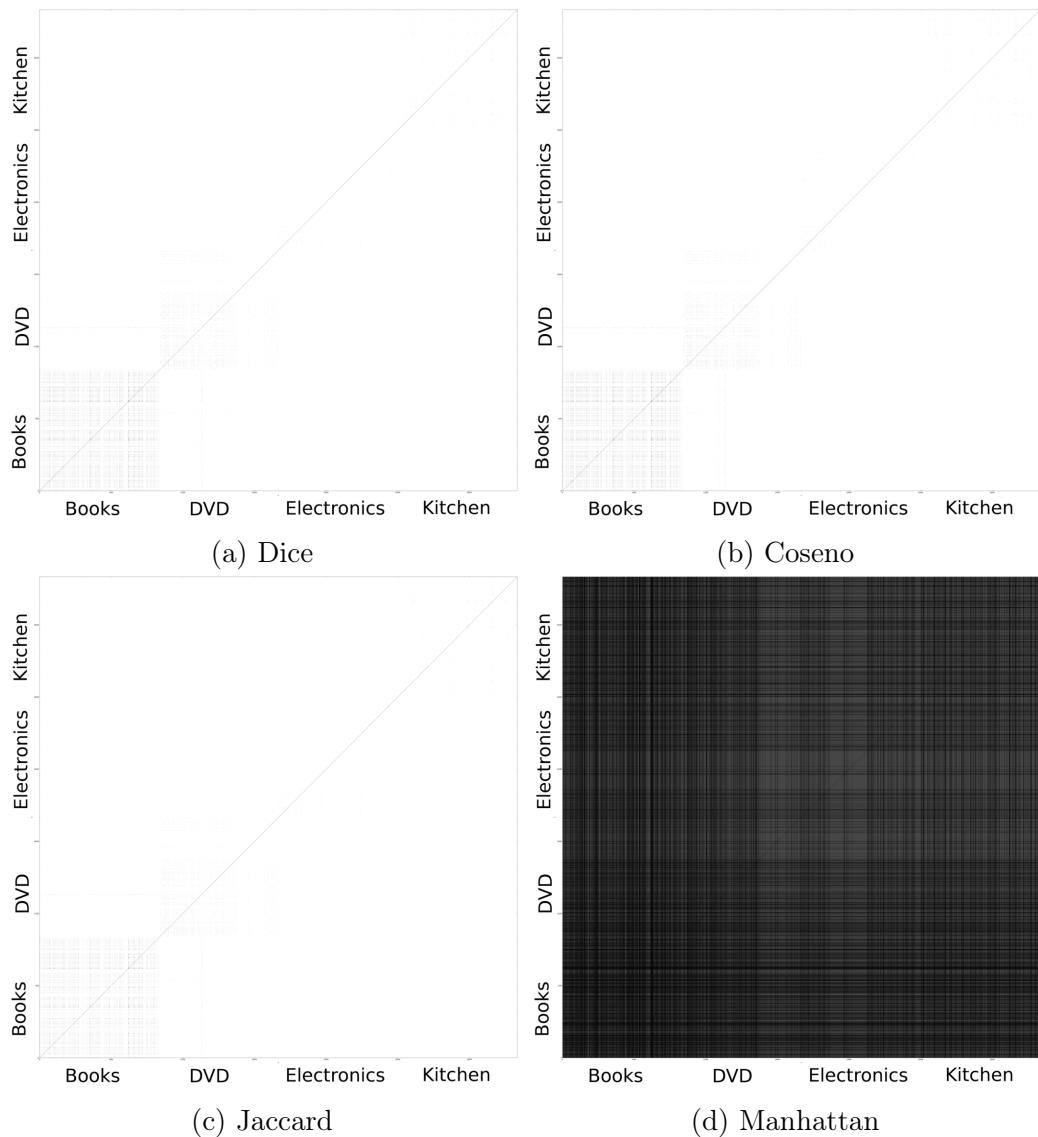


Figura 4.20: Métricas aplicadas al conjunto de 1600 documentos utilizando un umbral de frecuencia de 4.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0197683	0.0294726	0.0874552	1
electronics	0.0327482	0.0694405	1	0.0874552
dvd	0.0570256	1	0.0694405	0.0294726
books	1	0.0570256	0.0327482	0.0197683

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004647...	0.010522	0.0434208	1
electronics	0.005769...	0.0195536	1	0.0434208
dvd	0.0258171	1	0.0195536	0.010522
books	1	0.0258171	0.005769...	0.004647...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.17483	0.343778	0.640491	1
electronics	0.220489	0.390715	1	0.640491
dvd	0	1	0.390715	0.343778
books	1	0	0.220489	0.17483

(d) Manhattan

Figura 4.21: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 4.

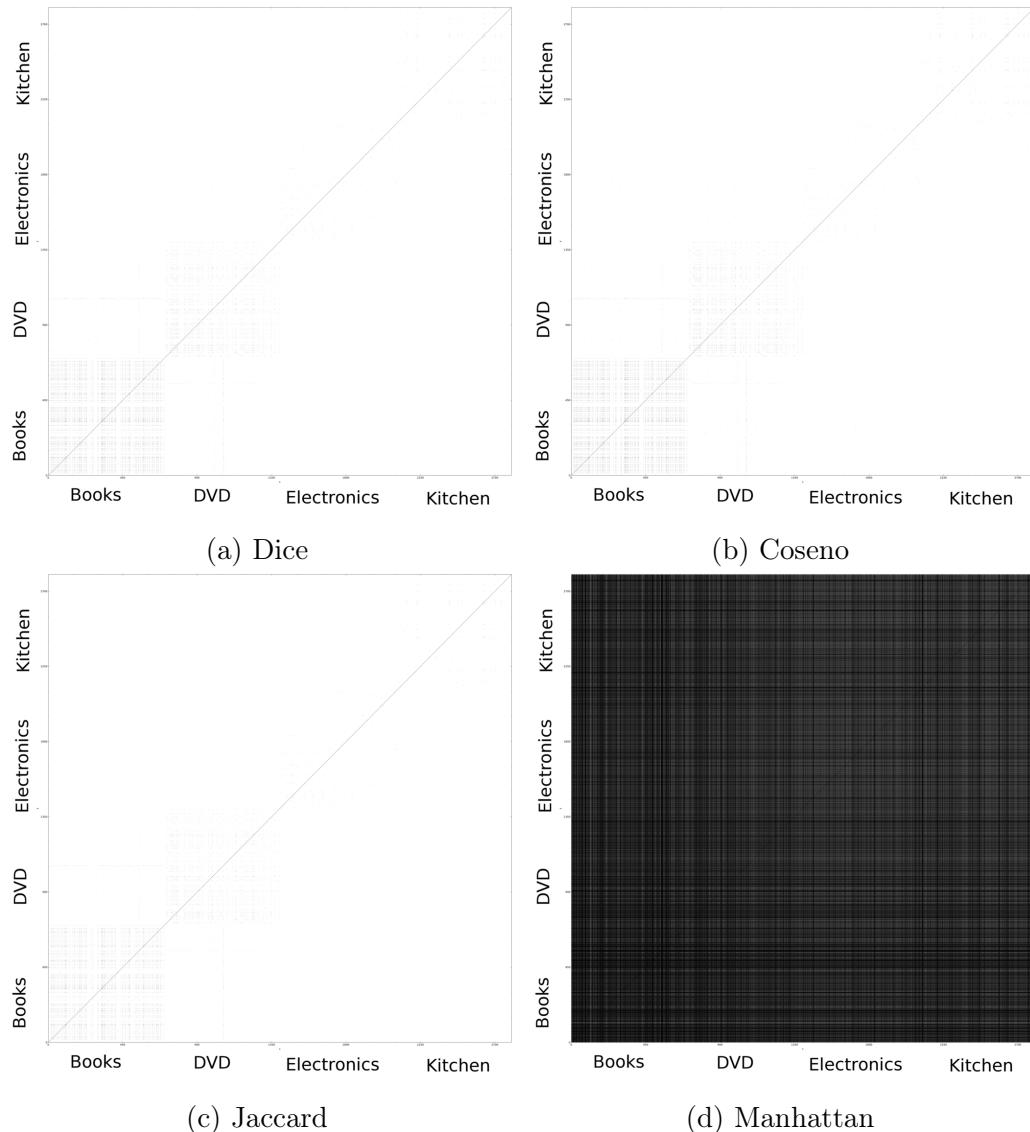


Figura 4.22: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 4.

	books	dvd	electronics	kitchen
kitchen	0.002917...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.002917...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.008157...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.008157...

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.001460...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.001460...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.730091	0.764425	1
electronics	0.215546	0.96185	1	0.764425
dvd	0.181211	1	0.96185	0.730091
books	1	0.181211	0.215546	0

(d) Manhattan

Figura 4.23: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 4.

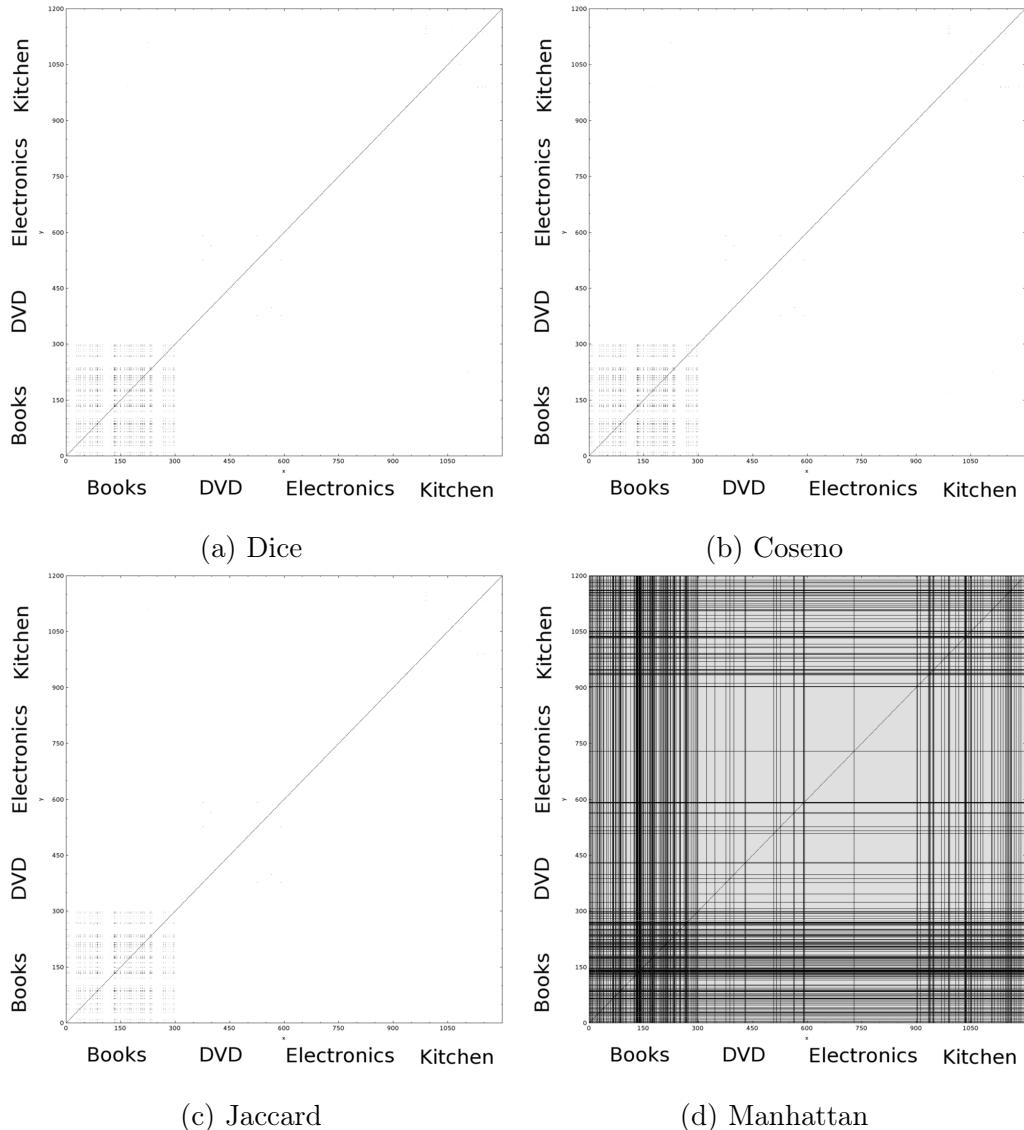


Figura 4.24: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 4.

#### 4.2.2. Gráficas de 4000 documentos

Las siguientes gráficas representan las pruebas realizadas al conjunto de 4000 documentos para los casos donde:

- Sólo se eliminan palabras de paro.
- Se aplica un umbral de frecuencia de 2.
- Se aplica un umbral de frecuencia de 3.
- Se aplica un umbral de frecuencia de 4.

Para cada caso, se muestran las gráficas del conjunto completo, y las gráficas de sus subconjuntos de prueba y entrenamiento.

	books	dvd	electronics	kitchen
kitchen	0.249855	0.402015	0.717055	1
electronics	0.204545	0.379918	1	0.717055
dvd	0.320256	1	0.379918	0.402015
books	1	0.320256	0.204545	0.249855

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.340418	0.427214	0.72559	1
electronics	0.311293	0.429679	1	0.72559
dvd	0.357125	1	0.429679	0.427214
books	1	0.357125	0.311293	0.340418

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.142763	0.251576	0.558913	1
electronics	0.113924	0.234506	1	0.558913
dvd	0.190657	1	0.234506	0.251576
books	1	0.190657	0.113924	0.142763

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.256555	0.562191	1
electronics	0.0317623	0.307405	1	0.562191
dvd	0.127063	1	0.307405	0.256555
books	1	0.127063	0.0317623	0

(d) Manhattan

Figura 4.25: Similitud entre clases resultante de cada métrica.

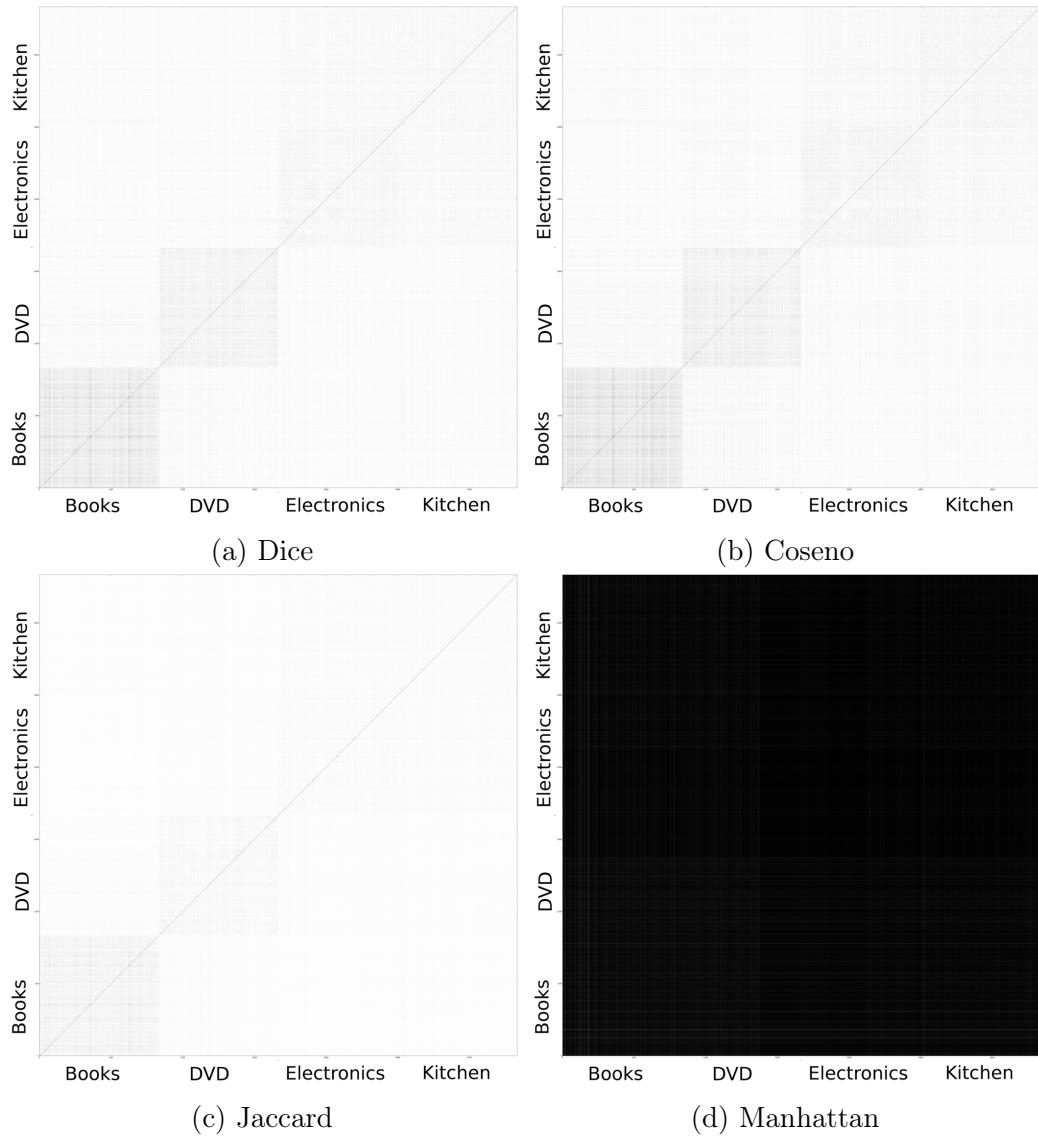


Figura 4.26: Métricas aplicadas al conjunto de 4000 documentos.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.345867	0.42206	0.693498	1
electronics	0.317042	0.41283	1	0.693498
dvd	0.370053	1	0.41283	0.42206
books	1	0.370053	0.317042	0.345867

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.146972	0.229722	0.5303	1
electronics	0.129527	0.219013	1	0.5303
dvd	0.214326	1	0.219013	0.229722
books	1	0.214326	0.129527	0.146972

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	5.96046e-...	0.146517	0.532455	1
electronics	0.0128672	0.163673	1	0.532455
dvd	0.0759009	1	0.163673	0.146517
books	1	0.0759009	0.0128672	5.96046e-...

(d) Manhattan

Figura 4.27: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento).

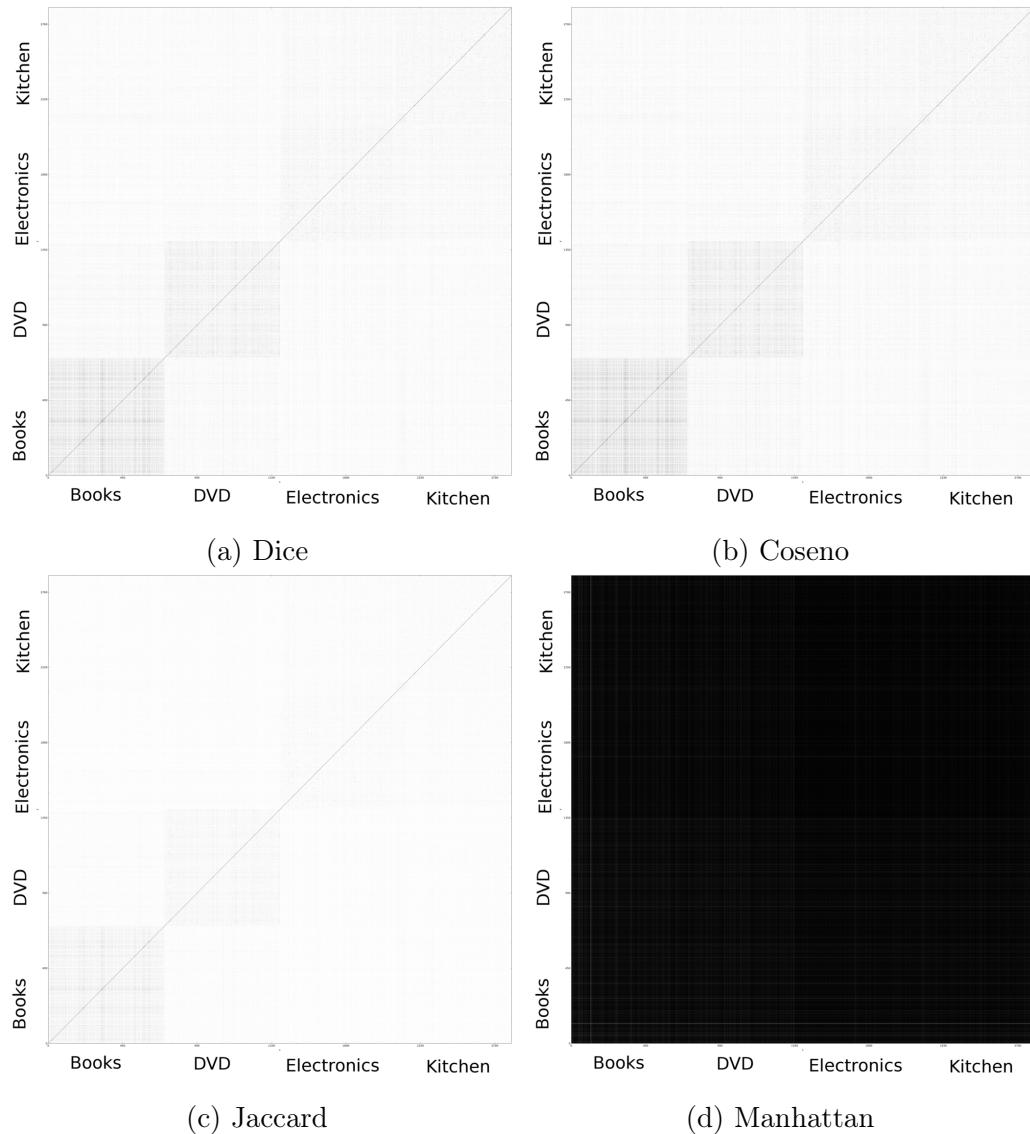


Figura 4.28: Métricas aplicadas al subconjunto de entrenamiento.

	books	dvd	electronics	kitchen
kitchen	0.218417	0.368144	0.595709	1
electronics	0.130192	0.325317	1	0.595709
dvd	0.178653	1	0.325317	0.368144
books	1	0.178653	0.130192	0.218417

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.296939	0.372652	0.653267	1
electronics	0.248067	0.338133	1	0.653267
dvd	0.271683	1	0.338133	0.372652
books	1	0.271683	0.248067	0.296939

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.122597	0.225598	0.424206	1
electronics	0.0696286	0.194256	1	0.424206
dvd	0.0980885	1	0.194256	0.225598
books	1	0.0980885	0.0696286	0.122597

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.46163	0.553622	1
electronics	0.0880362	0.647664	1	0.553622
dvd	0.110915	1	0.647664	0.46163
books	1	0.110915	0.0880362	0

(d) Manhattan

Figura 4.29: Similitud entre clases resultante de cada métrica (subconjunto de prueba).

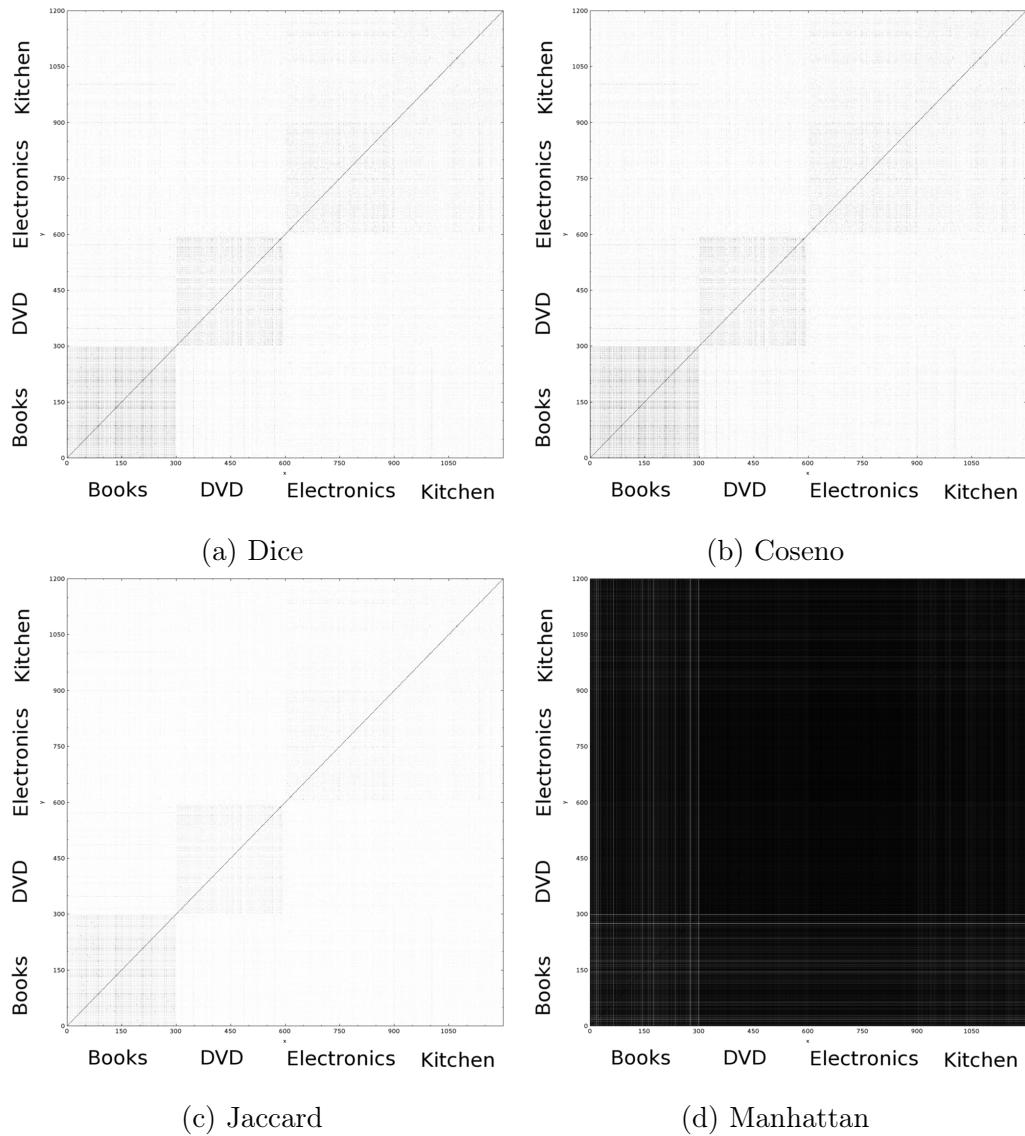


Figura 4.30: Métricas aplicadas al subconjunto de prueba.

	books	dvd	electronics	kitchen
kitchen	0.0681353	0.142873	0.420928	1
electronics	0.0500725	0.153189	1	0.420928
dvd	0.110637	1	0.153189	0.142873
books	1	0.110637	0.0500725	0.0681353

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.139786	0.176653	0.430826	1
electronics	0.124711	0.218162	1	0.430826
dvd	0.13653	1	0.218162	0.176653
books	1	0.13653	0.124711	0.139786

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0352692	0.0769325	0.266567	1
electronics	0.0256792	0.0829481	1	0.266567
dvd	0.0585581	1	0.0829481	0.0769325
books	1	0.0585581	0.0256792	0.0352692

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	5.96046e-...	0.276365	0.559805	1
electronics	0.0711843	0.359656	1	0.559805
dvd	0.0314453	1	0.359656	0.276365
books	1	0.0314453	0.0711843	5.96046e-...

(d) Manhattan

Figura 4.31: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 2.

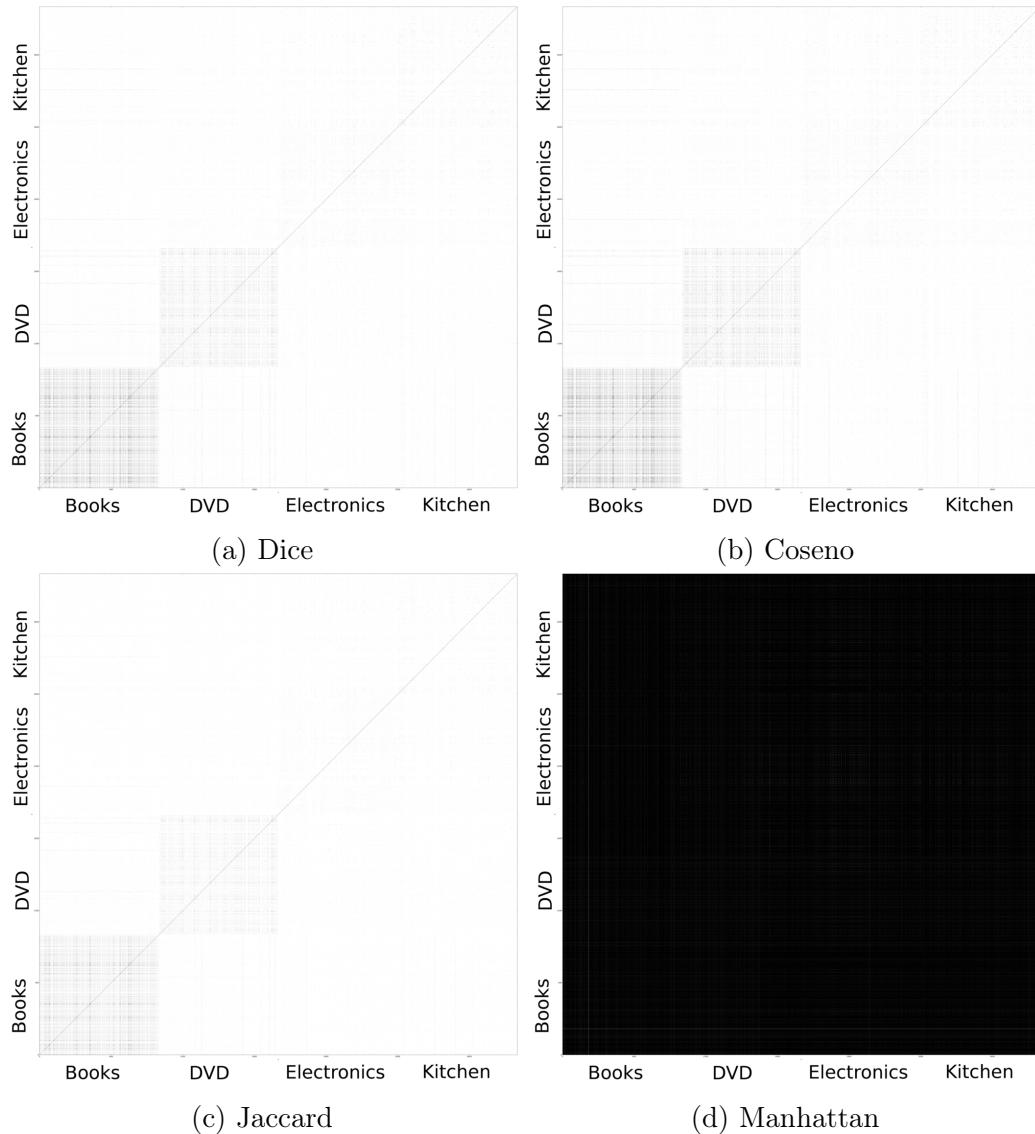


Figura 4.32: Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.147383	0.177523	0.403614	1
electronics	0.129357	0.209045	1	0.403614
dvd	0.145791	1	0.209045	0.177523
books	1	0.145791	0.129357	0.147383

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0400242	0.0706805	0.252374	1
electronics	0.0333284	0.0810087	1	0.252374
dvd	0.0704988	1	0.0810087	0.0706805
books	1	0.0704988	0.0333284	0.0400242

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0496819	0.195597	0.53297	1
electronics	0.0709886	0.229728	1	0.53297
dvd	0	1	0.229728	0.195597
books	1	0	0.0709886	0.0496819

(d) Manhattan

Figura 4.33: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 2.

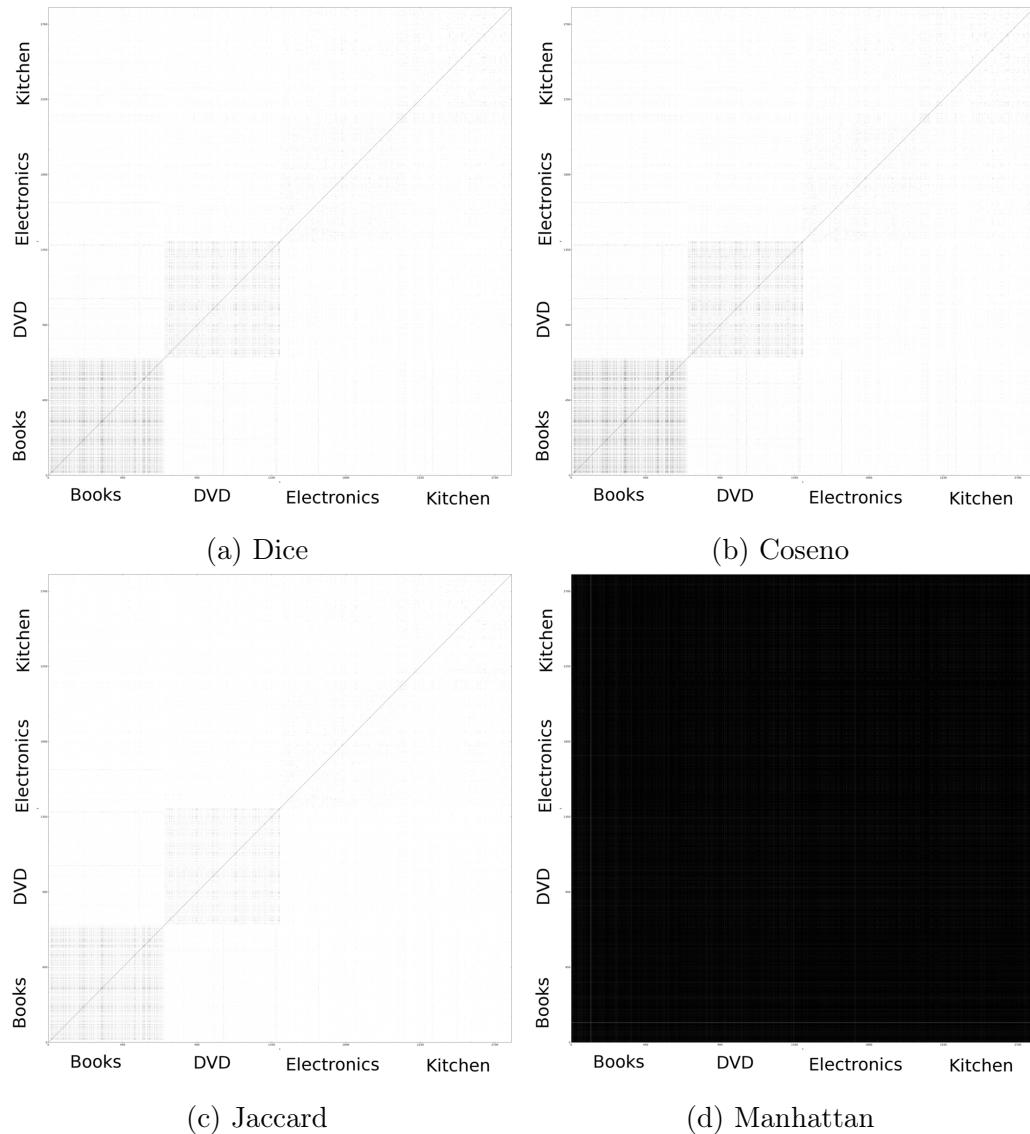


Figura 4.34: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.0434153	0.0942482	0.224436	1
electronics	0.0178012	0.0998605	1	0.224436
dvd	0.041529	1	0.0998605	0.0942482
books	1	0.041529	0.0178012	0.0434153

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0945472	0.0942781	0.320371	1
electronics	0.0904164	0.140027	1	0.320371
dvd	0.0924931	1	0.140027	0.0942781
books	1	0.0924931	0.0904164	0.0945472

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0221893	0.0494546	0.126403	1
electronics	0.008980...	0.0525543	1	0.126403
dvd	0.0212048	1	0.0525543	0.0494546
books	1	0.0212048	0.008980...	0.0221893

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.570443	0.63026	1
electronics	0.200282	0.81689	1	0.63026
dvd	0.167206	1	0.81689	0.570443
books	1	0.167206	0.200282	0

(d) Manhattan

Figura 4.35: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 2.

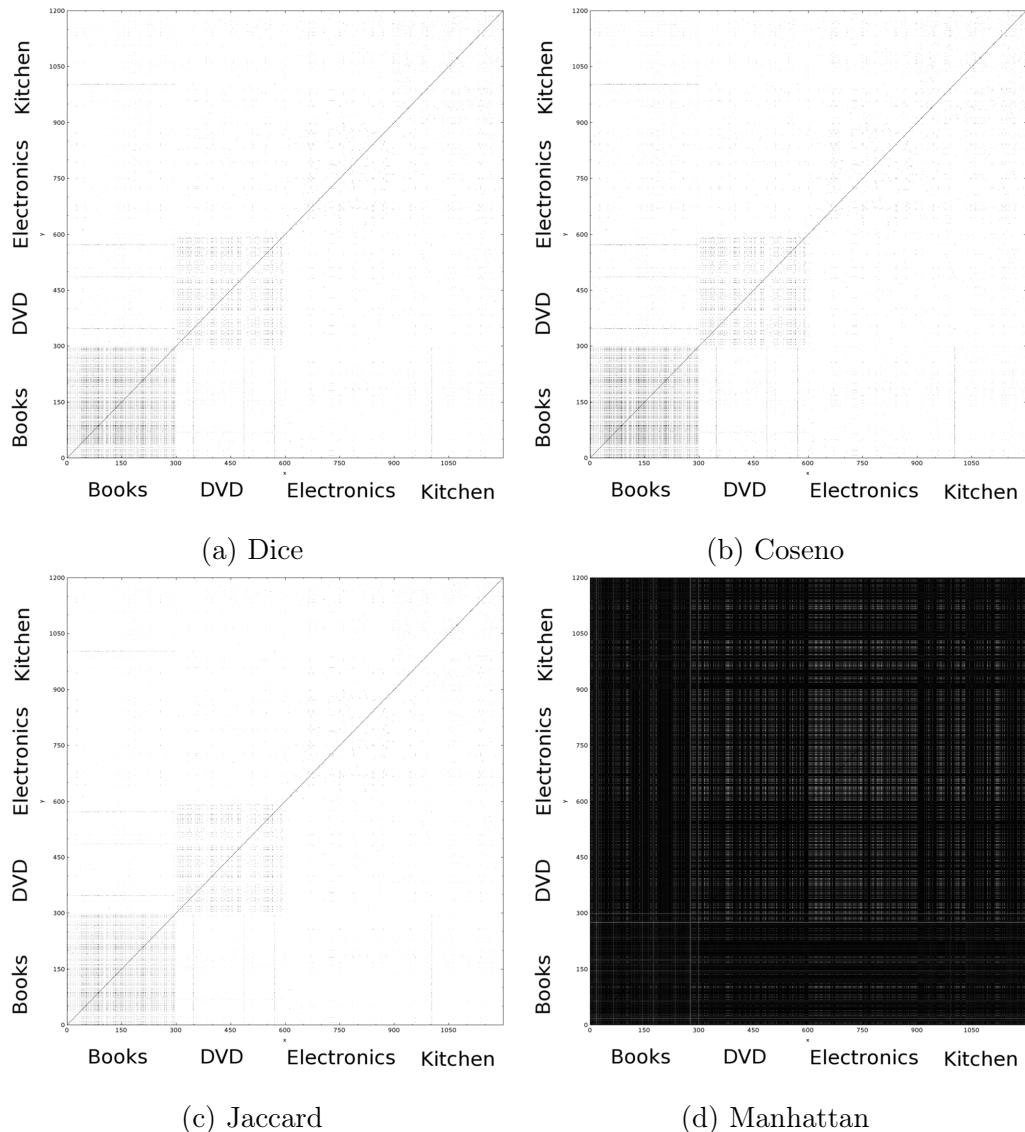


Figura 4.36: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 2.

	books	dvd	electronics	kitchen
kitchen	0.0234966	0.0536781	0.211011	1
electronics	0.0195469	0.07976	1	0.211011
dvd	0.0571288	1	0.07976	0.0536781
books	1	0.0571288	0.0195469	0.0234966

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0561667	0.0676669	0.223935	1
electronics	0.0646661	0.128455	1	0.223935
dvd	0.0769456	1	0.128455	0.0676669
books	1	0.0769456	0.0646661	0.0561667

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.011888	0.0275793	0.11795	1
electronics	0.0098699	0.0415365	1	0.11795
dvd	0.0294043	1	0.0415365	0.0275793
books	1	0.0294043	0.0098699	0.011888

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0390278	0.324128	0.59683	1
electronics	0.135188	0.42212	1	0.59683
dvd	0	1	0.42212	0.324128
books	1	0	0.135188	0.0390278

(d) Manhattan

Figura 4.37: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 3.

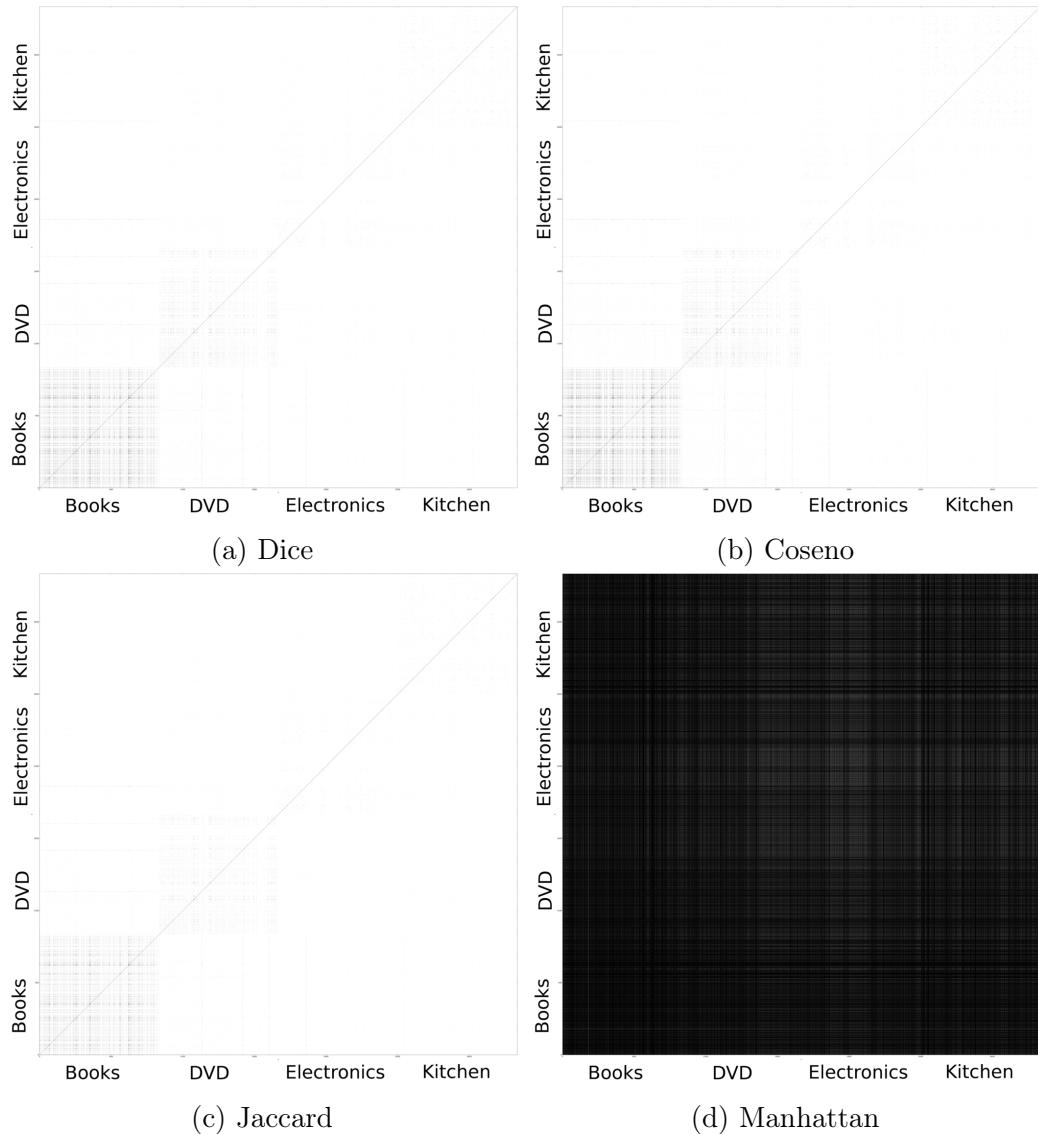


Figura 4.38: Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0605693	0.0629285	0.210102	1
electronics	0.071197	0.127156	1	0.210102
dvd	0.07975	1	0.127156	0.0629285
books	1	0.07975	0.071197	0.0605693

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.0142889	0.0231868	0.115865	1
electronics	0.0146407	0.0426981	1	0.115865
dvd	0.035861	1	0.0426981	0.0231868
books	1	0.035861	0.0146407	0.0142889

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.128249	0.275308	0.590971	1
electronics	0.176642	0.3237	1	0.590971
dvd	0	1	0.3237	0.275308
books	1	0	0.176642	0.128249

(d) Manhattan

Figura 4.39: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 3.

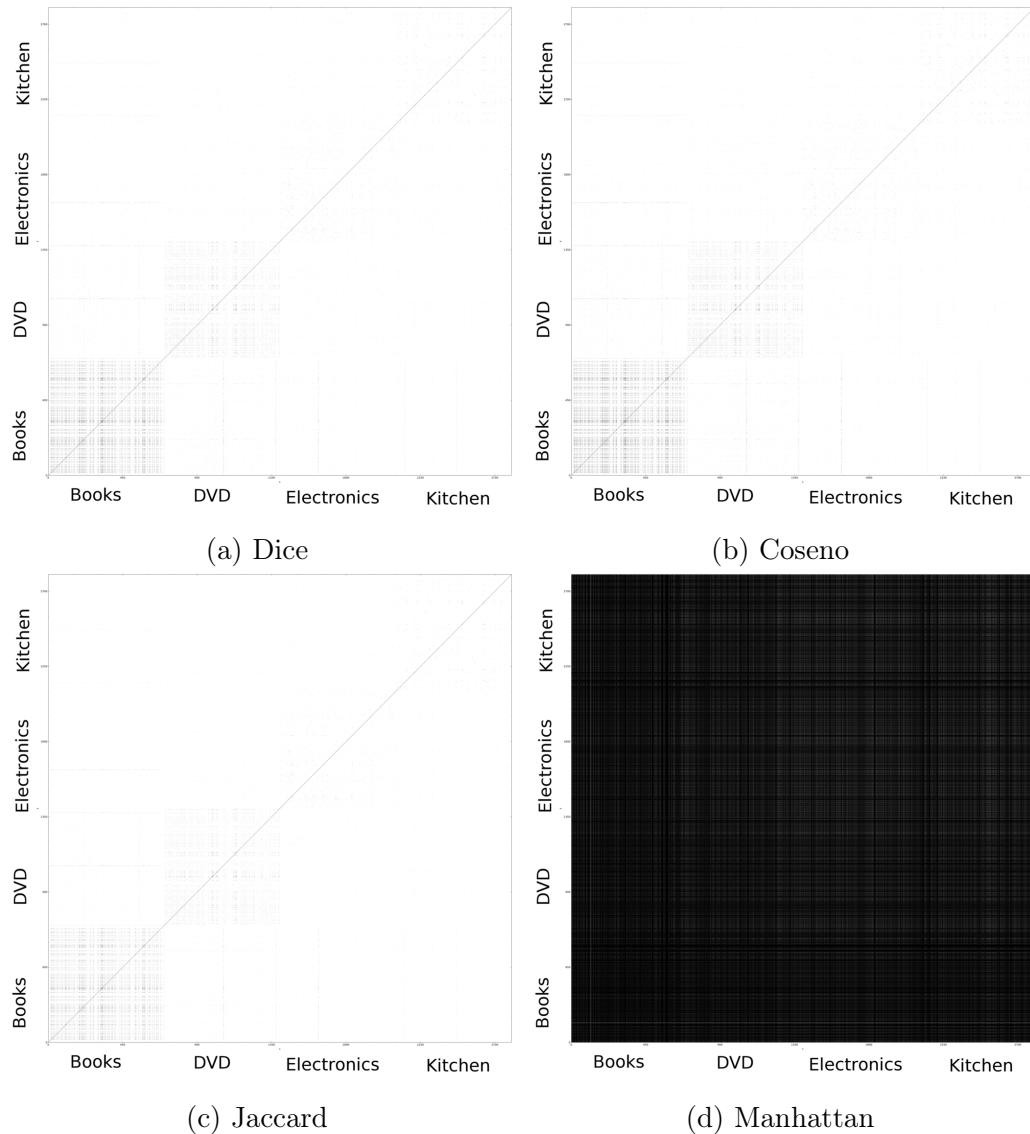


Figura 4.40: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.009383...	0.0337758	0.0203533	1
electronics	0.001668...	0.0381001	1	0.0203533
dvd	0.0171281	1	0.0381001	0.0337758
books	1	0.0171281	0.001668...	0.009383...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0244991	0.0366384	0.0560295	1
electronics	0.0223113	0.0726978	1	0.0560295
dvd	0.0658712	1	0.0726978	0.0366384
books	1	0.0658712	0.0223113	0.0244991

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004713...	0.017178	0.0102813	1
electronics	0.000834...	0.01942	1	0.0102813
dvd	0.008638	1	0.01942	0.017178
books	1	0.008638	0.000834...	0.004713...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.649064	0.696843	1
electronics	0.234423	0.911428	1	0.696843
dvd	0.200615	1	0.911428	0.649064
books	1	0.200615	0.234423	0

(d) Manhattan

Figura 4.41: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 3.

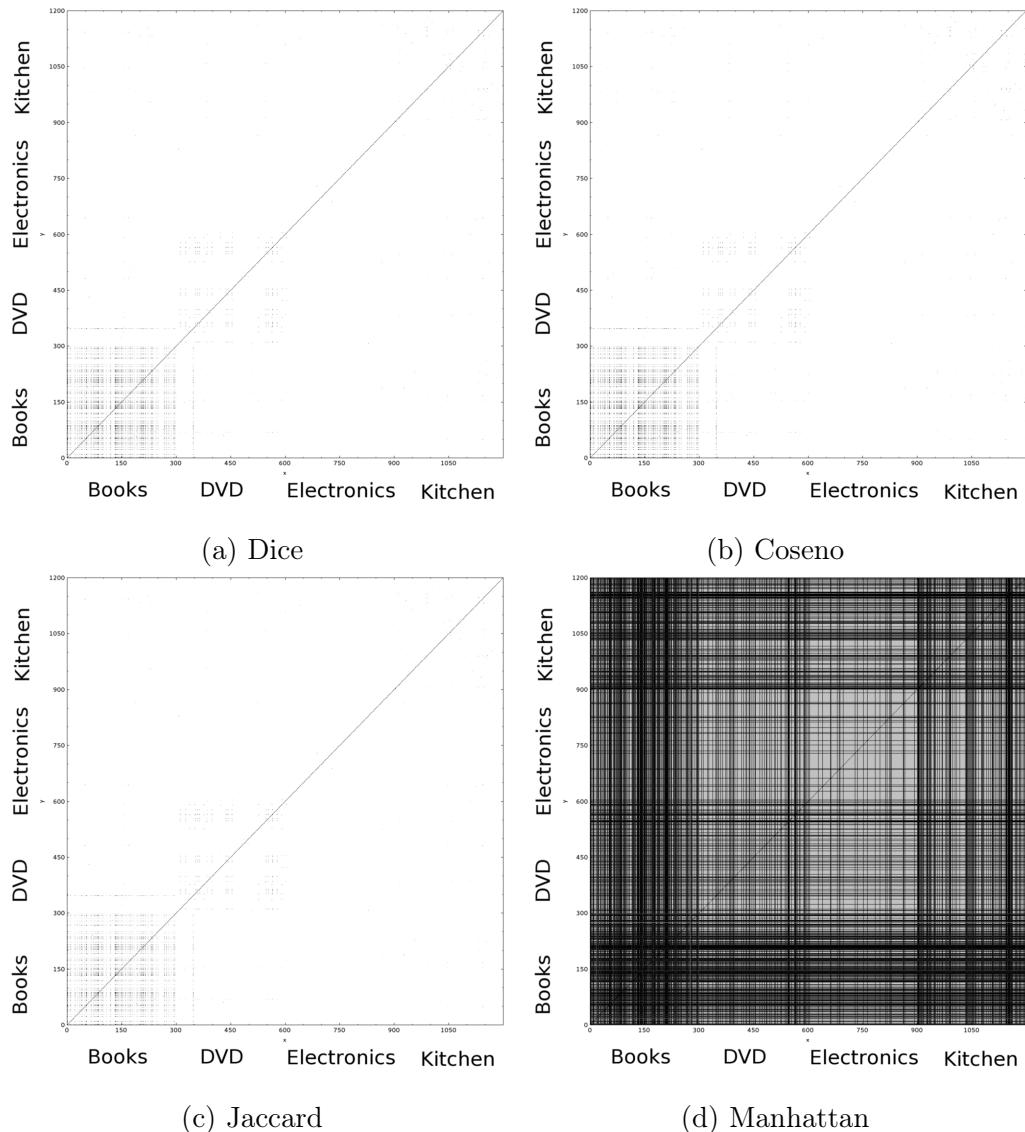


Figura 4.42: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 3.

	books	dvd	electronics	kitchen
kitchen	0.008259...	0.0279491	0.0945509	1
electronics	0.0069263	0.0370565	1	0.0945509
dvd	0.0375526	1	0.0370565	0.0279491
books	1	0.0375526	0.0069263	0.008259...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0201228	0.0354525	0.107024	1
electronics	0.0272603	0.0685419	1	0.107024
dvd	0.0509446	1	0.0685419	0.0354525
books	1	0.0509446	0.0272603	0.0201228

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004147...	0.0141726	0.0496213	1
electronics	0.003475...	0.018878	1	0.0496213
dvd	0.0191356	1	0.018878	0.0141726
books	1	0.0191356	0.003475...	0.004147...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.0989162	0.402571	0.654802	1
electronics	0.182958	0.485338	1	0.654802
dvd	0	1	0.485338	0.402571
books	1	0	0.182958	0.0989162

(d) Manhattan

Figura 4.43: Similitud entre clases resultante de cada métrica utilizando un umbral de frecuencia de 4.

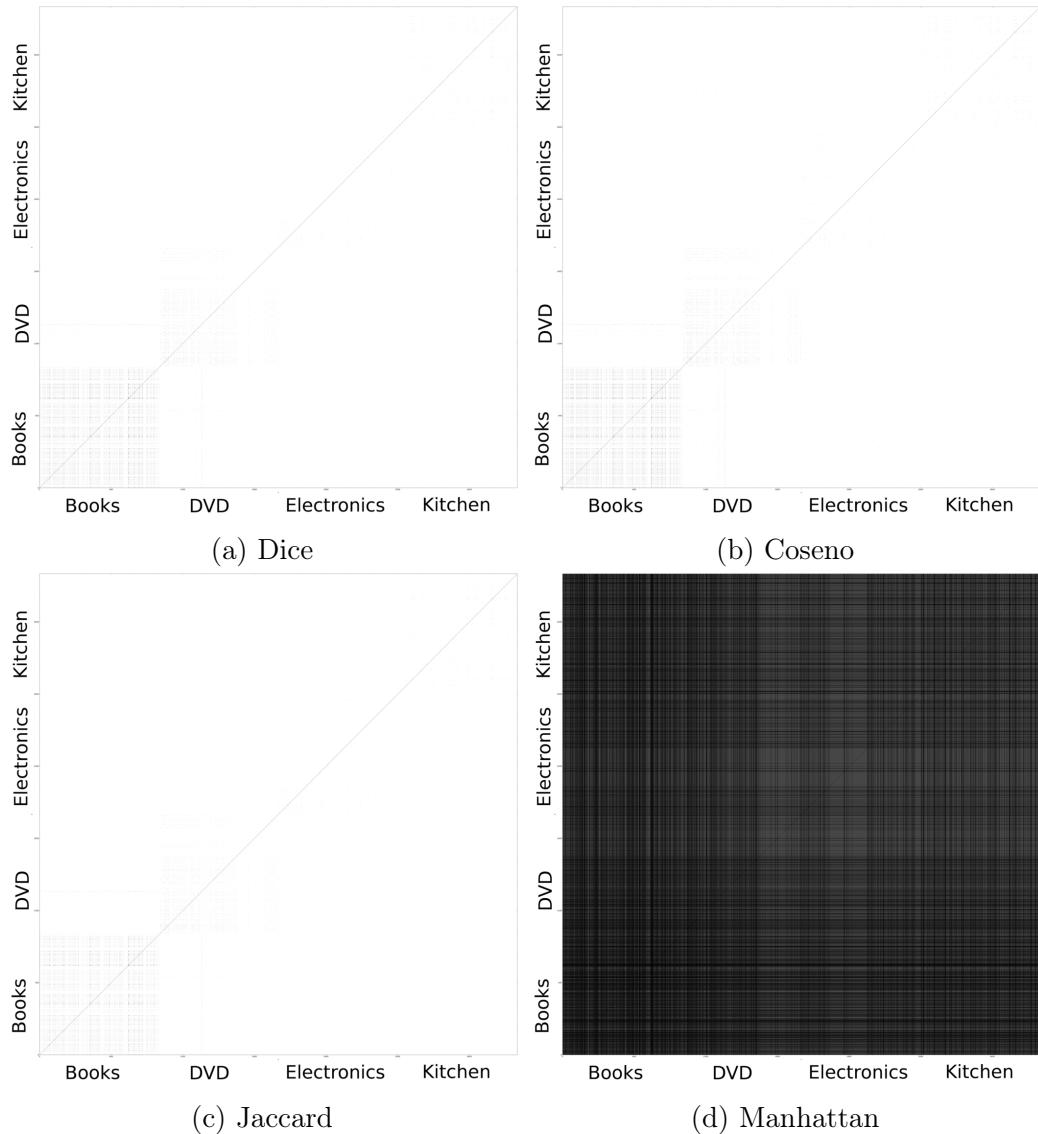


Figura 4.44: Métricas aplicadas al conjunto de 4000 documentos utilizando un umbral de frecuencia de 4.

	books	dvd	electronics	kitchen
kitchen	0.256279	0.373616	0.693067	1
electronics	0.229347	0.359328	1	0.693067
dvd	0.352995	1	0.359328	0.373616
books	1	0.352995	0.229347	0.256279

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.0197683	0.0294726	0.0874552	1
electronics	0.0327482	0.0694405	1	0.0874552
dvd	0.0570256	1	0.0694405	0.0294726
books	1	0.0570256	0.0327482	0.0197683

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.004647...	0.010522	0.0434208	1
electronics	0.005769...	0.0195536	1	0.0434208
dvd	0.0258171	1	0.0195536	0.010522
books	1	0.0258171	0.005769...	0.004647...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0.17483	0.343778	0.640491	1
electronics	0.220489	0.390715	1	0.640491
dvd	0	1	0.390715	0.343778
books	1	0	0.220489	0.17483

(d) Manhattan

Figura 4.45: Similitud entre clases resultante de cada métrica (subconjunto de entrenamiento) utilizando un umbral de frecuencia de 4.

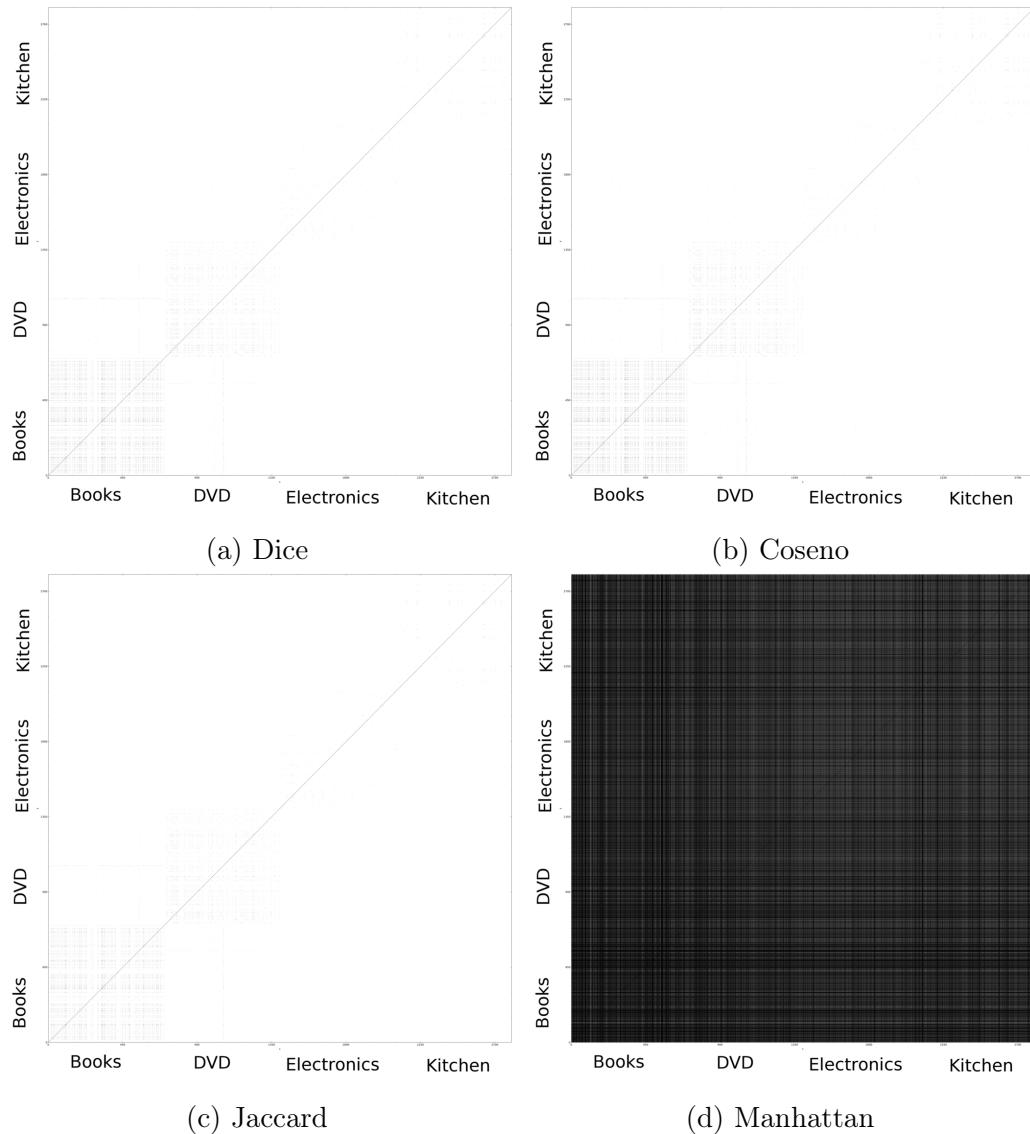


Figura 4.46: Métricas aplicadas al subconjunto de entrenamiento utilizando un umbral de frecuencia de 4.

	books	dvd	electronics	kitchen
kitchen	0.002917...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.002917...

(a) Dice

	books	dvd	electronics	kitchen
kitchen	0.008157...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.008157...

(b) Coseno

	books	dvd	electronics	kitchen
kitchen	0.001460...	0	0	1
electronics	0	0	1	0
dvd	0	1	0	0
books	1	0	0	0.001460...

(c) Jaccard

	books	dvd	electronics	kitchen
kitchen	0	0.730091	0.764425	1
electronics	0.215546	0.96185	1	0.764425
dvd	0.181211	1	0.96185	0.730091
books	1	0.181211	0.215546	0

(d) Manhattan

Figura 4.47: Similitud entre clases resultante de cada métrica (subconjunto de prueba) utilizando un umbral de frecuencia de 4.

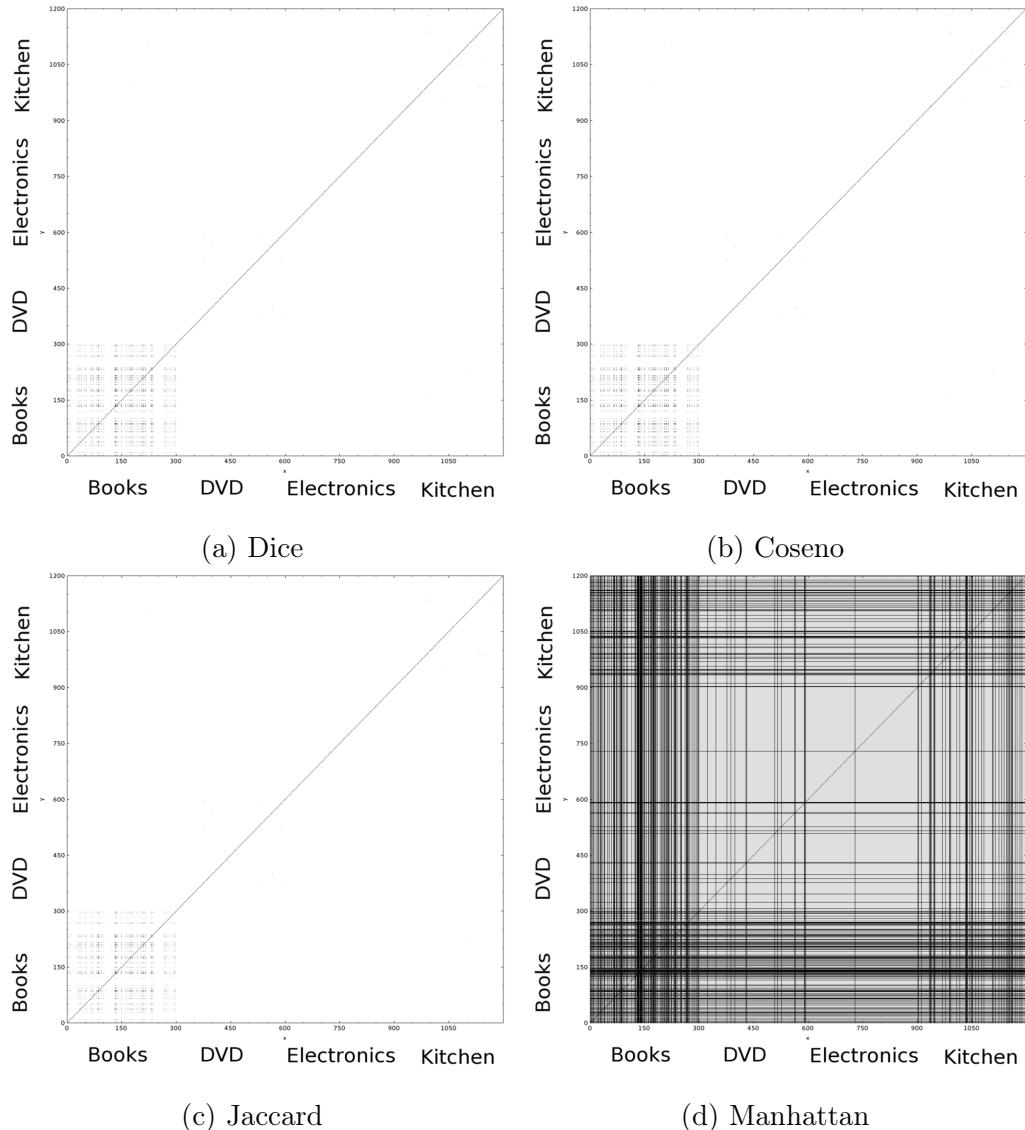


Figura 4.48: Métricas aplicadas al subconjunto de prueba utilizando un umbral de frecuencia de 4.

Como se puede apreciar de las figuras, entre mayor es el valor del umbral de la frecuencia, mayor es la eliminación de palabras en los documentos, llegando a un punto en el cual comienza a existir pérdida de información; por lo que se espera que el clasificador actúe mejor cuando cuenta con una cantidad de información relevante sin caer en la pérdida de ésta. La métrica de Manhattan presenta pésimos resultados a la hora de generar la gráfica de similitud, esto se debe a que, a diferencia de las demás métricas, ésta no se maneja en un rango entre 0 y 1 donde, 0 es que no existe similitud y 1 es que la similitud es total, sino que 0 es la similitud total, y entre más grande es el valor, mayor es la diferencia entre los documentos.

### 4.3. Pruebas de Clasificación

Para las pruebas de clasificación se utilizó el software WEKA de la Universidad de Waikato [40] el cual provee de diferentes técnicas de clasificación. Los métodos de clasificación seleccionados para las pruebas fueron, Naive Bayes y Máquinas de Vectores de Soporte. Las pruebas se realizaron para la muestra de 1600 documentos y 4000 documentos utilizando los umbrales presentados en las gráficas de la sección 4.2.1 y 4.2.2

#### 4.3.1. Clasificación 1600 documentos

En la tabla 4.1 se presentan los resultados obtenidos por los clasificadores bajo diferentes umbrales de frecuencia para la prueba con 1600 documentos. En el apéndice B se presentan los registros de resultados de cada clasificador. Como se puede apreciar, la precisión del clasificador disminuye, conforme aumenta la cantidad de palabras descartadas por el umbral de frecuencia.

Tabla 4.1: Resultados clasificación 1600 documentos.

Clasificador	Entrenamiento	Prueba	Precisión	Recuerdo	F-Measure
Naive Bayes Original	280	120	0.834	0.827	0.826
SVM Original	280	120	0.822	0.802	0.806
Naive Bayes TH2	280	120	0.725	0.684	0.691
SVM TH2	280	120	0.735	0.697	0.705
Naive Bayes TH3	280	120	0.739	0.602	0.619
SVM TH3	280	120	0.747	0.623	0.642
Naive Bayes TH4	280	120	0.734	0.604	0.603
SVM TH4	280	120	0.76	0.619	0.618

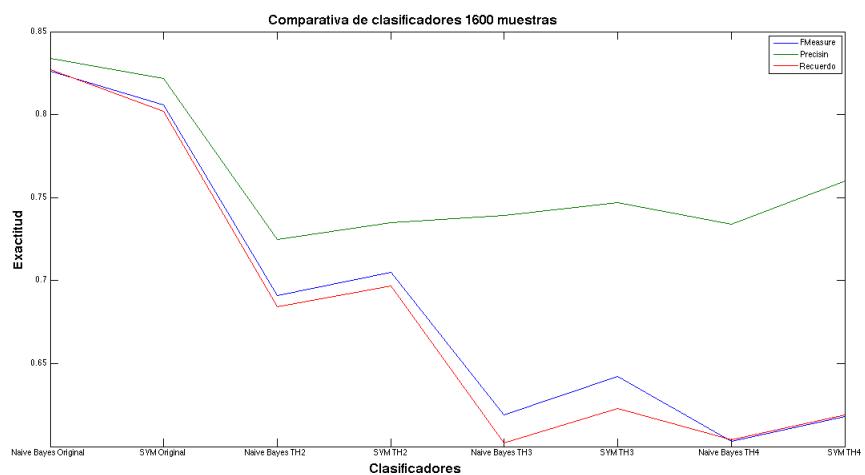


Figura 4.49: Exactitud de los clasificadores para 1600 documentos.

### 4.3.2. Clasificación 4000 documentos

En la tabla 4.2 se presentan los resultados obtenidos por los clasificadores bajo diferentes umbrales de frecuencia para la prueba con 4000 documentos. En el apéndice C se presentan los registros de resultados de cada clasificador. Como se puede apreciar, la precisión del clasificador disminuye, conforme aumenta la cantidad de palabras descartadas por el umbral de frecuencia. También se puede apreciar, que para este volumen de información, las Máquinas de vectores de soporte, presentan un mejor desempeño a la hora de clasificar.

Tabla 4.2: Resultados clasificación 4000 documentos.

Clasificador	Entrenamiento	Prueba	Precisión	Recuerdo	F-Measure
Naive Bayes Original	700	300	0.805	0.791	0.794
SVM Original	700	300	0.856	0.851	0.853
Naive Bayes TH2	700	300	0.701	0.635	0.643
SVM TH2	700	300	0.748	0.727	0.73
Naive Bayes TH3	700	300	0.728	0.666	0.672
SVM TH3	700	300	0.763	0.732	0.736
Naive Bayes TH4	700	300	0.768	0.698	0.653
SVM TH4	700	300	0.813	0.762	0.747

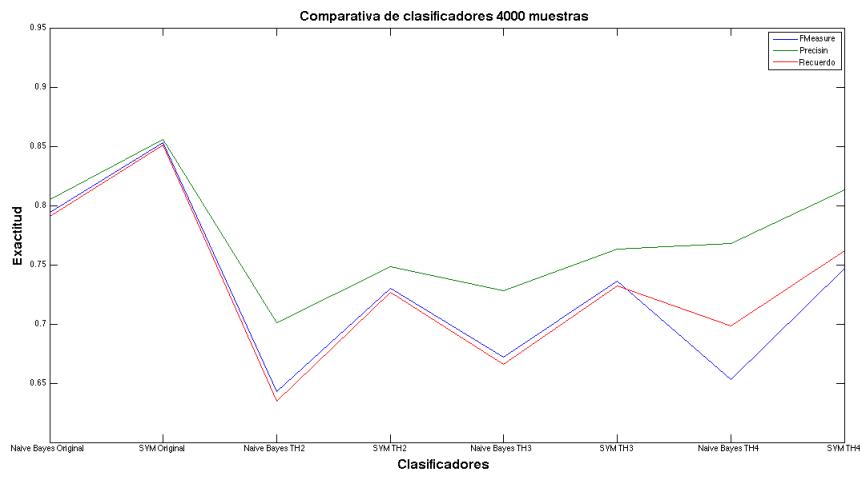


Figura 4.50: Exactitud de los clasificadores para 4000 documentos.

# Capítulo 5

## Conclusiones

Con el desarrollo de este proyecto se demostró que el resultado presentado por las gráficas de similitud, nos permite saber de una manera previa, cómo se comportará el clasificador. A su vez se pudo apreciar cómo el filtrado de información mediante un umbral de frecuencia puede contribuir a un mejor desempeño, siempre y cuando no se llegue a un grado donde exista una pérdida de información relevante.

En cuanto a las métricas implementadas, se puede apreciar que la métrica de Manhattan presenta pésimos resultados a la hora de generar las gráficas de similitud, esto debido a que su criterio para el 100 % de similitud es el 0 y no existe una normalización de la información en un rango específico, aunque se realizo programáticamente una normalización donde se tomó como 1 el valor máximo que representa el mayor grado de diferencia y posteriormente se aplicó una operación de negativo no se presentaron los resultados esperados. Para las métricas de Cosenos, Dice y Jaccard se puede apreciar que la métrica con mejores resultados fue la de Cosenos al mostrar una mejor nitidez ante las otras bajo el mismo conjunto de información.

Por la parte del desarrollo, El uso de las bibliotecas Boost y Qt5 contribuyeron a tener un desarrollo más eficiente al proveer de módulos para el desarrollo de la interfaz gráfica y el manejo de expresiones regulares, los cuales de otra forma hubieran implicado invertir un mayor tiempo en cuanto al desarrollo del programa.

# Trabajo Futuro

Con la finalidad de darle continuidad a este proyecto, se consideran realizar las siguientes pruebas y mejoras:

- Pruebas de clasificación en dominios cruzados.
- Pruebas de clasificación en corpus de multilenguaje.
- Mejoras en la interacción de la interfaz de usuario.

El objetivo de realizar estas mejoras, es el probar el rendimiento de los clasificadores bajo diferentes dominios una vez que se hayan realizado los análisis de las gráficas de similitud. En cuanto a la mejora en la interacción de la interfaz de usuario, se buscará dar soporte a más formas de presentar la información al sistema así como de buscar mejorar la experiencia del usuario al momento de utilizar la herramienta.



# Apéndice A

## Bibliotecas Boost

Tabla A.1: Listado de algunas bibliotecas Boost y su funcionalidad.

Nombre	Función
Boost.Accumulators	Ofrece acumuladores a los que pueden añadirse números para obtener, por ejemplo, la media o la desviación estándar.
Boost.Algorithm	Proporciona varios algoritmos que complementan los algoritmos de la librería estándar.
Boost.Aasio	Permite desarrollar aplicaciones de red en las cuales los datos trabajen de forma asíncrona.
Boost.Graph	Proporciona algoritmos para operaciones tales como encontrar el camino más corto entre dos puntos en un grafo.
Boost.Interprocess	Utiliza memoria compartida para ayudar a las aplicaciones comunicarse de forma rápida y eficiente.
Boost.Random	Proporciona generadores de números aleatorios.
Boost.Regex	Proporciona funciones para buscar cadenas mediante expresiones regulares.
Boost.Spirit	Permite generar programas de análisis utilizando una sintaxis similar a EBNF.
Boost.Thread	Permite crear aplicaciones multihilo.
Boost.Timer	Define relojes para medir el rendimiento del código.
Boost.Tokenizer	Permite iterar sobre tokens en una cadena.
Boost.TypeTraits	Proporciona funciones para comprobar propiedades de los tipos.
Boost.Unordered	Proporciona dos contenedores hash: boost::unordered_set y boost::unordered_map.
Boost.Utility	Engloba una serie de funciones las cuales son muy pequeñas para tener su propia biblioteca.
Boost.Uuid	Define la clase boost::uuids::uuid y generadores para crear UUIDs.

## Apéndice B

### Registros de resultados de los clasificadores 1600 documentos

Listing B.1: Navie Bayes 1600 documentos.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	396
82.6722 %	
Incorrectly Classified Instances	83
17.3278 %	
Kappa statistic	0.7689
Mean absolute error	0.0906
Root mean squared error	0.2615
Relative absolute error	24.1507 %
Root relative squared error	60.3937 %
Total Number of Instances	479

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.875 books	0.045	0.868	0.875	0.871	0.982
0.883 dvd	0.033	0.898	0.883	0.891	0.977
0.672 electronics	0.039	0.851	0.672	0.751	0.952
0.875 kitchen	0.114	0.719	0.875	0.789	0.959
Weighted Avg.	0.827	0.058	0.834	0.827	
0.826	0.967				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
105	7	2	6	a = books
7	106	3	4	b = dvd
7	1	80	31	c = electronics
2	4	9	105	d = kitchen

Listing B.2: SVM 1600 documentos.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	384
80.167 %	
Incorrectly Classified Instances	95
19.833 %	
Kappa statistic	0.7355
Mean absolute error	0.274
Root mean squared error	0.3481
Relative absolute error	73.0689 %
Root relative squared error	80.3899 %
Total Number of Instances	479

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.867 books	0.017	0.945	0.867	0.904	0.942
0.842 dvd	0.011	0.962	0.842	0.898	0.925
0.664 electronics	0.072	0.752	0.664	0.705	0.864
0.833 kitchen	0.164	0.629	0.833	0.717	0.864
Weighted Avg.	0.802	0.066	0.822	0.802	
0.806	0.899				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
104	3	4	9	a = books
4	101	2	13	b = dvd
2	1	79	37	c = electronics
0	0	20	100	d = kitchen

Listing B.3: Navie Bayes 1600 documentos, umbral de 2.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	269
68.4478 %	
Incorrectly Classified Instances	124
31.5522 %	
Kappa statistic	0.5803
Mean absolute error	0.1965
Root mean squared error	0.3261
Relative absolute error	52.4552 %
Root relative squared error	75.3591 %
Total Number of Instances	393

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
0.676	0.014	0.947	0.676	0.789	0.925
books					
0.771	0.097	0.743	0.771	0.757	0.934
dvd					
0.527	0.08	0.671	0.527	0.59	0.87
electronics					
0.756	0.224	0.5	0.756	0.602	0.866
kitchen					
Weighted Avg.		0.684	0.1	0.725	0.684
0.691	0.901				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
71	10	6	18	a = books
1	81	5	18	b = dvd
2	10	49	32	c = electronics
1	8	13	68	d = kitchen

Listing B.4: SVM 1600 documentos, umbral de 2.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	274
69.7201 %	
Incorrectly Classified Instances	119
30.2799 %	
Kappa statistic	0.5977
Mean absolute error	0.2947
Root mean squared error	0.376
Relative absolute error	78.682 %
Root relative squared error	86.9015 %
Total Number of Instances	393

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.648 books	0.021	0.919	0.648	0.76	0.839
0.752 dvd	0.063	0.814	0.752	0.782	0.855
0.645 electronics	0.11	0.645	0.645	0.645	0.801
0.744 kitchen	0.205	0.519	0.744	0.612	0.819
Weighted Avg.		0.697	0.095	0.735	0.697
0.705	0.83				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
68	10	5	22	a = books
2	79	9	15	b = dvd
3	5	60	25	c = electronics
1	3	19	67	d = kitchen

Listing B.5: Navie Bayes 1600 documentos, umbral de 3.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	142
60.1695 %	
Incorrectly Classified Instances	94
39.8305 %	
Kappa statistic	0.4789
Mean absolute error	0.2303
Root mean squared error	0.3391
Relative absolute error	62.121 %
Root relative squared error	78.7777 %
Total Number of Instances	236

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
0.553	0.025	0.913	0.553	0.689	0.91
books					
0.552	0.047	0.822	0.552	0.661	0.893
dvd					
0.438	0.048	0.7	0.438	0.538	0.873
electronics					
0.933	0.382	0.365	0.933	0.525	0.891
kitchen					
Weighted Avg.	0.602	0.104	0.739	0.602	
0.619	0.894				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
42	7	3	24	a = books
2	37	3	25	b = dvd
2	1	21	24	c = electronics
0	0	3	42	d = kitchen

Listing B.6: SVM 1600 documentos, umbral de 3.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	147
62.2881 %	
Incorrectly Classified Instances	89
37.7119 %	
Kappa statistic	0.5046
Mean absolute error	0.309
Root mean squared error	0.3951
Relative absolute error	83.3451 %
Root relative squared error	91.7853 %
Total Number of Instances	236

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.618 books	0.031	0.904	0.618	0.734	0.815
0.582 dvd	0.03	0.886	0.582	0.703	0.841
0.438 electronics	0.059	0.656	0.438	0.525	0.744
0.889 kitchen	0.356	0.37	0.889	0.523	0.759
Weighted Avg.	0.623	0.098	0.747	0.623	
0.642	0.797				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
47	3	2	24	a = books
2	39	5	21	b = dvd
3	1	21	23	c = electronics
0	1	4	40	d = kitchen

Listing B.7: Navie Bayes 1600 documentos, umbral de 4.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	84
60.4317 %	
Incorrectly Classified Instances	55
39.5683 %	
Kappa statistic	0.4389
Mean absolute error	0.2347
Root mean squared error	0.3407
Relative absolute error	64.3628 %
Root relative squared error	79.815 %
Total Number of Instances	139

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
books	0.033	0.903	0.596	0.718	0.88
dvd	0.5	0.442	0.884	0.589	0.867
electronics	0.009	0.889	0.333	0.485	0.84
kitchen	0.026	0.769	0.4	0.526	0.874
Weighted Avg.	0.604	0.172	0.734	0.604	
	0.603	0.868			

==== Confusion Matrix ====

a	b	c	d	<-- classified as
28	18	0	1	a = books
2	38	1	2	b = dvd
1	15	8	0	c = electronics
0	15	0	10	d = kitchen

Listing B.8: SVM 1600 documentos, umbral de 4.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	86
61.8705 %	
Incorrectly Classified Instances	53
38.1295 %	
Kappa statistic	0.4597
Mean absolute error	0.3088
Root mean squared error	0.3948
Relative absolute error	84.6749 %
Root relative squared error	92.4853 %
Total Number of Instances	139

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.574 books	0.022	0.931	0.574	0.711	0.812
0.93 dvd	0.5	0.455	0.93	0.611	0.717
0.417 electronics	0.009	0.909	0.417	0.571	0.701
0.36 kitchen	0.018	0.818	0.36	0.5	0.809
Weighted Avg.		0.619	0.167	0.76	0.619
0.618	0.763				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
27	19	0	1	a = books
1	40	1	1	b = dvd
1	13	10	0	c = electronics
0	16	0	9	d = kitchen

## Apéndice C

### Registros de resultados de los clasificadores 4000 documentos

Listing C.1: Navie Bayes 4000 documentos.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	949
79.0833 %	
Incorrectly Classified Instances	251
20.9167 %	
Kappa statistic	0.7211
Mean absolute error	0.1135
Root mean squared error	0.2842
Relative absolute error	30.277 %
Root relative squared error	65.6262 %
Total Number of Instances	1200

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.903 books	0.02	0.938	0.903	0.92	0.988
0.767 dvd	0.02	0.927	0.767	0.839	0.952
0.673 electronics	0.103	0.685	0.673	0.679	0.905
0.82 kitchen	0.136	0.668	0.82	0.737	0.933
Weighted Avg.		0.791	0.07	0.805	0.791
0.794	0.945				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
271	10	9	10	a = books
13	230	37	20	b = dvd
2	4	202	92	c = electronics
3	4	47	246	d = kitchen

Listing C.2: SVM 4000 documentos.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	1021
85.0833 %	
Incorrectly Classified Instances	179
14.9167 %	
Kappa statistic	0.8011
Mean absolute error	0.2668
Root mean squared error	0.3373
Relative absolute error	71.1481 %
Root relative squared error	77.8967 %
Total Number of Instances	1200

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.907 books	0.016	0.951	0.907	0.928	0.969
0.863 dvd	0.021	0.932	0.863	0.896	0.941
0.81 electronics	0.083	0.764	0.81	0.786	0.901
0.823 kitchen	0.079	0.777	0.823	0.799	0.91
Weighted Avg.		0.851	0.05	0.856	0.851
0.853	0.93				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
272	12	7	9	a = books
8	259	20	13	b = dvd
3	5	243	49	c = electronics
3	2	48	247	d = kitchen

Listing C.3: Navie Bayes 4000 documentos, umbral de 2.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	530
63.4731 %	
Incorrectly Classified Instances	305
36.5269 %	
Kappa statistic	0.5076
Mean absolute error	0.2246
Root mean squared error	0.3443
Relative absolute error	60.1497 %
Root relative squared error	79.7594 %
Total Number of Instances	835

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.709 books	0.023	0.935	0.709	0.807	0.934
0.462 dvd	0.039	0.784	0.462	0.581	0.867
0.536 electronics	0.11	0.519	0.536	0.528	0.822
0.766 kitchen	0.313	0.47	0.766	0.582	0.865
Weighted Avg.		0.635	0.12	0.701	0.635
0.643	0.88				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
188	7	11	59	a = books
8	91	22	76	b = dvd
2	11	81	57	c = electronics
3	7	42	170	d = kitchen

Listing C.4: SVM 4000 documentos, umbral de 2.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	607
72.6946 %	
Incorrectly Classified Instances	228
27.3054 %	
Kappa statistic	0.6309
Mean absolute error	0.2882
Root mean squared error	0.3675
Relative absolute error	77.1889 %
Root relative squared error	85.151 %
Total Number of Instances	835

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.789 books	0.033	0.917	0.789	0.848	0.9
0.68 dvd	0.053	0.798	0.68	0.734	0.848
0.543 electronics	0.077	0.607	0.543	0.573	0.817
0.82 kitchen	0.199	0.599	0.82	0.692	0.83
Weighted Avg.		0.727	0.09	0.748	0.727
	0.73	0.854			

==== Confusion Matrix ====

a	b	c	d	<-- classified as
209	16	8	32	a = books
10	134	12	41	b = dvd
6	14	82	49	c = electronics
3	4	33	182	d = kitchen

Listing C.5: Navie Bayes 4000 documentos, umbral de 3.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	263
66.5823 %	
Incorrectly Classified Instances	132
33.4177 %	
Kappa statistic	0.5085
Mean absolute error	0.2338
Root mean squared error	0.3428
Relative absolute error	64.5391 %
Root relative squared error	81.3019 %
Total Number of Instances	395

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
books	0.037	0.938	0.663	0.777	0.93
dvd	0.049	0.644	0.433	0.518	0.866
electronics	0.049	0.419	0.481	0.448	0.862
kitchen	0.327	0.529	0.842	0.65	0.91
Weighted Avg.	0.672	0.666	0.128	0.728	0.666
	0.908				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
120	8	3	50	a = books
3	29	4	31	b = dvd
2	3	13	9	c = electronics
3	5	11	101	d = kitchen

Listing C.6: SVM 4000 documentos, umbral de 3.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	289
73.1646 %	
Incorrectly Classified Instances	106
26.8354 %	
Kappa statistic	0.6026
Mean absolute error	0.2869
Root mean squared error	0.3626
Relative absolute error	79.206 %
Root relative squared error	85.9977 %
Total Number of Instances	395

==== Detailed Accuracy By Class ====

TP Rate Class	FP Rate	Precision	Recall	F-Measure	ROC Area
0.757 books	0.047	0.932	0.757	0.835	0.892
0.552 dvd	0.049	0.698	0.552	0.617	0.746
0.481 electronics	0.049	0.419	0.481	0.448	0.791
0.85 kitchen	0.225	0.622	0.85	0.718	0.831
Weighted Avg.		0.732	0.102	0.763	0.732
0.736	0.842				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
137	8	2	34	a = books
6	37	6	18	b = dvd
1	3	13	10	c = electronics
3	5	10	102	d = kitchen

Listing C.7: Navie Bayes 4000 documentos, umbral de 4.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	132
69.8413 %	
Incorrectly Classified Instances	57
30.1587 %	
Kappa statistic	0.3597
Mean absolute error	0.2212
Root mean squared error	0.3278
Relative absolute error	64.1212 %
Root relative squared error	80.7017 %
Total Number of Instances	189

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
books	0.965	0.613	0.705	0.965	0.931
dvd	0.375	0.052	0.4	0.375	0.863
electronics	1	0.011	0.333	1	0.968
kitchen	0.259	0	1	0.259	0.908
Weighted Avg.	0.653	0.918	0.698	0.374	0.768
					0.698

==== Confusion Matrix ====

a	b	c	d	<-- classified as
110	4	0	0	a = books
10	6	0	0	b = dvd
0	0	1	0	c = electronics
36	5	2	15	d = kitchen

Listing C.8: SVM 4000 documentos, umbral de 4.

==== Evaluation on test set ====

==== Summary ====

Correctly Classified Instances	144
76.1905 %	
Incorrectly Classified Instances	45
23.8095 %	
Kappa statistic	0.5159
Mean absolute error	0.276
Root mean squared error	0.3511
Relative absolute error	80.0187 %
Root relative squared error	86.4308 %
Total Number of Instances	189

==== Detailed Accuracy By Class ====

TP Rate	FP Rate	Precision	Recall	F-Measure	ROC Area
Class					
books	0.467	0.757	0.956	0.845	0.742
dvd	0.035	0.571	0.5	0.533	0.645
electronics	0.021	0.2	1	0.333	0.989
kitchen	0	1	0.448	0.619	0.856
Weighted Avg.	0.762	0.285	0.813	0.762	
0.747	0.77				

==== Confusion Matrix ====

a	b	c	d	<-- classified as
109	3	2	0	a = books
8	8	0	0	b = dvd
0	0	1	0	c = electronics
27	3	2	26	d = kitchen

# Bibliografía

- [1] Zhihang Chen, Chengwen Ni, and Y.L. Murphey. Neural network approaches for text document categorization. In *Neural Networks, 2006. IJCNN '06. International Joint Conference on*, pages 1054–1060, 2006.
- [2] Kamal Nigam, Andrew Kachites McCallum, Sebastian Thrun, and Tom Mitchell. Text classification from labeled and unlabeled documents using em. *Mach. Learn.*, 39(2-3):103–134, May 2000.
- [3] John Blitzer, Mark Dredze, and Fernando Pereira. Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205, 2007.
- [4] N. de Oliveira Gomes and E.P.L. Passos. Text categorization study case: Patents' application documents. In *Industrial Electronics and Applications (ICIEA), 2011 6th IEEE Conference on*, pages 446–450, June 2011.
- [5] Aurajo B. S. *Aprendizaje Automático: Conceptos básicos y avanzados*. Pearson-Prentice Hall, 2006.
- [6] Tom M. Mitchell. *Machine Learning*. McGraw-Hill Science/Engineering/Math, 1997.
- [7] G.J. Myatt. *Making Sense of Data: A Practical Guide to Exploratory Data Analysis and Data Mining*. Wiley, 2007.
- [8] Siddharth Patwardhan and Ellen Riloff. Effective information extraction with semantic affinity patterns and relevant regions. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-2007)*, pages 103–110, 2007.

- CoNLL*), pages 717–727, Prague, Czech Republic, June 2007. Association for Computational Linguistics.
- [9] Óscar Ferrández, Rafael Muñoz, and Manuel Palomar. Improving question answering tasks by textual entailment recognition. In Epaminondas Kapetanios, Vijayan Sugumaran, and Myra Spiliopoulou, editors, *Natural Language and Information Systems*, volume 5039 of *Lecture Notes in Computer Science*, pages 339–340. Springer Berlin Heidelberg, 2008.
  - [10] C. Namrata Korde, Vandana; Mahender. Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications; Mar 2012, Vol. 3 Issue 2, p85*, 2012.
  - [11] David D. Lewis. Naive (bayes) at forty: The independence assumption in information retrieval. In *Proceedings of the 10th European Conference on Machine Learning*, ECML '98, pages 4–15, London, UK, UK, 1998. Springer-Verlag.
  - [12] Yiming Yang and Xin Liu. A re-examination of text categorization methods. In *Proceedings of the 22Nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 42–49, New York, NY, USA, 1999. ACM.
  - [13] D. Jurafsky and Martin J. H. *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition*. Prentice Hall, 2000.
  - [14] Giorgio Maria Di Nunzio. Using scatterplots to understand and improve probabilistic models for text categorization and retrieval. *International Journal of Approximate Reasoning*, 50(7):945 – 956, 2009. Special Section on Graphical Models and Information Retrieval.
  - [15] Salahideen Mousa Samer Al Hawari Ghassan Kanaan Wa‘el Musa Hadi, Fadi Thabtah and Jafar Ababnih. A comprehensive comparative study using vector space model with k-nearest neighbor on text categorization data. *Asian Journal of Information Management*, 2: 14-22., 2008.
  - [16] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag New York, Inc., New York, NY, USA, 1995.

- [17] Thorsten Joachims. Text categorization with support vector machines: Learning with many relevant features. In Claire Nédellec and Céline Rouveiro, editors, *Machine Learning: ECML-98*, volume 1398 of *Lecture Notes in Computer Science*, pages 137–142. Springer Berlin Heidelberg, 1998.
- [18] Please explain support vector machines (svm) like i am a 5 year old. [http://www.reddit.com/r/MachineLearning/comments/15zrpp/please\\_explain\\_support\\_vector\\_machines\\_like\\_i\\_am\\_a\\_5\\_year\\_old/](http://www.reddit.com/r/MachineLearning/comments/15zrpp/please_explain_support_vector_machines_like_i_am_a_5_year_old/)
- [19] M. A Hearst, B Schölkopf, S Dumais, E Osuna, and J Platt. Trends and controversies - support vector machines. *IEEE Intelligent Systems*, 13(4):18–28, 1998.
- [20] S.J. Russell and P. Norvig. *Inteligencia artificial: un enfoque moderno*. Colección de Inteligencia Artificial de Prentice Hall. Pearson Educación, 2004.
- [21] G. y M. J. McGill. Salton. *Introduction to Modern Information Retrieval*. McGraw Hill, 1983.
- [22] W. B. y Baeza-Yates Frakes. *Information Retrieval: Data structures and Algorithms*. Englewood Cliffs, Prentice Hall, 1992.
- [23] Paul Jaccard. Étude comparative de la distribution florale dans une portion des Alpes et des Jura. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:547–579, 1901.
- [24] Ekkachai Naenudorn Suphakit Niwattanakul, Jatsada Singthongchai and Supachanun Wanapu. Using of jaccard coefficient for keywords similarity. *Proceedings of the International MultiConference of Engineers and Computer Scientists 2013 Vol I, IMECS 2013, March 13 - 15, 2013, Hong Kong*, 2013.
- [25] Bjarne stroustrup's homepage. <http://www.stroustrup.com>.
- [26] Microsoft site. <http://www.microsoft.com/>.
- [27] Matlab site. <http://www.mathworks.com/products/matlab/>.
- [28] Windows site. <http://windows.microsoft.com>.
- [29] Linux foundation site. <http://www.linuxfoundation.org>.

- [30] Apple site. <https://www.apple.com>.
- [31] Robert Lafore. *Object-Oriented Programming in C++ (4th Edition)*. Sams Publishing, 4 edition, 2001.
- [32] C++ boost library. <http://www.boost.org>.
- [33] Stl containers reference. <http://www.cplusplus.com/reference/stl>.
- [34] Stl algorithms reference. <http://www.cplusplus.com/reference/algorithm>.
- [35] Boris Schäling. *The Boost C++ Libraries*. XML Press, 2014.
- [36] Qt project site. <http://qt-project.org/>.
- [37] Qcustomplot site. <http://www.qcustomplot.com/>.
- [38] Multi-domain sentiment dataset (version 2.0).  
<http://www.cs.jhu.edu/~mdredze/datasets/sentiment/>.
- [39] Amazon site. <http://www.amazon.com>.
- [40] Weka 3: Data mining software in java.  
<http://www.cs.waikato.ac.nz/ml/weka/>.